



Hardware and Software
Engineered to Work Together



Custom CW: Oracle WebLogic Server 12c Administration

Student Guide - Volume 1

X95181GC10

Edition 1.0 | May 2016

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

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Disclaimer

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

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

Restricted Rights Notice

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

U.S. GOVERNMENT RIGHTS

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

Trademark Notice

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

Tony Albrecht (base2tech@yahoo.com) has a non-transferable
license to use this Student Guide.

Table of Contents

| | |
|--|-----------|
| Chapter 1: Lesson 1 - Course Overview | 9 |
| Objectives - 1 | 10 |
| Objectives - 2 | 11 |
| Target Audience | 12 |
| Introductions | 13 |
| Course Schedule | 14 |
| Course Practices | 16 |
| Classroom Guidelines | 17 |
| For More Information | 18 |
| Related Training | 19 |
| Chapter 2: Lesson 2 - WebLogic Server: Overview | 21 |
| Objectives | 22 |
| Distributed Systems | 23 |
| Java Platform Enterprise Edition | 24 |
| Oracle WebLogic Server | 25 |
| JVM | 27 |
| (Possible) System Architecture | 28 |
| WebLogic Server Domain | 29 |
| Administration Server | 30 |
| Managed Servers | 31 |
| Node Manager | 32 |
| Machines and Clusters | 33 |
| WebLogic Server Application Services | 34 |
| WebLogic Server Application: Example | 35 |
| WebLogic Server Administrative Tools | 36 |
| WebLogic Server Administration Console | 38 |
| WLST | 39 |
| WLDF | 42 |
| WLDF Monitoring Dashboard | 43 |
| Enterprise Manager Cloud Control | 44 |
| Quiz | 45 |
| Summary | 47 |
| Chapter 3: Lesson 3 - Installing and Patching WebLogic Server | 49 |
| Objectives | 50 |
| Determining Supported System Configurations | 51 |
| Ensuring Your System Meets Requirements | 52 |
| When Not All FMW Is the Same Version | 53 |
| WebLogic Server Installers | 54 |
| Generic Installers | 55 |
| What Is Oracle Coherence? | 57 |
| FMW Installation Flow | 58 |
| WebLogic Server Installation Modes | 59 |
| Installing WebLogic Server on Linux (Graphical Mode) | 60 |
| Installation Problems | 66 |

| | |
|--|------------|
| Sample Installation Directory Structure | 67 |
| Uninstalling WebLogic Server | 68 |
| Applying Patches by Using OPatch | 69 |
| Quiz | 70 |
| Summary | 72 |
| Practice 3-1 Overview: Installing WebLogic Server | 73 |
| Practice 3-2 Overview: Patching WebLogic Server | 74 |
| Chapter 4: Lesson 4 - Creating Domains | 75 |
| Objectives | 76 |
| Domain Planning Questions | 77 |
| Virtual IP Address and Virtual Host Name | 80 |
| Domain Mode: Development | 81 |
| Domain Mode: Production | 82 |
| Domain Creation Tools | 83 |
| Domains Are Created from Templates | 84 |
| Creating Domains | 85 |
| Where to Place the Domain | 86 |
| Creating a Domain with the Configuration Wizard | 87 |
| Admin Server Listen Address | 94 |
| Creating a Domain with the Configuration Wizard | 95 |
| Domain File Structure | 103 |
| Creating a Domain to Support FMW Components | 104 |
| The Domain on Other Hardware | 106 |
| Creating the Domain Archive: Pack | 107 |
| Using the Domain Archive: Unpack | 108 |
| Quiz | 109 |
| Summary | 111 |
| Practice 4-1 Overview: Creating a New Domain | 112 |
| Practice 4-2 Overview: Copying a Domain to a New Machine | 113 |
| Chapter 5: Lesson 5 - Starting Servers | 115 |
| Objectives | 116 |
| WebLogic Server Life Cycle | 117 |
| Starting WebLogic Server with a Script | 119 |
| Customizing the Scripting Environment | 120 |
| Creating a Boot Identity File | 122 |
| Stopping WebLogic Server | 123 |
| Suspend and Resume | 124 |
| Customizing Standard Scripts | 125 |
| WebLogic Server Options | 126 |
| Changing the JVM | 128 |
| JVM Options | 129 |
| Modifying the CLASSPATH | 130 |
| WebLogic Server Startup Issues | 133 |
| Failed Admin Server | 134 |
| Restarting a Failed Admin Server: Same Machine | 135 |
| Restarting a Failed Admin Server: Different Machine | 136 |
| Restarting a Failed Managed Server: Same Machine | 137 |
| Restarting a Failed Managed Server: Different Machine | 138 |

| | |
|---|------------|
| Quiz | 139 |
| Summary | 141 |
| Practice 5-1 Overview: Starting and Stopping Servers | 142 |
| Chapter 6: Lesson 7 - Configuring JDBC | 143 |
| Objectives | 144 |
| JDBC: Overview | 145 |
| WebLogic JDBC Drivers | 146 |
| Global Transactions: Overview | 147 |
| Two-Phase Commit | 148 |
| JDBC Data Source | 149 |
| Java Naming and Directory Interface (JNDI) | 151 |
| JNDI Duties of an Administrator | 152 |
| Deployment of a Data Source | 153 |
| Targeting of a Data Source | 154 |
| Types of Data Sources | 155 |
| Creating a Generic Data Source | 156 |
| Non-XA Driver Transaction Options | 159 |
| Creating a Generic Data Source | 160 |
| Connection Pool Configuration | 163 |
| Connection Properties | 165 |
| Testing a Generic Data Source | 166 |
| Oracle Real Application Clusters: Overview | 167 |
| GridLink Data Source for RAC | 168 |
| GridLink | 169 |
| GridLink and Services | 170 |
| GridLink and Single Client Access Name (SCAN) | 171 |
| Creating a GridLink Data Source | 172 |
| Common Data Source Problems | 178 |
| Basic Connection Pool Tuning | 182 |
| Quiz | 185 |
| Summary | 187 |
| Practice 7-1 Overview: Configuring a JDBC Data Source | 188 |
| Chapter 7: Lesson 8 - Monitoring a Domain | 189 |
| Objectives | 190 |
| WebLogic Server Logs | 191 |
| WebLogic Server Log Locations | 193 |
| Log Message Severity Levels | 194 |
| Understanding Log File Entries | 196 |
| Accessing the Logs from the Admin Console | 197 |
| Configuring Server Logging | 199 |
| Error Messages Reference | 202 |
| Log Filters | 203 |
| Creating a Log Filter | 204 |
| Applying a Log Filter | 207 |
| Subsystem Debugging | 208 |
| Debug Scopes | 209 |
| Debug Scopes: Examples | 210 |
| Admin Console: Monitoring Domain Resources | 211 |

| | |
|---|------------|
| Monitoring the Domain | 212 |
| Monitoring All Servers | 213 |
| Monitoring Server Health | 214 |
| Monitoring Server Performance | 215 |
| Monitoring Data Source Health | 216 |
| Example Data Source Performance Attributes | 217 |
| JMX | 218 |
| Monitoring Dashboard | 219 |
| Monitoring Dashboard Interface | 220 |
| Views | 221 |
| Built-in Views | 222 |
| Creating a Custom View | 223 |
| Anatomy of a Chart | 224 |
| Current or Historical Data | 225 |
| Quiz | 226 |
| Summary | 228 |
| Practice 8-1 Overview: Working with WebLogic Server Logs | 229 |
| Practice 8-2 Overview: Monitoring WebLogic Server | 230 |
| Chapter 8: Lesson 9 - Node Manager | 231 |
| Objectives | 232 |
| Node Manager | 233 |
| Two Types of Node Manager | 235 |
| Node Manager Architecture: Per Machine | 236 |
| Node Manager Architecture: Per Domain | 237 |
| How Node Manager Starts a Managed Server | 238 |
| How Node Manager Can Help Shut Down a Managed Server | 239 |
| Configuration Wizard and Node Manager | 240 |
| Configuring the Java-Based Node Manager | 242 |
| Configuring Server Start and Health Monitoring Parameters | 243 |
| Configuring the Java-Based Node Manager | 245 |
| Other Node Manager Properties | 247 |
| Node Manager Files | 248 |
| Enrolling Node Manager with a Domain | 251 |
| When Not to Use nmEnroll() | 252 |
| Reminder: Pack | 253 |
| Reminder: Unpack | 254 |
| Controlling Servers Through Node Manager | 255 |
| Node Manager: Best Practices | 256 |
| Quiz | 258 |
| Summary | 260 |
| Practice 9-1 Overview: Configuring and Using Node Manager | 261 |
| Chapter 9: Lesson 10 - Deploying Applications | 263 |
| Objectives | 264 |
| Deploying Applications to WebLogic Server | 265 |
| Software Life Cycle and WebLogic Server | 266 |
| Java EE Deployments | 267 |
| WebLogic Server Deployments | 268 |
| Other Deployments | 269 |

| | |
|---|------------|
| Deployment Terms | 271 |
| Deployment Descriptors | 274 |
| Deployment Plans | 275 |
| Exploded Versus Archived Applications | 276 |
| Autodeploy | 277 |
| Server Staging Mode | 278 |
| WebLogic Server Deployment Tools | 279 |
| Starting and Stopping an Application | 281 |
| Deploying an Application | 283 |
| Undeploying an Application | 288 |
| Redeploying an Application | 290 |
| Monitoring Deployed Applications: Admin Console | 292 |
| Monitoring Information Available from the Admin Console | 293 |
| Monitoring Deployed Applications: Monitoring Dashboard | 294 |
| Application Errors | 295 |
| Application Testing | 296 |
| Performance Testing Methodology | 297 |
| Load and Stress Testing | 298 |
| Load Testing Tools | 299 |
| The Grinder | 300 |
| The Grinder Architecture | 301 |
| The Grinder Proxy | 302 |
| Agent Properties | 303 |
| The Grinder Console | 304 |
| Finding Bottlenecks | 305 |
| Correcting Bottlenecks | 306 |
| Quiz | 308 |
| Summary | 310 |
| Practice 10-1 Overview: Deploying an Application | 311 |
| Practice 10-2 Overview: Load Testing an Application | 312 |
| Chapter 10: Lesson 12 - Clusters – Overview, Creation, and Configuration | 313 |
| Objectives | 314 |
| Cluster: Review | 315 |
| Benefits of Clustering | 317 |
| Basic (Single-Tier) Cluster Architecture | 318 |
| Multi-Tier Cluster Architecture | 319 |
| Architecture Advantages and Disadvantages | 320 |
| Cluster Communication | 322 |
| Creating a Cluster: Configuration Wizard | 324 |
| Creating a Cluster: Administration Console | 325 |
| Adding Servers to the Cluster: Administration Console | 326 |
| Server Templates and Dynamic Clusters | 327 |
| Creating a Dynamic Cluster | 329 |
| Editing the New Dynamic Cluster | 333 |
| Editing the New Server Template | 334 |
| Dynamic Server Calculated Attributes | 335 |
| Dynamic Server Calculated Attributes: Example | 337 |
| Comparing Configured and Dynamic Clusters | 338 |

| | |
|--|------------|
| Creating a Server Template | 339 |
| Server Templates and Configured Servers | 341 |
| Quiz | 342 |
| Summary | 344 |
| Practice 12-1 Overview: Configuring a Cluster | 345 |
| Practice 12-2 Overview: Configuring a Dynamic Cluster | 346 |
| Chapter 11: Lesson 13 - Clusters – Proxies and Sessions | 347 |
| Objectives | 348 |
| A Cluster Proxy for a Web Application Cluster | 349 |
| Proxy Plug-Ins | 350 |
| Oracle HTTP Server (OHS) | 351 |
| Installing and Configuring OHS (Part of Oracle Web Tier): Overview | 352 |
| Configuring OHS as the Cluster Proxy | 354 |
| httpd.conf and mod_wl_ohs.conf | 355 |
| mod_wl_ohs.conf | 356 |
| Some Plug-in Parameters | 357 |
| Starting and Stopping OHS | 359 |
| Verifying that OHS Is Running | 360 |
| Failover: Detecting Failures and the Dynamic Server List | 361 |
| HTTP Session Failover | 363 |
| Configuring Web Application Session Failover: weblogic.xml | 364 |
| In-Memory Session Replication | 367 |
| In-Memory Replication: Example | 368 |
| Configuring In-Memory Replication | 371 |
| Machines | 372 |
| Secondary Server and Replication Groups | 373 |
| Replication Groups: Example | 374 |
| Configuring Replication Groups | 375 |
| File Session Persistence | 376 |
| Configuring File Persistence | 377 |
| JDBC Session Persistence | 378 |
| JDBC Session Persistence Architecture | 379 |
| Configuring JDBC Session Persistence | 380 |
| JDBC Persistent Table Configuration | 381 |
| Configuring a Hardware Load Balancer | 383 |
| Hardware Load Balancer Session Persistence | 384 |
| Passive Cookie Persistence and the WebLogic Server Session Cookie | 385 |
| Quiz | 386 |
| Summary | 387 |
| Practice 13-1 Overview: Installing OHS (Optional) | 388 |
| Practice 13-2 Overview: Configuring a Cluster Proxy | 389 |
| Practice 13-3 Overview: Configuring Replication Groups | 390 |
| Chapter 12: Lesson 14 - Clusters – Communication, Planning, and Troubleshooting | 391 |
| Objectives | 392 |
| Review: Cluster Communication | 393 |
| How Multicast Works | 394 |
| How Unicast Works | 395 |
| How Unicast Works (Notes Only) | 396 |

| | |
|---|-----|
| Considerations for Choosing Unicast or Multicast | 397 |
| Configure Multicast | 398 |
| Configure Unicast | 401 |
| Replication Channel | 402 |
| Configure Replication Channels: Servers | 403 |
| Configure Replication Channels: Cluster | 406 |
| Configure Replication Channels | 407 |
| Planning for a Cluster | 408 |
| Managing a Cluster | 412 |
| Troubleshooting a Cluster | 413 |
| Monitoring a Cluster: Admin Console | 414 |
| WebLogic Server and OHS Logs | 415 |
| Common OHS to WLS Connectivity Issues | 416 |
| Multicast Communication Issues | 418 |
| Cluster Member Uniformity | 419 |
| Session Failover Issues | 420 |
| Quiz | 421 |
| Summary | 422 |
| Practice 14-1 Overview: Configuring a Replication Channel | 423 |

Unauthorized reproduction or distribution prohibited. Copyright© 2016, Oracle and/or its affiliates.

Tony Albrecht (base2tech@yahoo.com) has a non-transferable
license to use this Student Guide.

1

Course Overview

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives - 1

After completing this course, you should be able to:

- Install WebLogic Server
- Create and configure basic WebLogic Server resources
- Start and stop WebLogic Server
- Monitor WebLogic Server resources
- Configure, start, and use Node Manager to remotely start WebLogic Server
- Configure automatic recovery from a machine crash
- Deploy Java Enterprise Edition applications to WebLogic Server
- Replace WebLogic Server's default authentication provider with an external LDAP product



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives - 2

After completing this course, you should be able to:

- Back up and restore a WebLogic domain
- Write scripts using the WebLogic Scripting Tool (WLST)
- Create users, groups, roles, and policies in WebLogic's embedded LDAP
- Configure WebLogic Diagnostic modules
- Configure Coherence features that are integrated with WebLogic



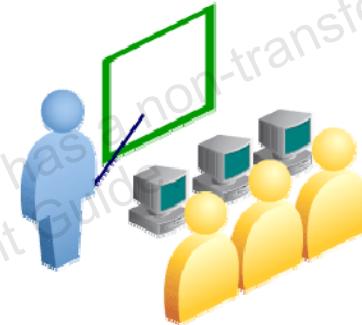
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Target Audience

This course is for administrators who will be responsible for administering Oracle WebLogic Server 12c.

Prerequisite skills include:

- Some system administration experience
- Basic knowledge of UNIX commands and navigation



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

If you are concerned about whether your experience fulfills the course prerequisites, ask the instructor.

Introductions

Introduce yourself and tell us about your:

- Company and role
- Experience with WebLogic Server
- Experience with other Oracle products

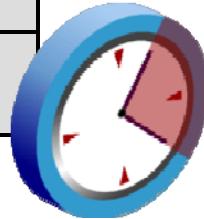


ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Course Schedule

| Day | | Ch | Lesson |
|-----|----|----|---|
| 1 | AM | 1 | 1. Course Overview |
| | | 2 | 2. WebLogic Server: Overview |
| | | 3 | 3. Installing and Patching WebLogic Server |
| | PM | 4 | 4. Creating Domains |
| | | 5 | 5. Starting Servers |
| 2 | AM | 6 | 7. Configuring JDBC |
| | | 7 | 8. Monitoring a Domain |
| 2 | PM | 8 | 9. Node Manager |
| | | 9 | 10. Deploying Applications |
| 3 | AM | 10 | 12. Clusters: Overview, Creation, and Configuration |



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The schedule might vary depending upon the instructor and the students in the class. The instructor will provide updates.

Course Schedule

Unauthorized reproduction or distribution prohibited. Copyright© 2016, Oracle and/or its affiliates.

| Day | Ch | Lesson |
|-----|----|--|
| 3 | PM | 11 12 13. Clusters: Proxies and Sessions 14. Clusters: Communication, Planning, and Troubleshooting |
| 4 | AM | 13 16. WebLogic Server Security 14 17. Backup, Recovery and Upgrading WebLogic Server |
| | PM | 15 5. WebLogic Server Startup and Crash Recovery 16 6. WebLogic Scripting Tool (WLST) |
| 5 | AM | 16 6. WebLogic Scripting Tool (WLST) (continued) 17 13. More on the Security Realm |
| | PM | 18 15. Diagnostic Framework 19 16. WebLogic and Coherence Integration |



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Course Practices

- Topics will be reinforced with hands-on exercises.
- Many exercises include a scripted solution to aid the students who fall behind.



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Any practice that is required by a later practice has a scripted solution.

Classroom Guidelines

- The instructor starts each session at the scheduled time.
- Ensure that cell phones are silent.
- When you ask questions, be respectful of the topic at hand and the interest of other students.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

We hope that these guidelines help the class proceed smoothly and enable you to get the maximum benefit from the course.

For More Information

| Topic | Website |
|--------------------------------|---|
| Oracle University | http://oracle.com/education |
| Oracle Learning Library | http://oracle.com/oll |
| Product Documentation | http://www.oracle.com/technetwork/documentation |
| Product Downloads | http://www.oracle.com/technetwork/indexes/downloads |
| Product Articles | http://www.oracle.com/technetwork/articles |
| Product Support | http://oracle.com/support |
| Product Forums | http://forums.oracle.com |
| Product Tutorials/Demos | http://www.oracle.com/technetwork/tutorials |
| Sample Code | http://www.oracle.com/technetwork/indexes/samplecode |
| WebLogic Blog | http://blogs.oracle.com/weblogicserver |
| WebLogic on Facebook | http://www.facebook.com/oracleweblogic |



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

This course and the instructor attempt to address any questions that you might ask, but after you complete the course, Oracle provides a variety of channels for developers and administrators to access additional information.

Related Training

| Course |
|---|
| <i>Oracle WebLogic Server 12c: Administration I</i> |
| <i>Oracle WebLogic Server 12c: Administration II</i> |
| <i>Oracle WebLogic Server 12c: JMS Administration</i> |
| <i>Oracle WebLogic Server 12c: Troubleshooting Workshop</i> |
| <i>Oracle WebLogic Server 12c: Performance Tuning Workshop</i> |
| <i>Oracle WebLogic Server 12c: Enterprise Manager Management Pack</i> |



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Your instructor can provide additional information about the availability and contents of the courses listed above.

Unauthorized reproduction or distribution prohibited. Copyright© 2016, Oracle and/or its affiliates.

Tony Albrecht (base2tech@yahoo.com) has a non-transferable
license to use this Student Guide.

WebLogic Server: Overview



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Define the WebLogic Server terms: domain, server, and cluster
- Describe the difference between the administration server and managed servers
- List WebLogic Server tools

Distributed Systems

- Distributed systems divide their work across similar modules.
- As demand increases, more modules can be added to the system. This makes the system more *scalable*.
- The failure of a single module has less impact on the overall system, which makes the system more *available*.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

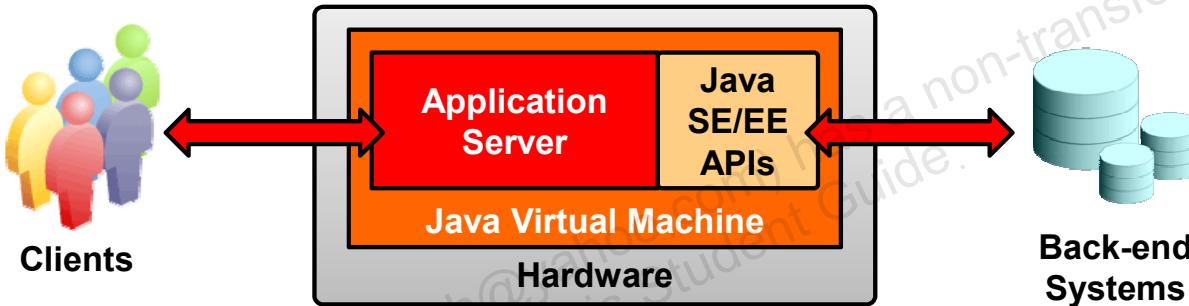
The main goal of a distributed system is to better manage the complexity and resulting cost of providing highly available and scalable systems.

Scalability is how well a system can adapt to increased demands. When a distributed system's capacity is reached, new equipment can be added fairly easily. These new modules should handle the increased demand. Distributed systems have the added advantage of lowering the initial costs of a new system, because additional equipment can be purchased as needed.

Availability is a measure of a system's ability to process client requests without downtime. *High availability* requires that a system is up and running as close to 24/7/365 as possible. This is achieved by using load balancing and failover techniques.

Java Platform Enterprise Edition

- Java Platform Enterprise Edition (Java EE) is the Java standard for distributed, enterprise computing.
- The Java EE platform consists of:
 - A Java Virtual Machine (JVM)
 - Java Platform Standard Edition (Java SE)
 - A Java EE application server
 - Java EE Application Programming Interfaces (APIs)



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The JVM is a software machine that converts compiled Java code into the machine code of the platform on which the virtual machine runs. Because Java source code is compiled into the code that runs on the virtual machine, that compiled code is portable.

Java Platform Standard Edition (SE) is a platform for developing portable, object-oriented applications. It includes a compiler, a debugger, and a JVM. It also includes thousands of already created classes (templates for kinds of objects) to make development easier. Java Platform Enterprise Edition is built on top of Java Platform Standard Edition.

An application server is software that handles application operations from the end user to the back-end business processes and databases. A Java EE application server complies with all the Java EE standards. It contains two main parts:

- The Web Container that processes Java web application components like Servlets and JSPs
- The EJB Container that processes the Java system components called an Enterprise JavaBeans

Java EE Application Programming Interfaces (APIs) define standards and provide callable code that can perform many server-side tasks from controlling transactions (Java Transaction API or JTA) to managing resources (Java Management Extensions or JMX).

Oracle WebLogic Server

Oracle WebLogic Server (WLS):

- Is a Java EE application server hosting Java EE applications (WLS 12c implements Java EE 6.0)
- Provides clustering for load balancing and high availability
- Offers an extensible security realm for authentication, authorization, and so on
- Runs the “Java components” of Oracle Fusion Middleware (FMW). These include Oracle SOA Suite, Oracle Service Bus, Oracle WebCenter Suite, and some Oracle Identity Management components
- Runs “System components” of FMW, such as Oracle HTTP Server (OHS)



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Oracle WebLogic Server is a scalable, enterprise-ready, Java Enterprise Edition application server. WebLogic Server enables enterprises to deploy mission-critical applications in a robust, secure, highly available, and scalable environment. These features enable enterprises to configure clusters of WebLogic Server instances to balance workload, provide extra capacity, and failover in case of hardware or other failures.

Extensive security features protect access to services and keep enterprise data secure.

Oracle Fusion Middleware is a collection of standards-based products that spans a range of tools and services: from Java EE, to integration services, business intelligence, and collaboration. Fusion Middleware products are organized into two general categories: Java components and system components. Java components generally are deployed to WebLogic Server as one or more Java Enterprise Edition applications and a set of resources. FMW 11g system components are not deployed as Java applications. Instead, system components are managed by the OPMN server. FMW 12c system components are deployed as part of a specialized WebLogic Server domain. OHS 12c runs as a WebLogic-managed system component either within a regular WebLogic domain or within a standalone domain that is dedicated to running OHS.

Some of the Fusion Middleware Java Components include:

- Oracle SOA Suite is a set of infrastructure components for designing, deploying, and managing service-oriented architecture applications. SOA Suite components run on top of WebLogic Server.
- Oracle Service Bus is an Enterprise Service Bus (ESB) that provides the infrastructure for service discovery, service provisioning and deployment, and governance. This service-infrastructure software adheres to the SOA principles of building coarse-grained, loosely coupled, and standards-based services.
- Oracle WebCenter Suite is an integrated set of components used for creating social applications, web portals, collaborative communities, and composite applications.
- Oracle Identity Management is an enterprise identity management system that manages user access privileges within the resources of an enterprise. Some of its components run on WebLogic Server, for example, Oracle Directory Integration Platform. This is a Java EE application that enables you to synchronize data between different repositories and Oracle Internet Directory.

JVM

- WebLogic Server, as Java code itself, runs within a JVM.
- The JVM and its settings can significantly affect WebLogic Server performance.
 - For example, the memory available to WebLogic Server and its applications is contained within the memory assigned to the JVM.
- Oracle provides the Oracle Hotspot JVM.
- JVM configuration options are set when the JVM starts.
 - This can be done by updating the scripts used to start WebLogic Server.



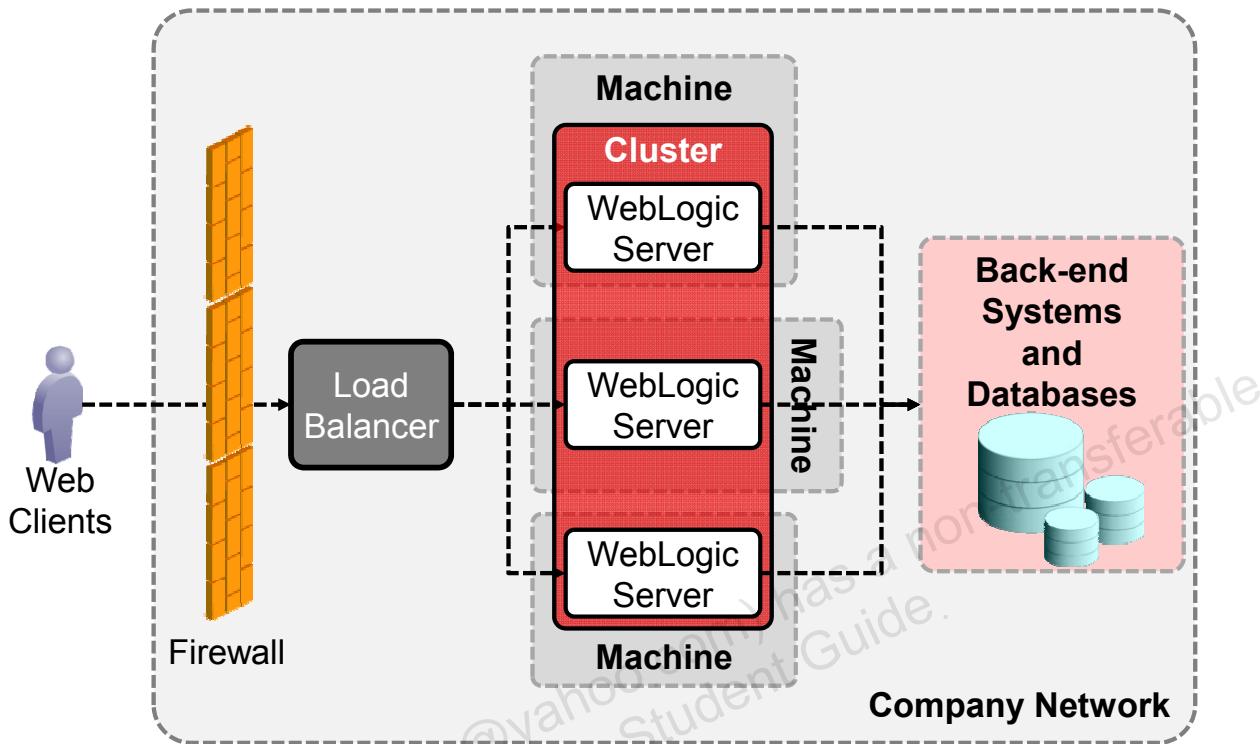
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The JVM is a “virtual computer” that executes the bytecode in compiled Java class files on a physical machine. How you tune your JVM affects the performance of WebLogic Server and the applications on it. Use only production JVMs on which WebLogic Server has been certified. Check the documentation for the latest list of JVMs and operating systems. The current release of WebLogic Server supports only those JVMs that are Java Platform Standard Edition 1.6 and higher.

Tuning the JVM to achieve optimal application performance is one of the most critical aspects of WebLogic Server performance. A poorly tuned JVM can result in slow transactions, long latencies, system freezes, and even system crashes. Ideally, tuning should occur as part of the system startup, by employing various combinations of the startup options.

(Possible) System Architecture



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A possible system architecture might contain:

- Clients using the World Wide Web to access your applications
- A firewall (hardware or software designed to prevent unauthorized access to or from a private network by using filtering or blocking ports)
- A cluster proxy of either a hardware load balancer or a web server like OHS
- A cluster of WebLogic Servers on various machines (each one running applications)
- Various back-end systems or databases accessed by the applications running on WebLogic Server

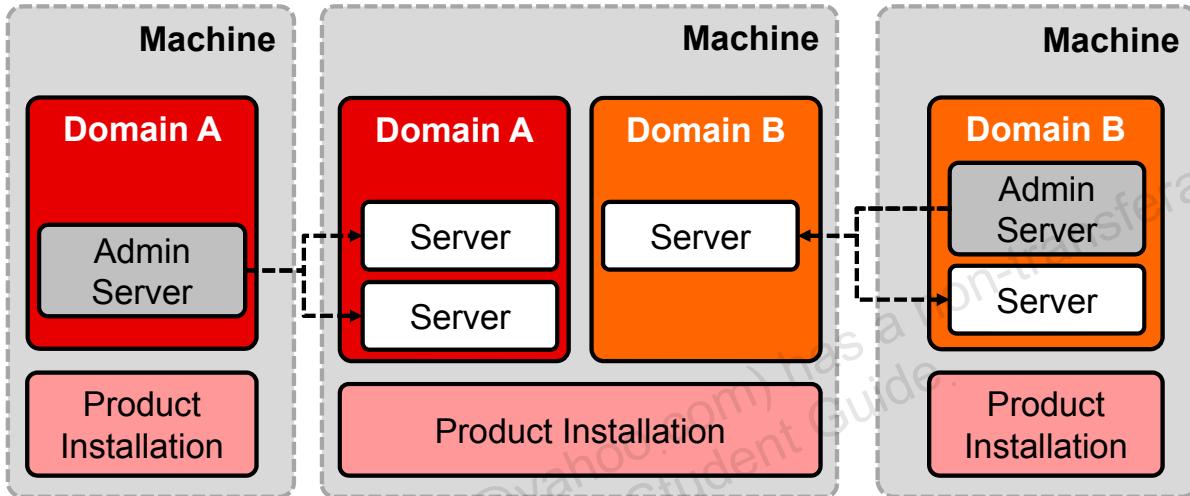
Other common architectural elements not shown:

- Additional firewalls (for example, between the Load Balancer and WebLogic Server or between WebLogic Server and the database)
- Multiple load balancers, or perhaps hardware load balancers in front of multiple web servers
- Multiple WebLogic Server clusters

WebLogic Server Domain

A *domain* is a collection of WebLogic Server resources.

- How many domains there are and how they are organized is completely up to you.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A *domain* is a collection of WebLogic Server resources. There are different kinds of resources in a domain, including WebLogic Servers, deployed applications, clusters, security providers, and Java Message Service and Java Database Connectivity elements.

How many domains you have and how you organize them is completely left to you. For example, domains may be organized by a logical division of types of applications, by physical location of hardware, by size and number of administrators, and so on.

All domains contain a special server called the *administration server*. You use the administration server to configure and manage all of the domain resources. Any other WebLogic Servers in the domain are called *managed servers*.

In most domains, the applications are deployed to the managed servers. The administration server is only used for domain configuration and management.

A single WebLogic Server product installation can be used to create and run multiple domains, or multiple product installations can be used to run a single domain. How domains are defined is up to you. You can define multiple domains based on different system administrators' responsibilities, application boundaries, or geographical locations of the machines on which servers run.

Administration Server

- A domain must have exactly one instance of WebLogic Server acting as the *administration server*. An administration server is part of exactly one domain.
- The administration server is:
 - The central point through which you configure and manage all domain resources
 - Solely in charge of the domain's configuration. It distributes configuration changes to other servers in the domain.
 - An instance of WebLogic Server and, therefore, a fully functional Java Enterprise Edition application server



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

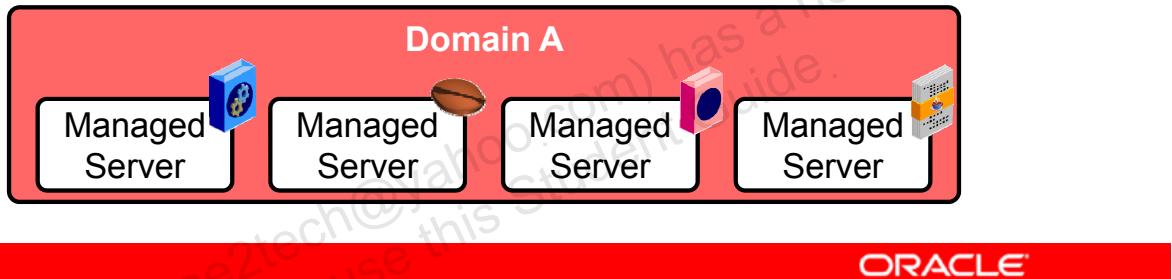
All domains contain a special server called the *administration server*. You use the administration server to configure and manage all of the domain resources. Any other WebLogic Servers in the domain are called *managed servers*.

In most domains, the applications are deployed to the managed servers. The administration server is only used for domain configuration and management.

Because an administration server is an instance of WebLogic Server, it can perform any task of a Java Enterprise Edition application server. Applications can be deployed and run on the administration server. For simplicity, often a development-time domain will only contain the administration server and no others. Developers deploy and test their applications on the administration server.

Managed Servers

- A domain can have zero or more *managed servers*.
- A managed server:
 - Is managed by the administration server
 - Is an instance of WebLogic Server and, therefore, a fully functional Java Enterprise Edition application server
 - Is where your Java Enterprise Edition applications run
 - Web applications, EJBs, web services, enterprise applications
 - Can be clustered with other cooperating managed servers for availability, scalability, and automatic failover



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

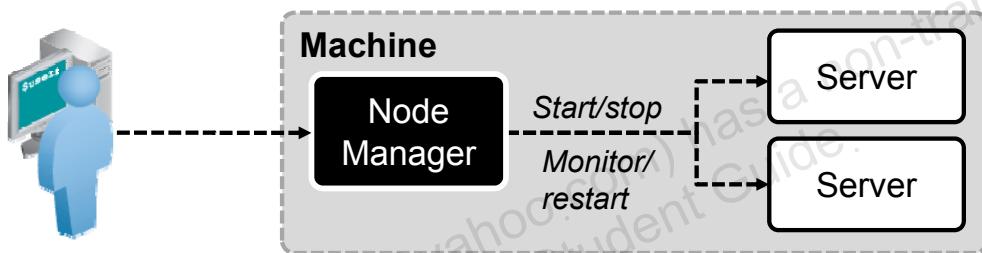
In almost all domains, the administration server is not the only server defined in the domain. Other servers are also defined. These others are called managed servers, because they are managed by the administration server.

A company's web applications, EJBs, web services, and other resources are deployed and run on the managed servers. That leaves the administration server free for configuration and management purposes.

For scalability, availability, and failover (when one server fails, requests are automatically sent to another server), managed servers can be placed together in a cluster.

Node Manager

- Is a separate process that accepts remote commands to start, stop, or suspend servers on its machine
- Monitors server availability and can restart failed servers
- Can be used to migrate servers on a failed machine to another machine



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Server instances in a WebLogic Server production environment are often distributed across multiple domains, machines, and geographic locations. Node Manager is a WebLogic Server utility that enables you to start, shut down, and restart both 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 runs on a particular machine.

There are two versions of Node Manager: the Java-based one that runs on any platform on which WebLogic Server runs and the script-based one that only runs on *nix operating systems. The Java-based Node Manager is recommended.

If Node Manager starts a server and that server later fails, Node Manager can be set to automatically restart it. If Node Manager fails or is explicitly shut down, upon restart, it determines the servers that were under its control when it exited. Node Manager can restart any failed servers as needed.

The WebLogic Server administration console can be used to issue commands to Node Managers running on remote machines. The WebLogic Scripting Tool (WLST) (in offline mode) also serves as a Node Manager command-line interface.

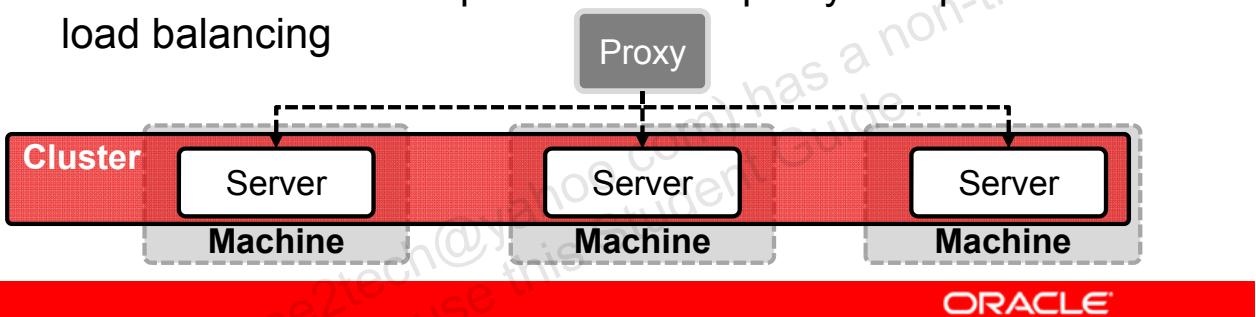
Machines and Clusters

A *machine*:

- Is defined within a domain to represent physical hardware
- Is required by Node Manager and used by clusters
- Has managed servers assigned to it

A *cluster*:

- Has multiple managed servers running cooperatively in it, which provides for failover
- With HTTP clients requires a cluster proxy that provides load balancing



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A machine definition is used to associate a computer with the managed servers that it hosts. Node Manager is always defined within the context of a machine. Machine definitions, and which servers are assigned to them, are also used by a clustered managed server in selecting the best location for storing replicated session data.

A cluster is a collection of multiple managed servers within a single domain running simultaneously and cooperatively to provide increased scalability and reliability. Resources and services are deployed identically to each server in a cluster, allowing for failover and load balancing. The session state of one clustered server is replicated on another server in the cluster. When a server fails, another server in the cluster takes over for it and retrieves the replicated data. No information is lost and customers do not realize that a different server is now fulfilling their requests. Clustered servers communicate with one another in two main ways: sending updates to their “backup server” when session state changes, and through cluster “heartbeats.” Each clustered server sends out a signal periodically to indicate that it is still viable. If a member of the cluster misses too many heartbeats, that server has “failed.”

A cluster with HTTP clients (a cluster with web applications) is always fronted by a proxy, which could be a web server, a hardware load balancer, or an instance of WebLogic Server. The proxy provides load balancing and enables failover by avoiding failed servers. Clusters that provide EJB or JMS applications do not require a cluster proxy.

WebLogic Server Application Services

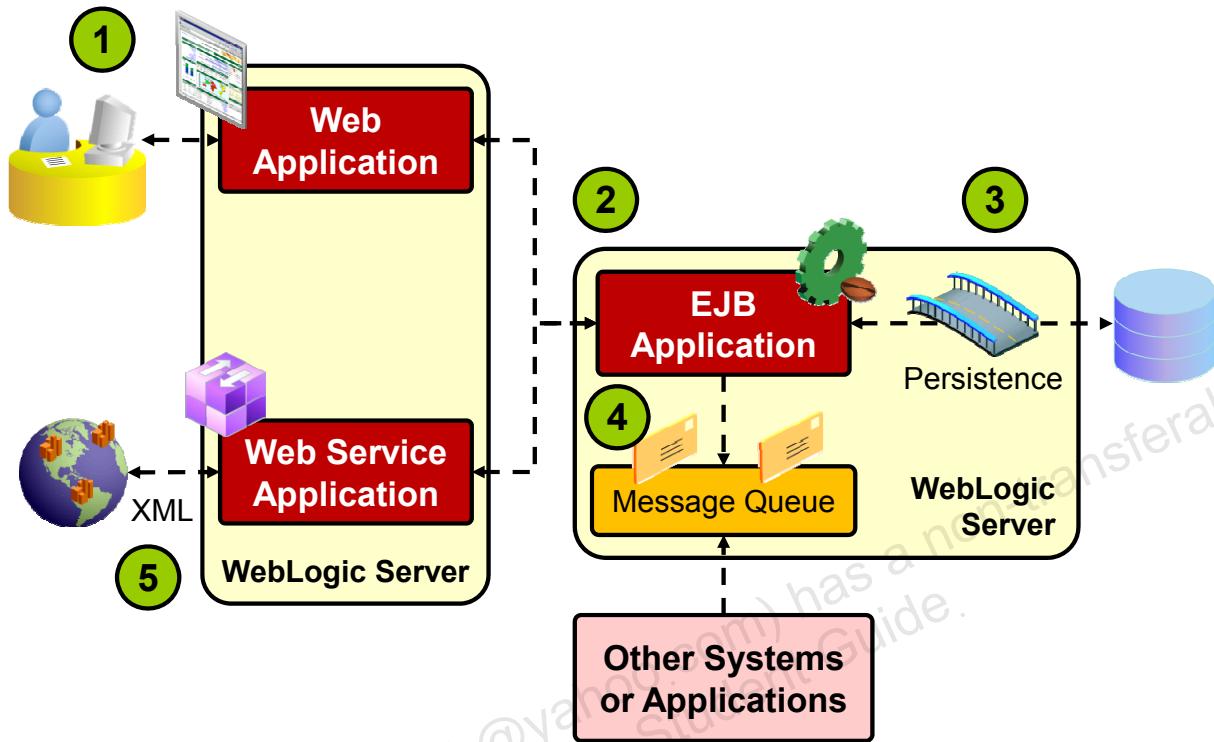
- Java Database Connectivity (JDBC)
 - The API for accessing relational databases
 - Data source objects are configured to provide database access.
- Java Message Service (JMS)
 - The API for and implementation of an enterprise messaging system
 - Multiple resources must be configured in WebLogic Server for JMS.
- Java Transaction API (JTA)
 - When transactions need to span resources, WebLogic Server can act as the transaction manager.



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

WebLogic Server Application: Example



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

1. Users interact with a web application by using a browser. The web application is responsible for rendering the website and for capturing user input through buttons, forms, links, and so on. It is possible that the web application contains all of the necessary business logic to perform the tasks that users request.
2. In this example, however, the web application accesses Enterprise JavaBeans (EJBs) to perform the business logic. These EJBs can be located on the same server as the web application or on a different server, as shown in this example.
3. Some of the EJBs shown include a persistence mechanism. They are writing newly placed orders to a relational database.
4. After the order is written to the database, an EJB uses the Java Message Service (JMS) to asynchronously communicate with other applications so that those applications can also process the order.
5. To expose the business logic of this application in a standard way to other applications, both within your organization and beyond, a web service is used. XML-based web services can be accessed by both Java and non-Java applications and are a cornerstone of service-oriented architecture.

WebLogic Server Administrative Tools

WebLogic Server can be administered and monitored by using:

- The WebLogic Server administration console
- The WebLogic Scripting Tool (WLST)
- The WebLogic Diagnostic Framework (WLDF)
- The WLDF Monitoring Dashboard
- Enterprise Manager Cloud Control



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

WebLogic Server Administrative Tools

WebLogic Server can be administered and monitored by using:

- The Java Management Extensions (JMX) API
 - WebLogic Server provides a large set of JMX managed beans (MBeans) for the resources that it manages.
 - These objects are used by the tools provided with the product.
 - These objects can also be used by your own custom JMX code to create, configure, manage, and monitor WebLogic Server resources.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A managed bean (MBean) is a Java bean that provides a Java Management Extensions (JMX) interface. JMX is the Java Enterprise Edition API for monitoring and managing resources. WebLogic Server provides a set of MBeans that its own tools use to configure, monitor, and manage WebLogic Server resources. You can also write custom JMX code to perform those same functions. For more information about WebLogic Server MBeans, see the *MBean Reference* document.

WebLogic Server Administration Console

- The Oracle WebLogic Server administration console is a web browser-based tool for configuring, administering, and monitoring the resources of a domain.
- The console application runs on the administration server.
- It is part of the normal installation of WebLogic Server.

The screenshot shows the Oracle WebLogic Server Administration Console interface. The title bar reads "ORACLE WebLogic Server Administration Console 12c". The main navigation bar includes "Home", "Log Out", "Preferences", "Record", "Help", and a search bar. The URL is "Welcome, weblogic" and it is connected to "wlsadmin". The left sidebar has sections for "Change Center" (with "View changes and restarts" and "Lock & Edit" buttons) and "Domain Structure" (listing "wlsadmin", "Environment", "Servers", "Clusters", "Coherence Clusters", "Machines", "Virtual Hosts", "Work Managers", "Startup and Shutdown Classes", and "Deployments"). The main content area is titled "Settings for AdminServer" under "Deployment". It shows tabs for General, Cluster, Services, Keystores, SSL, Federation Services, Deployment (which is selected), Migration, and Tuning. Below these tabs are sub-tabs for Overload, Health Monitoring, Server Start, Web Services, and Coherence. A note says "Click the Lock & Edit button in the Change Center to modify the settings on this page." There is a "Save" button. Another note says "Use this page to define the default deployment staging configuration for this server." A "Staging Mode:" dropdown is set to "hosted". A tooltip for "hosted" explains: "The mode that specifies whether an application's files are copied from a source on the Administration Server to the Managed Server's local disk during application deployment." The bottom right corner features the "ORACLE" logo.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Oracle WebLogic Server administration console (admin console) is a web browser-based, graphical user interface that can be used to manage a WebLogic Server domain. The admin console application runs on the administration server. The admin console can be used to:

- Configure, start, and stop instances of WebLogic Server
- Configure clusters
- Configure database connectivity (JDBC)
- Configure messaging (JMS)
- Configure WebLogic Server security
- Deploy applications
- Monitor server and application performance
- View server and domain log files

WLST

The WebLogic Scripting Tool (WLST) is:

- A scripting tool for creating, configuring, administering, and monitoring the resources of a WebLogic Server domain
- Part of the normal installation of WebLogic Server
- Capable of performing all the tasks available in the administration console, and more
- Based on the Jython programming language (the Java implementation of Python)
 - Python is an object-oriented, interpreted programming language.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

WLST is a command-line scripting environment that can be used 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 execution, and flow control statements, WLST provides a set of commands that are specific to WebLogic Server.

WLST can be run interactively (one command at a time) or in script mode (running a file of commands). It also can be run online (connected to an administration server that allows it to manage an active domain) or offline (accessing the configuration files of an inactive domain).

WLST

- WLST can run commands one at a time (interactive mode) or from files (script mode).
- To run WLST, set environment variables by running the `setWLSEnv.sh` script in `<WL_HOME>/server/bin`, and then call the Java Virtual Machine with the `WLST` class:

```
$> source /u01/app/fmw/wlserver/server/bin/setWLSEnv.sh
$> java weblogic.WLST
Initializing WebLogic Scripting Tool (WLST) ...
Welcome to the WebLogic Server Administration Scripting
Shell
Type help() for help on available command

wls:offline>
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Calling WLST without a script, as shown in the slide, puts you into interactive mode.

WLST

- To run a WLST script, after setting the environment variables, call the Java Virtual Machine with the `weblogic.WLST` class, followed by the name of the script:

```
...  
$> java weblogic.WLST myscript.py
```

```
myscript.py  
# Modify these values as necessary  
username = "weblogic"  
password = "Welcome1"  
url = "t3://myadminhost:7001"  
# Connect to the admin server  
connect(username, password, url)
```



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To call WLST with a script, simply put the name of the script after the WLST fully qualified class name, `weblogic.WLST`:

```
$> java weblogic.WLST myscript.py
```

WLDF

The WLDF is used to gather and analyze WebLogic Server runtime data:

- **Diagnostic images:** It creates snapshots of the server's configuration and runtime metrics.
- **Harvesters:** Metric collectors can be set to periodically collect and record data.
- **Watches and notifications:** Watches compare data to conditions you set, and when triggered, they can send out notifications.



WLDF is part of the normal installation of WebLogic Server and can be configured and used through the administration console or WLST.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

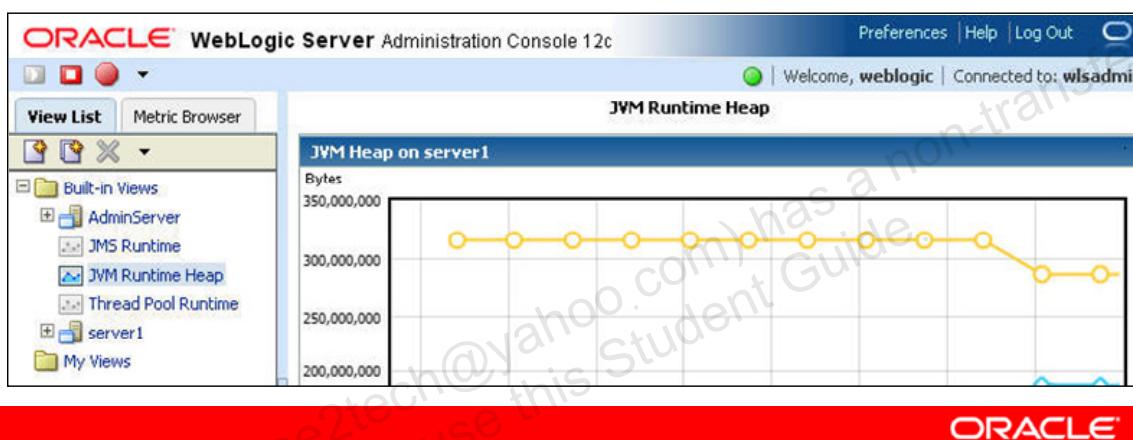
The WLDF provides features for generating, gathering, analyzing, and persisting diagnostic data from WebLogic Server instances and from applications deployed to them. Some WLDF features are configured as part of the configuration for a server in a domain. Other features are configured as system resources (diagnostic modules) that can be targeted to servers (or clusters).

You use the diagnostic image capture component of WLDF to create a diagnostic snapshot of a server's internal runtime state at the time of the capture. This information can help Oracle support personnel analyze the cause of a server failure. You can capture an image manually by using the WebLogic Server administration console or WLST, or you can generate one automatically as part of a watch notification. A diagnostic image zip file includes the server's configuration, log cache, JVM state, work manager state, JNDI tree, and most recent harvested data.

WLDF Monitoring Dashboard

The Monitoring Dashboard presents WLDF data graphically. It:

- Is part of the administration console application
- Can be launched from the administration console or with its own URL
- Can graphically display active runtime data or archived data



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

The Monitoring Dashboard provides views and tools for graphically presenting diagnostic data about servers and applications running on them. The underlying functionality for generating, retrieving, and persisting diagnostic data is provided by the WLDF. The Monitoring Dashboard provides additional tools for presenting that data in charts and graphs.

Enterprise Manager Cloud Control

- Is a tool for administering and monitoring your entire Oracle IT infrastructure, including WebLogic Server
 - Unlike other WebLogic tools, Cloud Control enables you to administer multiple domains.
- Requires its own installation (It is not installed as part of some other component.)
- Supplies a web browser-based user interface

The screenshot shows the Oracle Enterprise Manager Cloud Control 12c interface. The top navigation bar includes links for Grid, Targets, Favorites, History, Setup, Help, Log Out, and a search bar for 'Search Target Name'. The main content area is titled 'Enterprise Summary' and contains two main sections: 'Overview View' and 'Inventory and Usage'. The 'Overview View' section shows 'Targets Monitored: 2905' and 'Targets with Status: 2241', with a pie chart indicating the status distribution: 15% Down, 12% Metric Collection, 267 Error, and 93 Agent. The 'Inventory and Usage' section is set to 'Hosts' and shows a table of platforms with their respective host counts and OS patches. The table data is as follows:

| Platform | Hosts | OS Patches |
|--|-------|------------|
| Enterprise Linux Server release 5.6 (Carthage) | 39 | No |
| Enterprise Linux AS release 4 (October Update 8) | 15 | No |
| Enterprise Linux Server release 5.4 (Carthage) | 6 | No |
| SunOS | 5 | No |

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

Enterprise Manager Cloud Control is composed of three main components:

- The Oracle Management Repository
- One or more Oracle Management Services
- One or more Oracle Management Agents

An Oracle Management Agent (simply called an agent) is responsible for monitoring the health of a target and is installed on the host on which a target runs. An agent collects information about a target and sends it through the Oracle Management Service (OMS) to the Oracle Management Repository.

OMS is a Java Enterprise Edition web application. It receives information from agents and saves that information in the Oracle Management Repository. It also provides the Grid Control console.

The Oracle Management Repository contains a collection of Grid Control schema objects such as database jobs, packages, procedures, and views.

Enterprise Manager Grid Control is renamed Enterprise Manager Cloud Control in version 12c.

Quiz

The number of administration servers in a domain:

- a. 0
- b. 1
- c. 2
- d. 3 or more



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: b

Quiz

The minimum number of managed servers in a domain:

- a. 0
- b. 1
- c. 2
- d. 3



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a

Summary

In this lesson, you should have learned how to:

- Define the WebLogic Server terms: domain, server, and cluster
- Describe the difference between the administration server and managed servers
- List WebLogic Server tools

Unauthorized reproduction or distribution prohibited. Copyright© 2016, Oracle and/or its affiliates.

Tony Albrecht (base2tech@yahoo.com) has a non-transferable
license to use this Student Guide.

3

Installing and Patching WebLogic Server

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Determine supported configurations for WebLogic Server
- Install WebLogic Server
- Apply a patch to WebLogic Server

Determining Supported System Configurations

1. Know your system (operating system, processor).
2. Download the *System Requirements and Supported Platforms* spreadsheet for the version of FMW you plan to use.
3. Start on the “FMW on WLS - System” sheet. Find your operating system in the “OS Version” column.
 - Which version of FMW? Which JDK? What DB and version?

| Installation Type | Version Supported | Processor Type | OS Version | Oracle FM 32/64 bit | JDK Vendor Version 11gR1 (11.1.1.6) | JDK 32/64 bit | Oracle Database |
|-------------------|-------------------|----------------|-----------------------|---------------------|-------------------------------------|---------------|----------------------|
| ALL | 11gR1 (11.1.1.6+) | x64 | Oracle Linux 6 (UL1+) | 64 | Oracle JDK 1.7.0_02+ | 64 | ... Oracle 11.2.0.1+ |



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

On each spreadsheet “sheet,” there are multiple rows for each version of an operating system. In the “FMW on WLS - System” sheet, each row shows the operating system and its version, the version of FMW supported, which JDKs are supported, which Oracle databases are supported, and so on.

Note that the table shown, which represents part of that sheet, is for example purposes only. Also note that only some of the columns are shown and only one of the rows displayed.

There are multiple sheets in the supported systems spreadsheet. Some of them are:

- **FMW on WLS - Client and OER IDE:** Shows which browsers and what versions work with various FMW products. (OER stands for Oracle Enterprise Repository.)
- **FMW on WLS - Additional DB:** Shows which non-Oracle databases and what versions work with certain FMW products
- **FMW on WLS - Web Servers:** Shows which web servers and what versions work with WebLogic Server
- **FMW on WLS - Id&Access:** Shows which Oracle identity and access products and what versions work with WebLogic Server and other FMW products. Identity and access products include Oracle Access Management, Oracle Virtual Directory, and others.

Ensuring Your System Meets Requirements

1. Ensure that your system meets the minimum requirements for WebLogic Server:
 - At least a 1 GHz CPU
 - Sufficient disk space (~3.9 GB for a complete installation)
 - At installation, 2.5 x 3.9 GB of temporary space is needed.
 - Sufficient memory (1 GB of RAM is minimum, 2 GB is recommended)
2. If using other FMW components, ensure that your system meets their added requirements:
 - For example, Oracle SOA Suite requires a dual-core processor 1.5 GHz or better, 15 GB of disk space, and a minimum of 2 GB of physical memory with 4 GB of available memory.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

When Not All FMW Is the Same Version

1. If not everything is the same version, download the *System Requirements and Supported Platforms* spreadsheet for the WebLogic Server version you plan to use.
2. Start on the “System” sheet. Find your operating system in the “OS Version” column.

| Installation Type | Version Supported | Processor Type | OS Version | Oracle FM 32/64 bit | JDK Vendor Version 12c (12.1.2.0.0) | JDK 32/64 bit | Oracle Database |
|------------------------|-------------------|----------------|-----------------------|---------------------|-------------------------------------|-------------------------------|-----------------|
| Oracle WebLogic Server | 12c (12.1.2.x) | x64 | Oracle Linux 5 (UL6+) | 64 | Oracle JDK 1.7.0_02+ | 64 ... Oracle 11.2.0.3+ | |

3. Use the “WLS - WebServer” sheet for the OHS version.
4. Use the “Id&Access” sheet for OID & OAM versions.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

On each spreadsheet “sheet,” there can be multiple rows for each version of an operating system. In the “System” sheet, each row shows the operating system and its version, the version of WebLogic Server supported, which JDK is supported, which Oracle databases are supported, and so on.

Note that the table shown, which represents part of that sheet, is for example purposes only. Also note that only some of the columns are shown and only one of the rows displayed.

Oracle HTTP Server (OHS) is a web server based on the open-source Apache web server.

Oracle Internet Directory (OID) is an LDAP v3-compliant directory with meta-directory capabilities. It is built on Oracle database and is fully integrated into Oracle Fusion Middleware.

Oracle Access Manager (OAM) is a product that provides single sign-on capabilities.

WebLogic Server Installers

- Generic installer
 - JAR file
 - Choose the generic installer with the features you require:
 - WebLogic Server Core
 - Fusion Middleware Core
 - Requires you to first download and install an appropriate Java Development Kit (JDK) or Java Runtime Environment (JRE)
 - Choose a supported one for your operating system.
- ZIP installer
 - A ZIP file
 - JDK included
 - Only good for 32-bit operating systems
 - Cannot be used for production



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Generic Installers

- The WebLogic Server core installer (`wls_generic.jar`) contains:
 - WebLogic Server
 - Coherence
 - Administrative tools
 - Database support
 - Examples
- The FMW core installer (`fmw_12.1.3.0.0_infrastructure_Disk1_1of1.zip`) contains all of the above, plus:
 - Repository Creation Utility (RCU)
 - Java Required Files (JRF) and Enterprise Manager



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

WebLogic Server core installer:

- Coherence is an in-memory data management system for application objects shared across multiple servers.
- Administrative tools include the administration console, the Configuration Wizard, the Template Builder, WLDF, and WLST.
- Database support includes third-party JDBC drivers and an evaluation Derby database. The Apache Derby project is an open source relational database implemented entirely in Java.
- Examples include example applications for WebLogic Server, and examples of code using Coherence. Note that these examples are not installed as part of a “typical” installation. Some of the samples use the Derby evaluation database.
- Also available in this installer is a Web 2.0 HTTP Pub-Sub server, which can be used by Web clients to subscribe to channels and publish messages asynchronously to them by using the Bayeux protocol. Bayeux supports responsive bidirectional interactions between web clients, for example those using AJAX (Asynchronous JavaScript and XML).

- The last part of this installer is WebLogic SCA (Service Component Architecture), which is a Spring container. Spring is an open-source application development framework for Java.

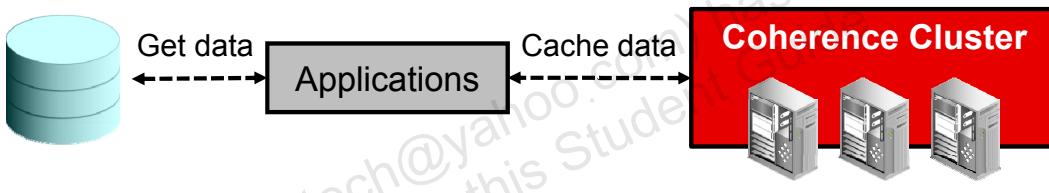
Fusion Middleware core installer:

- Note that one of the “administrative tools,” the Configuration Wizard, is updated by some Fusion Middleware components to ensure new WebLogic domains include artifacts required by these components.
- The RCU is the tool used to create database schemas for Fusion Middleware products.
- The JRF is a set of libraries providing common functionality for Oracle business applications and application frameworks.
- Enterprise Manager contains the administrative tool Fusion Middleware Control.

What Is Oracle Coherence?

Oracle Coherence:

- Is a JVM process
- Provides a distributed, in-memory data caching solution
- Offers high performance and scalability
- Is based on a cluster of cache servers
- Automatically distributes (partitions) cached data across the Coherence cluster
- Can be installed and managed independently or as part of a WebLogic Server domain



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

One of the primary uses of Oracle Coherence is to cluster an application's data. This means that the objects and data that an application delegates to Coherence clusters are automatically available to and accessible by all servers in the application cluster. None of the objects or data are lost in the event of server failure. Coherence thereby solves many of the problems related to achieving reliability and availability for clustered applications.

The partitioning feature dynamically load-balances data evenly across the Coherence cluster, whereas replication ensures that a desired set of data is always available and up-to-date in the Coherence cluster. Replication enables operations that are running on any server to obtain the data that they need locally, at basically no cost, because that data has already been replicated to that server. The only downside of partitioning is that it introduces latency for data access. To eliminate the latency associated with partitioned data access, Coherence can use local or "near caching" as well. Frequently and recently used data from the partitioned cache is maintained on the specific servers that are accessing that data, and this "near data" is kept up-to-date by using event-based invalidation.

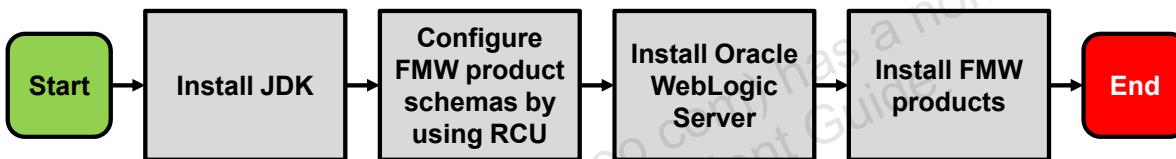
Coherence and Coherence*Web are included in the default installation of WebLogic Server. WebLogic Server includes features that enable deployed applications to use Coherence data caches and incorporate Coherence*Web for session management.

FMW Installation Flow

Installing Oracle Fusion Middleware products generally involves the following steps:

1. Install the JDK.
2. Create the database schemas required by the FMW products being installed by using the Oracle Repository Creation Utility (RCU).
3. Install Oracle WebLogic Server.
4. Install other Oracle Fusion Middleware products.

If WebLogic Server is used “stand alone” (without FMW components running on it), only steps 2 and 3 are needed.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Metadata Repository contains metadata for some Oracle Fusion Middleware components, for example, Oracle WebCenter. It can also contain metadata about the configuration of some Oracle Fusion Middleware components. Developers can also write code so that it can hold metadata for their own applications. The Metadata Repository can be database based or file based. The database-based repository can be created in an existing database by using the RCU.

Note that the installation of WebLogic Server and other FMW components does not require that the Metadata Repository be created first, but it often is created first.

WebLogic Server itself does not use the Metadata Repository.

Note: RCU requires an installed JDK in order to execute. If a JDK is installed on the machine by default, then you can install a different JDK after running RCU.

WebLogic Server Installation Modes

- Graphical:
 - An interactive, GUI-based method of installation
- Silent:
 - A noninteractive, command-line method of installation
 - Request silent mode with the `-silent` option
 - Installation configuration information must be placed in a file and referred to with the `-responseFile` option

```
$> java -d64 -jar wls_generic.jar  
          -silent  
          -responseFile /path/file.rsp
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To create a response file with the correct format, install in graphical mode. On the “Installation Summary” screen, click the **Save** button, which saves your responses in a response file.

Installing WebLogic Server on Linux (Graphical Mode)

1. Determine that your system is supported and meets requirements.
2. Log in as the user you want to use for installation. (Do *not* use root.)
3. Download the appropriate JDK tarball file and the WebLogic Server generic installer.
4. Install the JDK.
 - A. Create a directory for the JDK.
 - B. Extract the JDK archive file into the new JDK directory.

```
$> mkdir /u01/app/fmw/jdk
$> mv /download/name.tar.gz /u01/app/fmw/name.tar.gz
$> cd /u01/app/fmw/jdk
$> tar -zxvf name.tar.gz
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The name of the directory where files have been downloaded can be anything.

The JDK archive file name changes depending upon the version of the JDK downloaded.

You can put the JDK in the directory of your choice.

In the `tar` command, the options are:

- `z` = file was compressed, so unzip
- `x` = extract
- `v` = verbose
- `f` = file (extract from a file)

Note: There are also RPM-based 32-bit and 64-bit JDK installers for Linux. (RPM is a package management system.)

Installing WebLogic Server on Linux (Graphical Mode)

5. Run the WebLogic Server generic installer.
 - A. Set file permissions.
 - B. Set the PATH to include the bin directory of the JDK.
 - C. Run the JAR installer.

```
$> umask 027
$> export JAVA_HOME=/u01/app/fmw/jdk
$> export PATH=$JAVA_HOME/bin:$PATH
$> cd /download
$> java -d64 -jar wls_generic.jar
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

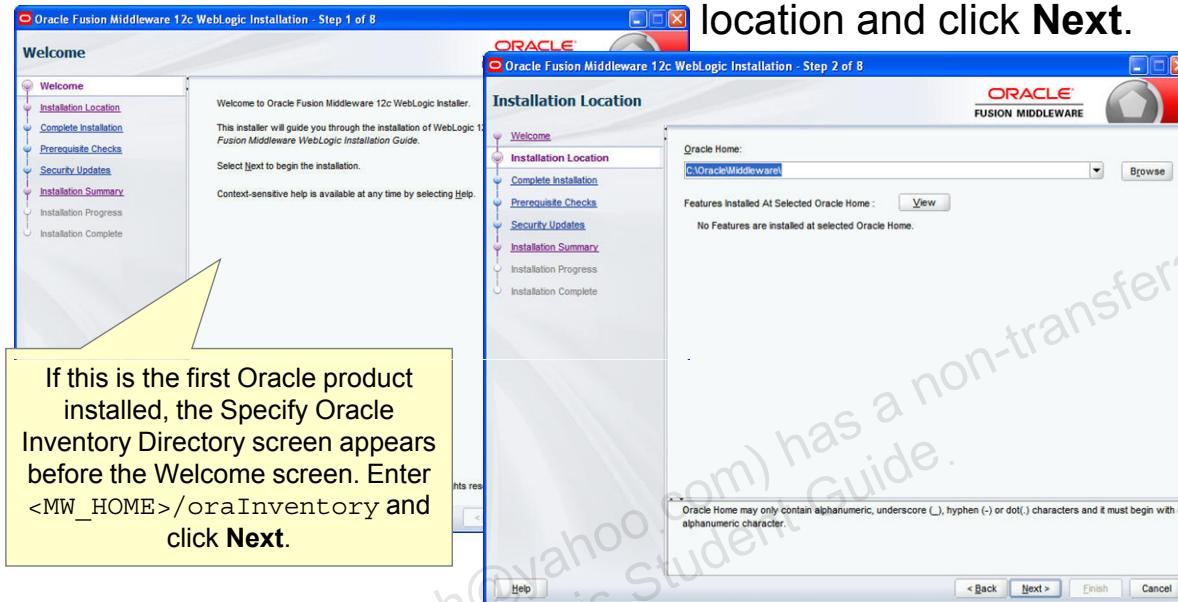
The umask command sets the file mode creation mask to effect the permissions of newly created files and directories. It is set to 027, which results in new files having the default permission of `rw- r-- ---` and new directories having the default permission of `rwx r-x ---`.

The `-d64` option is used for installing on 64-bit Linux systems.

The JAR file name may be different.

Installing WebLogic Server on Linux (Graphical Mode)

6. On the Welcome screen, click **Next**.
7. On the Installation Location screen, select a location and click **Next**.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

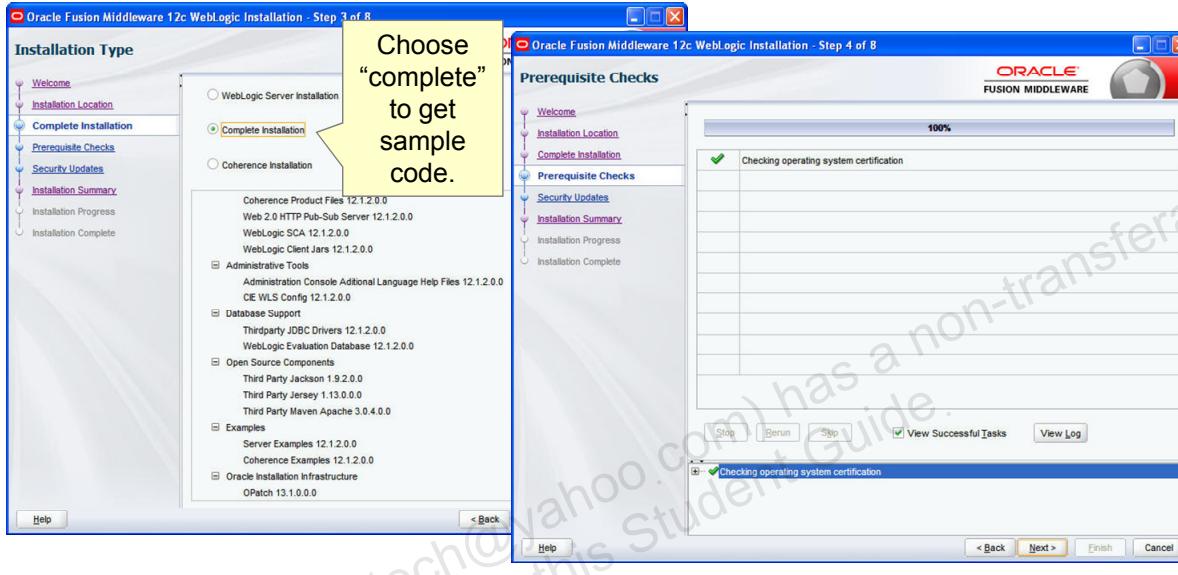
ORACLE

The Oracle Inventory directory (`oraInventory`) stores an inventory of all Oracle products installed on the system. It is required and shared by all Oracle products.

The Fusion Middleware documentation uses `/home/Oracle/Middleware` as the Fusion Middleware “Oracle Home” directory. This is an example, and the location of the installation is up to you.

Installing WebLogic Server on Linux (Graphical Mode)

8. On the Installation Type screen, select a type and click **Next**.
9. On the Prerequisite Check screen, at 100% success, click **Next**.



ORACLE

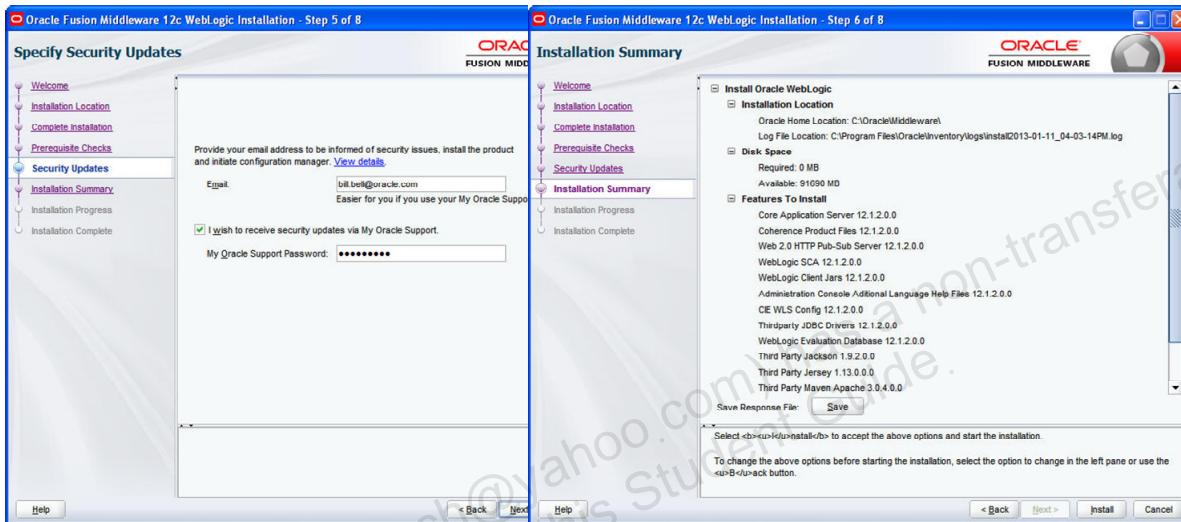
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The choices for the type of installation are:

- **WebLogic Server Installation:** WebLogic Server, Coherence, Web 2.0 Pub-Sub Server, WebLogic SCA (Spring), WebLogic client Jars, Administrative tools, 3rd party JDBC drivers, the Derby evaluation database, open source tools (Jackson, Jersey, and Maven), and OPatch.
- **Complete Installation:** All of the above, plus example code (WebLogic Server and Coherence)
- **Coherence Installation:** The same as the WebLogic Server installation, but missing WebLogic client Jars and 3rd party JDBC drivers.

Installing WebLogic Server on Linux (Graphical Mode)

10. On the Security Updates screen, enter a support email and password, and click **Next**.
11. On the Installation Summary screen, click **Install**.



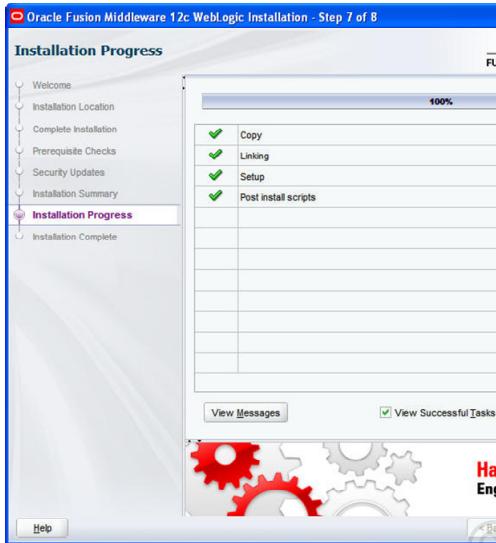
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

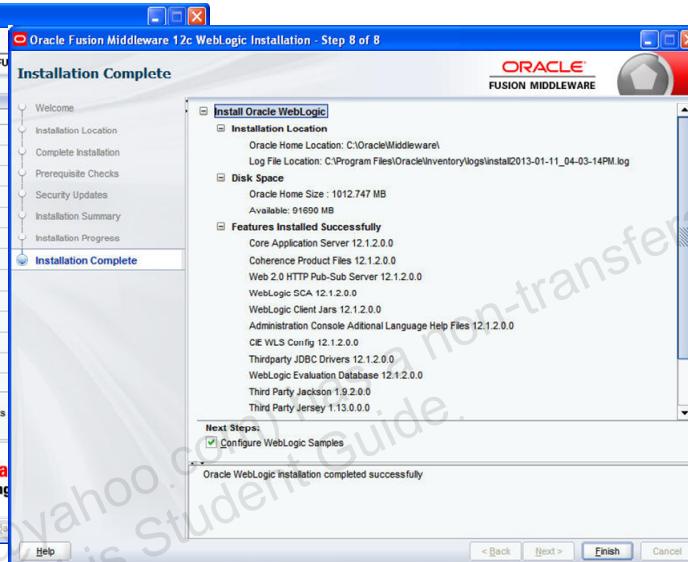
In the practice environment, you will not enter security update information, because you do not want to associate your email with a classroom installation.

Installing WebLogic Server on Linux (Graphical Mode)

12. On the Installation Progress screen, at 100%, click **Next**.



13. On the Installation Complete screen, click **Finish**.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Installation Problems

- The “prerequisite check” of the installer checks if the operating system meets requirements and lists any issues.
 - If you have determined your system is supported ahead of time, there should be no problems.
- If the generic installer does not come up, ensure that you have set the JAVA_HOME environment variable to the correct JDK and added the bin directory under it to the PATH.
- If you are having problems installing, create a verbose log file during the installation with the -logFile option:

```
$> java -d64 -jar wls_generic.jar -logFile /path/fn.log
```

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Sample Installation Directory Structure

| Directory | Description |
|---------------|----------------------|
| u01/app | Oracle base |
| db11g | Database home |
| fmw | Middleware home |
| coherence | Coherence home |
| jdk1.7.0_10 | Java home |
| oracle_common | Oracle common home |
| web | Web home (OHS) |
| wlserver | WebLogic Server home |

The Oracle Home you enter in the installer.(It is the “oracle home” of Fusion Middleware.)

Each Fusion Middleware product installed has its own “oracle home” directory under Middleware home. WebLogic Server’s “oracle home” is here.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

When you install the first Oracle Fusion Middleware product in a system, a directory referred to as middleware home is created. (In the practice environment, it is the `fmw` directory.) The path to this directory and its name is determined by the person installing the product.

Within middleware home, Oracle home directories are created for each FMW product that is installed. For example, in the slide, the Oracle home directory for WebLogic Server is `wlserver`.

A product's software binaries are installed into the product's Oracle home directory. You should not configure any component runtime environments (domains or instances) within this directory.

In addition to Oracle home directories, the Oracle common home directory (in the slide as `oracle_common`) is also located within the middleware home directory. The Oracle common home directory contains the binary and library files required for Oracle Enterprise Manager, Fusion Middleware Control, and the JRF. There can be only one Oracle common home directory within each middleware home directory.

Uninstalling WebLogic Server

1. Shut down any running servers.
2. Run the uninstall script:
`<MW_HOME>/oui/bin/deinstall.sh`.
3. Go through the screens, selecting the components to uninstall. By default, all are selected.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In Windows, the uninstaller can also be started by using the Windows Start menu: **Start > Programs > Oracle WebLogic > Uninstall Oracle WebLogic**.

When running the uninstaller script, if your system supports a graphical user interface, it starts in graphical mode. If not, it starts in console mode.

You can also choose console mode. To select console mode, add the `-mode=console` option when running the script.

Another option, usually used from scripts, is to run in silent mode. To run in silent mode, add the `-mode=silent` option when running the script. In silent mode, all components are uninstalled. Also, some files remain (for example, the domain directories). Those can be manually deleted.

Applying Patches by Using OPatch

- Follow these steps to apply a patch:
 1. Contact Oracle Support.
 2. Check for existing patches.
 3. Obtain the necessary patch from Oracle Support.
 4. Determine the Oracle home.
 5. Read the patch README file.
 6. Apply the patch by using OPatch.
 7. Based on the patch README file, perform postinstallation steps.
- OPatch example:

```
$> opatch apply -jdk $JAVA_HOME
```
- You can use the OPatch rollback command to remove an existing patch.

OPatch is a utility to help you apply software patches.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The OPatch executable (or batch file for Windows) is found in the `OPatch` directory under `<MIDDLEWARE_HOME>`.

To obtain the patch, you can contact your Oracle Support representative, or you can go to My Oracle Support (formerly OracleMetaLink) at <http://support.oracle.com/>.

If Oracle Support is not able to resolve the issue, you may be asked whether you have any patches already installed on your system. To determine this information, run the `lsinventory` command of OPatch.

The most common type of patch available in a Fusion Middleware environment involves patching a specific Oracle home directory. Some patches (for example, a patch pertaining to JRF) may apply to multiple Oracle home directories within a specific Middleware home. You can obtain the list of Oracle home directories that are registered in an Oracle inventory by using the `lshomes` command of OPatch. After you determine your Oracle home directories, you should run the `checkApplicable` command to make sure that the patch can actually be applied to them.

Quiz

The ZIP installer is OK to use for production installation in a 32-bit system.

- a. True
- b. False



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: b

Quiz

The name of the WebLogic home directory is _____.

- a. wlserver_12.1
- b. wlserver_12c
- c. wlserver_12.1.2
- d. wlserver



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: d

Summary

In this lesson, you should have learned how to:

- Determine supported configurations for WebLogic Server
- Install WebLogic Server
- Apply a patch to WebLogic Server

Practice 3-1 Overview: Installing WebLogic Server

This practice covers the following topics:

- Installing the JDK
- Installing WebLogic Server

Practice 3-2 Overview: Patching WebLogic Server

This practice covers using OPatch to patch WebLogic Server.



Creating Domains



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe a domain's file system
- Create a domain by using the Configuration Wizard
- Configure resources by using the Configuration Wizard
- Copy a domain to another computer with the pack and unpack utilities

Domain Planning Questions

- How many domains?
 - A domain is an arbitrary, administrative boundary.
 - Possible domain boundaries include:
 - Business unit
 - Cost center
 - Data center location
 - Administrator or administrative group
 - Application or application type (for example, one domain for end-user functions and another for back-end accounting)
 - Size (breaking up a large domain into smaller ones to manage them more efficiently)

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Domain Planning Questions

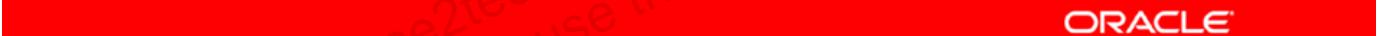
- For each domain:
 - What other FMW products are running in the domain?
 - What extra requirements do they impose?
 - See the *Fusion Middleware Enterprise Deployment Guides* which provide product installation recommendations.
 - What applications are running in the domain?
 - Do we need them to be highly available?
 - Will we be using WebLogic clustering or Coherence?
 - Do we need a database?
 - Do we need a highly available database like Oracle RAC?
 - Do our applications use JMS?
 - Do our applications contain EJBs?



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Domain Planning Questions

- What is the topology?
 - How many computers?
 - How many instances of WebLogic Server?
 - What hosts and ports?
 - Use virtual IP addresses or virtual host names
 - How many clusters?
 - What will proxy the web-tier clusters? A web server? A hardware load balancer?

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Virtual IP Address and Virtual Host Name

- Virtual IP
 - A network interface card typically binds to a single IP address. It can be set up to listen on extra addresses. These are called virtual IP (VIP) addresses.
 - Use VIP addresses when defining WebLogic Servers.
 - If the server must be brought up on new hardware, the VIP address can be moved over to the new hardware.
- Virtual host name
 - A host name is the primary name of a machine in the Domain Name System (DNS). Other (virtual) host names can be assigned to the same machine.
 - Use a virtual host name for each component in FMW. That way, if a component needs to be relocated, no URLs used to access that component must change.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

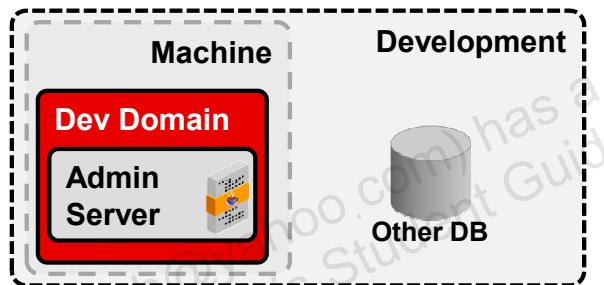
A virtual IP address is an unused IP address that belongs to the same subnet as the host's primary IP address. It is assigned to a host manually and Oracle WebLogic managed servers are configured to listen on this virtual IP address rather than a physical IP address. In the event of failure of the machine where the IP address is assigned, the virtual IP address is assigned to another machine in the same subnet, so that the new machine can take responsibility for running the managed servers assigned to it.

A host name is the primary name of the machine in Domain Name System (DNS). In setting up FMW components, it is a best practice to use virtual host names for each component, rather than the actual machine's host name. This simplifies migrating components to new hardware when you scale your system or when the hardware fails. Virtual host names are set in DNS or perhaps during development and test by updating the `hosts` file (with Linux it is found in the `/etc` directory). Virtual host names are especially important for any FMW that stores configuration information in the Metadata Repository. If real DNS names or IP addresses are used in the configuration, the repository would have to be updated when new hardware is used. By using virtual host names, the repository remains valid.

In case of a hardware failure, the virtual IP address and virtual host name are moved to a new machine along with the component binaries and configurations. Then, whatever components were running on the failed hardware are brought up on the new machine.

Domain Mode: Development

- Development mode
 - It allows applications to be auto-deployed.
 - It is OK to use demonstration digital certificates for SSL.
 - You are not prompted for a username and password to start (or stop) the admin server.
 - The admin console auto-locks the configuration by default.
 - Often no managed servers are defined in the domain.
 - The admin server handles administration and runs applications.



ORACLE

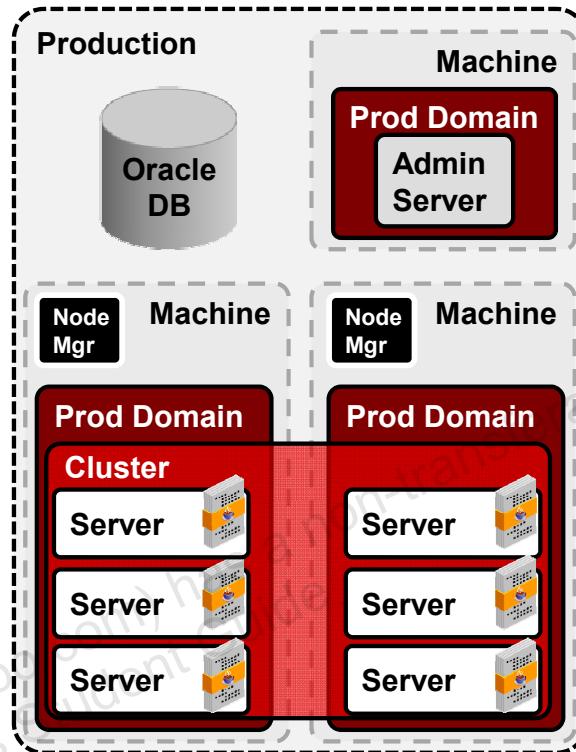
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In development mode, the administration server watches the directory in the domain called `autodeploy`. If you move an application archive file into that directory, the administration server detects it, deploys that application to itself, and starts the application servicing requests.

You are not prompted for a username and password when starting or stopping the admin server in development mode because a boot identity file (which contains the administrative credentials) is automatically created. There will be more on boot identity files in the “Starting Servers” lesson.

Domain Mode: Production

- Production mode
 - Auto-deploy is disabled.
 - You should not use the demo certificates for SSL.
 - You are prompted for a username and password to start (or stop) servers
 - The admin console does not allow auto-locking of the configuration.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Note that a test domain should mirror production as closely as possible. Sometimes fewer servers are used, there is less data in the test database, and so on.

Domain Creation Tools

- **The Configuration Wizard:** Is the graphical domain creation tool
- **WebLogic Scripting Tool (WLST):** Can create domains interactively or by running a WLST script
- **Pack and unpack utilities:** Is used to copy an existing domain to another machine



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Note that console mode of the Configuration Wizard no longer exists. To create domains from the command line, use a WLST script.

Also note that there is a Configuration Wizard script under the WebLogic Server installation directories here: <MW_HOME>/wlserver/common/bin. However, that script just calls the one under the oracle_common directory. This is important because some FMW components (for example, SOA Suite) update the Configuration Wizard so that the code required by that component is included when domains are created.

Also, when creating a domain with the Configuration Wizard, you should create the managed servers and clusters needed in the domain, so that all servers include any FMW component-required code.

In addition to the WLST commands used to create a domain, there is a WLST command called configToScript() that can read an existing domain and generate a script to re-create it.

Domains Are Created from Templates

- Domains are created from domain templates.
 - Domain templates based on FMW products are supplied with those products.
 - You can create custom domain templates by using the Template Builder tool.
- Domains can be extended with extension templates
 - Extension templates based on FMW products are supplied with those products.
 - You can create custom extension templates by using the Template Builder tool.
- The Template Builder graphical tool is found here:
 - <MW_HOME>/oracle_common/common/bin/config_builder.sh

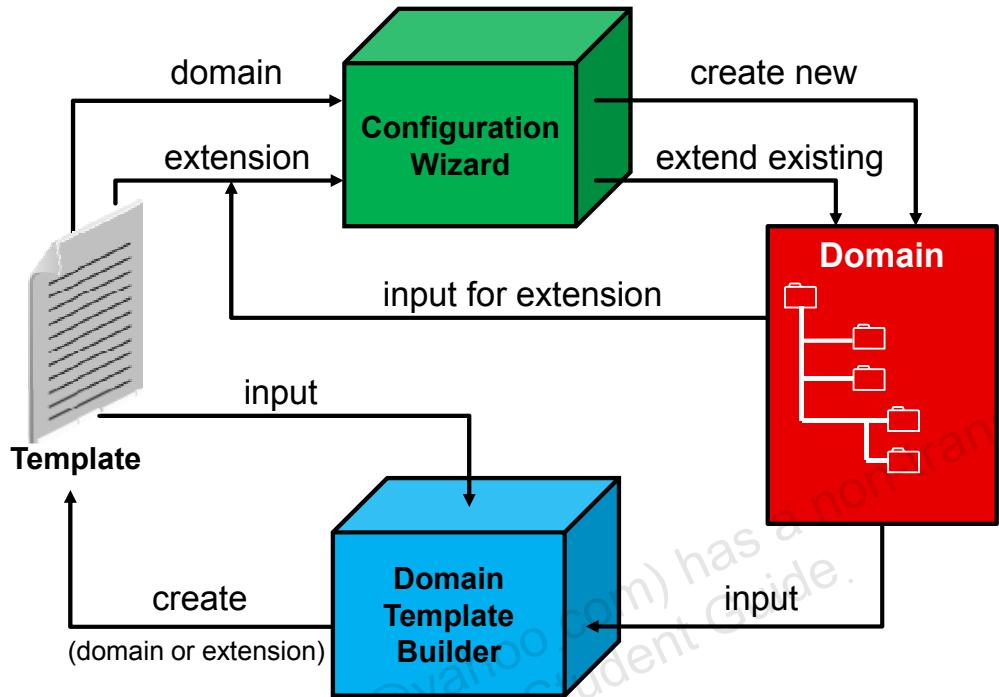


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Configuration Wizard knows which products are installed and where to locate the domain or extension templates. These template JAR files are often located under the product's Oracle home directory. For example:

- WebLogic Server templates are found here:
<MW_HOME>/wlserver/common/templates/wls
- The Oracle SOA Suite template is found here:
<ORACLE_HOME>/common/templates/applications
- Oracle Service Bus templates are found here:
<OSB_HOME>/common/templates/applications

Creating Domains



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Domains can be created by using the Configuration Wizard, which guides you through the domain creation process. It starts with either a template (a JAR file) or with built-in templates based on products. When you start the Configuration Wizard, it asks whether you want to create a new domain or extend an existing domain. You create a new domain with a domain template or prebuilt templates based on products selected from a list. You extend a domain by selecting the domain to extend, along with either an extension template or prebuilt extension templates based on products selected from a list. When you extend a domain, the domain must be “offline” (no servers running).

Templates can be built by using the Domain Template Builder. This wizard can build either domain templates, for creating new domains, or extension templates, used to extend existing domains. Templates are based on existing domains, other templates, or prebuilt product-based templates.

Where to Place the Domain

Each computer that has WebLogic Servers running on it will have a domain directory.

- The administration server domain directory is created by the Configuration Wizard.
 - It is a best practice to place the directory outside the installation directories.
 - This separates the product from your domain, which makes product upgrades and patching easier.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

All computers that run WebLogic Server have a domain directory.

- Do not place the domain directory under the product installation directories, even though that is the default. This separates the product from your domain, and should make product upgrades and patching go more smoothly.
- The main domain directory (for the administration server) is created by the Configuration Wizard.
- Make sure that each domain directory (for the same domain) is placed in the same location on each computer.
- It is a good practice to have separate domain directories for the administration server and managed servers, even on the same hardware. This isolates the administration server and make the recovery of it easier (if the computer on which it is running crashes).

Creating a Domain with the Configuration Wizard

The screenshot shows the 'Configuration Type' screen of the Oracle Fusion Middleware Configuration Wizard. On the left, a sidebar lists several options under 'Create Domain': Templates, Administrator Account, Domain Mode and JDK, Advanced Configuration, Configuration Summary, Configuration Progress, and End Of Configuration. The 'Create Domain' option is highlighted with a blue selection bar. The main panel contains the following text:
What do you want to do?
 Create a new domain
 Update an existing domain
Domain Location: /u01/domains/part1/wlsadmin
Below this is a progress bar labeled 'Create a new domain.' At the bottom are buttons for '< Back', 'Next >', 'Finish', and 'Cancel'. The Oracle logo is visible at the top right.

Run the configuration wizard script.

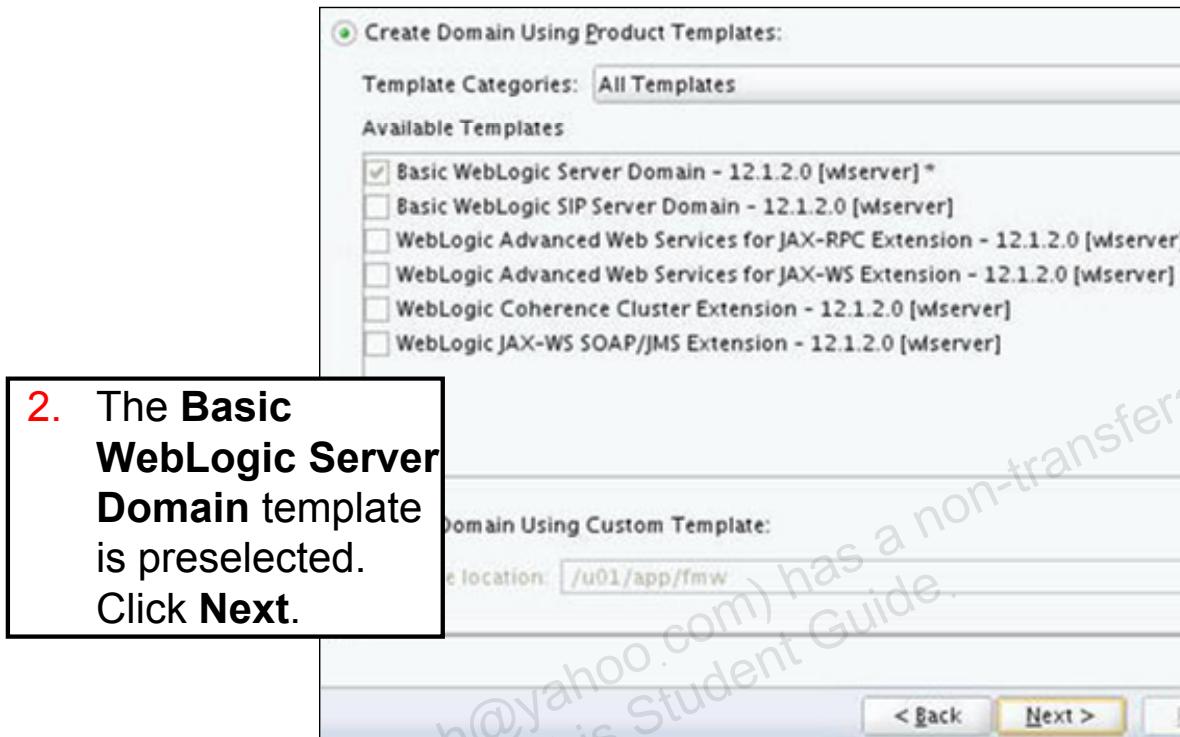
1. Select **Create a new domain** and enter a domain location. Click **Next**.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

When you create a domain, you place the domain files in the Domain Location. Oracle recommends you do not place a domain under the installation directories.

Choose **Update an existing domain** to extend a domain. In this case, the Domain Location is the location of a domain that already exists.

Creating a Domain with the Configuration Wizard



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

The drop-down list of Template Categories allows the list of templates to be shorter. By default **All Templates** is selected. The other choices are Oracle and Uncategorized Templates.

When the **Create Domain Using Product Templates** option is selected, the **Basic WebLogic Server Domain** check box is required and cannot be deselected.

The other check boxes on this wizard page are:

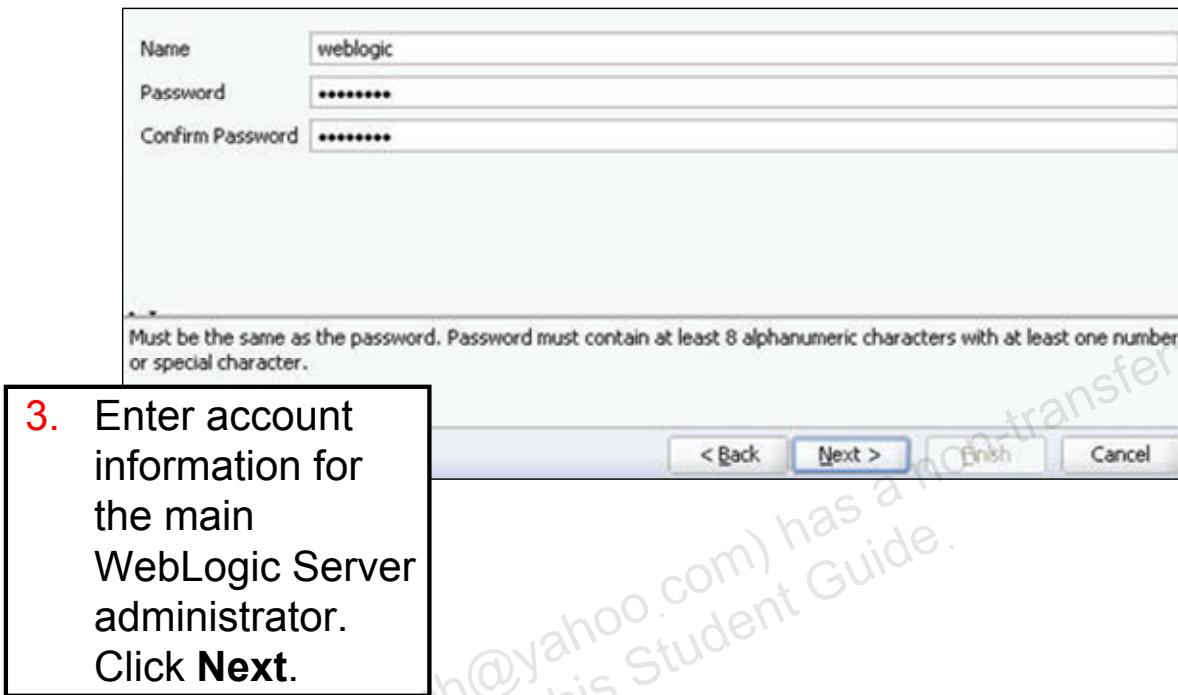
- **Basic WebLogic SIP Server Domain:** Oracle WebLogic Communication Services (OWLCS) is a comprehensive platform designed to integrate communication services with enterprise services and applications. OWLCS extends the core WebLogic Server platform with a SIP Container compliant with JSR 289. (SIP is Session Initiation Protocol, a telephony signaling protocol.) This enables the development of Java EE applications that process SIP in addition to HTTP for advanced communications applications.

- **WebLogic Advanced Web Services for JAX-RPC Extension:** It adds the functionality required for advanced JAX-RPC web services, including reliable messaging, buffering, and JMS transport.
- **WebLogic Advanced Web Services for JAX-WS Extension:** It adds the functionality required for advanced web services, including asynchronous messaging, web services reliable messaging, message buffering, web services atomic transactions, and security using WS-SecureConversation.
- **WebLogic Coherence Cluster Extension:** This adds a default Coherence cluster, `defaultCoherenceCluster`, to the WebLogic Server domain and sets the listen port for the cluster to 8088. A Coherence cluster is a collection of JVM processes. At run time, JVM processes that run Coherence automatically join and cluster. JVMs that join a cluster are called cluster members or cluster nodes. The Coherence*Web product builds on this technology and can be used to replicate session information for WebLogic Server.
- **WebLogic JAX-WS SOAP/JMS Extension:** It adds the functionality required for advanced web services using JMS.

Depending upon the installer, there can be other templates listed.

If you want to base a domain on a custom template, select **Create Domain Using Custom Template** and enter the Template Location.

Creating a Domain with the Configuration Wizard



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Creating a Domain with the Configuration Wizard



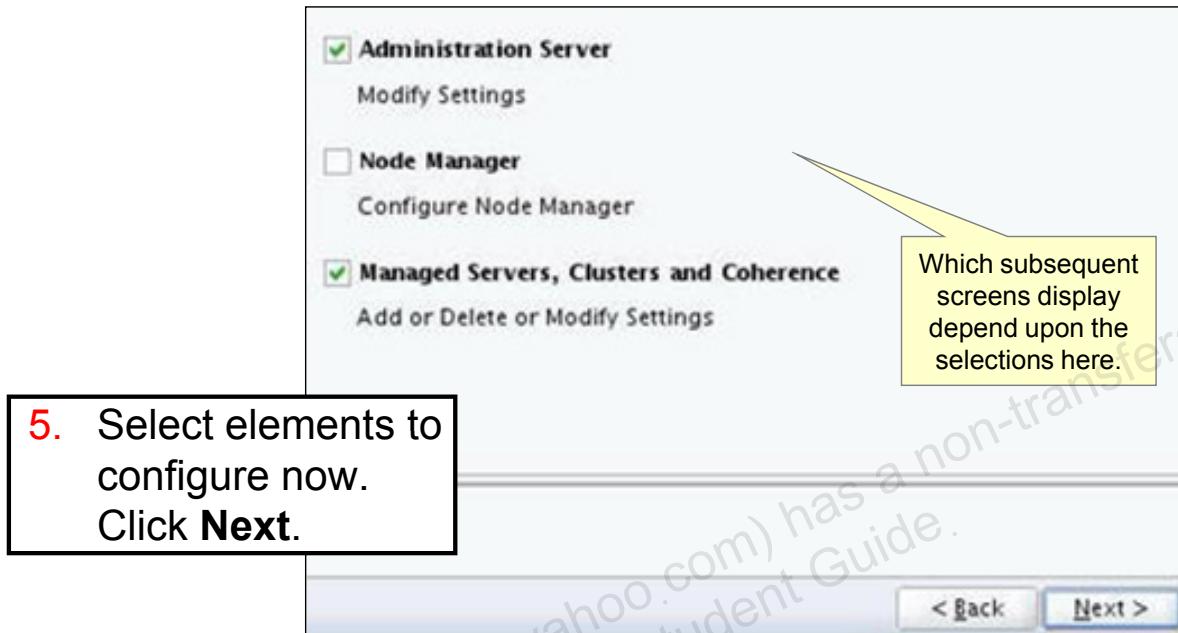
4. Select **Production** mode. Select a JDK. Click **Next**.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

The JDK selected by default is the one used to install WebLogic Server by using the generic installer. To select a different JDK, select **Other JDK Location** and enter its location.

Creating a Domain with the Configuration Wizard



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In the slide, Node Manager is not selected. It is configured later. If it is selected, the Node Manager screen is displayed, which allows you to select the Node Manager Type (Per Domain, Custom Location, or Manual Node Manager Setup), the Node Manager Home (if Custom Location is selected), and the Node Manager Credentials (Username, Password, and Confirm Password). The Node Manager credentials do not have to match the main WebLogic Server administrator credentials.

More information about Node Manager is provided in the lesson titled “Node Manager.”

Creating a Domain with the Configuration Wizard

The screenshot shows the 'Admin Server' configuration screen. It contains the following fields:

| | |
|-----------------|--------------------------|
| Server Name | AdminServer |
| Listen Address | host01.example.com |
| Listen Port | 7001 |
| Enable SSL | <input type="checkbox"/> |
| SSL Listen Port | Disabled |

At the bottom right are buttons for '< Back' and 'Next >'.

6. On the Admin Server screen, enter its name, Listen Address, Listen Port, if SSL is enabled (and, if so, the SSL Listen Port). Click **Next**.

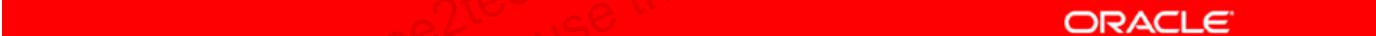
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

If you select Enable SSL, you must also enter the SSL Listen Port. It cannot be the same as the Listen Port.

Admin Server Listen Address

- By default, the Listen Address field for the administration server is “All Local Addresses.”
 - This means the server binds to all available IP addresses on the machine.
 - If the Listen Address is left blank, the effect is the same as choosing “All Local Addresses.”
- Another drop-down option is “localhost”
 - This is not a good option, since only processes that reside on this machine (local processes) can connect to this server.
- Best practice: Enter a virtual IP address or virtual host name for the Listen Address.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Creating a Domain with the Configuration Wizard

7. On the Managed Servers screen:

- A. Click **Add**.
- B. Enter the server's name, listen address, port, if SSL is enabled (and, if so, the SSL Listen Port).
- C. Do this for each one.
- D. Click **Next**.

| Server Name | Listen Address | Listen Port | Enable SSL | SSL Listen Port |
|-------------|--------------------|-------------|--------------------------|-----------------|
| server1 | host01.example.com | 7011 | <input type="checkbox"/> | Disabled |
| server2 | host02.example.com | 7011 | <input type="checkbox"/> | Disabled |

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

If you select Enable SSL on a managed server, you must also enter the SSL Listen Port. The SSL Listen Port cannot be the same as the Listen Port.

Creating a Domain with the Configuration Wizard

8. On the Clusters screen:

- A. Click **Add**.
- B. Enter the cluster's name and address (optional).
- C. Do this for each cluster.
- D. Click **Next**.

| Cluster Name | Cluster Address |
|--------------|-----------------|
| cluster1 | |
| | |
| | |

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

The Cluster Address is used in entity and stateless Enterprise JavaBeans to construct the host name portion of request URLs.

You can explicitly define the cluster address when you configure the cluster; otherwise, WebLogic Server dynamically generates the cluster address for each new request. Allowing WebLogic Server to dynamically generate the cluster address is easier, in terms of system administration, and is suitable for both development and production environments.

Creating a Domain with the Configuration Wizard

9. On the Assign Servers screen:
- A. Select a cluster.
 - B. Select a server.
 - C. Click the right arrow.
 - D. Repeat as needed.
 - E. Do this for each cluster.
 - F. Click **Next**.

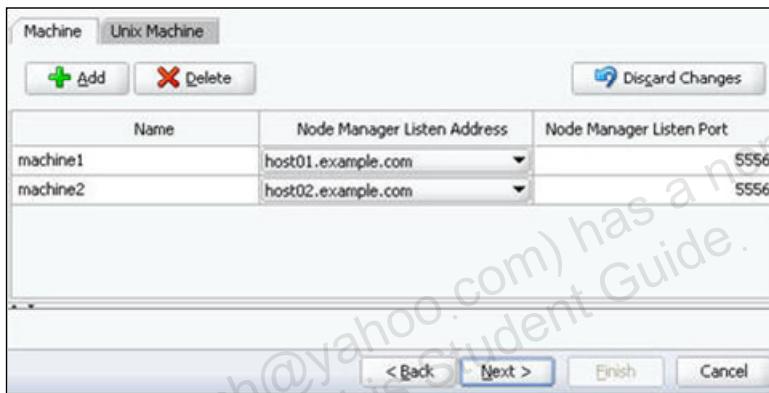


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Creating a Domain with the Configuration Wizard

10. On the Machines screen:

- A. Click the proper tab.
- B. Click **Add**.
- C. Enter the Name, Node Manager Listen Address, and Port.
- D. Do this for each machine.
- E. Click **Next**.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The wizard did not display the Coherence Clusters screen because Coherence was not selected earlier under Templates.

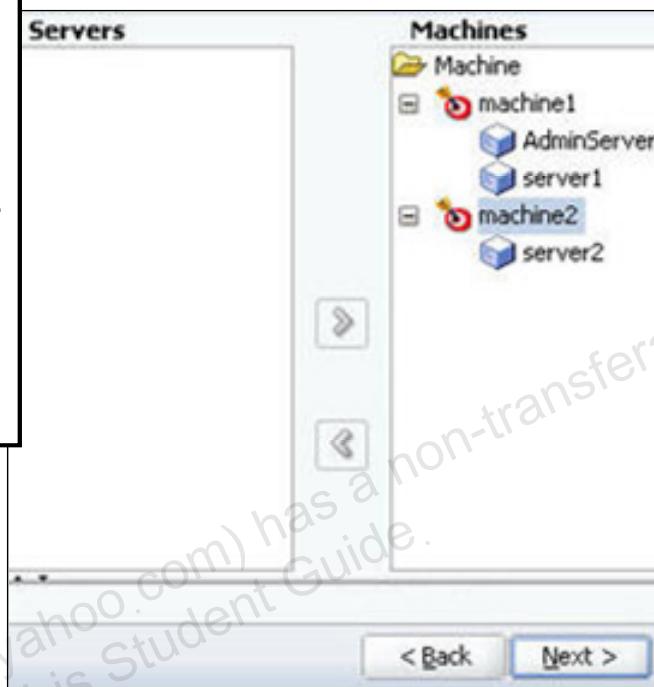
The Machine tab only has three columns: Name, Node Manager Listen Address, and Node Manager Listen Port.

The Unix Machine tab adds fields to allow the server process, after it finishes all privileged startup actions, to bind to a UNIX group ID (GID) and a UNIX user ID (UID). By default, this is not enabled, and if you do not need this capability, you can choose the Machine tab, even if this machine's operating system is UNIX (or Linux). Generally, you only need this feature when you configure your servers to low-order ports (less than 1024). The extra Unix Machine fields are:

- **Enable Post Bind:** If selected, the server should bind to a GID after it finishes all privileged startup actions.
- **Post Bind GID:** The GID
- **Enable Post Bind:** If selected, the server should bind to a UID after it finishes all privileged startup actions.
- **Post Bind UID:** The UID

Creating a Domain with the Configuration Wizard

11. On the Assign Servers to Machines screen:
- A. Select a machine.
 - B. Select a server.
 - C. Click the right arrow.
 - D. Repeat as needed.
 - E. Do this for each machine.
 - F. Click **Next**.

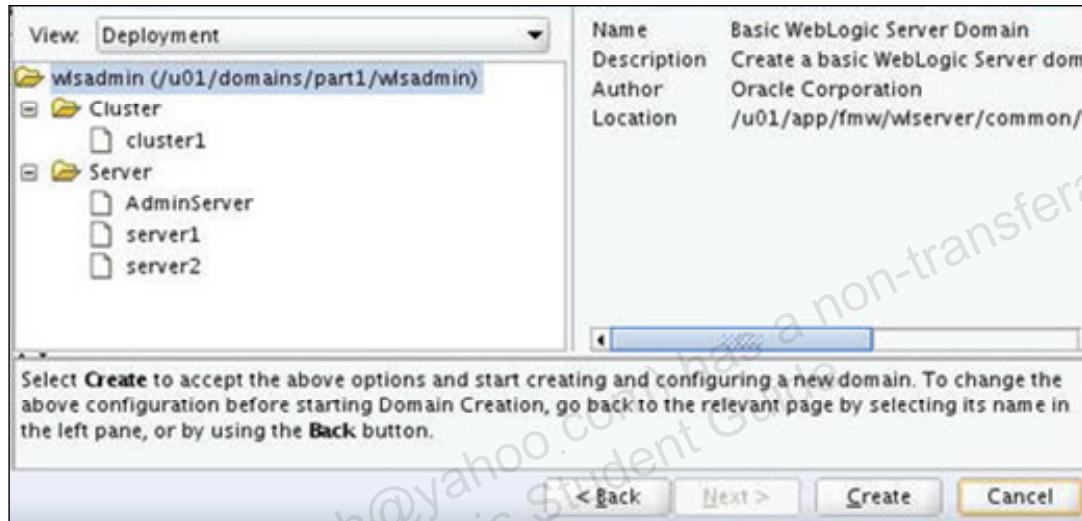


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

Creating a Domain with the Configuration Wizard

12. On the Configuration Summary screen, review the configuration and click **Create**.

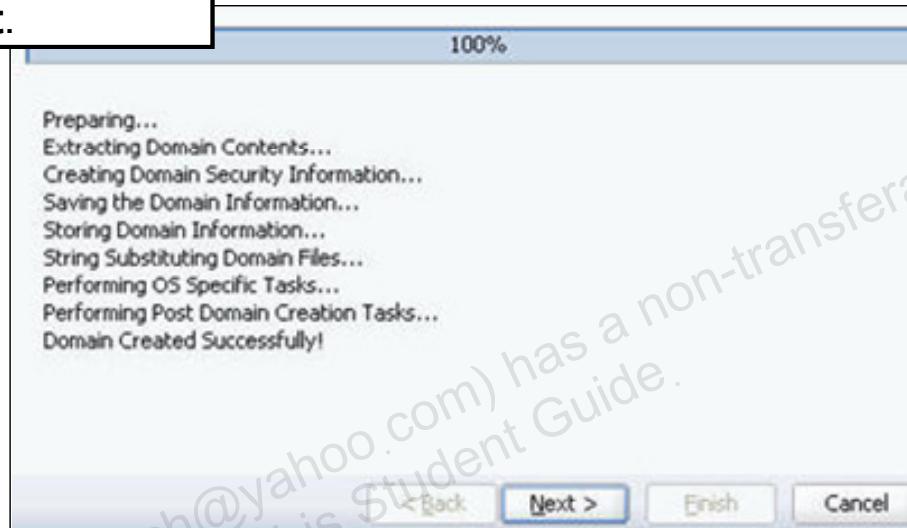


ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Creating a Domain with the Configuration Wizard

13. On the Configuration Progress screen, when the progress bar reaches 100%, click **Next**.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Creating a Domain with the Configuration Wizard

14. On the Configuration Success screen, click **Finish**.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Domain File Structure

| Directory | Description |
|----------------------|--|
| 📁 domain-name | The name of this directory is the name of the domain. |
| 📁 bin | Scripts for starting and stopping the servers in the domain |
| 📁 config | The saved configuration of the domain is contained in the config.xml file and other subdirectories and files. |
| 📁 lib | JAR files placed here are automatically added to the CLASSPATH of each WebLogic Server started on this machine. |
| 📁 nodemanager | The default location for the domain's Node Manager |
| 📁 pending | Domain configuration changes that have been saved but not yet activated are stored here temporarily. |
| 📁 security | Domain-wide security-related files |
| 📁 servers | One subdirectory for each server in the domain |
| 📁 server1 | The server directory for the server of the same name |
| 📁 data | Data for internal LDAP, Node Manager, and saved diagnostics |
| 📁 logs | Server log files |
| 📁 stage | Default staging directory for deployed applications |

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

There are more directories under a domain than those listed in the slide.

Creating a Domain to Support FMW Components

Domain templates are supplied when certain FMW components are installed, such as Oracle SOA Suite.

- If you add another FMW component, extend the domain with that product's extension template.

| Existing Domain Template | Components that Can Be Added/Registered |
|---|--|
| Oracle SOA Suite | Any other Oracle SOA Suite component Any Oracle WebCenter component Any Web Tier component |
| Oracle Identity Management | Other Identity Management components Any Web Tier component |
| Oracle Portal, Oracle Reports, Oracle Forms Services, Oracle Business Intelligence Discover | Any of these components Any Web Tier component |



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

When a WebLogic Server domain is created, it is created with a template. Usually that template is product based. In other words, it is the one that supports a particular component, such as Oracle SOA Suite. If you want to add other Fusion Middleware components to that domain, such as Oracle WebCenter, you extend the domain by using an extension template for the new component that you want to add. When you extend a domain, the domain must be offline (no servers running).

The Oracle Identity Management suite of components includes Oracle Internet Directory (a general purpose directory service that combines LDAP with the performance and scalability of an Oracle Database), Oracle Identity Federation (a self-contained federation server that enables single sign-on and authentication in a multiple-system identity network), Oracle Identity Manager (a user provisioning and administration product), Oracle Access Manager (a product that provides a full range of perimeter security functions that include single sign-on, authentication, authorization, policy administration, auditing, and more), Oracle Adaptive Access Manager (a product that provides risk-aware authentication, real-time behavior profiling, and risk analysis), and Oracle Authorization Policy Manager (a graphical interface tool to manage application authorization policies). If a domain is created to include a product in the Oracle Identity Management suite, it can be extended with other products in that suite.

Extending a domain for system components (like the Web Tier components) does not add them to the WebLogic Server domain, but rather registers them with the domain so that they may be controlled and monitored by Fusion Middleware Control.

Additionally, you may need to run the Repository Creation Utility (RCU) to add any required schemas for the new component, and, of course, the new component must also be installed. After it is installed, the prebuilt templates for that component will be available to the Configuration Wizard.

The Domain on Other Hardware

Remember, each computer that has instances of WebLogic Server running on it must have a domain directory.

- The administration server domain directory is created by the Configuration Wizard.
- To create the domain directory on other computers (for managed servers) use the pack utility to create a managed server template. Move the managed server template JAR file to the other computer, and then use the unpack utility.
- It is a best practice to place the domain directory in the same location on all computers that run that domain's servers.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

All computers that run WebLogic Server will have a domain directory.

- The main domain directory (for the administration server) will be created by the Configuration Wizard.
- To create domain directories for other computers (where the managed servers will be running), use the pack utility to create a managed server template. The pack utility is found here: <MW_HOME>/oracle_common/common/bin/pack.sh. After the managed server template JAR file has been created, move it to the machine where the managed server or servers will be running (and where WebLogic Server has already been installed), and use the unpack utility:
`<MW_HOME>/oracle_common/common/bin/unpack.sh`
- Make sure that each domain directory (for the same domain) is placed in the same location on each computer.

Creating the Domain Archive: Pack

1. On the administration server machine, use the `pack.sh` script with the managed option.

```
$> cd <MW_HOME>/oracle_common/common/bin  
$> ./pack.sh -domain=domain_path/domain_name  
              -template=name.jar  
              -template_name=somename  
              -managed=true
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Note that there are other available options for the `pack.sh` script.

Using the Domain Archive: Unpack

2. Move the JAR file to the machine where a managed server will run. (The WebLogic Server product must already be installed there.)
3. Before running the `unpack.sh` script on that machine, create the directory in which to place the domain. (In the example below, it is called `domain_path`.)
4. Run the `unpack.sh` script:

```
$> cd <MW_HOME>/oracle_common/common/bin  
$> ./unpack.sh -domain=domain_path/domain_name  
                  -template=pathtojar/name.jar
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Note that there are other available options for the `unpack.sh` script.

Quiz

Domains are created from _____.

- a. The administration server
- b. WAR files
- c. Templates
- d. The administration console



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: c

Quiz

To copy a domain from the administration server machine to a managed server machine, use the _____.

- a. Pack and unpack utilities
- b. Configuration Wizard
- c. Template builder
- d. Zip and unzip utilities



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a

Summary

In this lesson, you should have learned how to:

- Describe a domain's file system
- Create a domain by using the Configuration Wizard
- Configure resources by using the Configuration Wizard
- Copy a domain to another computer with the pack and unpack utilities

Practice 4-1 Overview: Creating a New Domain

This practice covers creating a new domain by using the Configuration Wizard.



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

This practice covers copying a domain to another machine that will run managed servers.



Unauthorized reproduction or distribution prohibited. Copyright© 2016, Oracle and/or its affiliates.

Tony Albrecht (base2tech@yahoo.com) has a non-transferable
license to use this Student Guide.

Starting Servers

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Start and stop servers with standard scripts
- Deal with server startup problems
- Customize start and stop scripts

WebLogic Server Life Cycle

As a server starts, stops, and runs, it finds itself in various states. The most important of these states are:

- **RUNNING:** The server is fully functional, offering its services to clients.
- **SHUTDOWN:** The server is configured, but not active.
- **ADMIN:** The server is running, but available only for administrative operations. Services are available to administrators, but are not available to other clients.
- **FAILED:** The server itself has failed (for example, it is out of memory) or it has detected that one or more critical subsystems are unstable.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Note that this is a simplification of the server life cycle, showing only the most important states. There are quite a few transitional states that are not shown. See the “Understanding Server Life Cycle” chapter in the *Administering Server Startup and Shutdown for Oracle WebLogic Server* document for the other states, as well as many details on how a server transitions from one state to another.

The states shown are:

- **RUNNING:** In the RUNNING state, WebLogic Server is fully functional, offering its services to its clients. It can also operate as a full member of a cluster. A server instance transitions to the RUNNING state as a result of a “start” command, or the “resume” command from the ADMIN state.
- **SHUTDOWN:** In the SHUTDOWN state, this instance of WebLogic Server is configured, but inactive. A server enters the SHUTDOWN state as a result of a “shutdown” or “force shutdown” command. You can transition a server in the SHUTDOWN state through transitional states to either RUNNING (with a “start” command) or ADMIN (with a “start in admin” command).

- **ADMIN:** In the ADMIN state, WebLogic Server is up and running, but available only for administration operations. You can start a server in the ADMIN state by using a “start in admin” command. Also, a server transitions through the ADMIN state when going from SHUTDOWN to RUNNING and vice versa. When a server is in the ADMIN state:
 - The administration console is available.
 - The server accepts requests from users with the Admin role. Requests from non-admin users are refused.
 - Applications are activated in the application ADMIN state. The applications only accept requests from users with the Admin and AppTester roles.
 - The subsystems are active, and administrative operations can be performed upon them.
 - Deployments are allowed, and take effect when the server goes back to the RUNNING state.
 - If the server is part of a cluster, the Cluster Service is active and listens for heartbeats and announcements from other cluster members, but this server is invisible to the other cluster members.
- **FAILED:** A running server can fail as a result of out-of-memory exceptions or stuck application threads. Also, as a server monitors its health, if it detects that one or more critical subsystems are unstable, it declares itself FAILED. When a server enters the FAILED state, it attempts to return to a nonfailed state. If it failed before reaching the ADMIN state, the server instance shuts itself down. If the server fails after reaching the ADMIN state, but before reaching the RUNNING state, by default, it returns to the ADMIN state if an administration port has been configured. A server can enter the FAILED state from any other state. However, once a server has entered the FAILED state, it cannot return to a RUNNING state directly. The FAILED state is fatal and a server must go into the ADMIN or SHUTDOWN state before returning to the RUNNING state.

Note: The commands in the explanation above are shown in double quotes (for example, “start in admin” command) to indicate that there are such commands, without giving what the command actually is. For example, the “start” command in the administration console is a button labeled **Start**. The “start” command in the WebLogic Scripting Tool (WLST) is either `start()` or `startServer()`.

Starting WebLogic Server with a Script

When a domain is created, scripts are generated and placed in the domain's bin directory.



- The administration server start script:
 - `startWeblogic.sh`
 - No parameters

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

- Managed servers start script:
 - `startManagedWeblogic.sh`
 - Parameter one: The name of the managed server (required)
 - Parameter two: The URL of the admin server (optional)

```
$> ./startManagedWebLogic.sh server1
http://host01.example.com:7001
```

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

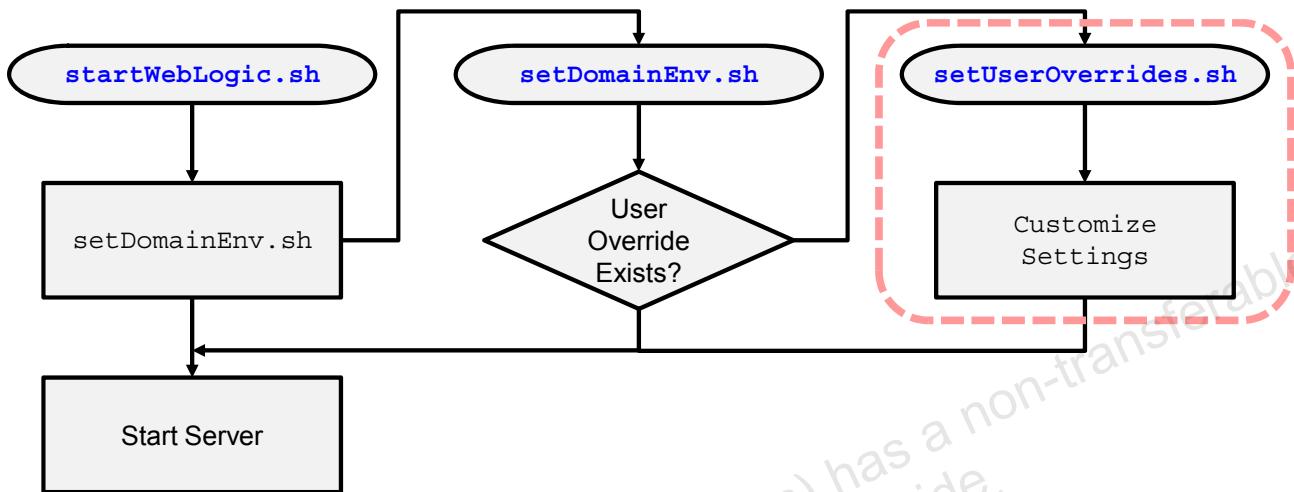
To start the administration server, open a terminal window and change to the `bin` directory under the domain directory. Run the `startWebLogic.sh` script. If prompted for a username and password, enter the administrator credentials set when the domain was created. Note that this script calls the `setDomainEnv.sh` script that sets the environment variables and options. Also note that there is a convenience script in the main domain directory, also called `startWebLogic.sh`. This script calls the script of the same name in the `bin` directory.

To start a managed server, open a terminal window and change to the `bin` directory under the domain directory. Run the `startManagedWebLogic.sh` script with its parameters. The first parameter is required and is the name of the managed server to start. The second parameter is the URL of the administration server of the domain. If omitted, it defaults to `http://localhost:7001`. If prompted for a username and password, enter the administrator credentials set when the domain was created. This managed server start script actually calls the `startWebLogic.sh` script, which calls the `setDomainEnv.sh` script.

The `setDomainEnv.sh` script calls a generic “set the environment” script under the install directories: `<WL_HOME>/common/bin/commEnv.cmd`.

Note that in a Microsoft Windows environment, you use the equivalent scripts that end in `.cmd`.

Customizing the Scripting Environment



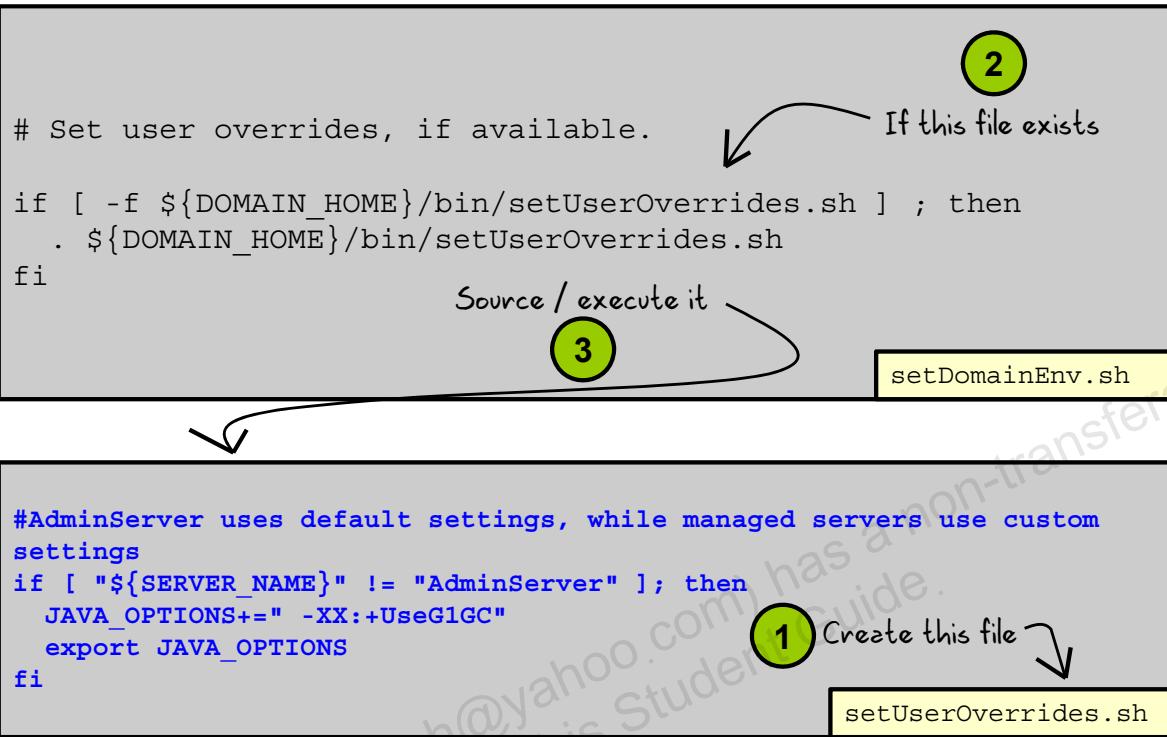
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You customize environment settings for starting and stopping servers by creating a script named `setUserOverrides` in the `DOMAIN_HOME/bin` folder. This script enables you to customize your domain safely because your changes are maintained outside of the `setDomainEnv` script. When your domain is updated, either to a new release or through some domain configuration tool, the `setUserOverrides` script is not affected.

Here is a flow chart that demonstrates how this works.

- When the `startWebLogic` script is executed, it sources the `setDomainEnv` script.
- When the `setDomainEnv` script is executed, it checks to see whether the `setUserOverrides` script exists.
- If it doesn't exist, then normal environment processing is performed and control is returned to the `startWebLogic.sh` script.
- If it exists, then it sources the `setUserOverrides` script.
- Any customized settings are set in the `setUserOverrides` script, and control is returned to `setDomainEnv`, and then to `startWebLogic`.
- The start script continues processing and the server is started.

Customizing the Scripting Environment



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

1. First, you create the `setUserOverrides.sh` file and set your custom settings.
2. When the `startWebLogic.sh` script is executed, it sources the `setDomainEnv.sh` script, which in turn checks to see if the `setUserOverrides.sh` script exists. This code shows the file check.
3. If the file exists, which it does because you just created it, it is sourced to include its settings as part of the initialization process to start the server.

Creating a Boot Identity File

Starting or stopping WebLogic Server requires administrative authority. The scripts prompt for a username and password.

A boot identity file contains administrator credentials, making prompting unnecessary.

1. Each server has its own directory in the domain. For example, servers/server1. Under that directory create a new subdirectory called security.
2. Create a text file there called boot.properties:

```
username=adminuser  
password=thepassword
```

3. Start the server with the script. The server reads the file for its credentials. If they are valid, the server starts. The server also encrypts both values in the file so that they are now secure.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A boot identity file contains the user credentials for starting and stopping an instance of WebLogic Server. An administration server or managed server can refer to this file for user credentials rather than prompt at the command line for them. Because the server encrypts the credentials in the boot identity file, it is more secure than storing plain text credentials in start or stop script. There can be a different boot identity file for each server in a domain. Both start and stop scripts can use a boot identity file.

If you choose Development Mode when creating a domain by using the Configuration Wizard, a boot identity file is automatically created for the administration server. Boot identity files must be manually created for a Production Mode domain.

If you use Node Manager to start managed servers, rather than running start scripts manually, you do not need to create boot identity files for those servers. Node Manager creates its own boot identity files and stores them under each server's directory in the data/nodemanager subdirectory.

Stopping WebLogic Server

A server can be stopped:

- By using the admin console. There are two options:
 - **When work completes** (sessions complete or time out)
 - You can set the **Graceful Shutdown Timeout** for a server (seconds to wait before switching to a forced shutdown)
 - **Force Shutdown Now** (session data lost)
- With a standard stop script
 - These standard domain scripts do a forced shutdown
- By killing its JVM process
 - `kill -2`
 - Does a forced shutdown
 - `kill -9`
 - “Hard” kill, no “shutdown” code; use only as a last resort

The same effect as pressing **Ctrl+C**
in the Terminal window in which a
server is running.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The `stopWebLogic.sh` script is for stopping the admin server. It takes no parameters.

The `stopManagedWebLogic.sh` script is for stopping a managed server. It takes two parameters:

- The managed server name (required)
- The URL of the admin server (optional—defaults to `t3://localhost:7001`)

See the “Understanding Server Life Cycle” chapter in the *Administering Server Startup and Shutdown for Oracle WebLogic Server* document for more details (for example, what the server normally does when shutting down and the possible server states).

Note: If the admin server is writing to the configuration files when its process is killed abruptly (`kill -9`), the configuration could be corrupted.

Suspend and Resume

A server that is suspended goes from the RUNNING state to the ADMIN state.

- A server can be suspended by using the admin console. There are two options:
 - When work completes
 - Force Suspend Now

A suspended server can be resumed. It then transitions from the ADMIN (or STANDBY) state to the RUNNING state.

- A server can be resumed by using the admin console.
- STANDBY is a state in which a server has completed its startup procedure, but does not process any requests. (Its regular listen port is closed.) This is used to keep a server ready as a “hot” backup.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To start a server in STANDBY, you can change its default startup mode by using the admin console. Select the server, then click the **Configuration > General** tab. Click **Advanced**. Change the Startup Mode field to **standby**. You must have an admin port defined for the domain, so that the server can listen on its admin port for the command to resume.

Customizing Standard Scripts

The standard start scripts can be customized to:

- Change WebLogic Server runtime options
- Change which JVM runs WebLogic Server
- Change JVM options
 - Performance tuning of the JVM
- Modify the CLASSPATH



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

WebLogic Server Options

The options of `weblogic.Server` are used to:

- Change the location of configuration data
 - Examples:
 - WebLogic home: `-Dweblogic.home=path`
 - Default: Determined by CLASSPATH
 - Server root directory: `-Dweblogic.RootDirectory=path`
 - Default: The directory from which the server was started
- Override a server's configuration
 - Examples:
 - `-Dweblogic.ListenAddress=host`
 - `-Dweblogic.ListenPort=portnumber`
 - Instead, make changes by using the admin console/WLST.

The Java class that starts WLS

This line could be added to a start script such as `startWebLogic.sh`.

```
JAVA_OPTIONS="${JAVA_OPTIONS} -Dweblogic.ListenPort=9001"
```

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Setting environment variables is shown as it is done in Linux.

There are many more `-D` options to override a server's configuration. See the *Command Reference for Oracle WebLogic Server*.

WebLogic Server Options

The options of `weblogic.Server` are used to:

- Change the server type
 - Normally all server services are started.
 - If you do not need EJBs, JCA (Java EE Connector Architecture), or JMS, you can run the lighter version of WebLogic Server that does not start those services.
 - Use the `-DserverType=wlx` option.

```
JAVA_OPTIONS="${JAVA_OPTIONS} -DserverType=wlx"
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The lighter version of WebLogic Server takes less memory and starts up faster.

The Java EE Connector Architecture is used by tool vendors and system integrators to create resource adapters. A *resource adapter* is a software component that allows Java EE applications to access and interact with the underlying resource manager of an enterprise information system (EIS).

Changing the JVM

- There are environment variables used to indicate which JVM runs WebLogic Server. In `setDomainEnv.sh`, there are the following variables:
 - `VM_TYPE`: Values such as "HotSpot" or "JRockit"
 - `JAVA_VENDOR`: Values such as "Oracle" or "Sun"
 - `JAVA_HOME`: Location of the JDK
- If, for example, both the HotSpot and JRockit JDKs are installed, ensure that the `SUN_JAVA_HOME` and `BEA_JAVA_HOME` variables are set properly. Then, near the top of `setDomainEnv.sh`, add a line setting `VM_TYPE`:

`VM_TYPE="JRockit"` or `VM_TYPE="HotSpot"`

 - Logic in the script will set the other variables accordingly.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

JVM Options

- The options that are available depend upon the JVM.
- For memory options, set the `USER_MEM_ARGS` variable in `setDomainEnv.sh`.
 - `Xms` is the minimum heap size.
 - `Xmx` is the maximum heap size.
 - `XX:MaxPermSize` sets the maximum permanent space for a HotSpot JVM.
- For other options, update the `JAVA_OPTIONS` variable in `setDomainEnv.sh`.
 - For example, to choose the “concurrent mark-sweep” garbage collection algorithm of Hotspot:

Where Java
classes are kept

```
USER_MEM_ARGS="-Xms64m -Xmx200m -XX:MaxPermSize=350m"
```

```
JAVA_OPTIONS="${JAVA_OPTIONS} -XX:+UseConcMarkSweepGC"
```

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The heap is the area of memory where objects are created and deleted dynamically. It has a configurable minimum and maximum size and is organized into different areas called generations. Objects that exist for a certain amount of time are promoted to the next generation. Older generations are garbage collected less frequently. The permanent generation, in a HotSpot JVM, is where application Java classes are stored. The allocation and deallocation of memory for objects is handled by the JVM. Garbage collection is the process of detecting when an object is no longer referenced, clearing the unused object from memory, and making reclaimed memory available for future allocations. JVM garbage collection happens automatically.

There are different algorithms for how JVMs perform garbage collection. One algorithm is called concurrent mark-sweep. For an application where fast response time is the most important performance criteria, this algorithm might be chosen. Generally, garbage collection for a young generation does not cause long pauses. However, old generation collections, though infrequent, can sometimes create longer pauses. With the concurrent mark-sweep algorithm most of the collection of old generations is done concurrently with the execution of the application so those pauses do not affect the application.

A JVM often has many options that can be set. See the particular JVM documentation.

Modifying the CLASSPATH

- The Java CLASSPATH tells the JVM where to find user-created packages and classes.
- There are many environment variables relating to the CLASSPATH. They are used to update the CLASSPATH for patches and libraries. These variables are set in:
 - *domain_name/bin/startWebLogic.sh*
 - *domain_name/bin/setDomainEnv.sh*
 - *<MW_HOME>/oracle_common/common/bin/commEnv.sh*
- To modify the CLASSPATH for:
 - All domains, edit the *commEnv.sh* script
 - A particular domain, edit that domain's *setDomainEnv.sh* script

These files may need to be modified on multiple hosts.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The CLASSPATH makes available to the JVM (and WebLogic Server) classes that are required for the server. The classes that make up the server itself, classes that patch the current version of the server, classes that make up tools such as Log4J or Ant, startup and shutdown classes, and database driver classes are the kinds of classes that go into the CLASSPATH. If the classes for the CLASSPATH are compiled in the expanded format, the directory that contains their package directory is added to the CLASSPATH. If the compiled classes are in a Java archive (JAR) file, the path to and the name of the JAR file is added to the CLASSPATH.

The CLASSPATH is not for application classes. Those compiled classes reside within application archives. There are better ways to share classes between applications than modifying the CLASSPATH. Use Java Optional Packages or WebLogic Server Shared Libraries for sharing application classes among deployed applications.

If you run instances of WebLogic Server on multiple hosts, remember that the scripts you are modifying reside on each host—whether you are modifying the *commEnv.sh* script under the installation directories or the *setDomainEnv.sh* under the domain directories.

Modifying the CLASSPATH

To modify the `commEnv.sh` installation script:

- Add your JAR file to the `WEBLOGIC_CLASSPATH` environment variable.
 - If your classes need to be found first, place your JAR file as the first thing in the string.

```
WEBLOGIC_CLASSPATH="/tools/mytool.jar${CLASSPATHSEP}..."
```

To modify the `setDomainEnv.sh` domain script:

- Create the `PRE_CLASSPATH` environment variable, if it does not exist, and assign it to your JAR file.
- If the variable already exists, add your JAR file to it.

```
PRE_CLASSPATH="/tools/mytool.jar"
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The `CLASSPATHSEP` variable is used as the `CLASSPATH` delimiter so the same script can be used in multiple operating systems. It is set in the `commEnv.sh` script.

Modifying the CLASSPATH

- To add code in a JAR file to the CLASSPATH of all servers in the domain, without changing any of the startup or set environment scripts, place the JAR file in the `lib` directory of the domain.
 - All JAR files in this directory are added dynamically to the end of each domain server's CLASSPATH the next time the server starts.
 - Note that the JAR file needs to be placed in the domain's `lib` directory on each host on which servers from this domain run.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

WebLogic Server Startup Issues

- Severe errors during server startup may cause:
 - WebLogic Server to exit prematurely
 - The JVM to crash (rare)
- Common causes include:
 - Missing or invalid server start parameters
 - Another process using the address and port of the server
 - Custom start scripts with errors
 - Missing or invalid boot credentials or SSL files
 - An invalid or corrupt domain configuration

```
<Critical> <WebLogicServer> <BEA-000362> <Server failed.
  Reason: [Management:141245] Schema validation error...>
...
<Error> <WebLogicServer> <BEA-000383> <A critical service
  failed. The server will shut itself down.>
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Possible server startup error messages include the following:

- Unable to create a server socket. The address <listenAddress> might be incorrect or another process is using port <port>.
- Cannot use SSL because no certificates have been specified in the WebLogic Server configuration
- Cannot read private key file <file>. Exception is <exception>.
- Server failed. Reason <reason>.
- Unable to initialize the server: <server>

The first time you start a managed server instance, it must be able to contact the administration server. Thereafter, the managed server can start even if the administration server is unavailable. This is called managed server independence (MSI) mode and is enabled by default. The managed server starts by using its local copy of the configuration.

To start and stop WebLogic Server, you must provide the credentials of a user who is permitted to start and stop servers for the domain. Users must fit the criteria of the global Admin role defined in the security realm. By default, this role includes all members of the Administrators group.

Failed Admin Server

- If the administration server fails:
 - There is no effect on running managed servers or their applications, so your users are unaffected.
 - When the admin server returns, managed servers reconnect.
 - A managed server not running can be started without the admin server, if it has:
 - Been started at least once before to obtain its own copy of the configuration
 - Managed server independence (MSI) mode enabled
 - No changes to the configuration can be made.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Restarting a Failed Admin Server: Same Machine

- To restart the admin server on the same machine:
 - Run the start script again
 - If the server fails again:
 - Analyze the reason for the failure (from the server log)
 - Correct the issue
 - Run the start script
 - If the admin server was started by Node Manager, Node Manager can be set to automatically restart it.
 - When the admin server has restarted, the running managed servers will reconnect to it.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Restarting a Failed Admin Server: Different Machine

- To restart the admin server on a different machine:
 - The admin server backup machine must have already been configured and contain:
 - WLS installation
 - Domain files
 - Deployed applications
 - If using a virtual IP address, move it to the backup machine.
 - Start the admin server (start script or Node Manager).
- When the admin server has restarted, the running managed servers reconnect to it if the admin server has:
 - A virtual IP address
 - A host name that maps to multiple IP addresses



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

If the admin server is started with a different IP address but has the same host name, its managed servers can find it. Therefore, it is good planning to use a virtual host or DNS name that maps to multiple IP addresses for the admin server.

Restarting a Failed Managed Server: Same Machine

- To restart a managed server on the same machine:
 - Run the start script again
 - If the server fails again:
 - Analyze the reason for the failure (from the server log)
 - Correct the issue
 - Run the start script
 - If the managed server was started by Node Manager, Node Manager can be set to automatically restart it.
 - As the managed server is restarted, it reconnects with its admin server.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Restarting a Failed Managed Server: Different Machine

- To restart a managed server on a different machine, it needs:
 - WLS installation
 - Domain files
 - Deployed applications
 - JTA and JMS artifacts
 - If using a virtual IP address, move it to the backup machine
- Start the managed server (start script or Node Manager).
- As the managed server is restarted (even if on a different IP address), it reconnects with its admin server.
- A clustered managed server that fails can be configured to migrate to another machine automatically or manually.
 - JTA and JMS artifacts must be available.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

WebLogic Server provides the capability to migrate clustered managed server instances. If a clustered managed server has been configured to be migratable and it fails and cannot be restarted on its current hardware, it can be moved to new hardware either manually by an administrator or automatically. The migration process makes all of the services running on the server instance available on a different machine, but not the state information for the JTA and JMS services that were running at the time of failure. For whole server migration or service-level migration to work properly, the artifacts of these services (TLogs for JTA and JMS stores for JMS) must be available to the new hardware. Therefore, these artifacts must be in a database that is accessible or in shared storage that is accessible.

For more information, see “Whole Server Migration” in *Oracle Fusion Middleware Administering Clusters for Oracle WebLogic Server*.

Quiz

The `startManagedWebLogic.sh` script takes two parameters. They are:

- a. The name of the admin server and the URL of the managed server
- b. The name of the managed server and the URL of the managed server
- c. The name of the admin server and the URL of the admin server
- d. The name of the managed server and the URL of the admin server



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: d

Quiz

The name of the boot identity file is:

- a. boot.identity
- b. identity.properties
- c. boot.properties
- d. password.properties



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: c

Summary

In this lesson, you should have learned how to:

- Start and stop servers with standard scripts
- Deal with server startup problems
- Customize start and stop scripts

Practice 5-1 Overview: Starting and Stopping Servers

This practice covers the following topics:

- Starting the administration server with a script
- Starting managed servers with a script
- Creating a boot identity file
- Modifying server start scripts



Configuring JDBC

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

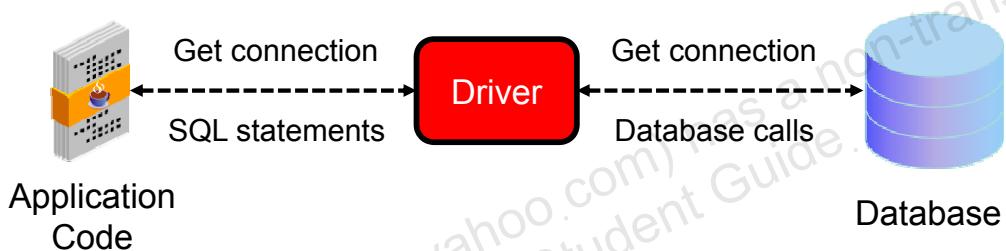
Objectives

After completing this lesson, you should be able to configure:

- A generic data source
- A GridLink data source

JDBC: Overview

- The Java Database Connectivity (JDBC) API:
 - Is a platform and vendor-independent mechanism for accessing and using a database
 - Provides transparency from proprietary vendor issues
 - Requires the use of a *driver* (a Java class)
- JDBC drivers are supplied with the WebLogic Server installation or by your database vendor.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The JDBC API is the way in Java code to work with the database Structured Query Language (SQL). It builds on Open Database Connectivity (ODBC), so developers familiar with ODBC find it easy to use.

By using JDBC, a Java application can access virtually any relational database and run on any platform with a Java Virtual Machine (JVM). That is, with JDBC, it is not necessary to write one program to access a Sybase database, another to access an Oracle database, another to access an IBM DB2 database, and so on. You can write a single program by using the JDBC API. Because the application is written in Java, you need not write different applications to run on different platforms, such as Windows and Linux.

JDBC accomplishes database connections by using a driver mechanism that translates the JDBC calls to native database calls. Although most available drivers are fully written in Java (Type 4 drivers), and are thus platform independent, some drivers (Type 2) use native libraries and are targeted to specific platforms.

WebLogic JDBC Drivers

- Oracle and third-party drivers for many databases are included in the WebLogic Server installation:
 - Oracle 11g and 12c
 - Sybase
 - Microsoft SQL Server
 - IBM DB2
 - Informix
 - MySQL
- By default, these drivers are added to the server's CLASSPATH. To use other drivers, you must update the server's CLASSPATH.
- XA (Extended Architecture) drivers provide support for global transactions.

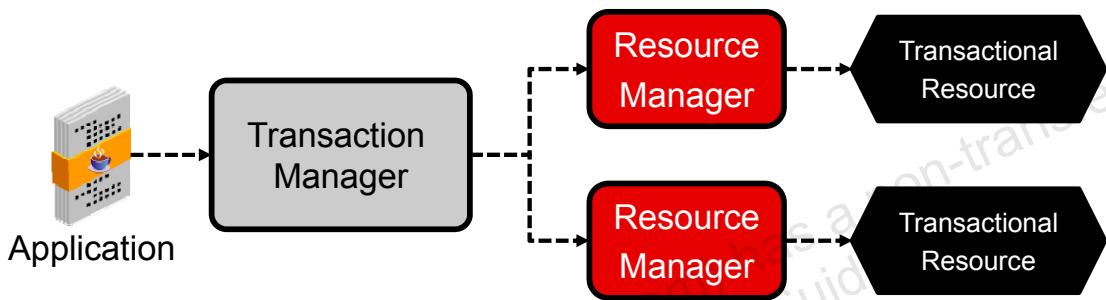


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The WebLogic JDBC drivers are installed with Oracle WebLogic Server in the <WL_HOME>/server/lib folder, where <WL_HOME> is the directory in which you installed Oracle WebLogic Server. Driver class files are included in the manifest CLASSPATH in weblogic.jar, so the drivers are automatically added to the server's CLASSPATH.

Global Transactions: Overview

- A global (distributed) transaction involves more than one transactional resource.
 - A transaction manager (TM) deals with each resource manager (RM). WebLogic Server can act as a TM.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

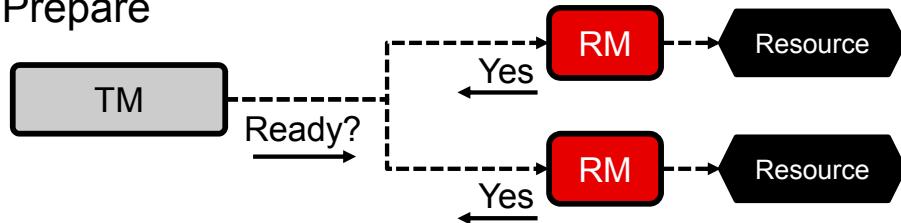
WebLogic Server can act as the transaction manager to the various transactional resource managers in a global or distributed transaction.

There will be more on global transactions in the lesson titled “Transactions.”

Two-Phase Commit

The Two-Phase Commit (2PC) protocol uses two steps to commit changes within a global transaction:

- Phase I: Prepare



- Phase II: If all resources are ready, the TM tells them to commit. Otherwise, the TM tells them to roll back.



Extended Architecture (XA) implements 2PC.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Two-phase commit protocol:

- **Phase 1:** TM asks RMs to prepare to make the changes.
- **Phase 2:** If all RMs report that they are ready to commit, TM tells the RMs to commit, which makes the changes permanent. If any RM is not ready to commit, TM tells all RMs to roll back (undo any changes).

The Extended Architecture (XA) specification comes from the Open Group (<http://www3.opengroup.org>), a global consortium that works on IT standards.

JDBC Data Source

- A data source is a Java object targeted to and managed by one or more instances of WebLogic Server.
- A deployed data source has connections to a particular database in its connection pool ready-to-go for applications running on those servers.
 - The connection pool configuration determines which database is used and how many connections are in the pool.
 - Applications locate a data source in a server's tree of resources by using the Java Naming and Directory Interface (JNDI) API.
 - After the application has a reference to the data source, it can request a database connection from the data source.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A data source is a Java object managed by WebLogic Server and used by application code to obtain a database connection. Retrieving a database connection from a data source is better than getting a connection directly from the database for two reasons:

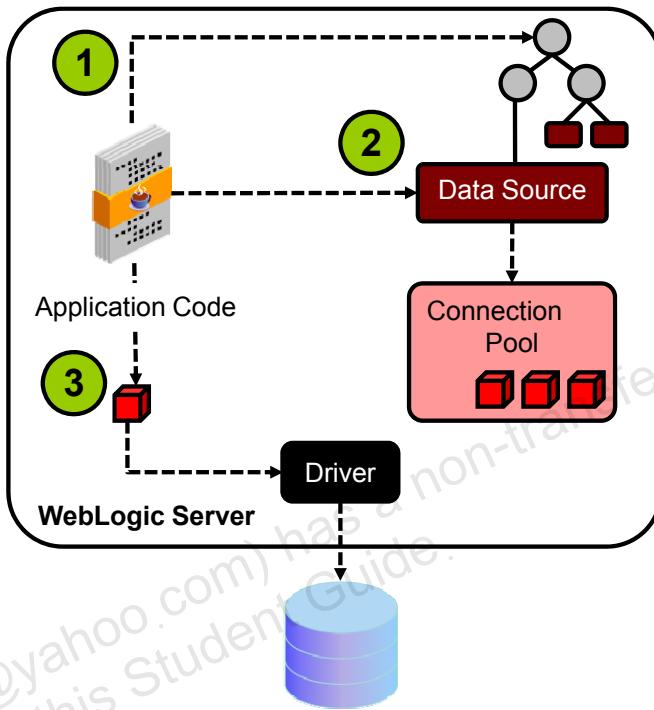
- Connections in a data source's connection pool have already been created. Therefore, the application does not have to wait for connection creation.
- All database-specific information moves out of the application code and into the WebLogic Server configuration, making the code more portable and robust.

Data sources can be created by using one of the WebLogic Server administration tools. A data source is configured with a connection pool that will contain connections to a particular database. It is also targeted to one or more instances of WebLogic Server.

For an application to use one of the connections in a data source's connection pool, first the application looks up the data source in the server's resource tree. The API used is the Java Naming and Directory Interface (JNDI). Once the data source is retrieved, the application asks it for a database connection. The data source gives the application one of the connections not currently being used, from its pool of connections. The application uses that connection to access the database. When the application is finished with the connection, it closes it. But rather than close, the connection is returned to the connection pool for some other application to use.

JDBC Data Source

1. An application looks up the data source in a server's resource tree by using the JNDI API.
2. It asks the data source for a connection.
3. It uses the connection (which uses a driver) to access the database.
4. When finished, it closes the connection (which returns it to the pool).



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A data source is a Java object managed by WebLogic Server and used by application code to obtain a database connection. Retrieving a database connection from a data source is better than getting a connection directly from the database for two reasons:

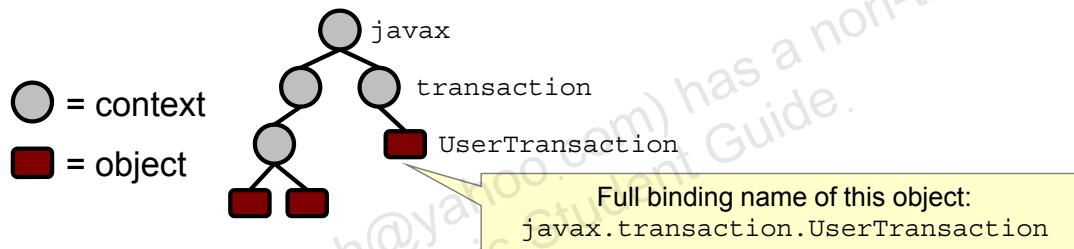
- Connections in a data source's connection pool have already been created. Therefore, the application does not have to wait for connection creation.
- All database-specific information moves out of the application code and into the WebLogic Server configuration, making the code more portable and robust.

Data sources can be created by using one of the WebLogic Server administration tools. A data source is configured with a connection pool that will contain connections to a particular database. It is also targeted to one or more instances of WebLogic Server.

For an application to use one of the connections in a data source's connection pool, first the application looks up the data source in the server's resource tree. The API used is the Java Naming and Directory Interface (JNDI). Once the data source is retrieved, the application asks it for a database connection. The data source gives the application one of the connections not currently being used, from its pool of connections. The application uses that connection to access the database. When the application is finished with the connection, it closes it. But rather than actually close it, the data source returns the connection to the connection pool for some other application to use.

Java Naming and Directory Interface (JNDI)

- An API for accessing directory or naming services.
- WebLogic Server keeps a tree of its resources in memory that can be accessed by using JNDI.
- JNDI terms:
 - **Context:** A “container” node in the JNDI tree that can contain other contexts and objects
 - **Object:** A leaf node in the JNDI tree. Resources are objects.
 - **Binding:** Associating an object with a name and a context



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

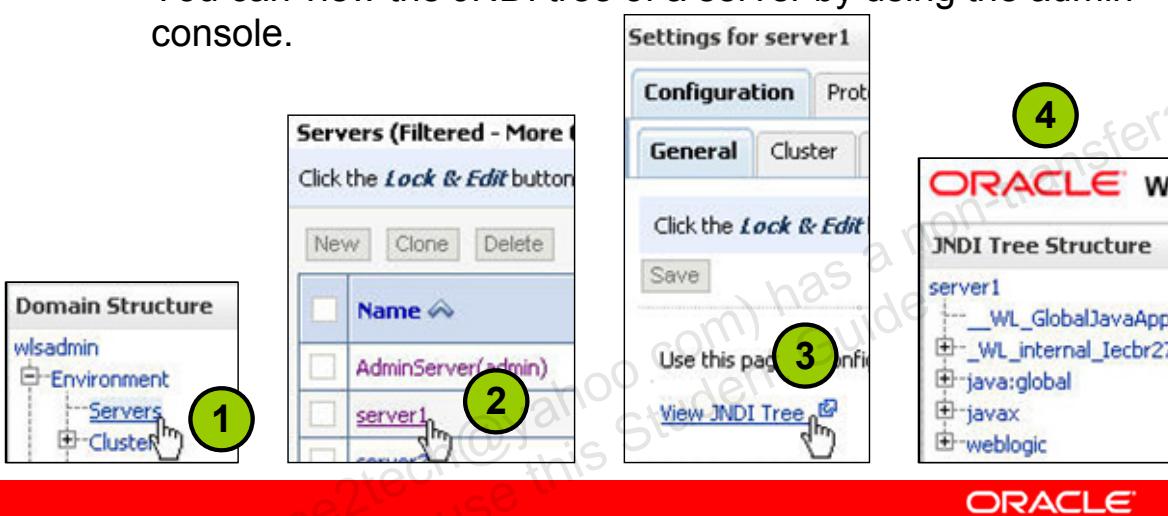
JNDI is an API that provides naming and directory functionality to Java applications independent of any specific naming or directory service implementation (such as the Lightweight Directory Access Protocol [LDAP], Domain Name Server [DNS], or a file system). WebLogic Server, upon starting, creates a tree of its resources in memory that can be accessed by using JNDI.

The terms associated with a JNDI tree include the following:

- **Context:** A “container” node in the JNDI tree. It can contain other contexts and objects.
- **Object:** A leaf in the JNDI tree, which is bound to a context. (It cannot contain other objects or contexts.) Resources are objects in the tree.
- **Root context:** The context at the top in the JNDI tree
- **Initial context:** A context that is chosen as the starting point for all future operations. It does not have to be the root context.

JNDI Duties of an Administrator

- Report to developers the JNDI names of resources you create, so they can look them up in their code.
- Check whether resources are in the JNDI tree of an instance of WebLogic Server.
 - You can view the JNDI tree of a server by using the admin console.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To view a server's JNDI tree by using the administration console:

1. In the Domain Structure, expand **Environment** and click **Servers**.
2. In the Servers table, select the server of interest. (In this example, **server1** is selected.)
3. Under **Configuration > General**, click the **View JNDI Tree** link.
4. The server's JNDI tree is displayed in a new web browser window (or tab). Use the plus signs to expand contexts. When you click an object in the tree, details about that object are displayed to the right.

Deployment of a Data Source

- The configuration of a WebLogic Server data source is stored in an XML document called a *JDBC module*:
 - A WLS-specific extension of Java EE modules
 - Targeted to certain instances of WebLogic Server
- A JDBC module is either a *system module* or an *application module*.
 - A JDBC system module is created by using the administration console or WLST. It resides in the domain configuration here:
domainname/config/jdbc/dsname-jdbc.xml
 - *domainname*: The main domain directory
 - *dsname*: The name of the data source
 - A JDBC application module can be deployed either within a Java EE application or stand-alone.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

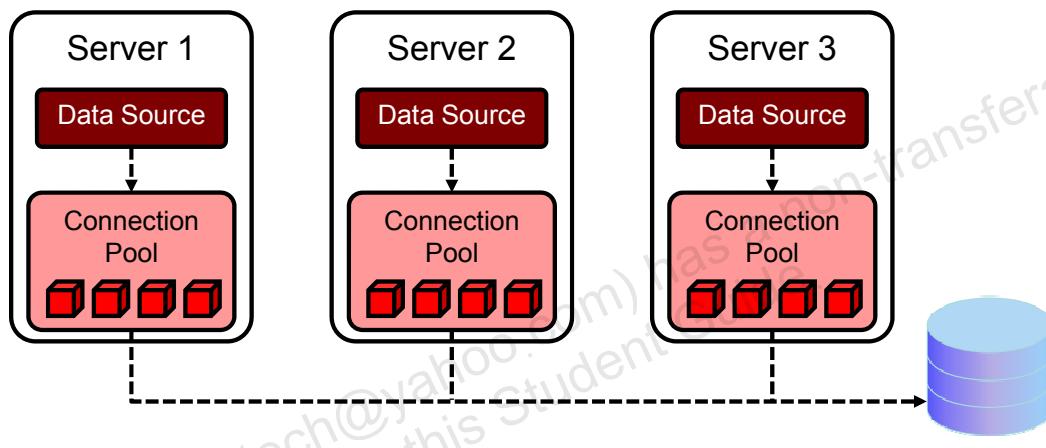
The connections to a database for the connection pool of a data source are created when the data source is first created (or deployed) to a targeted instance of WebLogic Server. Subsequently, connections are created when that server is restarted.

A JDBC application module can be part of a Java enterprise application. In that way, the data source is deployed along with the application code that needs it. Also, that data source is available only to the containing application.

Finally, a JDBC application module can be deployed stand-alone.

Targeting of a Data Source

- A data source is targeted to one or more instances of WebLogic Server.
 - Each targeted server has its own instance of that data source.
 - Each data source has its own connection pool.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

A JDBC module, which contains a data source, is targeted to one or more instances of WebLogic Server. When a server is targeted, it will have its own instance of that data source.

Types of Data Sources

- Generic data source
 - Is a standard data source with a connection pool tied to a particular database
- Multi data source
 - Is a collection of generic data sources tied to multiple database servers
 - Is looked up and used by applications like a generic data source
 - Transparently provides load balancing or failover across its member generic data sources
- GridLink data source
 - Provides connectivity between WebLogic Server and the Oracle Real Application Clusters (RAC) Database



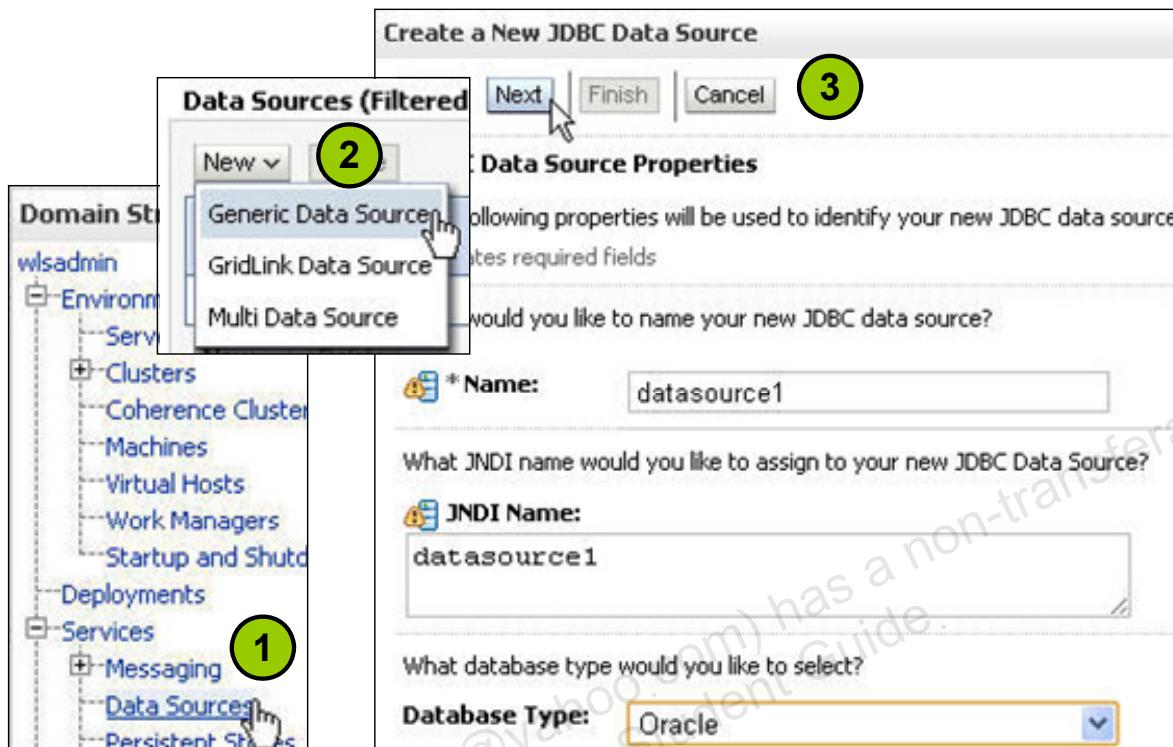
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A multi data source is an abstraction around a collection of generic data sources that provides load balancing or failover among that data source collection. Applications look up a multi data source in a server's JNDI tree just like they do a generic data source, and then ask the multi data source for a database connection. The multi data source determines which data source to use to satisfy the request depending on the algorithm selected in the multi data source configuration: load balancing or failover.

All generic data sources used by a multi data source to satisfy connection requests must be deployed on the same servers and clusters as the multi data source. A multi data source uses a data source deployed on the same server to satisfy connection requests. Multi data sources do not route connection requests to other servers in a cluster or in a domain. To deploy a multi data source to a cluster or server, you select the server or cluster as a deployment target.

A single GridLink data source provides connectivity between WebLogic Server and an Oracle Real Application Clusters (RAC) database. An Oracle RAC database allows database instances on different machines to access the same database. This provides scalability, load balancing, and failover. More information on Oracle RAC and GridLink data sources is found later in this lesson.

Creating a Generic Data Source



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

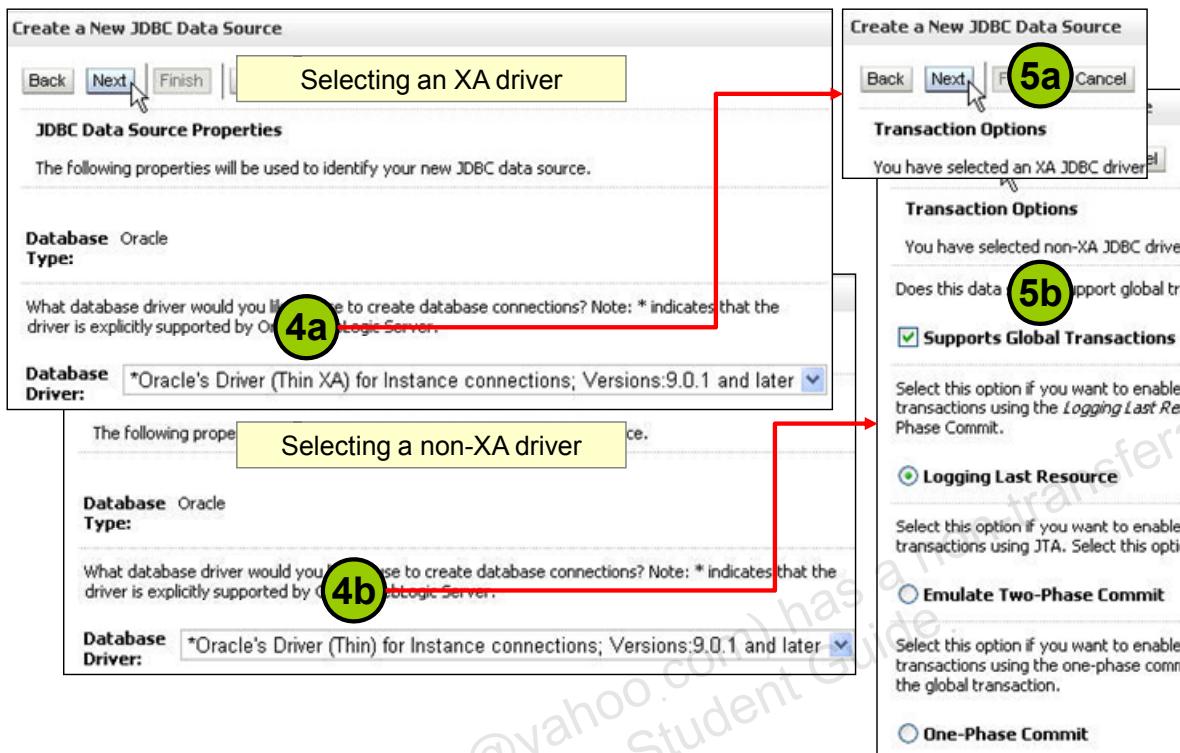
Make sure that the JDBC drivers that you want to use to create database connections are installed on all the servers on which you want to configure database connectivity and they are in the servers' CLASSPATH. Some JDBC drivers are installed with Oracle WebLogic Server and are placed in the CLASSPATH by default. To create a generic data source after clicking **Lock & Edit** in the Change Center, perform the following tasks:

1. In the Domain Structure tree, expand **Services** and then select **Data Sources**.
2. Above (or below) the table of data sources, click the **New** drop-down list and select **Generic Data Source**.
3. On the first page of the data source creation wizard, enter or select the following information and then click **Next**:
 - **Name:** The configuration name for this data source
 - **JNDI Name:** The JNDI "binding name" for this data source. Applications look up the data source on the server's JNDI tree by this name. The name can include contexts by placing dots in the string. For example, if the name is `jdbc.ds.myds`, the data source can be found in the context `jdbc`, subcontext `ds`, with the name `myds`. Note that the name and JNDI name can be different.

- **Database Type:** The database type. The database types in this drop-down list include Adabas, DB2, Derby, EnterpriseDB, Informix, Ingres, MS SQL Server, MySQL, Oracle, Sybase, and more. If your database is not listed, select **Other**.

Tony Albrecht (base2tech@yahoo.com) has a non-transferable
license to use this Student Guide.

Creating a Generic Data Source



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

4. Use the drop-down list to select a driver, and then click **Next**.
 - a. This example shows selecting an XA driver.
 - b. This example shows selecting a non-XA driver.
5. On the next wizard page, select the transaction options, and then click **Next**.
 - a. If you selected an XA driver, there are no transaction options because the data source supports global transactions by using the two phase commit protocol.
 - b. If you selected a non-XA driver, you can still allow the data source to participate in global transactions by selecting **Supports Global Transactions**. If selected, you then choose how the data source will participate, even though it is not XA-compliant. The choices are:
 - **Logging Last Resource (Recommended)**: This resource is processed last. If it succeeds, the other resources are told to commit, if it fails, they are told to roll back.
 - **Emulate Two-Phase Commit**: This resource always returns “ready” during phase one of 2PC. This can possibly cause heuristic conditions.
 - **One-Phase Commit**: Only this resource can participate in the global transaction.

Non-XA Driver Transaction Options

If a non-XA driver is selected, you can still choose for the data source to support global transactions. If you enable this option, you must specify *how* the data source will participate in those transactions:

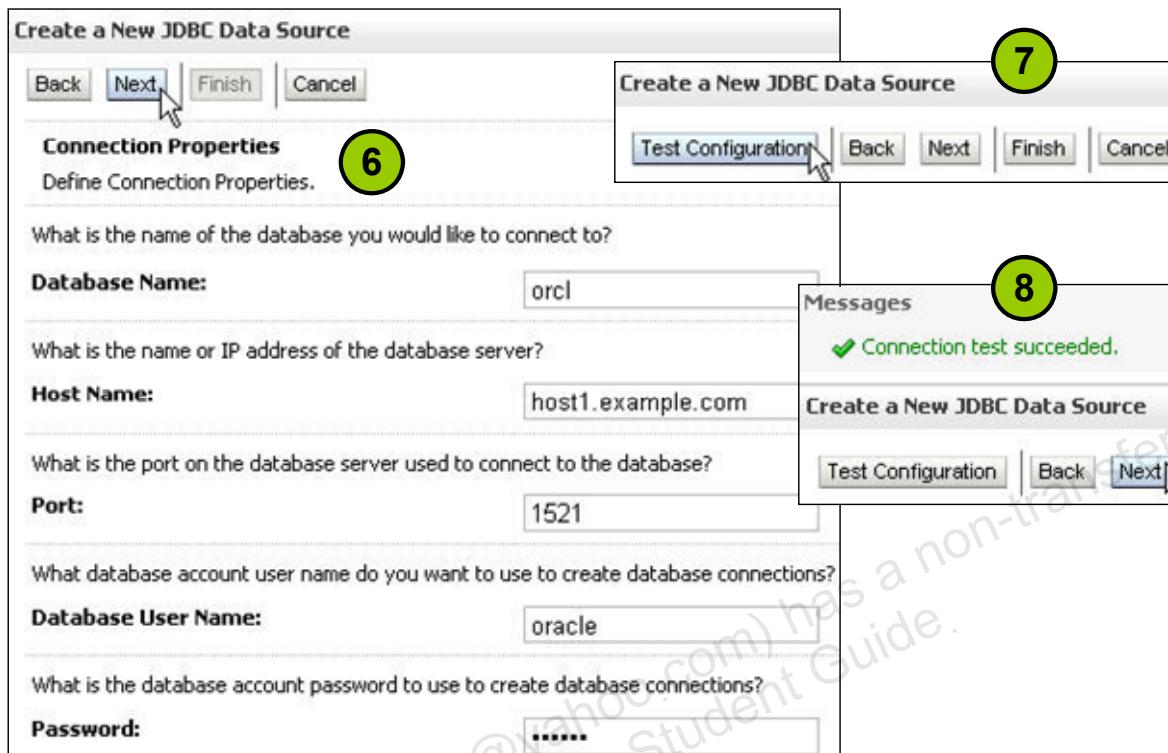
- Logging Last Resource: This resource is processed last. If it succeeds, the other resources are told to commit; if it fails, they are told to roll back.
- Emulate Two-Phase Commit: This resource always returns “ready” during phase one of 2PC. This can possibly cause heuristic conditions.
- One-Phase Commit: Only this resource can participate in the global transaction.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

- **Logging Last Resource (LLR):** With this option, the transaction branch in which the data source connection is used is processed as the last resource in the transaction and is processed as a one-phase commit operation. The result of the operation is written in a log file on the resource itself, and the result determines the success or failure of the prepare phase of the transaction. This option offers some performance benefits with greater data safety than Emulate Two-Phase Commit. There will be more information about LLR in the lesson titled “Transactions.”
- **Emulate Two-Phase Commit:** With this option, the transaction branch in which the connection is used always returns success (or “ready”) for the prepare phase of the transaction. This option offers performance benefits, but also has risks to data in some failure conditions.
- **One-Phase Commit:** Connections from the data source participate in global transactions using one-phase commit transaction processing. With this option, no other resources can participate in the global transaction.

Creating a Generic Data Source



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

6. Enter values for the following properties and click **Next**:

- **Database Name:** The database name (name requirements vary by DBMS)
- **Host Name:** The DNS name or IP address of the server that hosts the database
- **Port:** The port on which the database server listens for connections requests
- **Database User Name:** The database user account name that you want to use for connecting to the database
- **Password:** The password for the database user account
- **Confirm Password (not shown):** The password again
- **Oracle.jdbc.DRCPConnectionClass (not shown):** Database Resident Connection Pooling (DRCP) provides a connection pool at the database server. It is designed to work with applications that acquire a database connection, use it for a short while, and then release it. DRCP complements middle-tier connection pools. DRCP enables a significant reduction in key database resources that are required to support a large number of client connections. The DRCP connection class is a user-chosen string to distinguish this application from any others that also use DRCP. If this string is not set, the DRCP pool will not be able to be shared effectively.

7. Review the connection parameters (Driver Class Name, URL, Database User Name, Properties, System Properties, Test Table Name) and click **Test Configuration**. The test attempts to create a connection from the administration server to the database.
8. After the connection test, the result is displayed at the top of the page. If the test is unsuccessful, you should click **Back** and correct any configuration errors. If the test is successful, click **Next**.

Tony Albrecht (base2tech@yahoo.com) has a non-transferable
license to use this Student Guide.

Creating a Generic Data Source

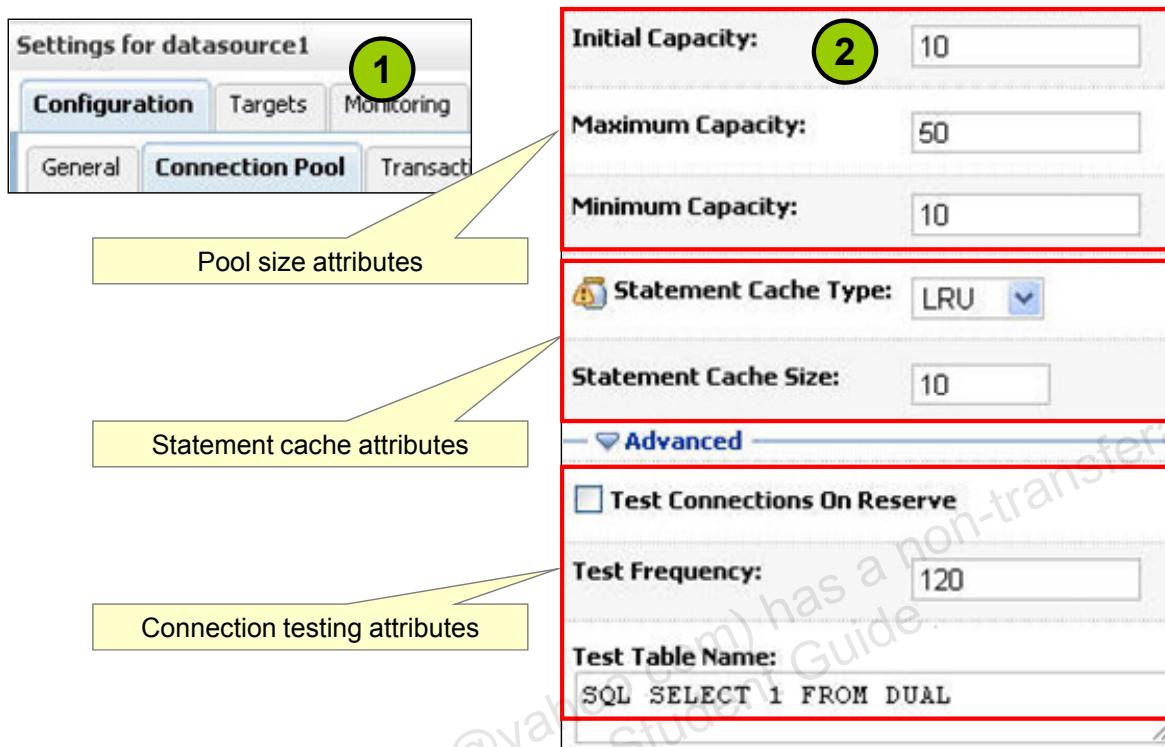


ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

9. Select the servers or clusters to which the data source should be deployed and then click **Finish**. If no servers are targeted, the data source is created, but not deployed. You will need to target it later. As usual, to confirm the changes, in the Change Center, click **Activate Changes**.

Connection Pool Configuration



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

After creating your initial data source configuration in the console, you can tune its connection pool settings (in the Change Center click **Lock & Edit** if you have not already done so):

1. In the Domain Structure tree, expand **Services** and select **Data Sources**. Click on your data source name in the table. Then click the **Configuration > Connection Pool** tab.
2. Enter values for any connection pool attributes you want to change (some are found under the Advanced options section), including:
 - **Initial Capacity:** The number of connections to create when deploying the connection pool. If unable to create this number of connections, creation of the data source will fail.
 - **Maximum Capacity:** The maximum number of connections that this connection pool can contain
 - **Minimum Capacity:** The minimum number of connections that this connection pool can contain after it is initialized. It is used only for connection pool shrinking calculations.

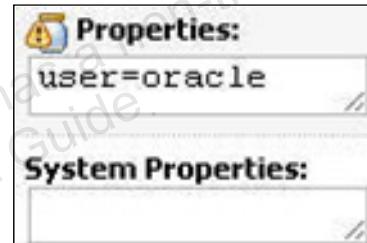
- **Statement Cache Type:** The algorithm used for maintaining the prepared statements stored in the statement cache. The options are LRU and Fixed. LRU: When a new prepared or callable statement is used, the least recently used statement is replaced in the cache. Fixed: The first fixed number of prepared and callable statements is cached.
- **Statement Cache Size:** The number of prepared and callable statements stored in the cache
- **Test Connections On Reserve:** WebLogic Server tests a connection before giving it to a client, which adds a small delay in serving the client requesting a connection, but ensures that the connection is viable.
- **Test Frequency:** The number of seconds between tests of unused connections. Connections that fail the test are closed and reopened to reestablish a valid connection. If the test fails again, the connection is closed.
- **Test Table Name:** The name of the database table to use when testing connections. This name is required when you specify a Test Frequency or enable Test Reserved Connections. The default SQL used in testing is: `select count(*) from tablename`. If the Test Table Name field begins with `SQL`, the rest of the string is taken as a literal SQL statement to use in testing connections.

Note: There will be more information on these connection pool settings later in this lesson.

Connection Properties

Connection properties are:

- Found in the configuration of the data source under **Configuration > Connection Pool**
- Property/value pairs entered in:
 - **Properties** as: *property=value*
 - **System Properties** as: *property=systemProperty*
- Used to configure JDBC connections
- Passed to the driver during connection setup



ORACLE

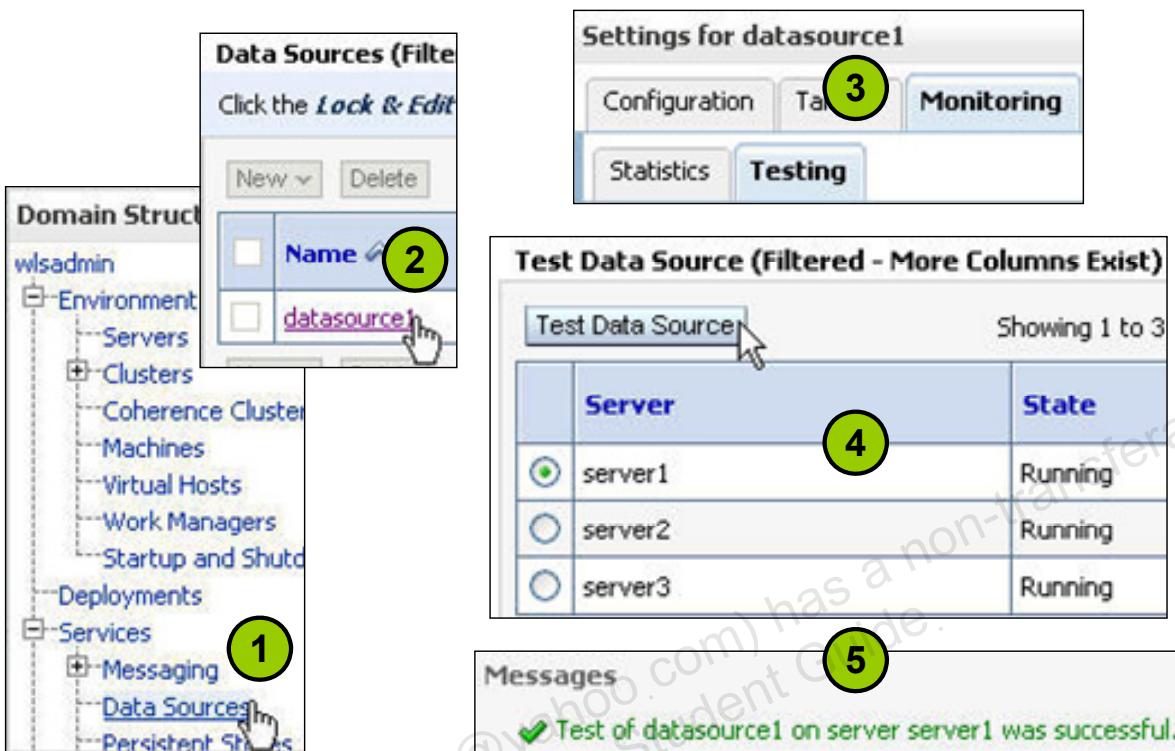
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can enable driver-level features by adding the property and its value to the **Properties** attribute of the connection pool of a data source.

Data sources also support setting driver properties by using the value of system properties. Add a system property to the **System Properties** attribute of the connection pool of a data source. The value of the property is derived at runtime from the named system property.

The available connection properties depend upon the database driver. For a complete list, see your driver documentation.

Testing a Generic Data Source



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

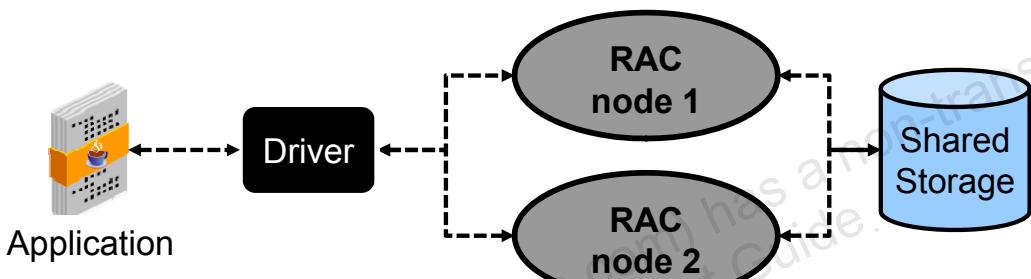
When you test a data source, the selected server reserves a connection from the data source and then releases it, thereby testing the connection pool. If the Test Connections on Reserve attribute of the connection pool is enabled, the acquired connection is also tested as part of the reserve operation. In that case you must have specified a table name or SQL query in the Test Table Name attribute of the data source's connection pool (as well as selected the Test Connections on Reserve attribute).

To test a generic data source, perform the following tasks:

1. In the Domain Structure tree, expand **Services**, and then select **Data Sources**.
2. Select the data source from the table of data sources.
3. Click the **Monitoring > Testing** tab.
4. Select the target on which to test the data source, and then click the **Test Data Source** button.
5. A message displays above the tabs to indicate whether the test was successful.

Oracle Real Application Clusters: Overview

- Oracle Real Application Clusters (RAC):
 - Supports multiple Oracle database servers for greater scalability and reliability
 - Relies on database servers having access to a shared and highly available storage device



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

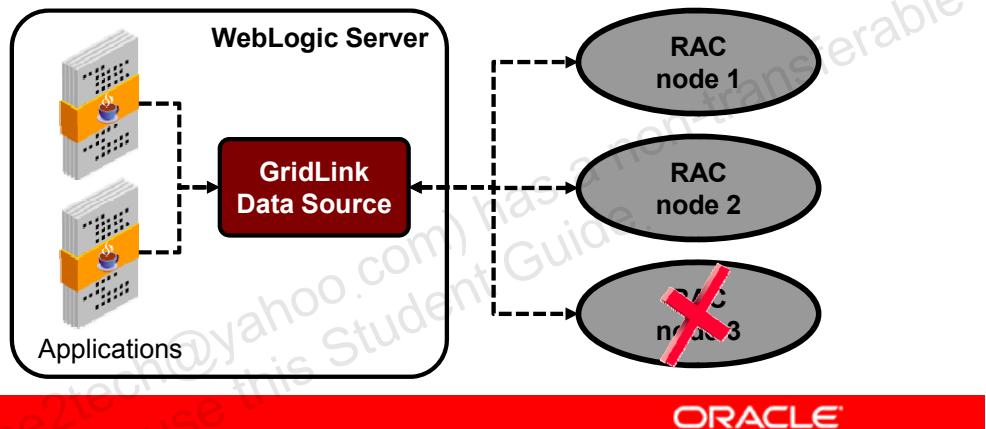
Oracle Real Application Clusters (RAC) is software that enables users on multiple machines to access a single database with increased reliability. RAC is made up of two or more Oracle database instances running on two or more clustered machines that access a shared storage device via cluster technology. To support this architecture, the machines that host the database instances are linked by a high-speed interconnect to form the cluster. This interconnect is a physical network used as a means of communication between the nodes of the cluster. Cluster functionality is provided by the operating system or compatible third-party clustering software.

Because every RAC node in the cluster has equal access and authority, the loss of a node may impact performance, but does not result in down time.

GridLink Data Source for RAC

WebLogic Server's *GridLink* data source is "RAC-aware." It:

- Performs intelligent load balancing based on the current RAC workload
 - Implements RAC's Fast Connection Failover (FCF) pattern
 - Ensures that all database operations within a global transaction are routed to the same RAC node ("XA affinity")



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A single GridLink data source provides connectivity between WebLogic Server and an Oracle database service that has been targeted to an Oracle RAC cluster. This type of data source automatically adjusts the distribution of work based on the current performance metrics reported by each RAC node, such as CPU usage, availability, and response time. If this capability is disabled, GridLink data sources instead use a round-robin, load-balancing algorithm to allocate connections to RAC nodes.

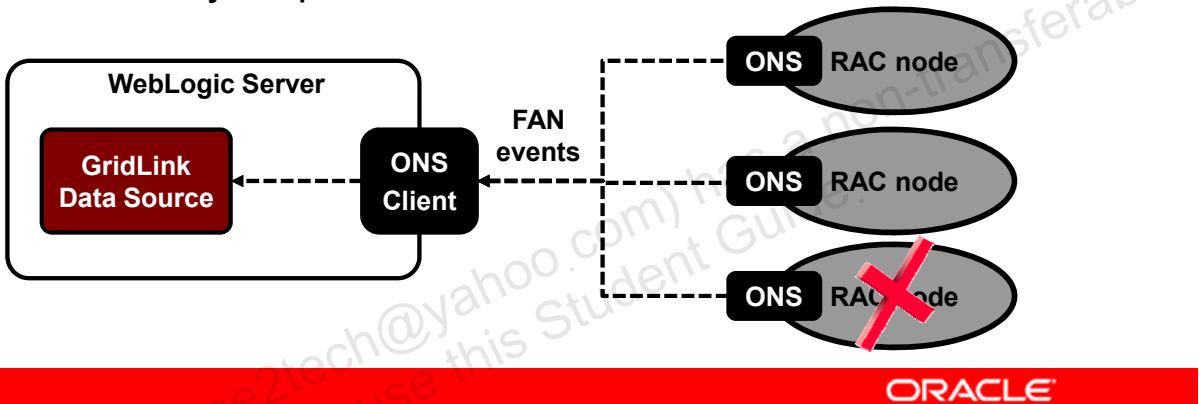
A GridLink data source implements Oracle's Fast Connection Failover (FCF) pattern, which:

- Provides rapid failure detection
 - Aborts and removes invalid connections from the connection pool
 - Performs graceful shutdown for planned and unplanned Oracle RAC node outages
 - Adapts to changes in topology, such as adding or removing a node
 - Distributes run-time work requests to all active Oracle RAC instances, including those rejoining a cluster

XA affinity ensures all the database operations performed on a RAC cluster within a global transaction are directed to the same RAC instance. This increases performance and also helps ensure data integrity after a failure.

GridLink, FCF, and ONS

- Typically, when a system goes down, applications must wait for the network to time out (perhaps minutes).
- FCF pattern: The Oracle Notification Service (ONS) delivers Fast Application Notification (FAN) events about RAC availability and workload to registered subscribers.
- The GridLink data source can subscribe to ONS and immediately respond to nodes that are added or removed.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

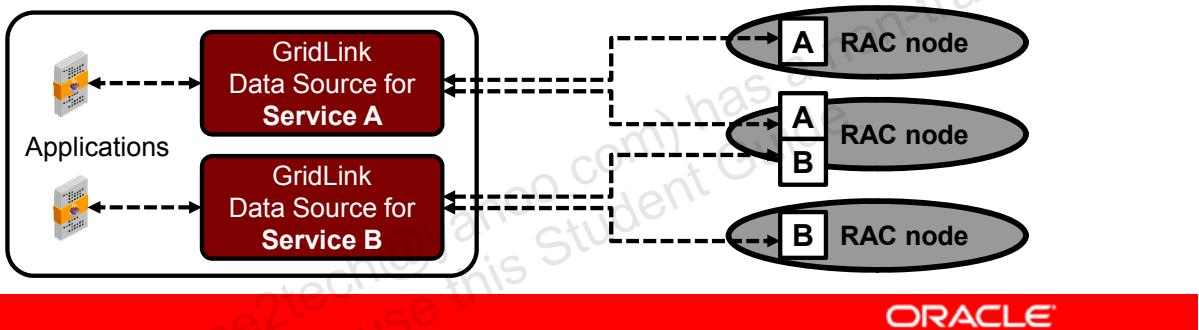
When a database outage occurs, two problems confront applications: errors and hangs. Applications encounter errors because the work they were doing (queries, transactions) is interrupted. Even worse, those errors may take some time to arrive. When the database outage occurs, the client application may have to wait for network timeouts (which may be minutes) before being informed of the outage. This can cause the application to hang for some time, leading to user frustration. Oracle Database provides several features that Java applications such as WebLogic Server can use to increase failure responsiveness and to help mask errors from the end user.

The Fast Connection Failover (FCF) pattern consists of the Oracle Notification Service (ONS) using a simple publish/subscribe method to produce and deliver Fast Application Notification (FAN) event messages. ONS daemon processes are automatically created during the RAC installation process and are configured to run on each node. ONS is a component of the Oracle Process Manager and Notification (OPMN) server, which manages Oracle Fusion Middleware system components.

A GridLink data source uses ONS to adaptively respond to state changes in an Oracle RAC instance. This ensures that the connection pool in the data source contains valid connections (including reserved connections) without the need to poll and test connections. The data source also distributes connections to Oracle RAC instances based on these FAN events.

GridLink and Services

- Oracle Database supports services that:
 - Act as gateways to a subset of RAC nodes
 - Automatically start on another node if the current one fails
 - Are accessed by applications by using the service name
 - Enable you to control and prioritize the available capacity on your grid for different clients
- FCF requires the use of database services.
- Create a separate GridLink data source for each service.



ORACLE

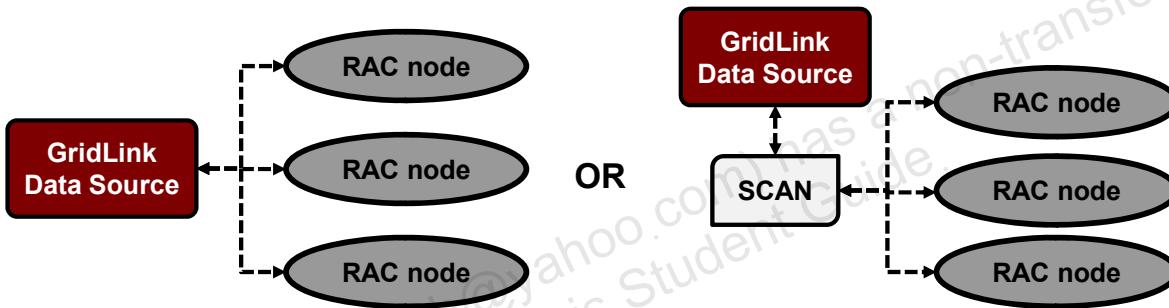
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Services are entities defined for a RAC database that enable its workload to be managed. Services also decouple any hardwired mapping between a connection request and a RAC instance. Each service represents a workload with common attributes, thresholds, and priorities. For example, online users can be one service, batch processing can be another service, and reporting can be a third service type. A service can span one or more database instances and a single instance can support multiple services. The use of FAN events and run-time connection load balancing *requires* the configuration of services.

Services hide the complexity of a cluster from the database client by providing a single logical entity for managing work. Applications or middleware such as WebLogic Server specify a service by supplying its name when making the initial connection. On WebLogic Server in particular, you scale the number of GridLink data sources as the number of services increases in the database, independent of the number of nodes in the cluster.

GridLink and Single Client Access Name (SCAN)

- Starting with Oracle RAC version 11gR2, a SCAN service is provided that:
 - Accepts a database cluster alias
 - Returns the locations of cluster members
 - Can run independently or integrate with your corporate DNS
- GridLink data sources can either use a list of database node locations or a single SCAN address.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

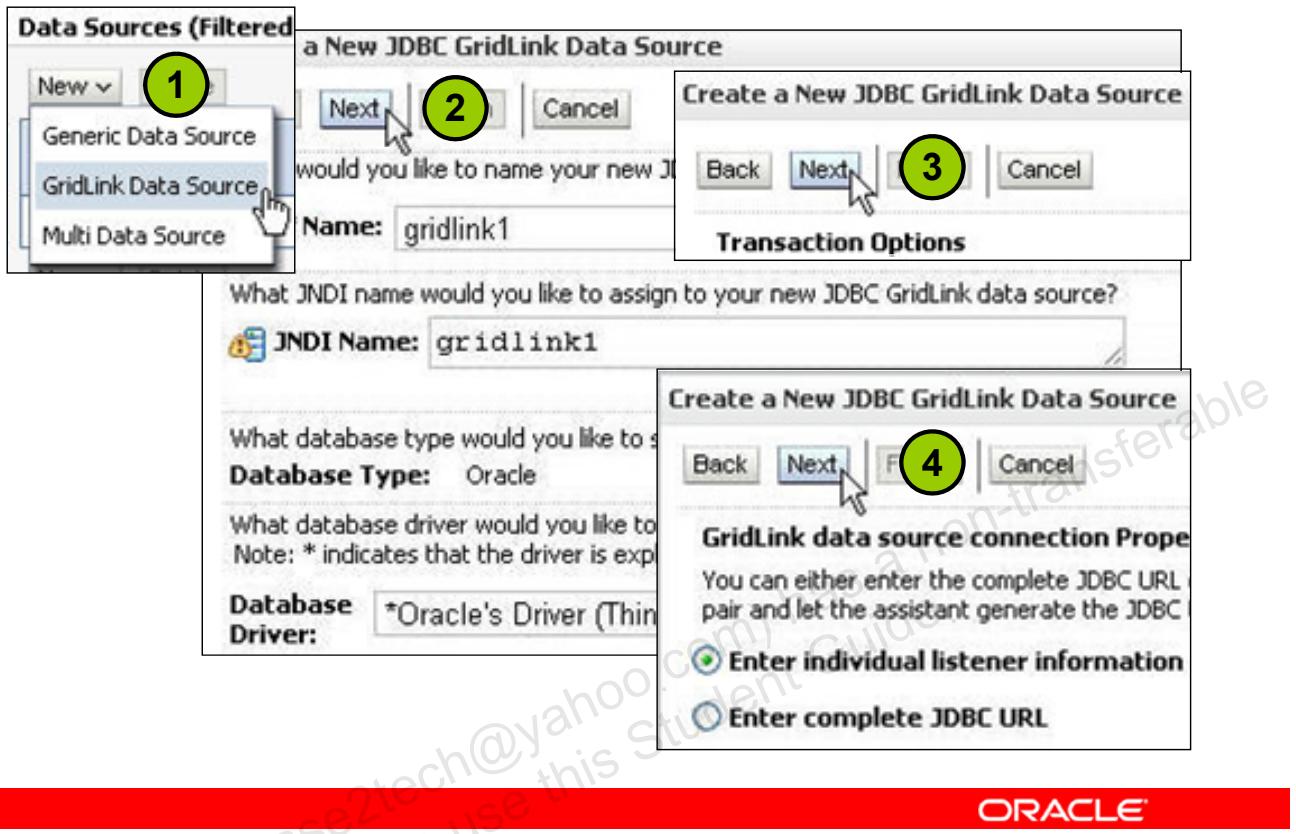
Single Client Access Name (SCAN) is a new feature starting with RAC 11g Release 2 that provides a single name for clients to access databases running in a cluster. The benefit is that the client's connection information does not have to change if you add or remove nodes in the cluster. Having a single name to access the cluster allows clients to access any database running in the cluster, independently of which server or servers in the cluster are active. SCAN provides load balancing and failover for client connections to the database. The SCAN works as a cluster alias for databases in the cluster.

During the installation of the Oracle Grid Infrastructure, you are prompted to provide a SCAN name. There are two options for defining the SCAN:

- Use an existing Domain Name Service (DNS) implementation
- Use the Oracle Grid Naming Service (GNS)

If you choose the DNS option, you must ask your network administrator to create a single name that resolves to three IP addresses by using a round-robin algorithm. Three IP addresses are recommended for the SCAN service for high-availability reasons, regardless of the number of servers in the cluster. The IP addresses must be on the same subnet as your public network in the cluster.

Creating a GridLink Data Source



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

1. As always, when making a change to the configuration in the administration console, you first click **Lock & Edit** in the Change Center. Then, in the Domain Structure, select **Services > Data Sources**. Then click **New > GridLink Data Source**.
2. Supply a Name, JNDI Name, and select the Database Driver. Then click **Next**. Notice the drivers are special GridLink drivers. Also notice that the Database Type is Oracle and cannot be changed.
3. As with generic data sources, on the next wizard page you select transaction options and then click **Next**. If you selected an XA driver, there are no transaction options because the data source supports global transactions. If you selected a non-XA driver, you can allow the data source to participate in global transactions by selecting **Supports Global Transactions** and then choose how the data source will participate.
4. Select **Enter individual listener information** and click **Next**. (You can instead choose to enter the complete JDBC URL manually, but it is easier to let the data source wizard generate it for you.)

Creating a GridLink Data Source

What is the service name of the database you would like to connect to?

Service Name: sales 5

Enter host and port of each listener separated by colon and click the add button.
In the case of a RAC DB listener, specify the SCAN address.

Host and Port: node2.example.com:1521 6

node1.example.com:1521

Up ▲ Down ▼ Add Remove

What database account user name do you want to use to create database connections?

Database User Name: oracle 7

What is the database account password to use to create database connections?

Password: 8

Create a New GridLink Data Source

Back Next Finish Cancel

ORACLE

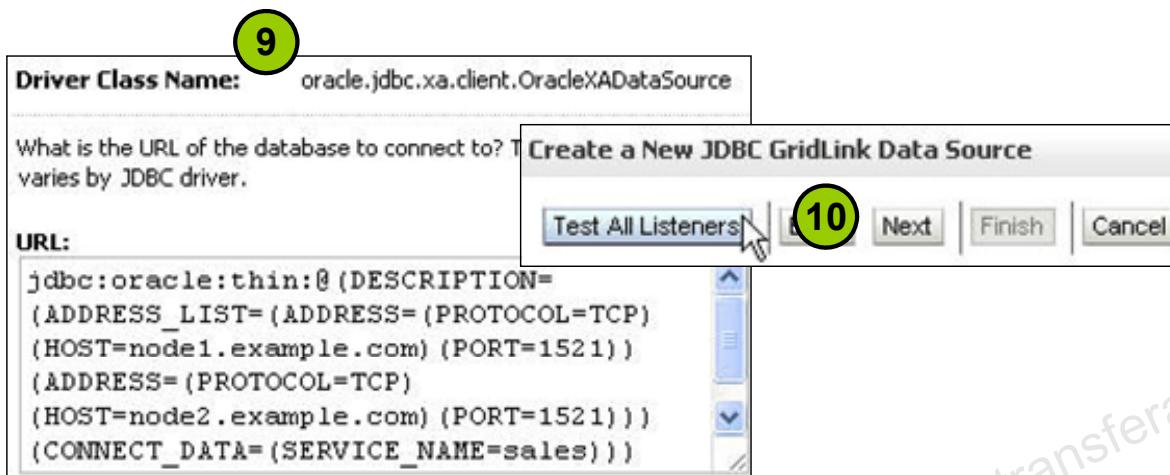
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

5. Enter the Service Name.
6. Use the **Add** button multiple times to specify a list of initial database listeners to connect to, or use the **Add** button once to specify a Single Client Access Name (SCAN) address. A GridLink data source containing SCAN addresses does not need to change if you add or remove Oracle RAC nodes.
7. Enter values for the following properties:
 - **Database User Name:** The database user account name that you want to use for connecting to the database
 - **Password:** The password for the database user account
 - **Confirm Password (not shown):** The password again

- **Protocol (not shown):** Leave the default value of TCP (Transmission Control Protocol) in this field, unless you are using Exalogic hardware. If you are using Exalogic, replace TCP with SDP (Sockets Direct Protocol). This protocol is used in InfiniBand high-speed networks. For example, InfiniBand can be used to connect an Exalogic machine to an Exadata machine. SDP is characterized by short-distance, high-performance communications between multiple server systems. Simply connecting the machines with InfiniBand cables is not sufficient, however, because WLS and RAC will still communicate by using standard TCP unless the value of SDP is entered here. Also note that in order to use SDP, you will need to add a command-line argument when starting WebLogic Server: -
`Djava.net.preferIPv4Stack=true`. One way to do that is to add the argument to the start scripts for WebLogic Server, like `startWebLogic.sh`.
- **Oracle.jdbc.DRCPConnectionClass (not shown):** See the explanation of the Database Resident Connection Pooling (DRCP) class in the “Creating a Generic Data Source” section.

8. Click **Next**.

Creating a GridLink Data Source

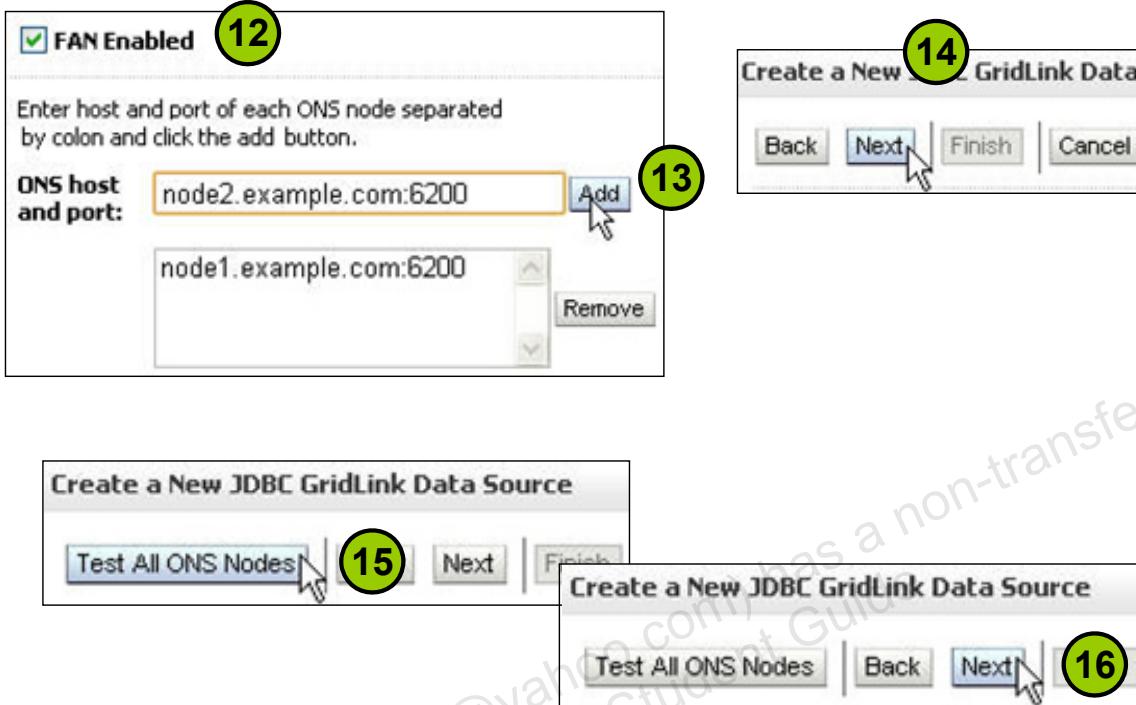


ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

9. Review the properties and values displayed. Notice that the **Driver Class Name** cannot be changed. The **URL** has been generated from what was entered earlier, but could be modified. The **Database User Name** (not shown) and **Password** (not shown) can be modified. As with generic data sources, driver-level features can be enabled by adding properties and values to the **Properties** (not shown) and **System Properties** (not shown) text areas. The **Test Table Name** (not shown) can be updated.
10. Optionally, test whether the URL is correct by clicking the **Test All Listeners** button. Or, you can test one listener at a time as each one is given a **Test Listener** button (not shown).
11. When you have confirmed the properties (and, if you tested, the test was successful), click **Next**.

Creating a GridLink Data Source

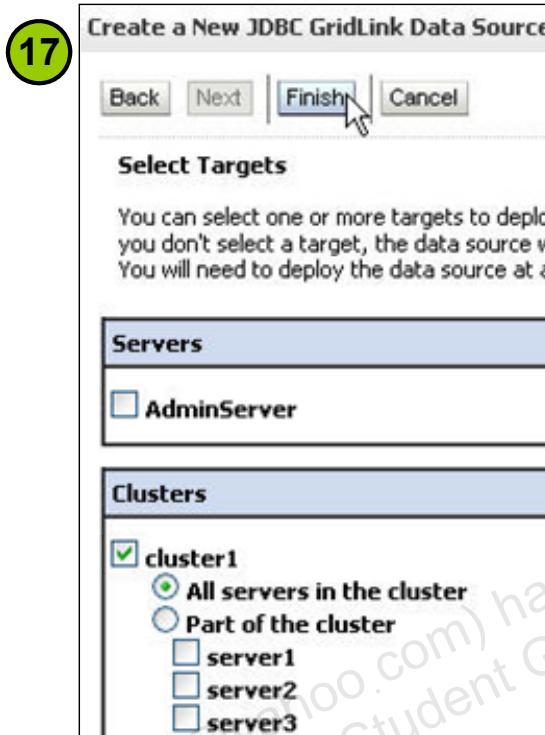


ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

12. Confirm that **FAN Enabled** is selected if you want WebLogic Server to subscribe to Oracle FAN events.
13. Use the **Add** button multiple times to specify a list of initial ONS daemon processes to connect to. You can also optionally configure the ONS client on WebLogic Server to receive events over SSL by configuring an **ONS Wallet File Directory** (not shown) and **ONS Wallet Password** (not shown). Oracle Wallet manages SSL certificates.
14. Click **Next**.
15. Optionally, test whether the ONS hosts and ports are correct by clicking the **Test All ONS Nodes** button. Or, you can test one ONS node at a time as each one is given a **Test ONS Node** button (not shown).
16. Click **Next**. (If you tested, ensure that the test was successful.)

Creating a GridLink Data Source



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

17. Select the servers or clusters to which the data source should be deployed, and then click **Finish**. If no servers are targeted, the data source is created, but not deployed. You will need to target it later. As usual, to confirm the changes, in the Change Center, click **Activate Changes**.

Common Data Source Problems

- Data source configuration errors:
 - Invalid JDBC URL
 - When the data source is created or deployed it will fail. The underlying exception is `java.net.ConnectException`.
 - Invalid credentials
 - When the data source is created or deployed, it will also fail. Not because a network connection could not be established, but because the database rejects the connection attempt.
 - Wrong driver version in the CLASSPATH
 - Errors can be subtle and hard to catch if the version of the driver is not the one expected. Ensure that the drivers you want to use are in the CLASSPATH before the standard ones. For example, add this line in `startWebLogic.sh` after the CLASSPATH has been set:

```
CLASSPATH="/path/customdriver.jar:${CLASSPATH}"
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Note that the first two errors can be caught when the data source is created in the administration console by using the **Test Configuration** button.

Drivers are installed in the `<WEBLOGIC_HOME>/server/lib` directory by using `weblogic.jar`. The manifest file found in `weblogic.jar` lists driver JARs to be loaded when the JAR is loaded (when the server starts). Therefore, you do not need to add these JDBC drivers to your CLASSPATH. If you plan to use a third-party JDBC driver that is not installed with WebLogic Server, you must install the drivers, which includes updating your CLASSPATH with the path to the driver files. If you plan to use a different version of any of the drivers installed with WebLogic Server, you can replace the driver file in `<WEBLOGIC_HOME>/server/lib` with an updated version of the file or add the new file to the front of your CLASSPATH.

Common Data Source Problems

- Insufficient connections



- When the maximum capacity is reached, the next application to request a connection will wait Connection Reserve Timeout seconds for a free one.
 - If none are free at that time, an exception is returned.
- If this happens often, see whether more connections from the database are available, and increase the Maximum Capacity.
- If the applications can wait longer, perhaps increase the Connection Reserve Timeout seconds.

ORACLE

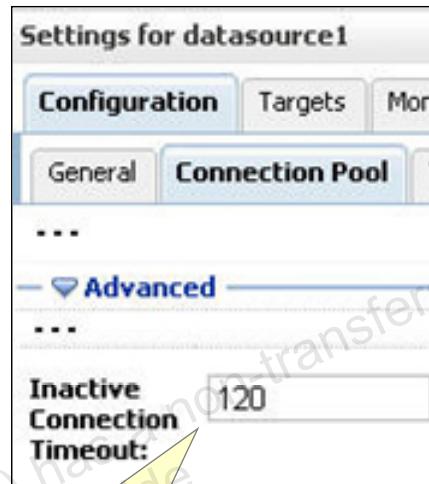
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

When an application requests a connection from a data source, if all connections in the data source are in use and if the data source has expanded to its maximum capacity, the application gets `ConnectionUnavailableSQLException`. To avoid this, you can configure the Connection Reserve Timeout value (in seconds) so that connection requests wait for a connection to become available. After the Connection Reserve Timeout has expired, if no connection becomes available, the request fails and the application gets `PoolLimitSQLException`.

If you set Connection Reserve Timeout to `-1`, a connection request will time out immediately if there is no connection available. If you set Connection Reserve Timeout to `0`, a connection request will wait indefinitely.

Common Data Source Problems

- Connection leaks
 - Poorly implemented applications can starve the data source of connections.
 - Hold a connection too long, even when not currently using it
 - Forget to close a connection (so it can return to the pool)
 - To provide a failsafe, WebLogic Server can automatically reclaim a connection after a certain amount of inactivity.



ORACLE

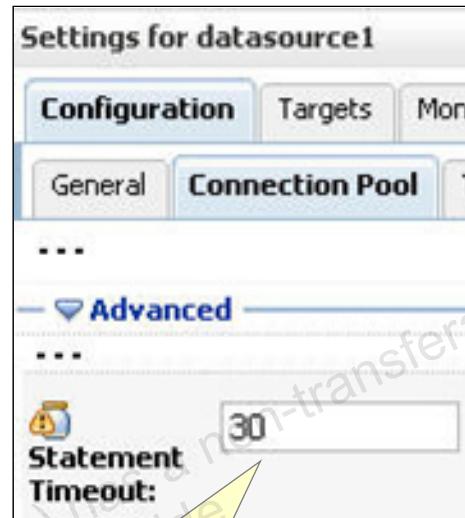
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A leaked connection is a connection that was not properly returned to the connection pool in the data source. To automatically recover leaked connections, you can specify a value for Inactive Connection Timeout. Find this attribute in the **Advanced** section of the **Connection Pool** tab. WebLogic Server forcibly returns a connection to the data source when there is no activity on a reserved connection for the number of seconds that you specify. When set to 0 (the default value), this feature is turned off.

Note that the actual timeout could exceed the configured value for Inactive Connection Timeout. The internal data source maintenance thread runs every five seconds. When it reaches the Inactive Connection Timeout (for example 30 seconds), it checks for inactive connections. To avoid timing out a connection that was reserved just before the current check or just after the previous check, the server gives an inactive connection a “second chance.” On the next check, if the connection is still inactive, the server times it out and forcibly returns it to the data source. On average, there could be a delay of 50% more than the configured value.

Common Data Source Problems

- Statement timeout
 - Most JDBC drivers support a maximum time limit for SQL statements that can be set:
 - By using the Statement Timeout attribute
 - Programmatically by the application
 - Increase this value if your applications require complex, long-running database operations.

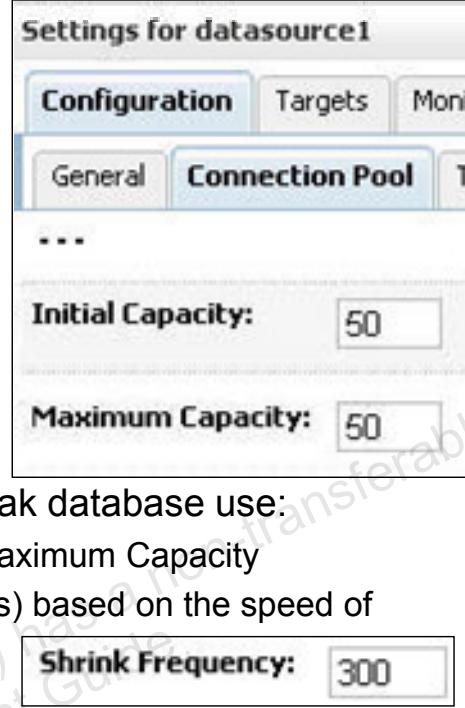


ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

With the Statement Timeout option, you can limit the amount of time that a statement is allowed to execute. When you set a value for Statement Timeout, WebLogic Server passes the time specified to the JDBC driver by calling the Statement's `setQueryTimeout()` method. WebLogic Server makes the call, and if the driver throws an exception (this is unsupported, for example), the Statement Timeout value is ignored. In some cases, the driver may silently not support the call or may document limited support. Oracle recommends that you check the driver documentation to verify the expected behavior. When Statement Timeout is set to -1, (the default) statements never time out. Find the Statement Timeout attribute in the **Advanced** section of the **Connection Pool** tab.

Basic Connection Pool Tuning

- Pool Capacity
 - Connection creation is expensive.
 - For applications with consistent, heavy database use:
 - Determine Maximum Capacity experimentally
 - Set Initial Capacity equal to Maximum Capacity
 - For applications with intermittent peak database use:
 - Use different values for Initial and Maximum Capacity
 - Tune the Shrink Frequency (seconds) based on the speed of the load changes
- 

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Each data source has a pool of database connections that are created when the data source is deployed or at server startup. Applications use a connection from the pool and then return it when they have finished using it. Connection pooling enhances performance by eliminating the costly task of creating new database connections for the application.

Creating a database connection is a relatively expensive process in any environment. Typically, a connection pool starts with a small number of connections. As demand for more connections grows, there may not be enough in the pool to satisfy the requests. WebLogic Server creates additional connections and adds them to the pool until the maximum pool size is reached.

One way to avoid connection-creation delays is to initialize all connections at server startup, rather than on-demand as applications need them. Set the Initial Capacity equal to the Maximum Capacity on the Connection Pool tab of your data source configuration. However, you still need to determine the optimal value for Maximum Capacity as part of your preproduction performance testing.

Basic Connection Pool Tuning

- Connection testing helps to ensure that connections are viable, but can degrade performance.
 - WebLogic Server tests a connection before giving it to an application when you enable Test Connections On Reserve.
 - WebLogic Server tests connections periodically based on the Test Frequency value.
 - If you use Test Frequency, do not test too often.
 - To help minimize the performance impact of testing, use Seconds to Trust an Idle Pool Connection
 - It is the number of seconds after a connection has been proven viable that WebLogic Server trusts it and skips testing it again.



ORACLE

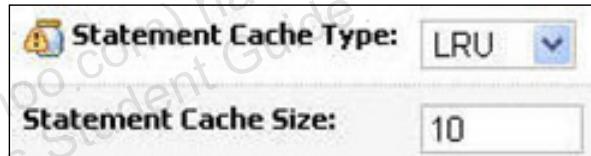
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To make sure that the database connections in a data source remain healthy, you can periodically test the connections. WebLogic Server provides automatic testing that you configure with attributes of the connection pool.

- **Test Connections on Reserve:** Enable this to test each connection before assigning it to a requesting application. This may add a slight delay to the request, but it guarantees that the connection is viable. If this is enabled, you must also set a **Test Table Name**. You can minimize the impact of testing reserved connections by tuning **Seconds to Trust an Idle Pool Connection**.
- **Test Frequency:** Use this attribute to specify the number of seconds between tests of unused connections. When unused connections are tested, WebLogic Server closes and replaces any connections that prove faulty. Setting this also requires you to set the **Test Table Name**.
- **Seconds to Trust an Idle Pool Connection:** Use this option to specify the number of seconds after a connection has been proven to be okay that WebLogic Server trusts the connection is still viable and skips the connection test, either before delivering it to an application or during the periodic connection testing process. This option is an optimization that minimizes the performance impact of connection testing, especially during heavy traffic.

Basic Connection Pool Tuning

- Statement caching
 - Prepared and callable statements can be cached to improve overall performance through reuse.
 - The Statement Cache Type determines the algorithm.
 - LRU: Once the cache is full, the least recently prepared/callable statement is replaced by a new one.
 - FIXED: Once the cache is full, *no* new prepared/callable statement is cached.
 - Determine the size of the cache through experimentation
 - Warning: Some databases maintain an open cursor for each open statement, so if the cache is too large, you could exceed the open cursor limit.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

If you use statement caching, each connection in a data source has its own individual cache of prepared and callable statements. However, you configure statement cache options per data source. That is, the statement cache for each connection in a data source's connection pool uses the statement cache options specified for the data source, but each connection caches its own statements.

Statement Cache Type: The algorithm used for maintaining the prepared statements stored in the statement cache. The options are LRU and Fixed.

- **LRU:** When a new prepared or callable statement is used, the least recently used statement is replaced in the cache.
- **Fixed:** The first fixed number of prepared and callable statements is cached. After the cache is full, new prepared or callable statements are no longer cached. With this statement cache algorithm, you can inadvertently cache statements that are rarely used.

Statement Cache Size: The number of prepared and callable statements stored in the cache. Caching increases performance, however, you must consider how your DBMS handles open prepared and callable statements. In many cases, the DBMS maintains a cursor for each open statement. This applies to the prepared and callable statements in the statement cache. If you cache too many statements, you may exceed the limit of open cursors on your database server. Setting the size of the statement cache to 0 turns off statement caching.

Quiz

Developers look up a server's data source by using which API?

- a. JDBC
- b. EJB
- c. SQL
- d. JNDI

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: d

Quiz

The connection pool of a generic data source has its Initial Capacity set to 2 and its Maximum Capacity set to 10. The data source is targeted to three managed servers. What is the most number of database connections that this can cause to be used at one time?

- a. 10
- b. 12
- c. 30
- d. 36



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: c

Summary

In this lesson, you should have learned how to configure:

- A generic data source
- A GridLink data source

Practice 7-1 Overview: Configuring a JDBC Data Source

This practice covers creating and configuring a generic data source.



Monitoring a Domain

8

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Configure and access WebLogic Server logs
- Enable WebLogic Server debugging output
- Monitor WebLogic Server health and performance
- Monitor JDBC data sources
- Access diagnostic charts in the Monitoring Dashboard

WebLogic Server Logs

The subsystems of an instance of WebLogic Server publish information about themselves into logs.

| Log | Description |
|---------------------|--|
| Server log | Used by server subsystems to record events |
| Standard out | Some server log messages are printed to standard out. |
| Domain log | Some server messages are gathered by the administration server for inclusion into the domain-wide log. |
| Access log | Used by the HTTP subsystem to track HTTP communication |
| Audit log | Tracks security requests. Requires configuring an Auditing provider (not configured by default). |



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Each WebLogic Server instance writes the messages from its subsystems and applications into a server log file that is located on the local host computer.

In addition to writing messages to a log file, each server instance prints a subset of its messages to standard out. Usually, standard out is the shell (command prompt) in which you are running the server instance. However, some operating systems enable you to redirect standard out to some other location. By default, a server instance prints only messages of a NOTICE severity level or higher to standard out.

Each server instance also forwards a subset of its messages for the administration to collect and place in the domain-wide log file. By default, servers forward messages of severity level NOTICE or higher. While you can modify the type of messages that are forwarded, servers can never forward messages of the DEBUG severity level.

The HTTP subsystem keeps a log of all HTTP transactions in a text file. The default location and rotation policy for HTTP access logs is the same as the server log. You can set the attributes that define the behavior of HTTP access logs for each server.

WebLogic Server Logs

| Log | Description |
|------------------------|---|
| Transaction log | <ul style="list-style-type: none">Contains information about transactions being managed by WebLogic ServerIs used by that server when recovering from crashesIs in binary format |
| JMS Server log | <ul style="list-style-type: none">Is enabled when a JMS Server is createdMessage destinations must be specifically enabled.Contains information on basic message lifecycle events |



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The WebLogic Auditing provider records information from a number of security requests, which are determined internally by the WebLogic Security Framework. The WebLogic Auditing provider also records the event data associated with these security requests, and the outcome of the requests. Configuring an Auditing provider is optional. The default security realm does not have an Auditing provider configured.

Each server has a transaction log, which stores information about committed transactions managed by the server that may not have been completed. WebLogic Server uses the transaction log when recovering from system crashes or network failures. You cannot directly view the transaction log; the file is in a binary format. The Transaction Manager uses the default persistent store to store transaction log files. You can change where the default store is located.

JMS logging is enabled by default when you create a JMS Server, however, you must specifically enable it on message destinations in the JMS modules targeted to this JMS server (or on the JMS template used to create the destinations). JMS server log files contain information on basic message lifecycle events, such as message production, consumption, and removal. When a JMS destination is configured with message logging enabled, then each of the basic message lifecycle events generate a message log event in the JMS message log file.

WebLogic Server Log Locations

| Directory | Description |
|-------------------------------|--|
| <code>domainname</code> | |
| <code>servers</code> | |
| <code>AdminServer</code> | Admin server (named <i>AdminServer</i>) directory |
| <code>logs</code> | |
| <code>AdminServer.log</code> | The server log file for <i>AdminServer</i> |
| <code>domainname.log</code> | The domain log |
| <code>server1</code> | Directory for managed server named <i>server1</i> |
| <code>logs</code> | |
| <code>server1.log</code> | The server log file for <i>server1</i> |
| <code>access.log</code> | HTTP subsystem log |
| <code>jmsServers</code> | |
| <code>jmsservername</code> | |
| <code>jms.messages.log</code> | JMS lifecycle events of the JMS Server called <i>jmsservername</i> created on <i>server1</i> |



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Note that the administration server also has an `access.log` file, which is not shown here.

The auditing log location is also not shown. If the WebLogic Auditing provider is configured, the audit log is here:

`domainpath/domainname/servers/servername/logs/
DefaultAuditRecorder.log`.

The location of the transaction log (also not shown) in the default store is:

`domainpath/domainname/servers/servername/data/store/default/_WLS_SERVERNAMExxxxxx.DAT.` (Where `xxxxxx` is a generated count.)

The JMS subsystem is enabled once JMS Servers are created. There is a log file for each JMS Server.

Log Message Severity Levels

Severity levels from low to high impact:

| Severity | Description |
|----------|---|
| TRACE | Used for messages that are part of WebLogic Diagnostic Framework |
| DEBUG | Messages from enabled "debug flags" |
| INFO | Normal operation information |
| NOTICE | More important operation information |
| WARNING | Something suspicious occurred, but it might not affect normal operation. |
| ERROR | A user level error has occurred, but the system or application can handle it with no interruption and limited degradation of service. |



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

There is more on debug flags later in this lesson.

Log Message Severity Levels

| Severity | Description |
|-----------|--|
| CRITICAL | A system or service level error has occurred. The system can recover, but there may be momentary loss or permanent degradation of service. |
| ALERT | A particular service is unusable, while other parts of the system still function. Automatic recovery is not possible. Immediate attention of an administrator is needed. |
| EMERGENCY | The server is unusable. This indicates a severe system failure. |

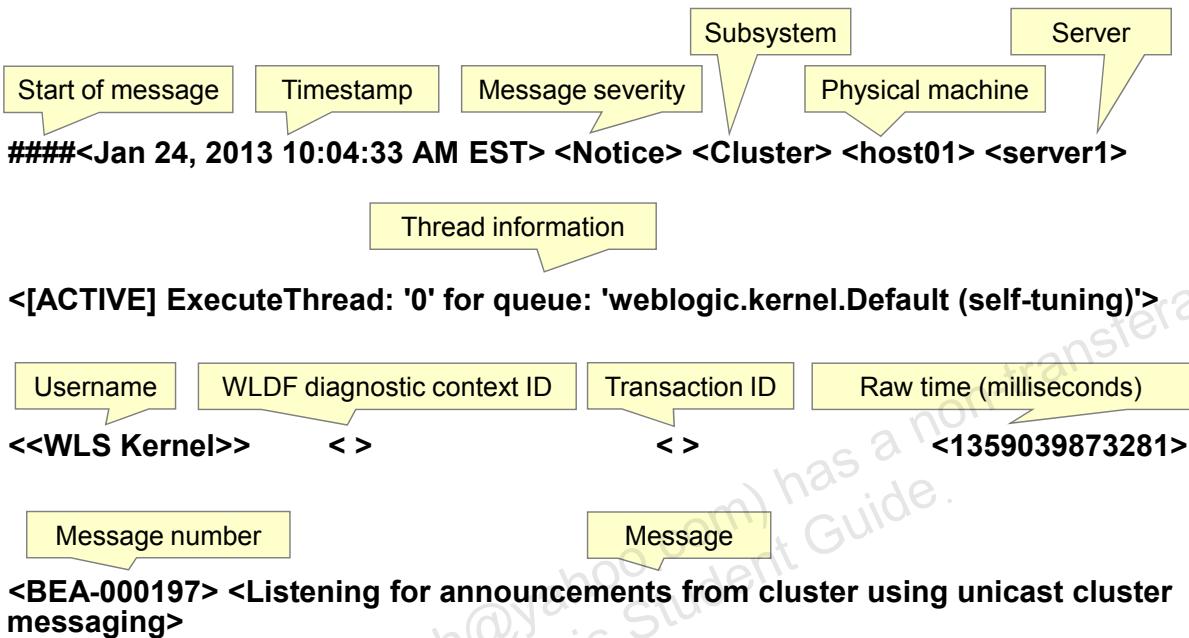


ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Understanding Log File Entries

Log message format:



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Log files can be viewed in any text editor.

In this example, extra blanks are placed in the log entry for readability. The username being used is WLS Kernel, an internal ID used by the server itself. Also, there is no WebLogic Diagnostic Framework (WLDF) context ID (it is blank), nor a transaction ID (also blank).

Accessing the Logs from the Admin Console

Screenshot 1: Domain Structure tree showing 'Log Files' selected (marked with a green circle).

| Name | Type | Server |
|-----------|------------|-------------|
| ServerLog | Server Log | AdminServer |

Screenshot 2: Log Files table showing 'ServerLog' selected and a 'View' button highlighted (marked with a green circle).

| Date | Subsystem | Severity | Message ID | Message |
|-----------------------------|-----------|----------|------------|--|
| Jan 24, 2013 3:40:24 PM EST | Health | Info | BEA-310002 | 12% of the total memory in the server is free. |

Screenshot 3: Server Log Entries table showing a single log entry (marked with a green circle).

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To access logs from the WebLogic Server administration console:

1. Under Domain Structure, select **Diagnostics** and then **Log Files**.
2. Select the log of interest and click **View**.
3. View log entries. To see more details, select an entry and click **View**.

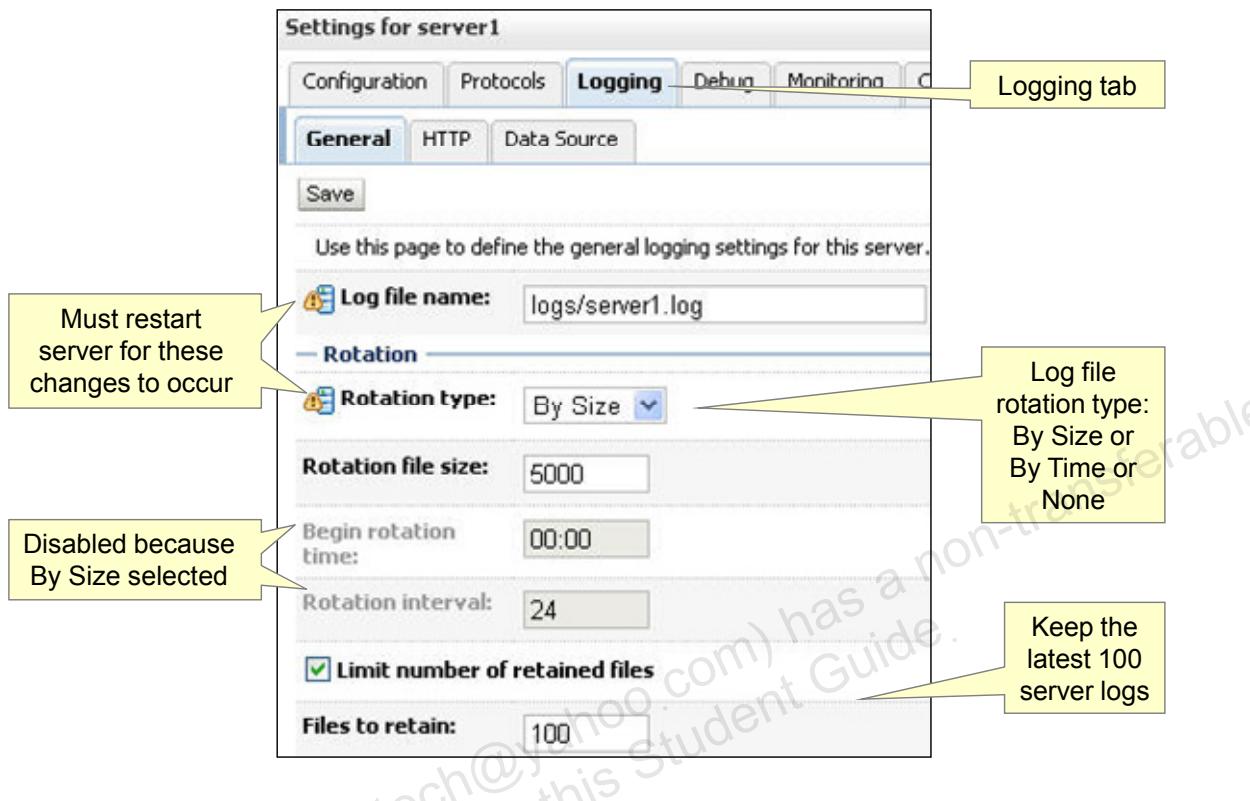
The logs available for viewing in the administration console are:

- Server logs
- Domain logs
- HTTP access logs
- JMS Server logs

The logs available for viewing in the administration console are:

- WebLogic Diagnostic Framework (WLDF) diagnostic logs:
 - **Data source profile log:** Data source diagnostic data that in earlier versions of WebLogic Server was written to the events data archive. What is contained in the data source profile log is configured at the data source level. In the administration console, select the particular data source and then the **Configuration > Diagnostics** tabs. The configuration of the log file itself is done at the server level. In the administration console, select the particular server and then click the **Logging > Data Source** tab. The default location and name of the file is:
domainpath/domainname/servers/servername/logs/datasource.log
 - **Events data archive:** Diagnostic data from WLDF instrumentation. WLDF instrumentation is a mechanism for adding diagnostic code to instances of WebLogic Server or applications to trigger actions at specific code locations and record data from those actions in the archive. This data is placed in the WLDF archive (see below).
 - **Harvested data archive:** Diagnostic data from a WLDF “harvest.” WLDF data can be collected by a WLDF artifact called a data harvester, which is configured to periodically collect diagnostic data and store it in this archive. Harvesters are configured in a WLDF diagnostic module. The default location of the WLDF archive file is:
domainpath/domainname/servers/servername/data/store/
diagnostics/WLS_DIAGNOSTICSxxxxxx.DAT
(Where *xxxxxx* is a generated count.)

Configuring Server Logging



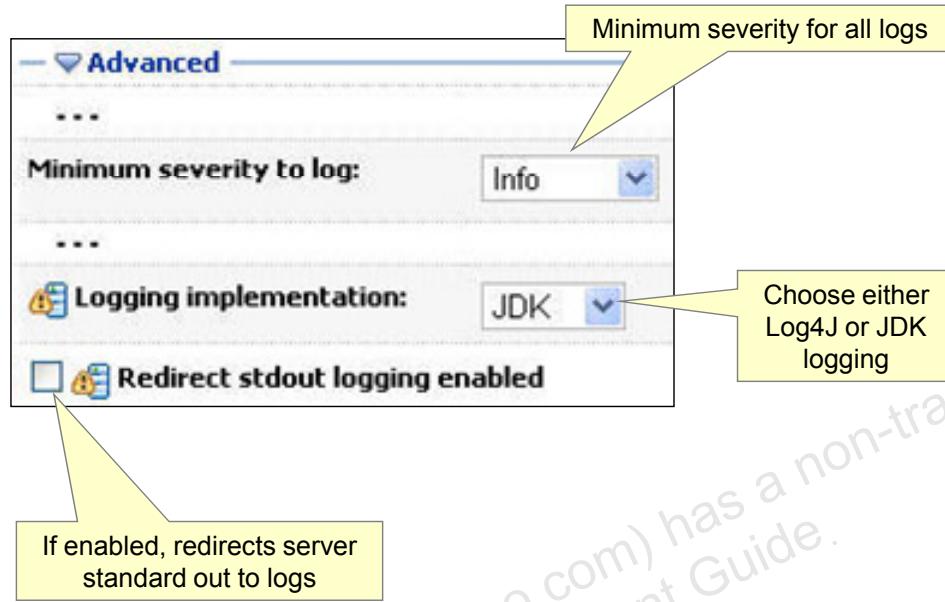
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

After you click **Lock & Edit** in the Change Center, in the Domain Structure of the administration console, expand **Environment** and click **Servers**. In the Servers table, click the name of the server you want to configure. Click the **Logging > General** tab. The available options include:

- **Log file name:** The name of the file that stores the server log. The default is to place the file in the `log` directory under the server directory and name it the server's name `.log`. If you change it and specify a relative path, it is relative to the server's main directory.
- **Rotation type**
 - **None:** Messages accumulate in a single file. You must erase the contents of the file when it grows too large. Note that Oracle WebLogic Server sets a threshold size limit of 500 MB before it forces a hard rotation to prevent excessive log file growth.
 - **By Size:** When the log file reaches the size that you specify in "Rotation file size," the server renames the file to `servername.lognnnnn`.
 - **By Time:** At each time interval that you specify in "Begin rotation time" and "Rotation interval," the server renames the file to `servername.lognnnnn`.
- **Limit number of retained files:** After the server reaches the **File to retain** limit, it deletes the oldest log file and creates a new log file with the latest suffix.

Configuring Server Logging



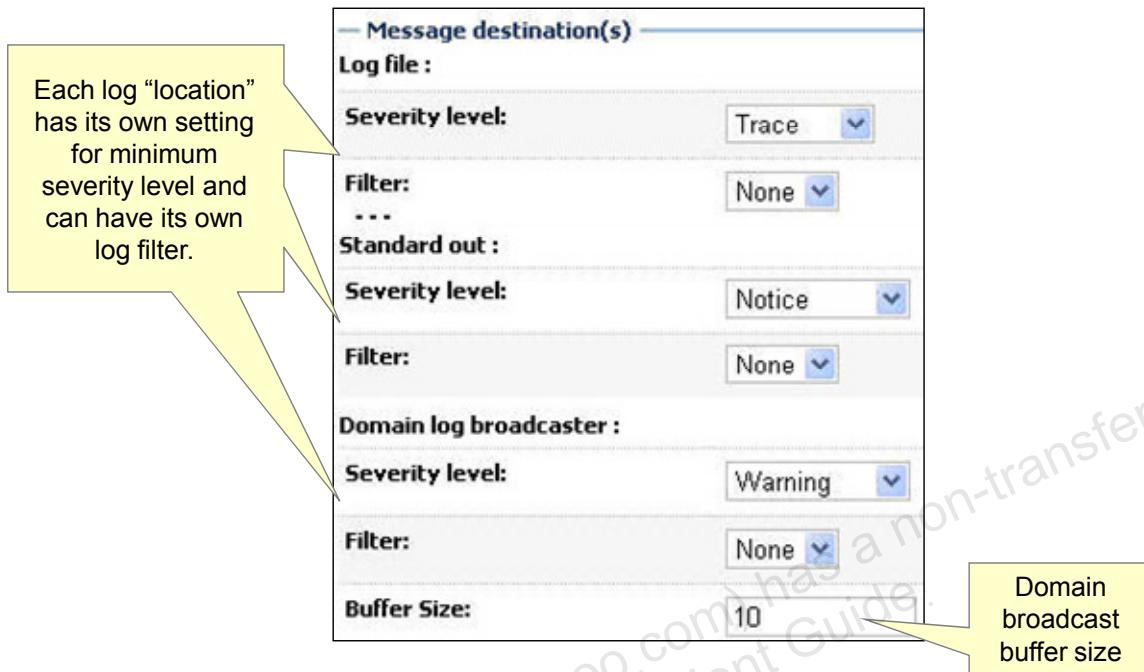
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The available Advanced options include:

- **Minimum severity to log:** The minimum severity of log messages going to all log destinations. By default all messages are published.
- **Logging implementation:** Specifies whether the server logging is based on a Log4J implementation or the default, the logging based on the Java Logging APIs in the JDK
- **Redirect stdout logging enabled:** When enabled, redirects the standard out of the JVM in which a WebLogic Server instance runs to the logging system

Configuring Server Logging



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The available Advanced options include:

- **Log file:** The server log attributes:
 - **Severity level:** The minimum severity of log messages for this log. By default all messages go.
 - **Filter:** Specifies the filter configuration for this log. A filter configuration defines simple filtering rules to limit the volume of log messages written.
- **Standard out:** The standard out attributes:
 - **Severity Level** and **Filter**. (Same explanations as before.) The default severity for standard out is NOTICE.
- **Domain log broadcaster:** The domain log (from this server) attributes:
 - **Severity Level** and **Filter**. (Same explanations as before.) The default severity for the domain log is also NOTICE.
 - **Buffer Size:** Specifies the size of the buffer for the log messages that are sent to the domain log. The buffer is maintained on the Managed Server and is broadcast to the domain log when it gets full. If you notice performance issues due to a high rate of log messages being generated, set this value higher.

Error Messages Reference

Use the WebLogic Server online Error Messages Reference document to obtain more information about a specific log message based on its ID.

Oracle® WebLogic Server Error Messages Reference
12c Release 1 (12.1.2)
Part Number E26117-01

[Home](#) [Book List](#) [Contents](#) [Contact Us](#)

[Previous](#) [View PDF](#)

1 BEA-000001 to BEA-2160002

BEA-000001: Server must be started by Node Manager when consensus leasing is enabled.
Cause: The server was not started by Node Manager.
Action: Start the server using Node Manager.
Level: 1
Type: INTERNAL_ERROR
Impact: ConsensusLeasing

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

For a detailed description of log messages if you only have a message number, use the online document titled *Oracle WebLogic Server Error Messages Reference*.

Log Filters

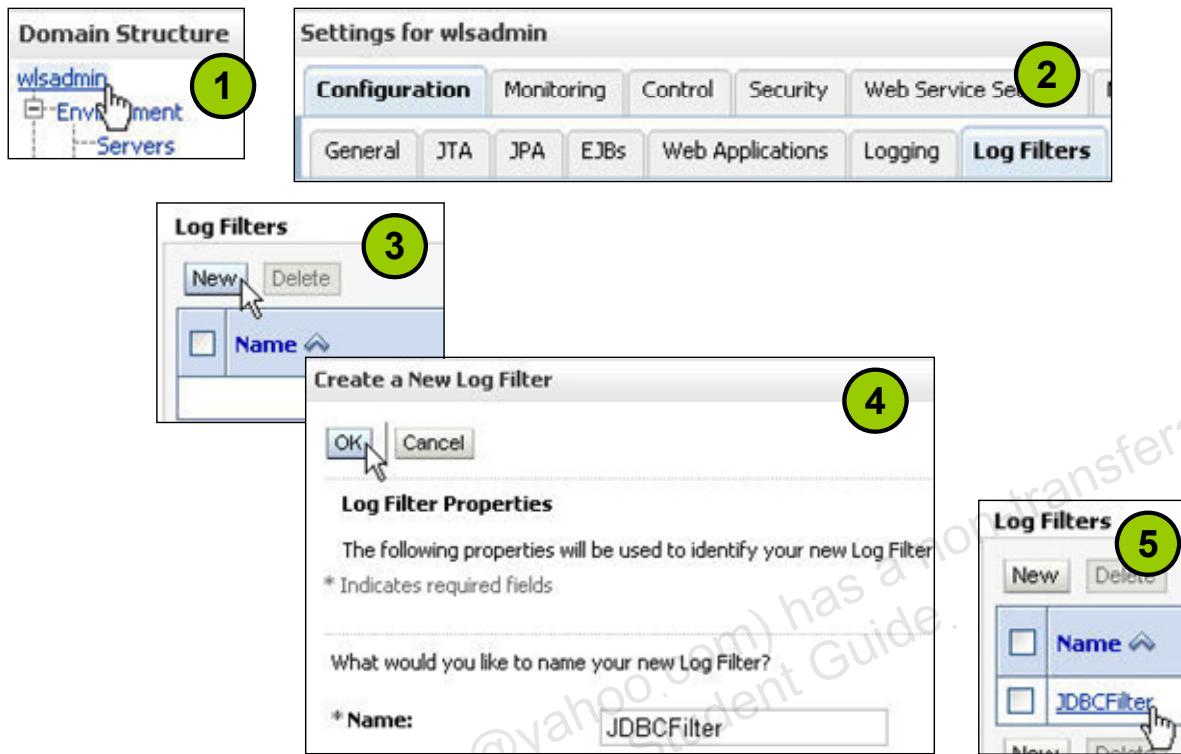
- Provide finer tuned control of the log messages that are published
- Are based on the values of message attributes
- Are created at the domain level
- Can be applied to different log message destinations:
 - Server log
 - Server log memory buffer
 - Server standard out
 - Domain log broadcaster

 ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Log filters provide control over the log messages that are published. A filter uses custom logic to evaluate the log message content and accept or reject a message. For example, you can filter messages of a certain severity level from a particular subsystem. Only the log messages that satisfy the filter criteria are published. You can create separate filters for the messages that each server instance either writes to its server log file, standard out, memory buffer, or broadcasts to the domain-wide log.

Creating a Log Filter

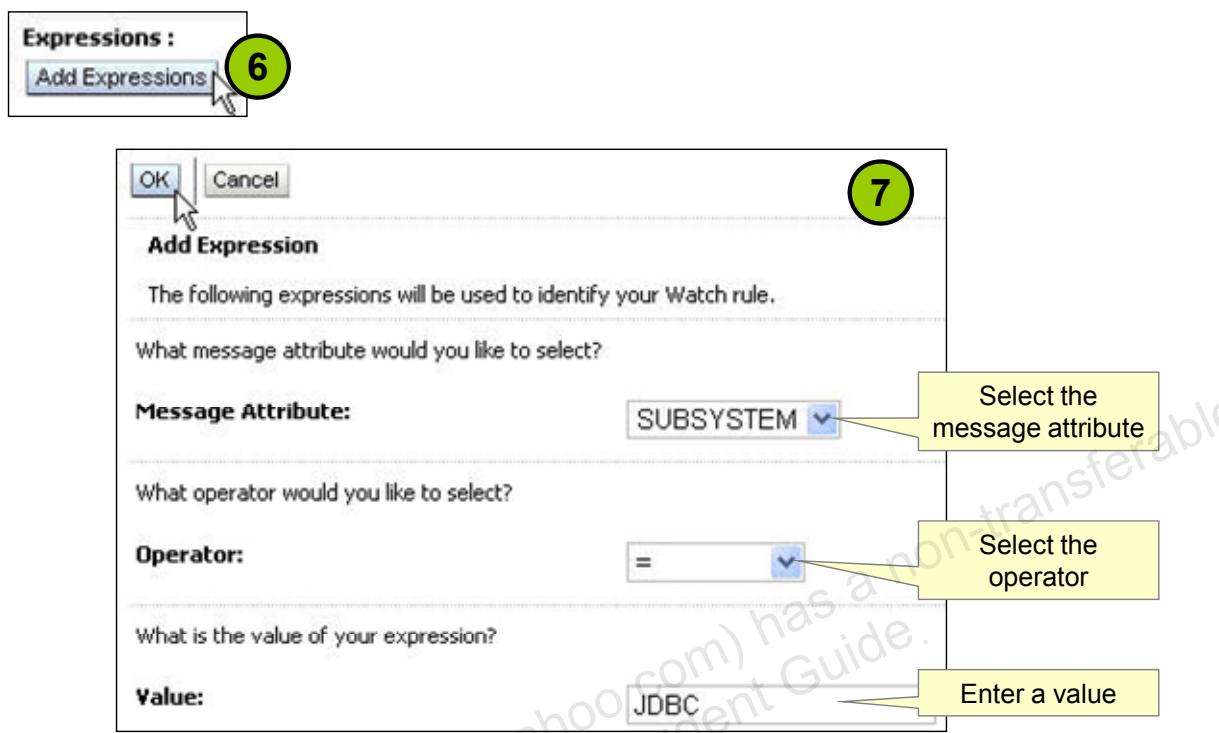


ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

1. After locking the configuration, in the Domain Structure, select the name of the domain.
2. Click the **Configuration > Log Filters** tab.
3. Click the **New** button.
4. On the Create a New Log Filter page, enter a name for the filter and click **OK**.
5. The new log filter appears in the Log Filters table. To configure a filter expression, click the log filter name.

Creating a Log Filter



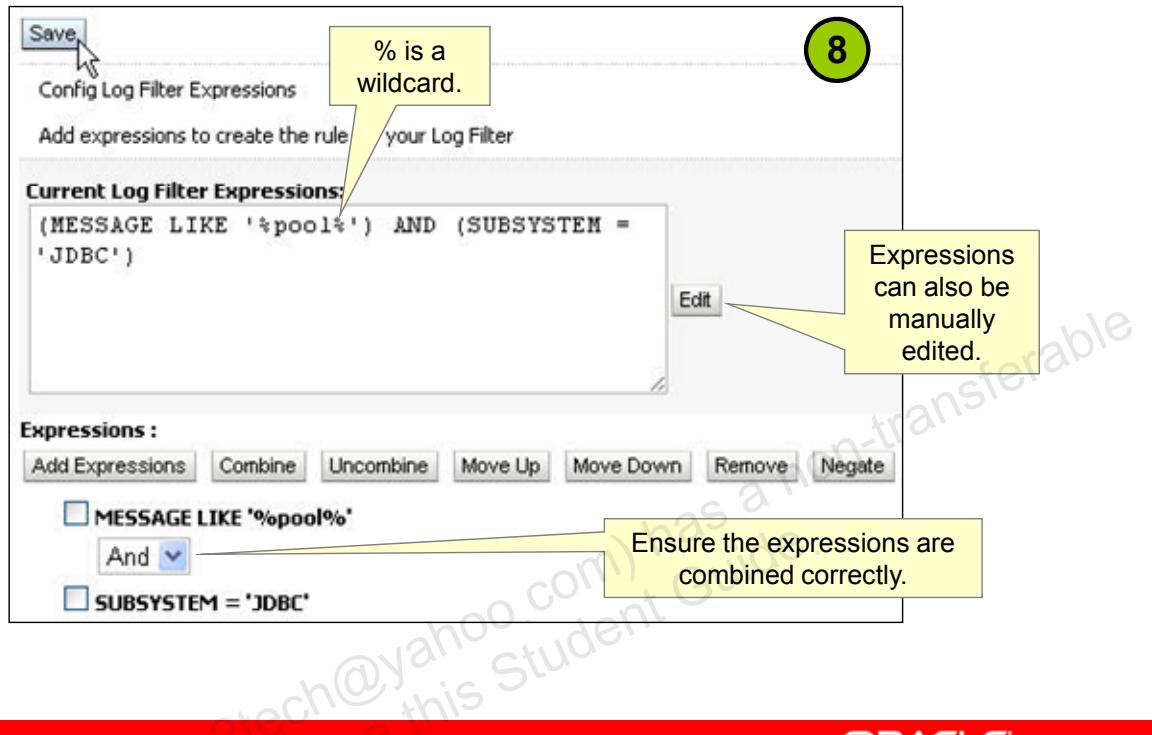
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

6. On the Configuration page, click the **Add Expressions** button.
7. Specify an expression as part of the criteria for qualifying messages. A filter expression defines simple filtering rules to limit the volume of log messages that are written to a particular log destination. Select a Message Attribute, an Operator, and a Value. Then click **OK**. Continue to click the **Add Expressions** button and saving expressions until you have entered all the expressions needed.

Note: The names of the subsystems can be found by looking through the "Impact" values in the error messages in *Oracle WebLogic Server Error Messages Reference*.

Creating a Log Filter



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

8. After the expressions have been entered, ensure that the Boolean operators that connect them are correct. Use the drop-down list to choose **And** or **Or**. Then click the **Save** button. Finally, click **Activate Changes** in the Change Center.

Log filter expressions are WLDF queries. For more information about the WLDF Query Language see the appendix titled "WLDF Query Language" in the *Configuring and Using the Diagnostics Framework for Oracle WebLogic Server* document.

Applying a Log Filter



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

1. After the configuration has been locked, in the Domain Structure, expand **Environment** and select **Servers**. In the Servers table, click the name of the server that you want to configure. Click the **Logging > General** tab.
2. Click **Advanced**.
3. Under the **Message destination(s)** section, specify an existing filter for messages going to any of the four log message destinations:
 - Log file (the server log, being filtered in the slide)
 - Standard out
 - Domain log broadcaster
 - Memory buffer
4. Click the **Save** button. Then, in the Change Center, click **Activate Changes**.

Subsystem Debugging

- Various WebLogic Server subsystems have the ability to generate detailed log messages to facilitate debugging.
- You can enable debugging on specific servers for individual subsystems.

The screenshot shows the 'Settings for server1' interface. The 'Debug' tab is highlighted with a red box and circled with number 1. Number 2 points to the 'Debug settings for this Server' table header. Number 3 points to the table body where 'application' is selected and 'Enabled'.

| Debug Scopes and Attributes | | State |
|-------------------------------------|-------------|----------|
| <input type="checkbox"/> | default | Disabled |
| <input type="checkbox"/> | weblogic | Disabled |
| <input checked="" type="checkbox"/> | application | Enabled |

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

1. After the configuration has been locked, in the Domain Structure, expand **Environment** and select **Servers**. In the Servers table, click the name of the server you want to configure.
2. Click the **Debug** tab.
3. Select one or more available debugging scopes by using the supplied check boxes. Then click **Enable** or **Disable**. For convenience, a **Clear** button is also provided to deselect the debug scopes or attributes that are currently selected.

Debug Scopes

- Debug flags (attributes) for WebLogic Server subsystems are organized into **scopes**.
 - You can enable entire debug scopes or individual attributes.
 - When a parent scope is enabled, all child scopes and attributes are also enabled, unless they are overridden.

| | | |
|--------------------------|---|---------------------|
| <input type="checkbox"/> | <input checked="" type="checkbox"/> weblogic | Disabled |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> application | Enabled |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> classloader | |
| <input type="checkbox"/> | <input type="checkbox"/> cluster | Enabled |
| <input type="checkbox"/> | <input type="checkbox"/> leasing | Enabled (Inherited) |
| <input type="checkbox"/> | <input type="checkbox"/> databaseless | Enabled (Inherited) |
| <input type="checkbox"/> | DebugConsensusLeasing | Enabled (Inherited) |

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Changing debugging is “dynamic,” in that you can enable debugging while a server is running. Many debug flags can also be set as command-line arguments when starting a server. Some examples:

- `-Dweblogic.debug.DebugCluster=true` (cluster debugging)
- `-Dweblogic.debug.DebugJDBCSQL=true` (SQL debugging)

Debug Scopes: Examples

| Subsystem | Scopes (weblogic.*) |
|---------------------|--|
| JDBC | jdbc.connection, jdbc.internal, jdbc.sql |
| Cluster | core.cluster |
| Deployment | deploy, ejb.deployment |
| Applications | application.library, ejb.caching, ejb.invoke, ejb.pooling, servlet, servlet.internal, servlet.internal.session |
| Transactions | transaction.recovery, transaction.twopc, transaction.xa |
| Security | security, security.ldap, security.ssl |

All “debug scope” messages are the DEBUG severity level, so ensure the log location severity level is set appropriately.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

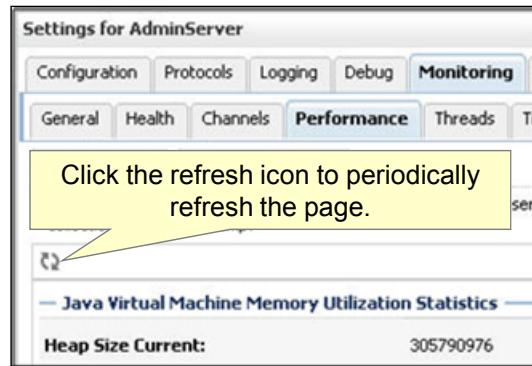
Some examples of attributes under a scope:

- **weblogic.jdbc.connection.DebugJDBCCConn**: Traces all connection reserve and release operations in data sources as well as all application requests to get or close connections
- **weblogic.jdbc.sql.DebugJDBCSQL**: Prints information about all JDBC methods invoked, including their arguments and return values, and thrown exceptions
- **weblogic.core.cluster.DebugCluster**: Prints information about basic cluster lifecycle events

Admin Console: Monitoring Domain Resources

The administration console can monitor domain resources:

- Servers
- Clusters
- Machines
- Deployments
- JDBC data sources
- And more



Use the Domain Structure to locate the type of resource. Select a particular instance. Then click the **Monitoring** tab. The **Monitoring** tab of some elements have subtabs.

- When data is displayed in a table, use the **Customize this table** link to modify the columns displayed.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

- **Servers:** The subtabs under the Monitoring tab are: General (state, version, OS, and so on), Health, Channels (statistics on hosts/ports), Performance (Java memory information), Threads (Java thread information), Timers (internal timers used by the server), Workload (information on Work Managers), Jobs (Job Scheduler information), Security (invalid logins and locked out users), Default Store (information on the default persistent store used by JMS, JTA, WLDF, and others), JMS (information on JMS connections and servers), SAF (information on Store and Forward agents), JDBC (data sources), and JTA (transactions).
- **Clusters:** Information about the servers in the cluster, how often they drop out of the cluster, failover data, and more
- **Machines:** Monitoring of the Node Manager under the machine
- **Deployments:** The state and other information about deployed applications
- **JMS destinations (queues and topics):** Information about messages on the destination
- **JDBC data sources:** Information about connections and capacity

If a monitoring page has a refresh icon, click it to have the administration console periodically poll the resource and update the display. To change the rate at which this occurs, update the Refresh Interval under the admin console's User Preferences.

Monitoring the Domain

The domain itself has a **Monitoring** tab, which can show you an overview of the domain's health, servers, clusters, and migration.

Settings for wlsadmin

Configuration Monitoring Control Security Web Service Security

Health Servers Clusters Migration

This page allows you to monitor health information for this domain.

Health: All Filter

Health Information

| Server/Subsystem Name | State | Health | Reason |
|-----------------------|----------|---------------|--------|
| AdminServer | RUNNING | OK | |
| server1 | RUNNING | OK | |
| server2 | RUNNING | OK | |
| server3 | SHUTDOWN | Not reachable | |

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

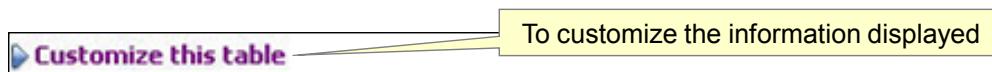
To access domain monitoring, in the Domain Structure, click the domain name, and then click the **Monitoring** tab. Its subtabs are **Health**, **Servers**, **Clusters**, and **Migration**.

The options for the Health filter are:

- **All:** Show all servers.
- **OK:** Show only servers that are functioning without any problems.
- **Warning:** Show servers that have issued warnings and that might have problems in the future.
- **Overloaded:** Show servers that are overloaded; these servers have gone below their free memory threshold percentage. (This threshold is set for a server in the admin console under the **Configuration > Overload** tabs. The field is called “Free Memory Percent Low Threshold.”)
- **Critical:** Show servers that are about to fail. Action must be taken immediately or a server in this state will soon fail.
- **Failed:** Show servers that have failed. A failed server’s Health will display as “Not reachable.”

Monitoring All Servers

The servers table in the admin console lists all the servers in a domain. The information displayed can be customized so you can use this table to see the information important to you.



| Servers (Filtered - More Columns Exist) | | | | | | | |
|--|----------|----------|---------|--------|-----------------------------------|-------------------|--|
| Click the Lock & Edit button in the Change Center to activate all the buttons on this page. | | | | | | | |
| | | New | Clone | Delete | Showing 1 to 4 of 4 Previous Next | | |
| Name | Cluster | Machine | State | Health | Heap Free Current | Heap Size Current | |
| AdminServer(admin) | | machine1 | RUNNING | ✓ OK | 165433584 | 342294528 | |
| server1 | cluster1 | machine1 | RUNNING | ✓ OK | 173135168 | 349372416 | |
| server2 | cluster1 | machine2 | RUNNING | ✓ OK | 245126888 | 331153408 | |

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To access the servers table, in the Domain Structure, expand **Environment** and select **Servers**. The servers table is on the **Configuration** tab.

The data in the servers table can be customized (as many tables can in the administration console). Click **Customize this table** and you can filter the data displayed and adding or subtracting the attributes that are shown. You can also sort the data in the tables by clicking the column headers.

Monitoring Server Health

The admin console server health monitoring page shows the state of the server's subsystems and deployments.

The screenshot shows the 'Settings for server1' interface. The 'Monitoring' tab is active. Under the 'Health' tab, it says 'Server Health: OK' and 'Reason: (No value specified)'. Below that, under 'Health information details', there is a table:

| Subsystem | Health | Reason |
|-----------|--------|--------|
| JDBC | ✓ OK | |
| JTA | ✓ OK | |

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To access the server health information, in the Domain Structure, expand **Environment** and select **Servers**. In the servers table, select the server of interest by clicking its name. Click the **Monitoring > Health** tab.

Monitoring Server Performance

The admin console server performance monitoring page shows information on the JVM.

The screenshot shows the 'Monitoring' tab selected in the top navigation bar. Below it, the 'Performance' tab is also selected. At the top right, there are two buttons: 'Garbage Collect' and 'Dump Thread Stacks'. The main content area displays 'Java Virtual Machine Memory Utilization Statistics' with the following data:

| Statistic | Value |
|--------------------|-----------|
| Heap Size Current: | 320471040 |
| Heap Free Current: | 123103200 |
| Heap Free Percent: | 58 |
| Heap Size Max: | 477233152 |

Three callout boxes point to specific elements:

- A box points to the 'Garbage Collect' button with the text: "Request the JVM do a garbage collection now."
- A box points to the 'Dump Thread Stacks' button with the text: "Display the current stacks for each thread."
- A box points to the 'Refresh' icon (a circular arrow) with the text: "Have the admin console refresh the screen periodically."

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To access the server health information, in the Domain Structure expand **Environment** and select **Servers**. In the servers table, select the server of interest by clicking its name. Click the **Monitoring > Performance** tab.

Monitoring Data Source Health

The admin console data source monitoring lets you view data source state and many statistics about its health and activity.

The screenshot shows the 'Settings for datasource1' page. At the top, there are tabs: Configuration, Targets, **Monitoring**, Control, Security, and Notes. Below these are two sub-tabs: Statistics (selected) and Testing. A callout bubble points to the 'Customize this table' link with the text: 'Many data source attributes are available when you customize the table.' The main content area is titled 'Deployed Instances of this Data Source (Filtered - More Columns Exist)' and shows a table with one row. The table has columns: Server, Enabled, State, and JDBC Driver. The data in the table is:

| Server | Enabled | State | JDBC Driver |
|---------|---------|---------|--|
| server1 | true | Running | oracle.jdbc.xa.client.OracleXADatasource |

Below the table, it says 'Showing 1 to 1 of 1 Previous | Next'.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To access data source monitoring, in the Domain Structure expand **Environment**, then **Services**, and select **Data Sources**. In the data sources table, select the data source of interest by clicking its name. Click the **Monitoring > Statistics** tab.

Many health, activity, and performance data source attributes are available to display by using the **Customize this table** link.

Example Data Source Performance Attributes

Customize the data source monitoring table to display performance data. For example:

| Attribute | Description |
|---|--|
| Active Connections Current Count | The number of database connections currently in use |
| Current Capacity | The total number of connections in the connection pool |
| Failed Reserve Request Count | The running total of connection requests that could not be fulfilled |
| Leaked Connection Count | The number of connections reserved but not returned to the connection pool |
| Number Available | The number of connections idle and available for use |



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The following are some of the performance attributes available under data source monitoring :

- **Active Connections Current Count:** The number of connections currently in use by applications
- **Current Capacity:** The current count of JDBC connections in the connection pool in the data source
- **Failed Reserve Request Count:** The cumulative, running count of requests for a connection from this data source that could not be fulfilled
- **Leaked Connection Count:** The number of leaked connections. A leaked connection is a connection that was reserved from the data source but was not closed. Because it was not closed, it was not returned to the connection pool.
- **Number Available:** The number of database connections that are currently idle and available to be used by applications in this instance of the data source

JMX, MBeans, Managing, and Monitoring

WebLogic Server manages and monitors its resources by using the Java Management Extensions (JMX) API.

- JMX provides a standardized way of managing and monitoring resources through objects called MBeans (managed beans).
- WebLogic Server provides a large set of MBeans for all the resources that it manages and monitors.
 - These MBeans are used by WebLogic Server tools like the administration console, WLST, and the Monitoring Dashboard.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Configuration MBeans contain information about the configuration of WebLogic Server resources. They represent the information that is stored in the domain's XML configuration documents. Each instance of WebLogic Server has an in-memory representation of its configuration as a collection of these read-only Configuration MBeans.

In addition to the read-only Configuration MBeans, the administration server maintains a collection of editable Configuration MBeans. To edit them, you use a JMX client (either the administration console, WLST, or Enterprise Manager Cloud Control). This client goes through the administration server to use its editable Configuration MBeans.

Runtime MBeans contain information about the runtime state of a server and its resources. They generally contain only data about the current state of a server or resource, and they do not persist this data. When you shut down a server instance, all runtime statistics and metrics from the runtime MBeans are destroyed. It is these runtime MBeans that are used by the Monitoring Dashboard to get real-time data.

Monitoring Dashboard

The Monitoring Dashboard:

- Is accessible from a link on the administration console home page
- Graphically presents current or historic WebLogic Server diagnostic data
 - Multiple graph types are available
- Allows you to monitor WebLogic Server MBean attributes
 - From active runtime MBeans (*polled metrics*)
 - From an archive collected by WLDF (*collected metrics*)



The dashboard opens in a new window or tab

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Monitoring Dashboard provides views and tools for graphically presenting diagnostic data. The underlying functionality for generating, retrieving, and persisting diagnostic data is provided by the WebLogic Diagnostics Framework (WLDF).

You can launch the Monitoring Dashboard from the administration console, or you can run it independently. To launch it from the admin console, go to the Home page and under “Charts and Graphs” click the **Monitoring Dashboard** link. The dashboard opens in its own window (or tab). To access the Monitoring Dashboard directly, use the URL:

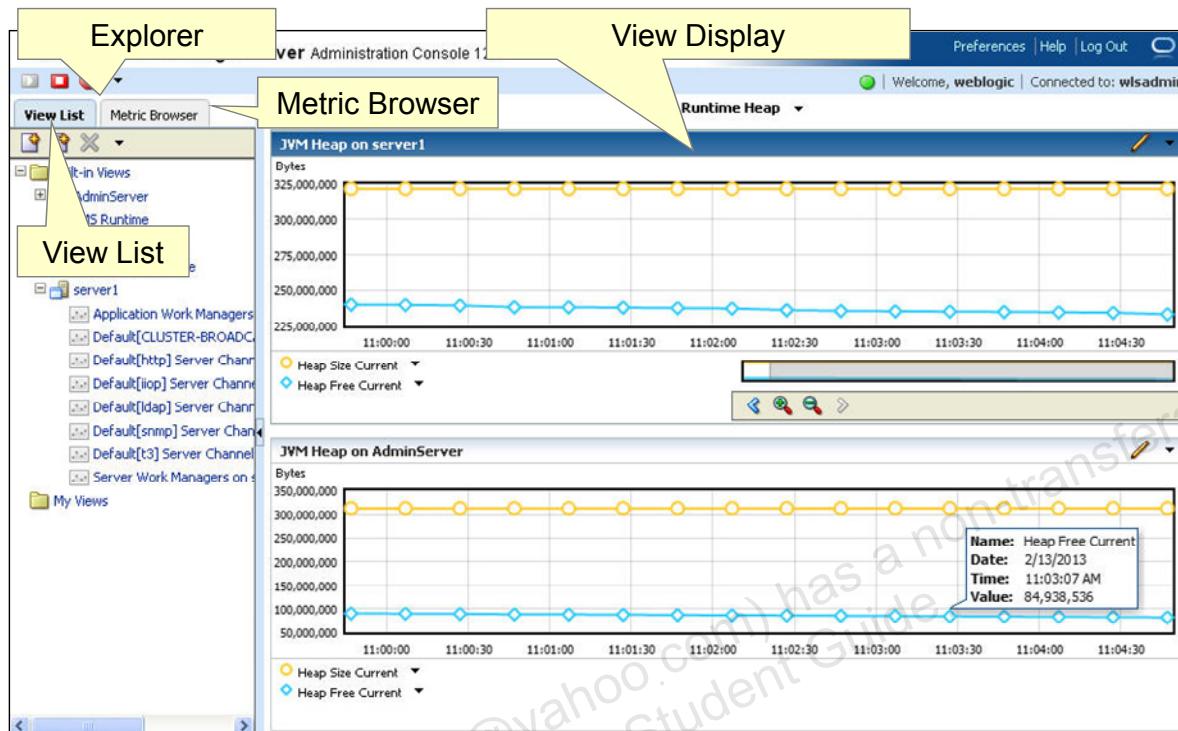
`http : admin_host : admin_port / console / dashboard`

The dashboard, like the admin console, requires you to log in with administrative credentials.

The diagnostic data displayed by the Monitoring Dashboard consists of runtime MBean attributes. These values are referred to in the Monitoring Dashboard as *metrics*. The dashboard obtains metrics from two sources:

- Directly from active runtime MBean instances. These metrics are referred to as *polled metrics*.
- From Archive data that has been collected by a WLDF (by a WLDF artifact called a Harvester). These metrics are referred to as *collected metrics*.

Monitoring Dashboard Interface



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

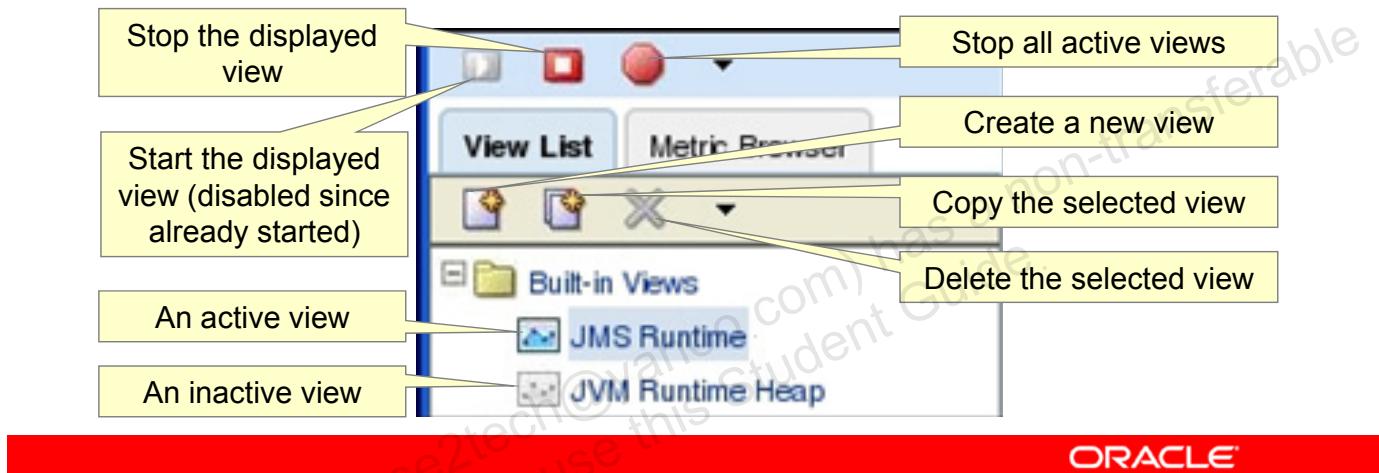
The Monitoring Dashboard has two main panels: the explorer panel and the view display panel.

The explorer panel has two tabs:

- **View List:** A list of existing built-in and custom views. It also contains controls for creating, copying, renaming, and deleting views.
- **Metric Browser:** A way of navigating to and selecting the specific MBean instance attributes whose metric values you want to display in a chart in a view

Views

- Are a way to organize your charts and graphs
- Typically display metrics that are related in some way
- Are individually started (to collect data) and stopped
- Continue to collect data even when not being displayed



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

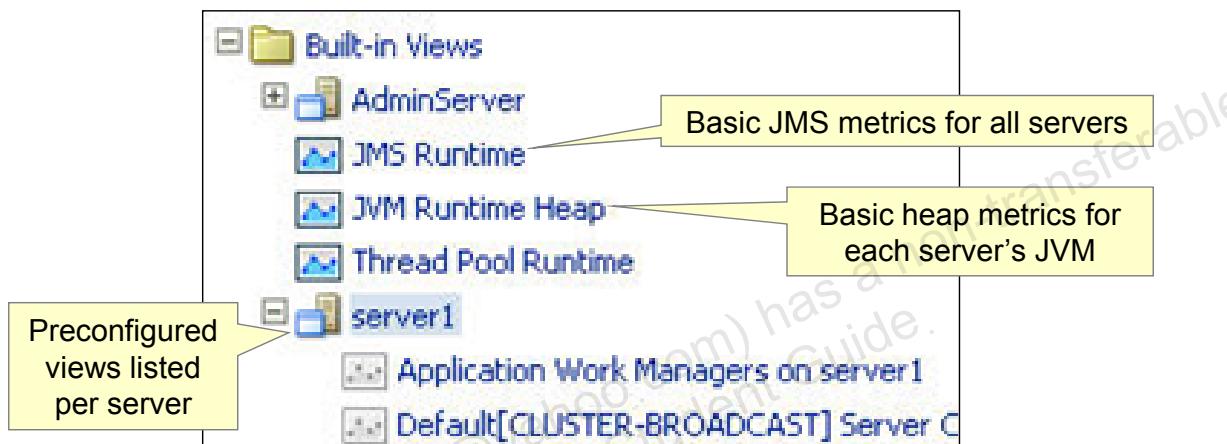
The View List tab lists views. A view is a collection of one or more charts that display captured monitoring and diagnostic data. You can access, create, and update views from the View List tab. When you click the name of a view on the View List tab, that view is displayed in the View Display on the right.

The dashboard uses icons to indicate the status of a view. A gray icon indicates that the view is inactive and data polling is not occurring for the charts in that view. A color icon indicates that the view is active and data polling is occurring for all charts in that view (this is true whether or not the view is currently displayed in the View Display).

To start the data collection for a view, click the view name in the list and click the green Start button above the tabs. To stop data collection, click the red-and-white rectangular Stop button. To stop all active views, click the red octagonal Stop All button.

Built-in Views

- The dashboard defines built-in views for some of the more critical runtime performance metrics.
- Built-in views cannot be modified (or deleted), but they can be copied and the copy modified.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

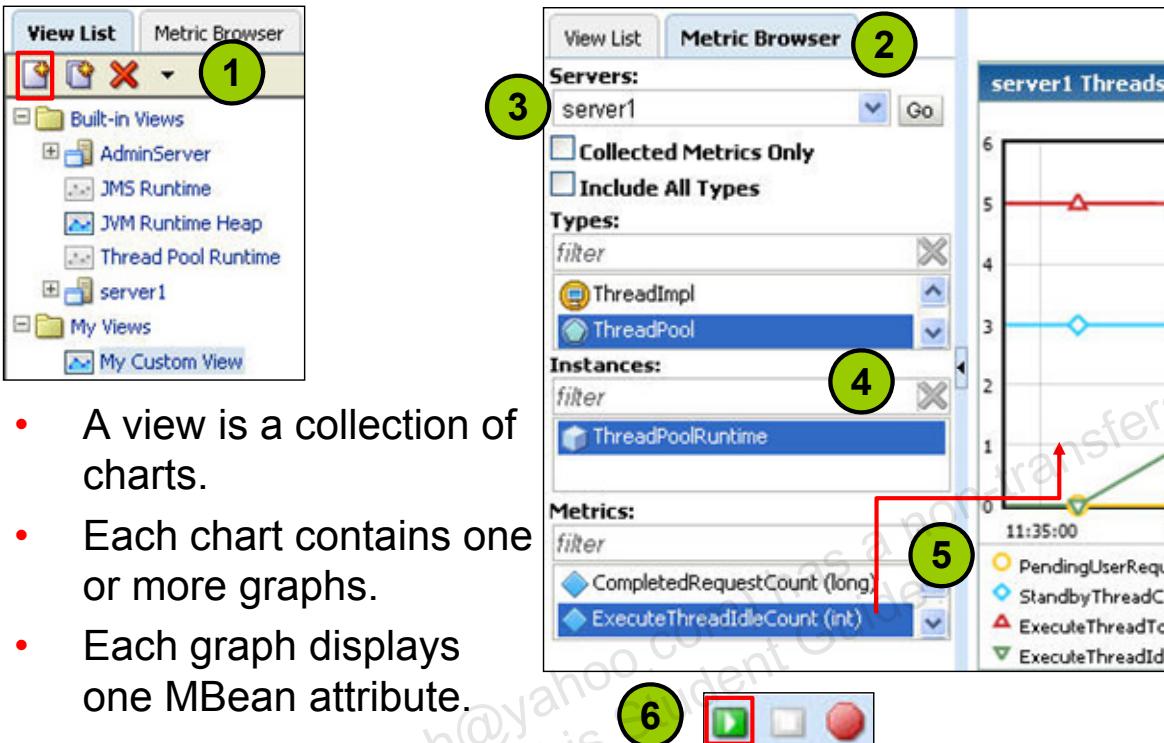
The built-in views are a set of predefined views of available runtime metrics for all running WebLogic Server instances in the domain. These views display some of the more critical runtime WebLogic Server performance metrics and serve as examples of the dashboard's chart and graph capabilities.

You cannot modify a built-in view, but you can copy it. This copy is now one of your custom views. As a custom view, the copy can be modified, renamed, saved, and later deleted.

Built-in views also cannot be deleted.

Custom views are available only to the user who created them and only within the current domain.

Creating a Custom View



- A view is a collection of charts.
- Each chart contains one or more graphs.
- Each graph displays one MBean attribute.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

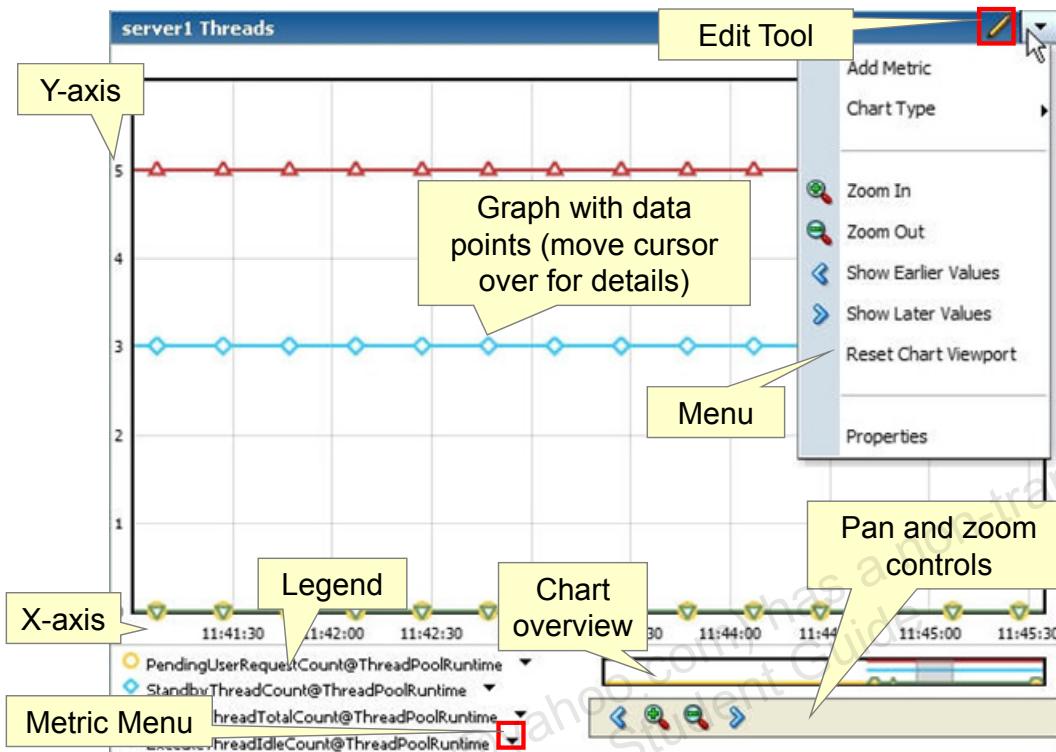
A custom view is any view created by a user. Custom views are available only to the user who created them. You can access a custom view again when needed.

To create a new custom view with a chart and graphs:

1. Click the **View List** tab. Then click the New View button. A new view appears in the list named New View. Replace the default name with something meaningful. Also, a new empty view appears in the View Display area. To add charts to the custom view, use the drop-down menu above the View Display area and click New Chart, or just drag in the first metric (MBean attribute) and a new chart is created for you.
2. To add graphs to a chart, first click the **Metric Browser** tab.
3. Select a server in the Servers drop-down list and click **Go**.
4. Select an MBean type and an MBean instance.
5. In the Metrics list for that instance, drag an MBean attribute to a chart. Note that a view may have as many charts as you like and a chart can graph as many metrics as you like. Also, if a metric is dragged into a view that contains no charts, the dashboard automatically creates a new chart to contain the graph.
6. When the metrics are in place, click the green Start button to start collecting data.

To delete a custom view, select the name of the view and click the Delete (red "X") button.

Anatomy of a Chart



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A chart contains one or more graphs that show data points over a specified time span. A chart also includes a legend that lists the sources of data for each graph along with their associated icons and colors.

When working with a view, you can do the following:

- Add charts to views
- Add graphs to charts
- Pan and zoom
- Edit labels and legends by using the Edit Tool
- Start and stop data collection for charts in a view

Current or Historical Data

- To view real-time metrics, no set up is needed. When a view is started, the runtime MBean instances are polled.
- To view historical (collected) metrics, WLDF must have been previously configured to collect data. Metrics collected by WLDF are placed into the diagnostic archive.
To view harvested data:
 1. In the View List, click the New View button.
 2. In the Metric Browser, select a Server.
 3. To see only harvested data, select Collected Metrics Only.
 4. Drag some attribute from the Metrics list to the new view.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Monitoring Dashboard displays two kinds of diagnostic metrics: real-time data directly from active runtime MBeans (called *polled metrics*) and historical data collected by a previously configured WLDF artifact called a Harvester (called *collected metrics*).

Note that with polled metrics, if polling has been taking place long enough for old data to be purged, a view will not contain all data from the time polling started.

If a WLDF Harvester was configured to harvest data for a particular metric, that historical data is available and can be displayed. Harvesters, or metric collectors, are configured within Diagnostic Modules. To create a Diagnostic Module, from the Domain Structure expand **Diagnostics** and select **Diagnostic Modules**, and then create one by clicking the **New** button. Select the new module and click the **Configuration > Collected Metrics** tabs to set up a collector. Select the **Enabled** check box to enable this collector. Set the period (in milliseconds) between collections in the **Sampling Period** field. Define the metric to collect by clicking the **New** button and use the Create a Metric wizard to select the **MBean Server location**, the **MBean Type**, and its **Attributes**. Target the Diagnostic Module to servers from which you want it to collect data. Harvested data is placed into the diagnostic archive, which can either be a WLDF file store or WLDF JDBC store. By default, the file store is used. The file can be found here: `server_name/data/store/diagnostics`.

Quiz

Which list of severity levels is in order from bad to worse?

- a. ERROR, CRITICAL, ALERT, EMERGENCY
- b. ALERT, ERROR, CRITICAL, EMERGENCY
- c. ERROR, ALERT, CRITICAL, EMERGENCY
- d. ERROR, CRITICAL, EMERGENCY, ALERT



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a

Quiz

A log filter can be applied to only one log message destination at a time.

- a. True
- b. False

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: b

Summary

In this lesson, you should have learned how to:

- Configure and access WebLogic Server logs
- Enable WebLogic Server debugging output
- Monitor WebLogic Server health and performance
- Monitor JDBC data sources
- Access diagnostic charts in the Monitoring Dashboard



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice 8-1 Overview: Working with WebLogic Server Logs

This practice covers the following topics:

- Accessing the server log by using the admin console
- Creating and applying a log filter

Practice 8-2 Overview: Monitoring WebLogic Server

This practice covers the following topics:

- Monitoring a server by using the admin console and the Monitoring Dashboard
- Monitoring JDBC data sources by using the admin console
- Enabling debugging by using the admin console

9

Node Manager

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Configure WebLogic Server machines
- Set up and configure Node Manager
- Start managed servers through Node Manager



Node Manager

Node Manager is a WebLogic Server utility that:

- Can remotely start and stop the administration server or managed servers
- Can monitor the health of an instance of WebLogic Server and automatically restart it if it fails
- Runs as a separate process on the same machines as instances of WebLogic Server
- Is available as either a Java-based or a script-based process (for UNIX or Linux)
- Can be set up as an operating system service to start automatically when a system is rebooted



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Instances of WebLogic Server are often distributed across multiple domains, machines, and geographic locations. Node Manager is a WebLogic Server utility that enables you to start, shut down, and restart the administration server or managed servers from a remote location. Although Node Manager is optional, it is recommended if your WebLogic Server environment hosts applications with high availability requirements.

If Node Manager starts an instance of WebLogic Server and that instance fails, Node Manager can automatically restart it. Node Manager can restart only a failed server that it started. This restart feature is configurable. Node Manager's default behavior is to:

- Automatically restart WebLogic Server instances under its control that fail. You can disable this feature.
- Restart failed server instances no more than a specific number of times. You define the number of restart tries.

If Node Manager fails or is explicitly shut down, upon restart, it determines which WebLogic Server instances were under its control when it exited. Node Manager can then restart any failed server instances if need be.

To control instances of WebLogic Server on a particular machine through Node Manager, there must be a Node Manager process running on that machine.

WebLogic Server provides two versions of Node Manager: Java based and script based (for UNIX or Linux). The two versions have different configuration and security considerations. The Java-based Node Manager contains the latest functionality and runs on any platform that supports WebLogic Server, and so it is the one covered in this lesson.

The administration console can communicate with Node Manager to start, shut down, or restart instances of WebLogic Server.

WLST (in offline mode) can serve as a Node Manager command-line interface.

Node Manager is also involved in automatic server migration within a cluster. If a server fails and cannot be restarted (as in the case of hardware failure), that server can be migrated to new hardware and restarted there.

It is recommended that you run Node Manager as an operating system service, so that it restarts automatically if its host machine is restarted.

Two Types of Node Manager

- Java based
 - Runs within its own Java Virtual Machine
 - Runs on any operating system that runs WebLogic Server
 - Can use secure SSL communication, if desired
- Script based
 - Runs in its own process
 - Available only for UNIX or Linux systems
 - Uses Remote Shell (RSH) or Secure Shell (SSH)

Either type should be set up to start automatically when the computer is booted (as a UNIX daemon or a Windows service).



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Oracle WebLogic Server provides two versions of Node Manager: Java based and script based.

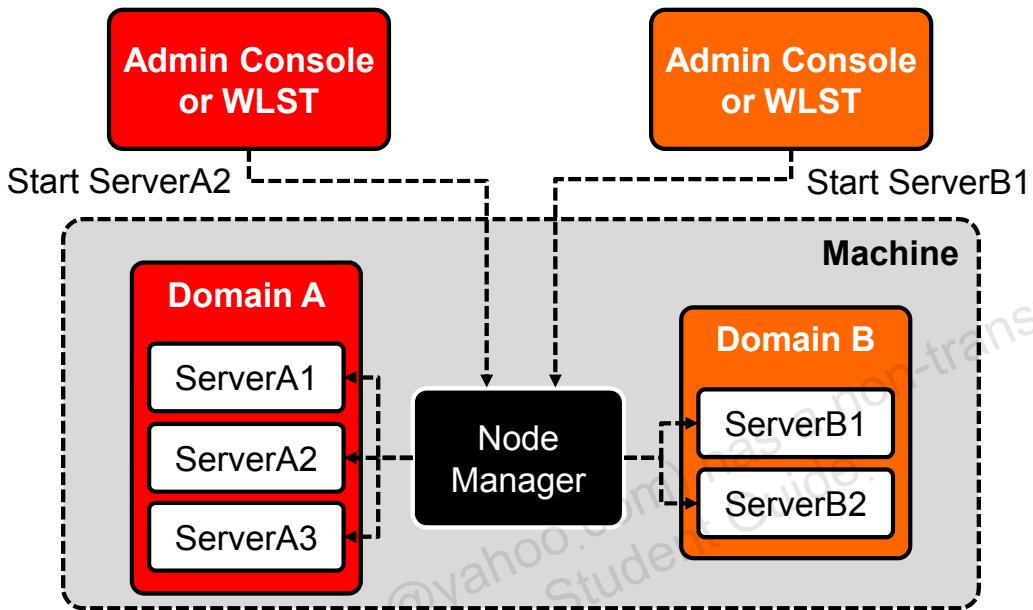
- A Java-based Node Manager runs within its own JVM process. The Java-based version of Node Manager determines its configuration from the `nodemanager.properties` file.
- For UNIX or Linux systems, WebLogic Server also provides a script-based version of Node Manager. This script is based on UNIX shell scripts. It can use SSH for increased security. SSH uses user ID-based security.

Automatic Server Migration is supported by both script-based and Java-based versions of Node Manager.

It is recommended that you run Node Manager as a Windows service on Windows platforms and as a daemon on UNIX platforms, allowing Node Manager to restart automatically when the system is rebooted.

Node Manager Architecture: Per Machine

A single Node Manager can control servers from multiple domains.



ORACLE

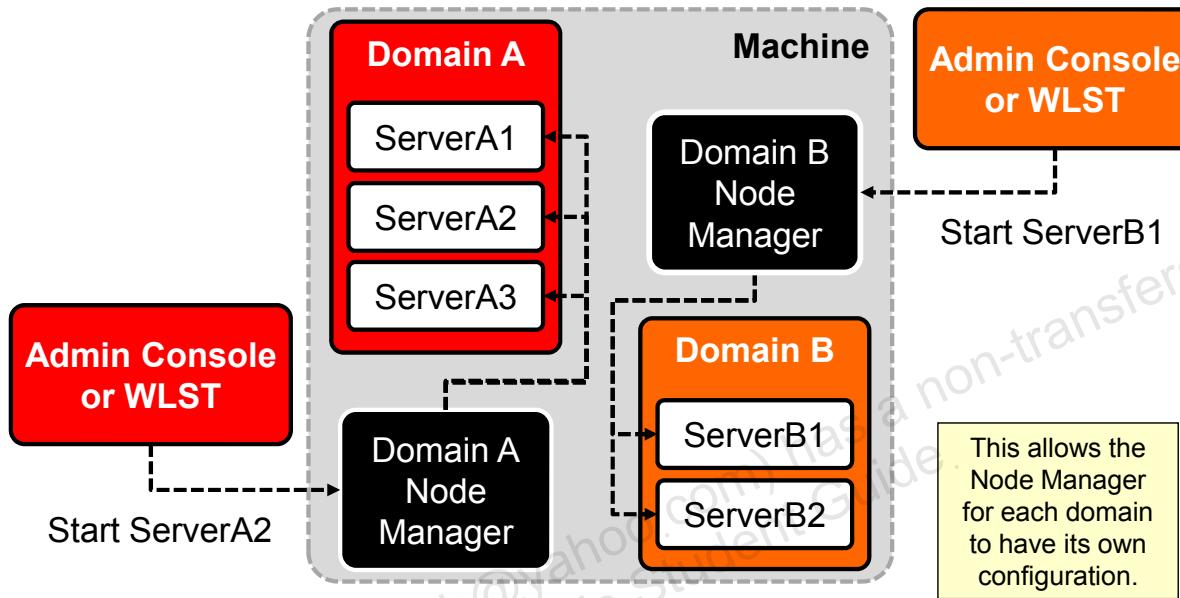
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In previous releases of WebLogic Server, a Node Manager process was not associated with a specific WebLogic Server domain, but instead with a machine. You used the same Node Manager process to control server instances from any WebLogic Server domain, as long as those server instances ran on the same machine.

This configuration is still possible.

Node Manager Architecture: Per Domain

The default is to have one Java-based Node Manager per domain.



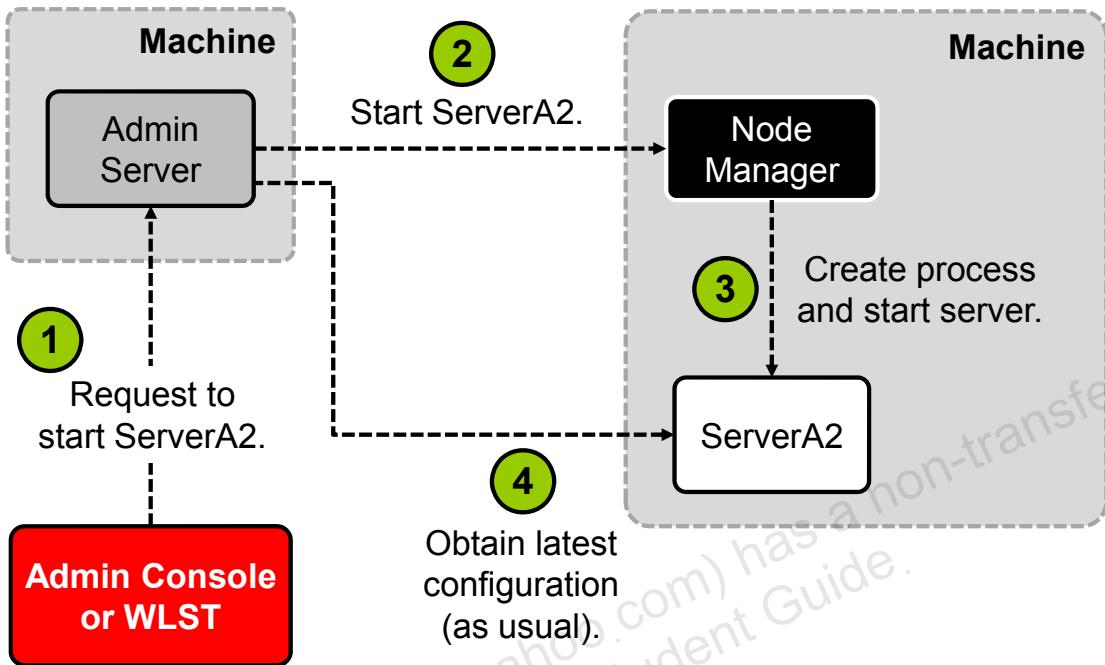
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Starting with WebLogic Server 12.1.2, the default is for a Java version of Node Manager to control all WebLogic Server instances belonging to the same domain.

Having domain-specific (Java-based) Node Managers allows you to have different configurations for different domains.

How Node Manager Starts a Managed Server



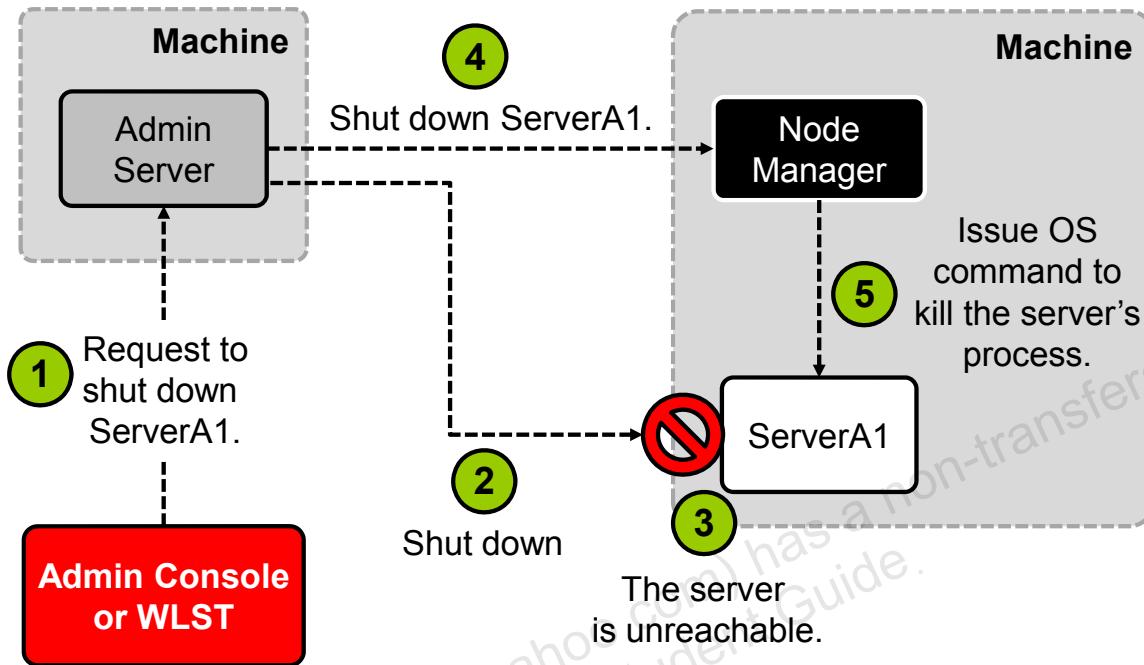
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

1. The Node Manager client, the administration console or WLST, asks to start the managed server.
2. The administration server contacts the Node Manager running on that server's machine, asking Node Manager to start that server. The admin server verifies its identity by sending along the Node Manager credentials (username and password). The server's startup properties are also sent (from the server's **Configuration > Server Start** page). If those properties have not been set, Node Manager uses the defaults found in the `nodemanager.properties` file.
3. Node Manager creates the server process and starts the server.
4. As usual, when a managed server comes up, it obtains the latest configuration from its administration server.

Note: This is the usual procedure used to start a managed server through Node Manager. It is also possible to start a managed server by using WLST and connecting directly to Node Manager (bypassing the administration server).

How Node Manager Can Help Shut Down a Managed Server



ORACLE

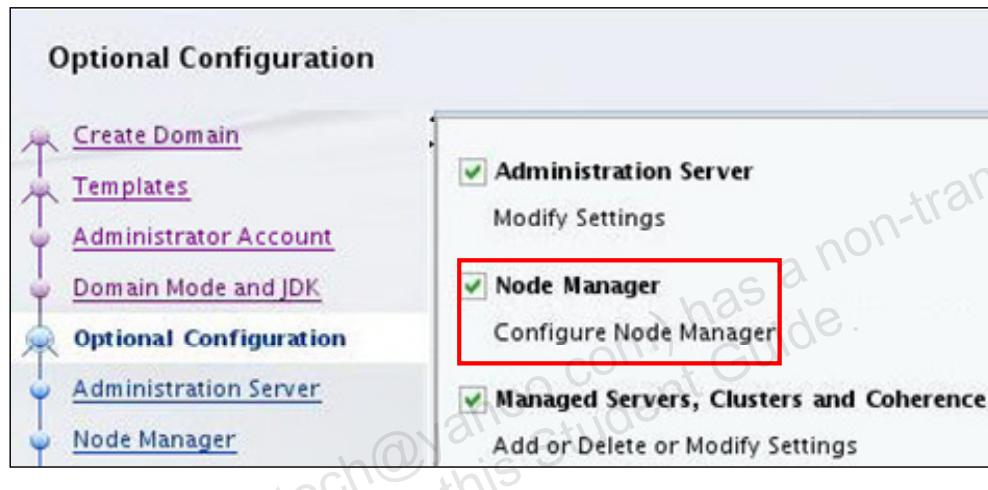
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

1. The Node Manager client, the administration console or WLST, asks to shut down the managed server.
2. The administration server issues the shutdown command directly to the managed server. Normally, the server then performs the shutdown sequence.
3. If the server is unreachable (perhaps it is “hung”), the admin server contacts Node Manager.
4. The administration server asks Node Manager to shut down the server.
5. Node Manager requests that the operating system kill the server’s process.

Configuration Wizard and Node Manager

When creating a domain with the Configuration Wizard, there is an option for Node Manager.

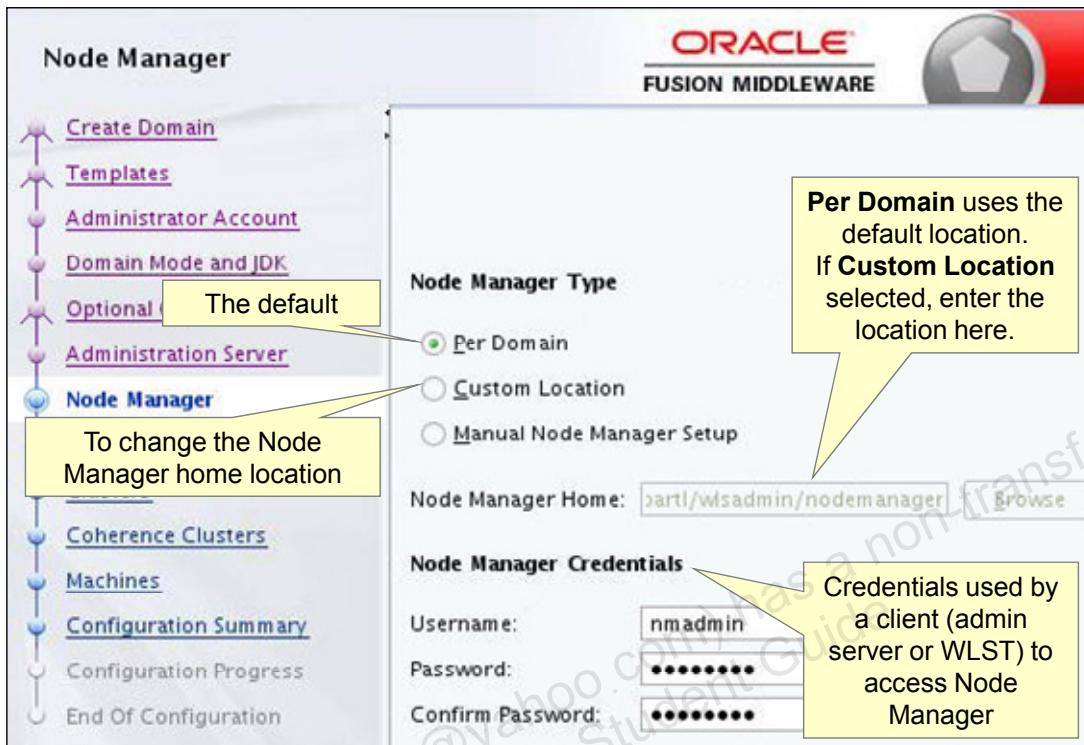
- You can select the **Node Manager** check box on the Optional Configuration page of a wizard.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Configuration Wizard and Node Manager



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

On the Node Manager page:

- Select:
 - Per Domain: Node Manager home is the default location of <domain>/nodemanager.
 - Custom Location: Enter the home directory in the Node Manager Home field.
 - Manual Node Manager Setup: This bypasses any Node Manager configuration by the wizard.
- Enter the Node Manager Credentials:
 - Username/Password/Confirm Password: Enter the credentials that the administration server or WLST will use to authenticate with Node Manager. Note that these are not the same as the WebLogic Server administrator username and password.
- Then click **Next**.

Configuring the Java-Based Node Manager

1. Define machines in the domain to represent each computer.
2. Under each machine configuration:
 - A. Assign servers to the machine.
 - B. Set the Java Node Manager type to Plain or SSL.
 - C. Set the listen address to the Node Manager computer's IP address or its host or DNS name. Select a listen port.
3. In each managed server configuration, set the server start parameters and the monitoring and restart parameters used by Node Manager. (If not set by you, these parameters have default values.)

Note that some of the configuration can be done when creating the domain.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

It is recommended that you use the SSL type of the Java-based Node Manager in a production environment. Configuring SSL involves obtaining identity and trust for the Node Manager and each administration and managed server with which the Node Manager will be communicating, and then configuring the Node Manager, the admin server, and the managed servers with the proper identity and trust. For more information, see “Configuring SSL” in the *Administering Security for Oracle WebLogic Server* document.

Configuring Server Start and Health Monitoring Parameters

Settings for server1

| Configuration | Protocols | Logging |
|-------------------------------------|----------------------|---------|
| Health Monitoring | Server Start | Web |
| <input type="button" value="Save"/> | | |
| Java Home: | <input type="text"/> | |
| Java Vendor: | <input type="text"/> | |
| BEA Home: | <input type="text"/> | |
| Root Directory: | <input type="text"/> | |

Settings for server1

| Configuration | Protocols | Logging | De |
|--|-----------------------------------|---------|----|
| Health Monitoring | Server Start | Web S | |
| <input type="button" value="Save"/> | | | |
| Health Check Interval: | <input type="text" value="180"/> | | |
| <input type="checkbox"/> Auto Kill If Failed | | | |
| <input checked="" type="checkbox"/> Auto Restart | | | |
| Restart Interval: | <input type="text" value="3600"/> | | |
| Max Restarts Within Interval: | <input type="text" value="2"/> | | |
| Restart Delay Seconds: | <input type="text" value="0"/> | | |



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Server Start parameters are the kinds of things you can set in server start scripts. They are:

- **Java Home:** Where the JVM is located to use for running this server. Enter the parent directory of the JDK `bin` directory.
- **Java Vendor:** The Java vendor. This needs to be entered if different Java vendors are used in the same cluster.
- **BEA Home:** The home directory of “BEA” products
- **Root Directory:** The server’s root directory. The domain directory is used if this is not specified.
- **Class Path (not shown):** The `CLASSPATH` for this server
- **Arguments (not shown):** Arguments to use when starting the server (for example, JVM memory arguments)
- **Security Policy File (not shown):** The Java security policy file and the path to it
- **User Name (not shown):** The username to use when starting this server
- **Password (not shown):** The password to use when starting this server. Note that the password is stored in the configuration encrypted.

Note that if these attributes are left blank, they are set to default values.

Health Monitoring parameters:

- **Health Check Interval:** The server monitors the health of its subsystems every Health Check Interval seconds and changes the server's overall state if it changes.
- **Auto Kill If Failed:** Specifies whether Node Manager should automatically kill this server if its health state is “failed”
- **Auto Restart:** Specifies whether Node Manager should automatically restart this server if it crashes or otherwise goes down unexpectedly
- **Restart Interval:** The number of seconds during which Node Manager can try to restart this server
- **Max Restarts Within Interval:** The number of times that Node Manager tries to restart this server within the time specified in Restart Interval
- **Restart Delay Seconds:** The number of seconds Node Manager must wait before trying to restart this server

Note that if these attributes are left blank, they are set to default values.

Configuring the Java-Based Node Manager

4. Install WebLogic Server on the computers where you plan to run Node Manager (and managed servers).
5. *Create a directory for Node Manager on each computer:
 - A. Copy the `startNodeManager.sh` script to the directory from `<WL_HOME>/server/bin`.
 - B. Edit the script and set `NODEMGR_HOME` to the current path.
 - C. Create and edit a text file named `nodemanager.properties`.
 - D. Set (at least) the following properties:
 - `ListenAddress=value`
 - `ListenPort=value`

These are done for you if Node Manager is set up through the Configuration Wizard.

* The default Node Manager home is the `nodemanager` directory under the domain directory. This is the location used when Node Manager is configured during domain creation by using the Configuration Wizard and the “Per Domain” Node Manager type is selected.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Creating a unique directory for Node Manager is not a requirement, but a recommended practice, and the default for the Java-based “per domain” Node Manager.

When you start the Node Manager configuration during domain creation by using the Configuration Wizard, the default Node Manager Type is “Per Domain.” The default location of Node Manager Home is the `nodemanager` directory under the domain directory. The `nodemanager.properties` file is created and placed under that directory, and the values for the `ListenAddress` and `ListenPort` properties are set for you. If you set them yourself or change the values, ensure that they match the values set in the machine configuration.

Configuring the Java-Based Node Manager

6. On each computer, enroll Node Manager with the domain.
7. Start Node Manager by running `startNodeManager.sh`.
8. *Set up a UNIX daemon to start Node Manager automatically on computer startup.

* Not required, but recommended



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

More on enrolling Node Manager with a domain is found later in this lesson.

Setting up a UNIX daemon (or a Windows service) to start Node Manager is not a requirement, but a recommended practice.

Other Node Manager Properties

The nodemanager.properties file has many properties:

| Property | Description |
|-----------------------|--|
| AuthenticationEnabled | Node Manager authenticates the admin server against the credentials defined in the domain. |
| StartScriptEnabled | Start servers by using a script. |
| StopScriptEnabled | Stop servers by using a script. |
| StartScriptName | Name of the script used to start servers |
| StopScriptName | Name of the script used to stop servers |
| CrashRecoveryEnabled | Automatically restart failed servers after machine restart. |
| StateCheckInterval | Time in between checks of a server's state |



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

There are many more properties than those listed in the slide. See the “Configuring Java Node Manager” chapter of *Node Manager Administrator’s Guide for Oracle WebLogic Server*.

Node Manager Files

Under the Node Manager home directory:

- **nodemanager.properties**: Is used to set Node Manager attributes
- **nodemanager.domains**: Stores the domains with which this Node Manager communicates
- **nodemanager.log**: Is used by a running Node Manager to log information about its work

Under the domain's config/nodemanager directory:

- **nm_password.properties**: Stores the encrypted Node Manager username/password that the Node Manager client uses to authenticate itself to Node Manager



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Note that the domain directory being discussed in this and the next few slides is the domain directory on the machine where Node Manager and the managed servers it controls are running.

Node Manager Files

Under the domain's security directory:

- **SerializedSystemIni.dat**: Is used by Node Manager for encryption and decryption

Under the domain's directory called
servers/*servername*/data/nodemanager:

- **boot.properties**: Is created by Node Manager to hold a server's encrypted credentials
- **startup.properties**: Keeps track of server start and health monitoring options
- ***servername.lck*, *servername.pid*,
*servername.state***: Are files used internally by Node Manager to track a server it started



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Note that the **boot.properties** file used by Node Manager is created automatically by Node Manager. It is not the same file (or in the same location) as a **boot.properties** file you can create and place in a server's security directory.

Node Manager Files

Under the domain's directory called
servers/*servername*/logs:

- ***servername.out***: Is the log file for a server started by Node Manager that contains `stdout` and `stderr` messages generated by that server
 - You cannot set up “rotation” for this file, as you can with a server log file, which can be an issue. An easy solution:
 - For servers that are started by Node Manager, set the standard out “severity level” to `Critical` or higher. Then very few messages go into this file.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Enrolling Node Manager with a Domain

Enrolling Node Manager with a domain copies files required by Node Manager to the remote computer (or updates existing files). Then the Node Manager running on this machine can accept requests from that domain. To enroll with a domain:

1. Start WLST on the computer where you plan to run Node Manager.
2. Run the `connect()` command to connect to the administration server of the domain:

```
connect('username', 'pw', 't3://host:port')
```

3. Run the `nmEnroll()` command:

```
nmEnroll('domain_home', 'node_mgr_home')
```

Domain location

Node Manager home

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

For the `connect()` command, the first two arguments are the username and password of an administrator. The third argument is the URL to access the administration server.

The `nmEnroll()` command downloads two files from the administration server to this computer: `nm_password.properties` and `SerializedSystemIni.dat`.

The Node Manager “secret file,” `nm_password.properties`, contains the encrypted username and password used by the administration server to authenticate itself to the Node Manager process. This file is placed under the domain directory in `config/nodemanager/`.

The file used for encryption and decryption, `SerializedSystemIni.dat`, is placed under the domain directory in `security/`.

Also, the `nmEnroll()` command creates the `nodemanager.domains` file (or updates an existing file) found in the Node Manager home directory. It adds the current domain to the list of domains with which this Node Manager communicates.

When Not to Use nmEnroll()

- `nmEnroll()` is for registering Node Manager with a “WebLogic Server only” domain on a remote machine. It can be part of the process of preparing that machine for running the managed servers of the domain.
 - You should *not* use `nmEnroll()` if your domain uses other FMW components, because it is insufficient to ready a machine to run the managed servers in such a domain.
 - The `nmEnroll()` command does not transfer the required FMW component-specific files to the new machine.
- Using `nmEnroll()` is not needed if you have copied the domain to the new machine by using the pack and unpack utilities.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

There is a scenario in which you use `nmEnroll()` *after* the pack and unpack utilities have been used to copy the domain to a remote machine. If, after the domain has been copied over, the Node Manager username and password are changed, the credentials in the file on the remote machine (`nm_password.properties`) will be out-of-date. This means that the administration server fails when it tries to contact this Node Manager. However, if you issue the `nmEnroll()` command from the remote machine, the command updates the credentials in that file.

Reminder: Pack

1. On the machine where the domain files reside (and the administration server runs), use the `pack.sh` script with the `-managed=true` option:

```
cd <MW_HOME>/oracle_common/common/bin  
./pack.sh -domain=domain_path/domain_name  
           -template=name.jar  
           -template_name=somename  
           -managed=true
```

2. Move the archive file to the machine where you plan to run Node Manager and the managed servers (and the products are already installed).



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

For details of readying a machine for running managed servers in a domain that contains Fusion Middleware Components (beyond WebLogic Server), see the *Enterprise Deployment Guide* for that component. For example, for Oracle SOA Suite, see the chapter titled “Creating a Domain for an Enterprise Deployment” in the *Enterprise Deployment Guide for Oracle SOA Suite*. This document details preparing the network, file system, and database for SOA Suite. It covers product installation, domain creation (the chapter referenced above), domain extension for SOA Suite (and other components), setting up Node Manager, and more.

Reminder: Unpack

3. Before running the `unpack.sh` script, ensure that the directory exists in which to place the domain: `new_path`
4. Run the `unpack.sh` script:

```
cd <MW_HOME>/oracle_common/common/bin  
./unpack.sh -domain=new_path/domain_name  
             -template=name.jar  
             -app_dir=new_path/applications  
             -nodemanager_home=nodemgr_home_dir
```

- The `-app_dir` argument is optional. It specifies the application directory (used by some FMW components).
- The `-nodemanager_home` argument is also optional. Use this to place Node Manager somewhere other than the default location: `<domain>/nodemanager`.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The `-app_dir` option specifies the full path to the directory used to store the applications defined in the template.

The `-nodemanager_home` option specifies the full path to the Node Manager home directory. If not used, it defaults to the `nodemanager` directory under the domain.

Controlling Servers Through Node Manager

After the configuration is set and Node Manager is running on the remote machines, servers can be started by using the WebLogic Server administration console or WLST.

Summary of Servers

Configuration Control

Customize this table

Servers (Filtered - More Columns Exist)

| | Server | Machine | State |
|-------------------------------------|--------------------|----------|----------|
| <input type="checkbox"/> | AdminServer(admin) | machine1 | RUNNING |
| <input type="checkbox"/> | server1 | machine1 | RUNNING |
| <input checked="" type="checkbox"/> | server2 | machine2 | SHUTDOWN |

Start Resume Suspend Shutdown Restart SSL



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Depending upon the settings of Node Manager, servers that it starts can be monitored and automatically restarted if they fail.

Node Manager: Best Practices

- Do not place the Node Manager home under the WebLogic Server installation directories.
 - The default location for the per-domain Java-based Node Manager is under the domain directories.
- Use the Java-based Node Manager.
 - It is portable and has the latest features.
- Use the “per domain” Node Manager.
 - The default with the Java-based Node Manager
 - Allows each domain to have a different Node Manager configuration
- Use Node Manager to start the administration server, too.
 - If it fails, Node Manager can automatically restart it.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

For Node Manager to be able to start the administration server, you must use WLST. For more information about how to do that, see the section titled “Using Node Manager to Start Servers on a Machine” in *Understanding the WebLogic Scripting Tool*.

Node Manager: Best Practices

- Use a start script to start servers.
 - Use the script to set WebLogic Server parameters or prepare resources before the server starts.
 - The default for `StartScriptEnabled` is `true`.
 - The default `StartScriptName` is `startWebLogic.sh`.
- In production, use SSL between the admin server and the Java-based Node Manager.
 - This is one-way SSL.
 - Set up SSL by obtaining and configuring Identity and Trust keystores on the admin server.
 - Configure the admin server to listen on a specific IP, not “all local addresses.”
 - Set `SecureListener=true` in `nodemanager.properties`.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The parameters that can be set in a script are JVM and WebLogic Server options. Other things could be done in the script as well (for example, mounting a drive used by the server).

For more information about setting up SSL, see the “Configuring SSL” chapter in *Securing Oracle WebLogic Server*.

Because of host name verification, the admin server needs to be configured to listen on a specific IP address, rather than the default of “all local addresses.” Otherwise, SSL communication can fail and Node Manager reject the admin server commands.

If you have configured an admin port for the domain, SSL must be configured for all servers in the domain. Communication to the servers is through the secure admin port, including communication from Node Manager.

Quiz

To configure a Node Manager by using the admin console, what resource is selected before you click its **Node Manager** tab?

- a. A managed server
- b. The administration server
- c. A machine
- d. A cluster

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: c

Quiz

When you shut down a managed server by using the admin console, Node Manager is involved:

- a. Each time
- b. If the managed server cannot be reached by the admin server
- c. Never



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: b

Summary

In this lesson, you should have learned how to:

- Configure WebLogic Server machines
- Set up and configure Node Manager
- Start managed servers through Node Manager

Practice 9-1 Overview: Configuring and Using Node Manager

This practice covers the following topics:

- Configuring the Java-based Node Manager
- Starting Node Manager
- Starting servers through Node Manager by using the administration console

Unauthorized reproduction or distribution prohibited. Copyright© 2016, Oracle and/or its affiliates.

Tony Albrecht (base2tech@yahoo.com) has a non-transferable
license to use this Student Guide.

10

Deploying Applications

ORACLE®

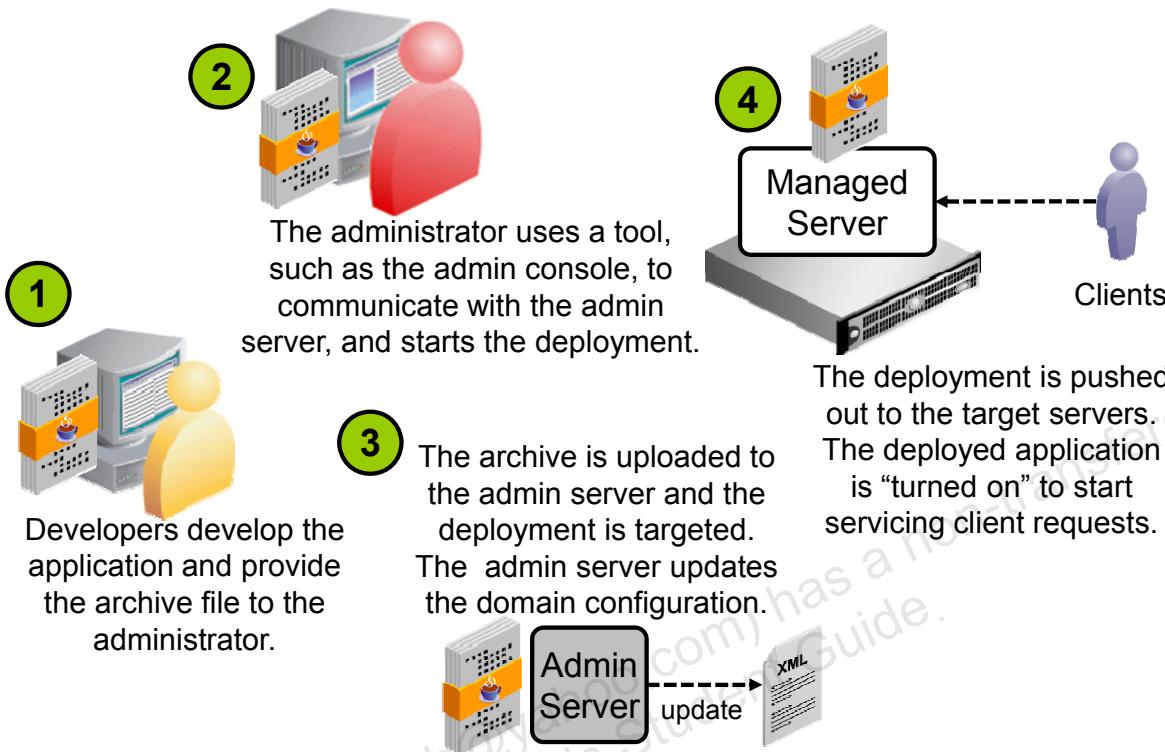
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Deploy an application
- Test a deployed application
- Monitor a deployed application
- Load test an application

Deploying Applications to WebLogic Server

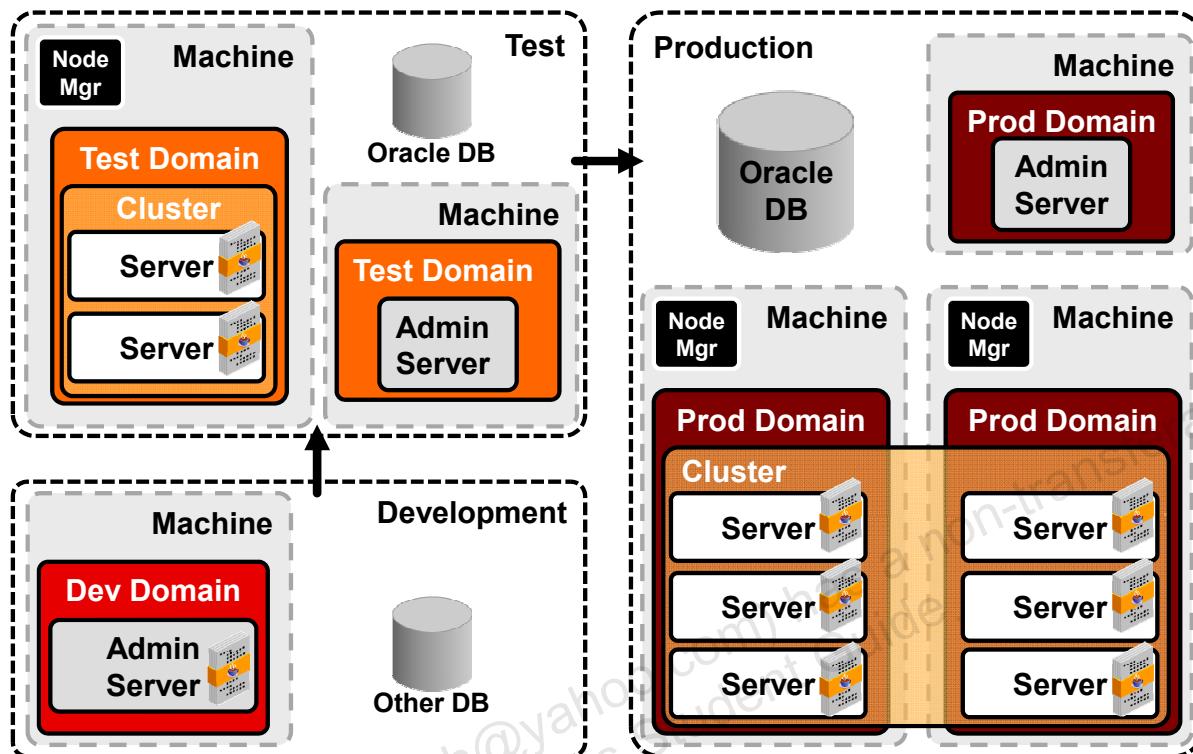


ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

1. First, an application must be developed, tested, and packaged (usually as an application archive).
2. The application archive file is placed where an administrator in charge of deployment has access to it. A deployment tool, such as the administration console, is used to communicate with the administration server of the domain, which is in charge of any configuration changes, including application deployment.
3. The deployment tool gives the administration server access to the application, and allows the deployment administrator to target a server (or servers) on which to run the application. The administration server updates the domain's configuration.
4. The administration server pushes the application's code out to the target server (or servers). After the application is “activated” (told to start servicing requests), it can be accessed by clients.

Software Life Cycle and WebLogic Server



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Applications are created by developers, who are responsible for unit-level testing. After the developers are satisfied with individual components, those components are combined and moved into a test environment. The system is tested there by a quality organization. After the application has passed the system-level quality tests, it is moved into production.

With WebLogic Server, each environment has its own domain:

- A development-time domain often defines only the required administration server. The portion of the application being developed by a single developer is deployed to the admin server, which is running on a local machine. The database used may be from a different vendor than the production database. No managed servers or clusters are created or tested. Completed development elements are combined on a shared machine, almost always by using a version control system. After all development elements are completed and put together on the shared machine, the system is moved to the test environment.
- A test domain mimics production as closely as possible, although on a smaller scale. Rather than have as much hardware or as many instances of WebLogic Server, it defines a small cluster, perhaps all servers running on one machine. The database is the same vendor as production, but contains a much smaller amount of data than its production counterpart. Test users and groups are created that differ from actual production users and groups.

Java EE Deployments

- Java Platform, Enterprise Edition deployment units:

| Java EE Deployment | Archive Name | Archive File Extension |
|-----------------------------|------------------------|------------------------|
| Web application | Web archive | .war |
| Enterprise JavaBeans (EJBs) | EJB Java archive | .jar |
| Web service | Web archive or EJB JAR | .war / .jar |
| Resource adapter | Resource archive | .rar |
| Optional package | Java archive | .jar |
| Enterprise application | Enterprise archive | .ear |

A collection of web applications, EJBs, and resource adapters



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A web application can contain static files (such as images and HTML pages) as well as JavaServer Pages (JSPs), Servlets, and other Java code.

EJBs are the Java Enterprise Edition component architecture. They are often used as containers of custom business logic for middleware.

Web services are part of service-oriented architecture (SOA) and are deployed either as a web application or as an EJB JAR file, depending upon how the web service was implemented.

Resource adapters implement the Java EE Connector Architecture to allow Java EE components to access Enterprise Information Systems. Examples of Enterprise Information Systems include enterprise resource planning systems or mainframe transaction processing systems.

An optional package is the Java Enterprise Edition unit for sharing code. It allows you to share the code in a Java Archive (JAR) file. An optional package can be referenced from any Java Enterprise Edition module.

An enterprise application is a collection of web applications, EJBs, and resource adapters. It makes deployment easier by allowing you to deploy one element that contains many separate entities.

WebLogic Server Deployments

- WebLogic-specific deployment units:
 - Enterprise applications, web applications, or Java archives as shared libraries
 - JMS and JDBC modules

| WLS Deployment | Archive Name | File Extension |
|------------------------|--|--------------------|
| Shared library | Enterprise application, web application, or Java archive | .ear / .war / .jar |
| JMS module/JDBC module | XML file | .xml |

These can be included in an enterprise application.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Oracle WebLogic Server has some deployment units of its own. WebLogic Server has a deployable unit, very similar to a Java optional package, called a shared library. A shared library is a web application, an enterprise application, or a JAR file that is deployed so that web applications and enterprise applications can share its code.

You can deploy JMS and JDBC modules to define JMS and JDBC resources. These modules can also be packaged within an enterprise application.

There is another type of WebLogic-specific module not listed here, the WLDF module. These modules contain WebLogic Diagnostic Framework components, such as data harvesters.

Other Deployments

| Product | Deployment | Archive Name | Archive File Extension |
|--|--|---|--|
| Oracle Application Development Framework (ADF) | ADF application | Enterprise application | .ear |
| Oracle SOA Suite | SOA Suite composite application (SAR stands for SOA archive) | Single composite application or multiple composite applications | Single: .jar or .sar Multiple: .zip |
| Oracle WebCenter | WebCenter application | Enterprise application | .ear |
| Oracle Coherence | Coherence artifacts | Grid archive | .gar |



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Oracle Application Development Framework (ADF) is an end-to-end Java EE framework built on JavaServer Faces (JSF). JSF is a user interface framework for Java EE. ADF applications can be deployed by using the development environment, JDeveloper, or can be placed in an enterprise application archive (EAR) and deployed with WLST or the admin console.

Oracle SOA Suite composite applications can also be “deployed” by using JDeveloper, or can be placed in a JAR file or a SAR file (for a single composite application) or a ZIP file (for multiple composite applications) and deployed by using WLST. A SAR (SOA archive) file is a special JAR file that requires the prefix of `sca_`. Note that SOA composite applications are not Java EE applications, but rather instructions “registered” with the SOA Suite infrastructure. Therefore, “deploying” a SOA composite application does not change the domain configuration, nor is the administration server involved. Oracle Service Bus (OSB) resources, such as a proxy service, are similar, in that they do not modify the domain’s configuration. (However, unlike SOA composite application deployment, OSB resource changes are accomplished through the administration server.)

Oracle WebCenter applications are deployed as enterprise application archives (EARs), but the EAR must be specially prepared, and the target environment must contain WebCenter Shared Libraries. Also the Metadata Services (MDS) repository must be created and registered. MDS is a repository for Fusion Middleware Component metadata.

Coherence is integrated within WebLogic Server as a container subsystem. Like other Java EE containers, the Coherence container supports its own application module, which is called a Grid Archive (GAR). The GAR contains the artifacts of a Coherence application and includes a deployment descriptor. Coherence is typically set up in tiers. A proxy server tier should be set up by using Coherence*Extend. An HTTP session tier should be set up by using Coherence*Web. WebLogic managed servers that are associated with a Coherence cluster are referred to as managed Coherence servers. Managed Coherence servers in each tier can be individually managed, but are typically associated with respective WebLogic Server clusters. A GAR must be deployed to each data and proxy tier server. The same GAR is then packaged within an EAR and deployed to each application server and client tier server.

Deployment Terms

- **Deploy:**
 1. Developers provide application files, usually as an archive, to an administrator, who moves the file to a desired location.
 2. An administrator adds the application to the domain configuration and target servers (or clusters). The deployment is distributed to those servers.
 3. An administrator starts the application (so the application starts servicing requests).
- **Undeploy:**
 1. An administrator stops the application (making it unavailable to clients).
 2. An administrator removes the application from the configuration.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Deploying an application:

1. Developers package tested application files, usually in an archive. They provide the application to an administrator in charge of deployment. The administrator moves the file to a location from which it can be accessed by the deployment tool.
2. The deployment administrator uses a tool that adds the deployment to the domain configuration. Part of deploying is choosing deployment targets, either servers or clusters. As part of this process, the deployment is distributed to the targeted servers.
3. An administrator (the one that deployed the application, or another) starts the application. Normally the application starts servicing all requests (requests from clients). It is also possible, if the domain has an administration port, to start the application so that it only services administration requests (also called starting the application in administration mode). This allows administrators to test the application before it is made available to clients.

Undeploying an application:

1. An administrator uses a tool to stop the application (making it no longer accessible to clients).
2. An administrator uses a tool to remove the deployment from the configuration. Note that this does not mean that the deployment files (or the archive file) are deleted.

Deployment Terms

- **Redeploy:**
 1. Developers provide a new version of the application.
 2. An administrator copies over the deployment files (or archive file) of the deployed application with the new version.
 3. An administrator deploys the new version of the application.
 - Note that with redeployment there is no explicit “start the application” step, because the application is automatically started.
 - WebLogic Server has a strategy called “Production Redeployment” that allows both versions of the application to be active simultaneously.
- **Distribute:** Similar to **deploy**, but the application is not started. The application is pushed out to its targets, ready to be started later.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Redeploying an Application:

1. Developers create and test a new version of the application. Once tested, they provide it to an administrator in charge of deployment.
2. The administrator copies over the deployment files (or archive file) with the new version of the application.
3. The administrator uses a tool to redeploy the application, so that the new version is now the version used.
 - Note that with redeployment, the application does not have to be explicitly started, it starts servicing requests automatically.
 - WebLogic Server has a redeployment strategy called “production redeployment.” Using this strategy, an application has an explicit version number. When a versioned application is redeployed, the old version of the application remains active, if any clients are using it. The new version of the application is also active and is used by all new clients. Eventually, as existing clients finish using the old version, it is retired and all clients are using the new version.

Distributing an application prepares it for deployment by copying its deployment files to all target servers and validating it. This is the same as deploying without starting the application.

Deployment Descriptors

- **Deployment descriptor:** An XML file packaged within a deployment that sets properties of the application
 - Each Java EE deployment has both a standard and a WebLogic-specific deployment descriptor.
 - Since Java EE 5, deployment descriptors can be replaced with annotations in the application code.

web.xml

```
...  
<web-app ...>  
  <servlet>  
    <servlet-name>  
      BenefitsServlet  
    </servlet-name>  
    <servlet-class>  
      stcurr.Benefi  
    </servlet-class>  
  ...
```

weblogic.xml

```
...  
<weblogic-web-app ...>  
  <session-descriptor>  
    <cookie-path>  
      benefits  
    </cookie-path>  
    <descriptor-root>  
      ...
```

Developers are responsible for creating the deployment descriptors (or code annotations) as part of application development.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Each Java EE deployment unit has a standard and a WebLogic-specific deployment descriptor:

- **Web application:** `web.xml` and `weblogic.xml`
- **Web service:** `webservices.xml` and `weblogic-webservices.xml`
- **Enterprise JavaBean:** `ejb-jar.xml` and `weblogic-ejb-jar.xml`
- **Resource adapter:** `ra.xml` and `weblogic-ra.xml`
- **Enterprise application:** `application.xml` and `weblogic-application.xml`

Deployment Plans

- A WebLogic Server deployment plan is an XML document that can override an application's configuration (deployment descriptors). A deployment plan is:
 - Optional
 - Used to update deployment descriptor values
 - Useful when moving an application from one environment to another (such as from development to test or test to production)
 - A separate file, outside of the deployment archive `plan.xml`

```
<?xml version='1.0' encoding='UTF-8'?>
<deployment-plan ... >
  <application-name>timeoff.war</application-name>
  <variable-definition>
    <variable> ...
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

For more information about deployment plans, see the section “Understanding WebLogic Server Deployment Plans” in the *Administering Server Environments for Oracle WebLogic Server* document.

Deployment plans are covered in the *Oracle WebLogic Server 12c: Administration II* course.

Exploded Versus Archived Applications

- An application can be deployed as a set of directories and files. This is called an “exploded directory” application.
- An application file and directory structure can be placed within the appropriate archive file, and that single file can be deployed.
- Exploded directory applications are most often used during development, and archived applications during test and production.
 - There is nothing that prevents exploded application deployments in test and production or archive file deployments during development, however.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Autodeploy

- A development mode domain can automatically deploy applications:
 1. Place the application's exploded directories and files or archive file in the domain's `autodeploy` directory.
 2. The administration server watches that directory. When it detects a new application, it automatically:
 - A. Adds the application to the configuration
 - B. Targets the application to itself, the admin server
 - C. Starts the application (the application starts servicing requests)
- Autodeploy is a convenient feature for developers, which allows them to quickly deploy and test an application.
- Autodeploy is disabled in a production mode domain.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Note that autodeployment works to redeploy an application, too. When you copy over an application archive file in the `autodeploy` directory, the administration server notices the new timestamp and automatically redeploys the application. To achieve the same automatic redeployment of an exploded directory application, create a file called `REDEPLOY` and place it in the `WEB-INF` or `META-INF` directory of the application. When that file's timestamp changes, the administration server knows to redeploy the application.

Server Staging Mode

The Staging Mode of a server determines how the server accesses deployed applications:

- **stage**: During the deployment process the application is pushed out to the remote server's Staging Directory.
- **nostage**: The server accesses the application from an accessible location (often shared storage). You must ensure the application is placed in this location. The location is specified during deployment.
- **external_stage**: Similar to `stage`, except the application must be copied into the Staging Directory before beginning the deployment process (manually or by some other tool).

The Staging Mode's default is `stage`.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Staging Mode of a server is found in the admin console under the server's **Configuration > Deployment** tab.

The default Staging Directory is the `stage` directory under the server's directory under the domain. This can be modified, if desired.

When deploying an application, the Staging Mode of the target servers can be overridden.

WebLogic Server Deployment Tools

The following tools can be used to deploy applications:

- Administration console
- WLST
 - Can be used interactively or to run a script
 - Example:

```
deploy('app', '/apps/app.ear', targets='cluster1')
```

- The `weblogic.Deployer` class
 - For command-line deployment
 - Example:

```
java weblogic.Deployer
    -adminurl http://host01.example.com:7001
    -username weblogic -password Welcome1
    -deploy -source /apps/app.ear -targets cluster1
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Deploying by using the administration console is shown later in this lesson.

The WebLogic Scripting Tool (WLST) has many commands related to the deployment of applications:

- `deploy()`: Deploys an application, targets it, distributes it, and starts it
- `redeploy()`: Redeploys a previously deployed application
- `distributeApplication()`: Distributes the application to the targets, but does not start it
- `startApplication()`: Starts an application, so it can service requests
- `stopApplication()`: Stops an application from servicing requests
- And more

To use the `weblogic.Deployer` class, ensure that the Java Virtual Machine executable is in the PATH and the WebLogic Server classes have been added to CLASSPATH. The latter can be accomplished by running the `setWLSEnv.sh` script found in the `<WL_HOME>/server/bin` directory. For more information about the `weblogic.Deployer` class, see the chapter titled “Deploying Applications and Modules with `weblogic.Deployer`” in the *Deploying Applications to Oracle WebLogic Server* document.

WebLogic Server Deployment Tools

The following tools can be used to deploy applications:

- Ant
 - It is an Apache open-source, Java-based “make utility”
 - Ant and WebLogic-specific Ant tasks come with the WebLogic Server installation.
 - The `wldeploy` task is the Ant version of the `weblogic.Deployer` utility.
- Maven
 - It is an Apache open-source tool for managing Java projects.
 - A plug-in is available with the WebLogic Server installation.
- Enterprise Manager Cloud Control
 - Cloud Control can deploy, undeploy, and redeploy applications to WebLogic Server.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Apache Ant is an open-source Java library designed to build projects. WebLogic Server comes with an Ant installation, and provides custom Ant tasks. To use Ant, ensure that the Ant executable (or batch file) is in the PATH and the WebLogic Server classes have been added to the CLASSPATH. The Ant executable (for Linux) or batch file (for Windows) can be found here: `<WL_HOME>/oracle_common/modules/org.apache.ant_1.7.1/bin`. (Note that the exact directory name may change with other releases of WebLogic Server.) Setting the CLASSPATH can be accomplished by running the `setWLSEnv.sh` script found in the `<WL_HOME>/server/bin` directory.

Apache Maven is an open-source software management tool. WebLogic Server comes with two Maven plug-ins. One of them came out with WebLogic Server 11g and is essentially the Maven version of the `weblogic.Deployer` class. The second plug-in is new with WebLogic Server 12c and provides enhanced functionality to not only deploy applications, but also start and stop servers, create domains, run WLST scripts, and more. For how to configure and use either Maven plug-in for deployment, see the appendix titled “Using the WebLogic Maven Plug-In for Deployment” in the *Deploying Applications to Oracle WebLogic Server* document.

Enterprise Manager Cloud Control is part of Oracle Enterprise Manager, Oracle’s integrated enterprise cloud management product line. Cloud Control enables you to monitor and manage the complete Oracle IT infrastructure from a single console.

Starting and Stopping an Application

- Newly deployed applications must be started. There are two possible “start levels:”
 - Start servicing all requests: This gives regular users access to the application.
 - Start servicing only administration requests: The application is available only to administrators through the domain administration port.

A special port set up for inter-server communication
- Applications must be stopped before they are undeployed.
 - When work completes: Allows current users of the application to complete their work and disconnect
 - Force stop now: Stops the application immediately, whether or not it is being used



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

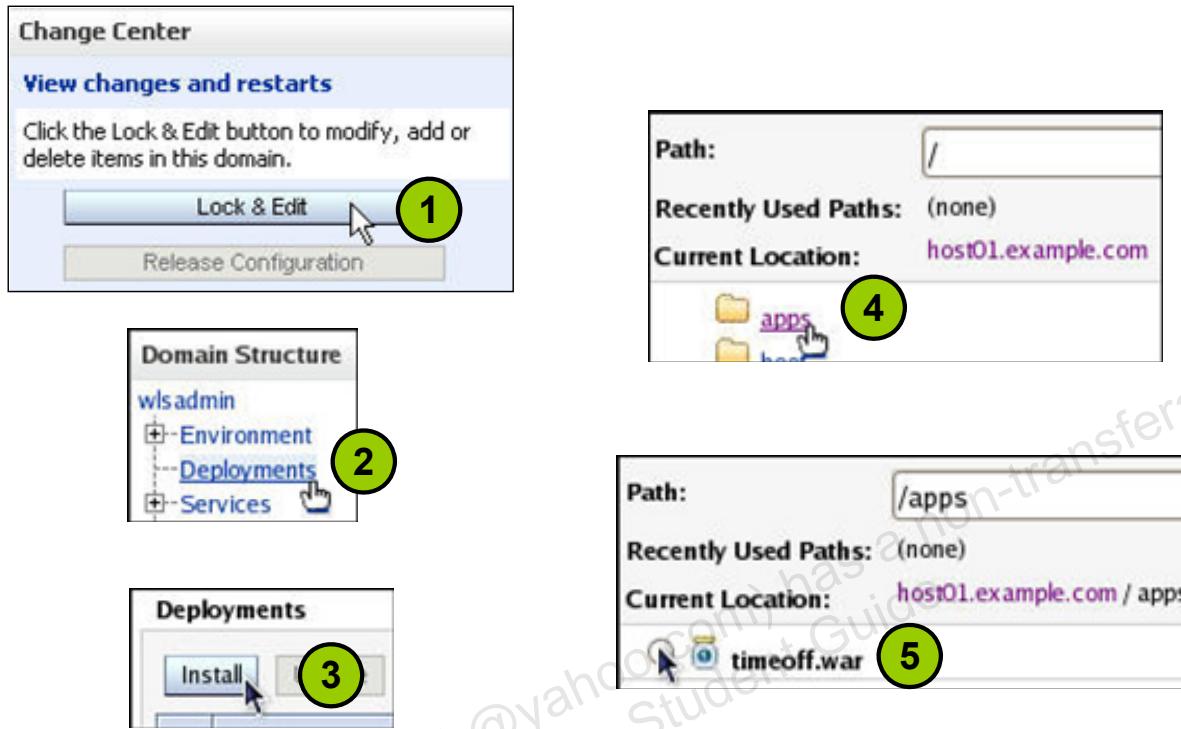
When you start an application, you have two options:

- **Servicing all requests:** WebLogic Server makes the application immediately available to all clients.
- **Servicing only administration requests:** WebLogic Server makes the application available in administration mode only. This means that administrators have access to the application through the target servers’ administration port. The administration port is a domain option. (In the admin console, select the domain and then click the **Configuration > General** tab. Select the **Enable Administration Port** check box. Then enter the Administration Port number.) The administration port uses SSL, so enabling it requires that SSL is configured for all servers in the domain. After the administration port is configured, all communication between the administration server and managed servers is through this secure port. Also, the administration console can be reached only through this secure port. For more information about the administration port, see the “Configuring Network Resources” chapter in the *Administering Server Environments for Oracle WebLogic Server* document.

When you stop an application, there are three options:

- **When work completes:** WebLogic Server waits for the application to finish its work and for all currently connected users to disconnect. It also waits for all HTTP sessions to time out.
- **Force stop now:** WebLogic Server stops the application immediately, regardless of the work that is being performed and the users that are connected.
- **Stop, but continue servicing administration requests:** WebLogic Server stops the application after all its work has finished, but then puts the application in administration mode. Note that the application has not actually stopped, but has been placed in administration mode. That means that the application cannot be undeployed at this time.

Deploying an Application

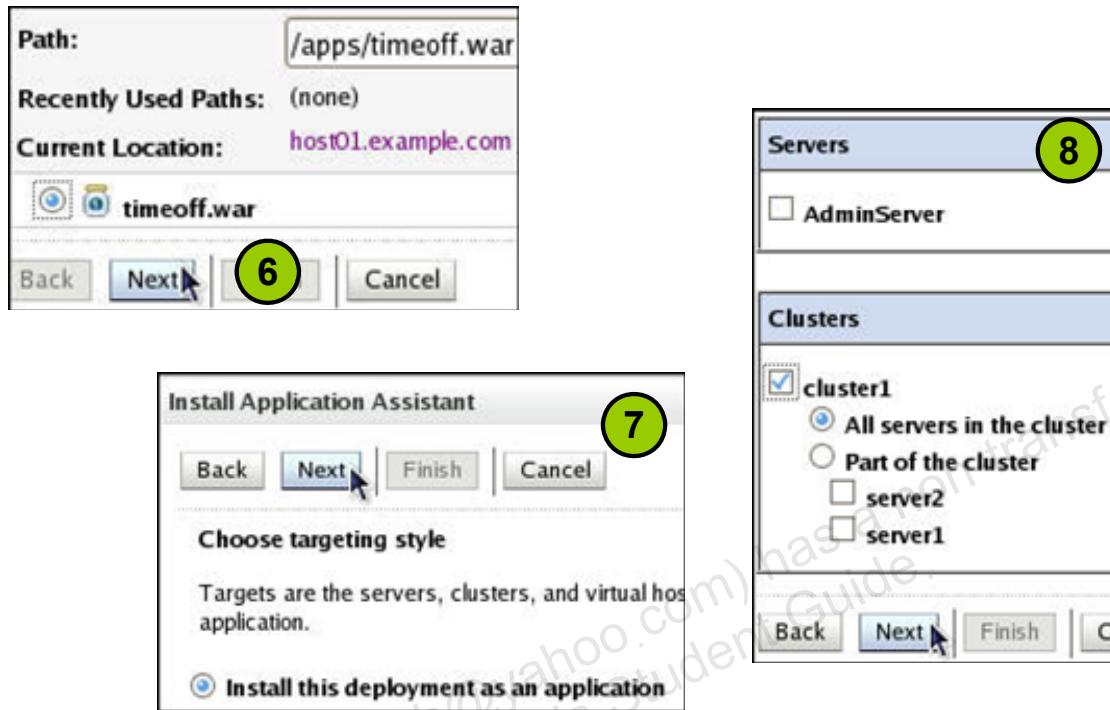


ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

1. Lock the configuration.
2. In the Domain Structure, select **Deployments**.
3. Above the Deployments table, click the **Install** button.
4. Enter the path and name of the deployment in the Path field, or use the links to navigate to the location of the deployment.
5. If using the links to navigate to the deployment, after you have found a deployable unit, an option button appears next to it. Select the deployment.

Deploying an Application

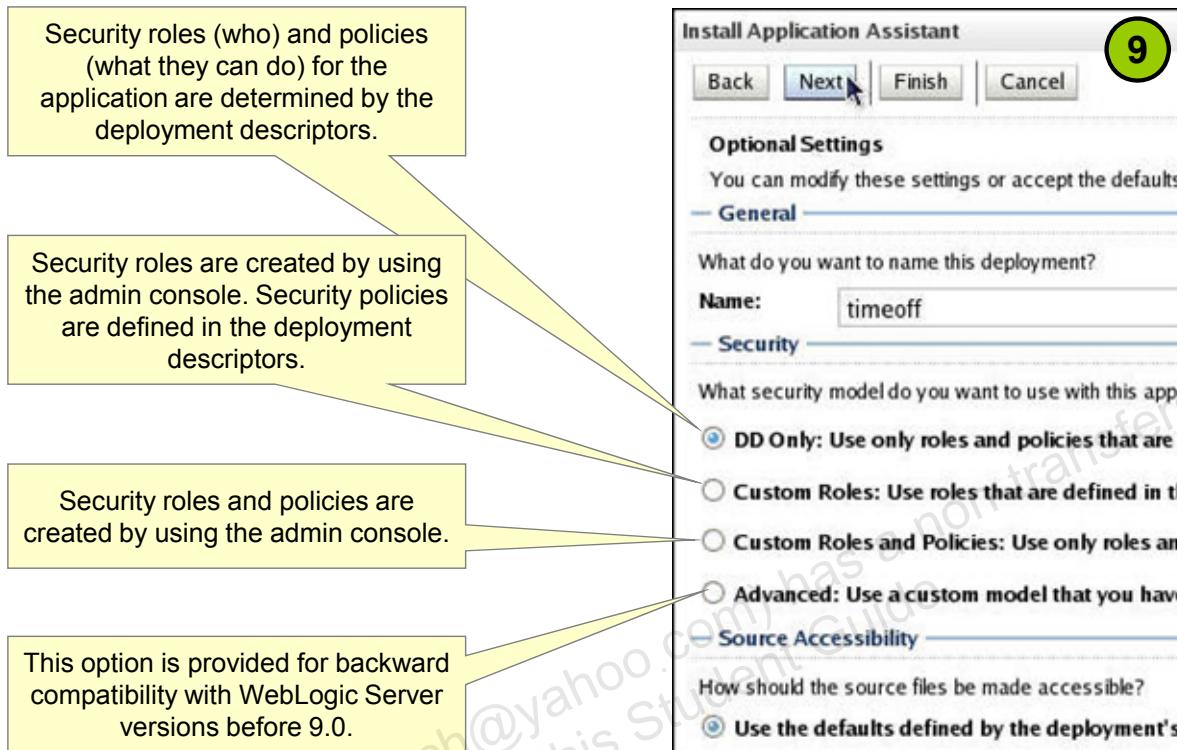


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

6. After the deployment is selected, click **Next**.
7. Select **Install this deployment as an application** and click **Next**. (The other option is to install the application as a library.)
8. Select the targets and click **Next**.

Deploying an Application



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

9. This page of the wizard contains optional configuration items. If the defaults are acceptable, click **Next**. Otherwise, enter or select the following, and then click **Next**:

- **Name:** The name of the deployment as it is displayed in the admin console
- **Security model** (There is more information on security roles and policies in the lesson titled “WebLogic Server Security.”):
 - **DD Only:** Security roles (who) and policies (what they are allowed to do) come from the application deployment descriptors. This is the default and usually the option you want.
 - **Custom Roles:** Security roles are created by using the admin console, but policies come from the deployment descriptors.
 - **Custom Roles and Policies:** Ignore the deployment descriptors. Roles and policies are created by using the admin console.
 - **Advanced:** It is provided for backward compatibility with earlier versions of WebLogic Server (prior to 9.0).

- **Source accessibility:**
 - Use the defaults defined by the deployment's targets: This is the default and the recommended option.
 - Copy this application onto every target for me: This ignores the Staging Mode defined by the targets and does a copy to each target's Staging Directory. This is the same as each server's Staging Mode being set to stage.
 - I will make the deployment accessible from the following location: This option indicates that the deployment will be placed in a central location that is accessible to all targets. You must ensure the deployment is copied to that location. This is for the nostage Staging Mode. If this option is selected, the Location (that all targets can reach) must be entered.
- **Plan source accessibility (for the deployment plan, which is optional):**
 - Use the same accessibility as the application.
 - Copy this plan onto every target for me.
 - Do not copy this plan to targets.

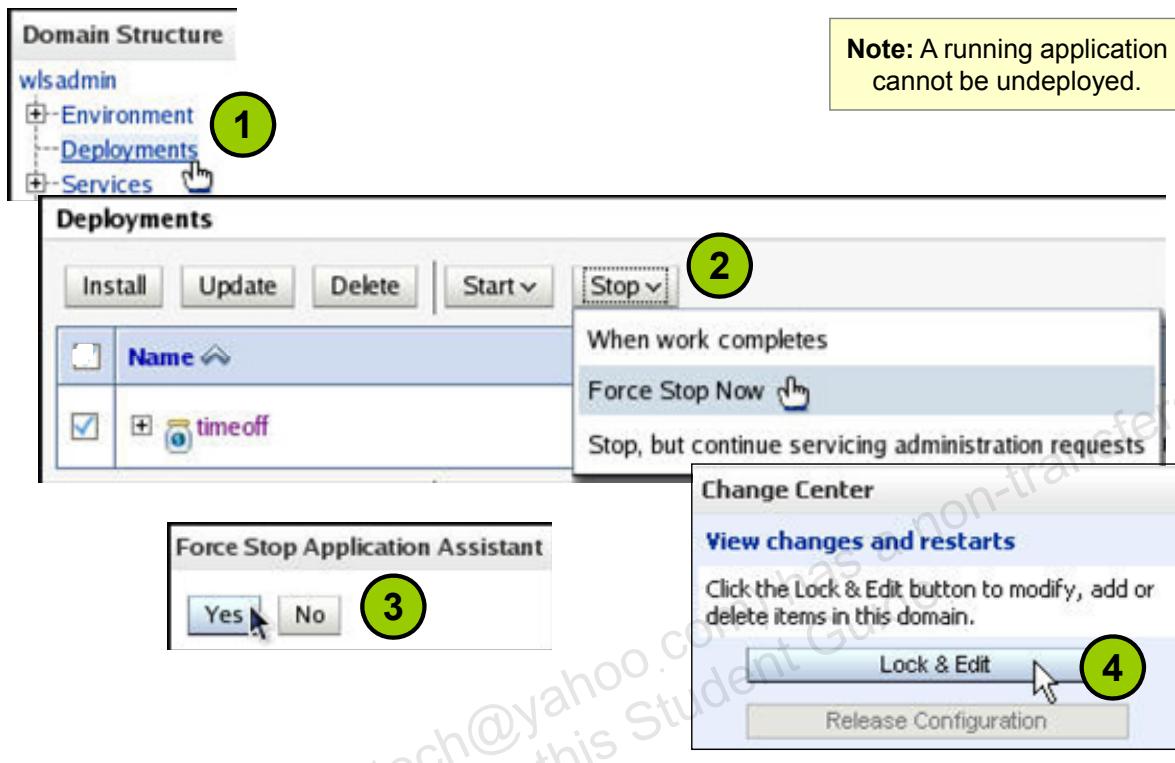
Deploying an Application

The screenshot shows the Oracle WebLogic Server Administration Console. On the left, the 'Install Application Assistant' window has a 'Finish' button highlighted with a green circle labeled '10'. Below it, a message says 'Review your choices and click Finish' and 'Click Finish to complete the deployment. This may take a few minutes.' A radio button for 'Additional configuration' is selected. On the right, the 'Change Center' window shows a message 'Pending changes exist. They must be activated to take effect.' with a 'Activate Changes' button highlighted with a green circle labeled '11'. Below it is an 'Undo All Changes' button. At the bottom, a 'Deployments' table lists two deployments: 'Name' (unchecked) and 'timeoff' (checked). The 'timeoff' row has a dropdown menu open with options 'Servicing all requests' (selected) and 'Servicing only administration requests'. The Oracle logo is at the bottom right.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

10. Choose whether to view the deployment configuration screen next or not, review the deployment, and click **Finish**.
11. In the Change Center, click **Activate Changes**.
12. To start the application, select the new deployment in the Deployments table. Click the **Start** button, and select **Servicing all requests**. When asked to confirm the application start, click **Yes**. The State of the application is changed to "Active."

Undeploying an Application



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

1. In the Domain Structure, select **Deployments**.
2. Select the deployment. Click the **Stop** button and select either **When work completes** or **Force Stop Now**. A running application cannot be undeployed. Stopping an application when work completes can take quite a while. For example, with a web application, it will be at least as long as the session timeout.
3. When asked if you are sure, click **Yes**.
4. Lock the configuration.

Undeploying an Application

The screenshot shows the Oracle WebLogic Server Administration Console interface. At the top, there is a toolbar with buttons for Install, Update, Delete, Start, and Stop. The 'Delete' button is highlighted with a green circle and labeled '5'. Below the toolbar is a table titled 'Deployments' with columns for Name and State. A row for 'timeoff' is selected, indicated by a checked checkbox and a plus sign icon. The state of the application is 'Prepared'. In the center, a modal dialog titled 'Delete Application Assistant' has two buttons: 'Yes' and 'No', with 'Yes' highlighted with a green circle and labeled '6'. To the right, a 'Change Center' window is open, showing a message: 'Pending changes exist. They must be activated to take effect.' It contains two buttons: 'Activate Changes' (highlighted with a green circle and labeled '7') and 'Undo All Changes'. A note in a yellow box states: 'Note that although the deployment is no longer in the configuration, the deployment files (or the archive file) have not been deleted.'

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

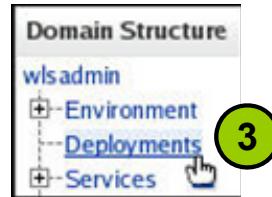
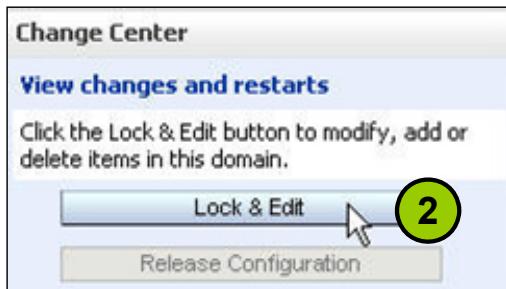
ORACLE

5. Now that the application is stopped and the configuration is locked, select the application in the Deployments table again. Then click the **Delete** button.
6. When asked if you are sure, click **Yes**.
7. In the Change Center, click **Activate Changes**.

Redeploying an Application

1

```
$> cp /uploaded/timeoff.war /apps/timeoff.war
```



4

The Deployments table lists one application: 'timeoff' which is 'Active'. The table has columns for Name and State. A green circle containing the number 4 is placed over the 'Update' button in the toolbar.

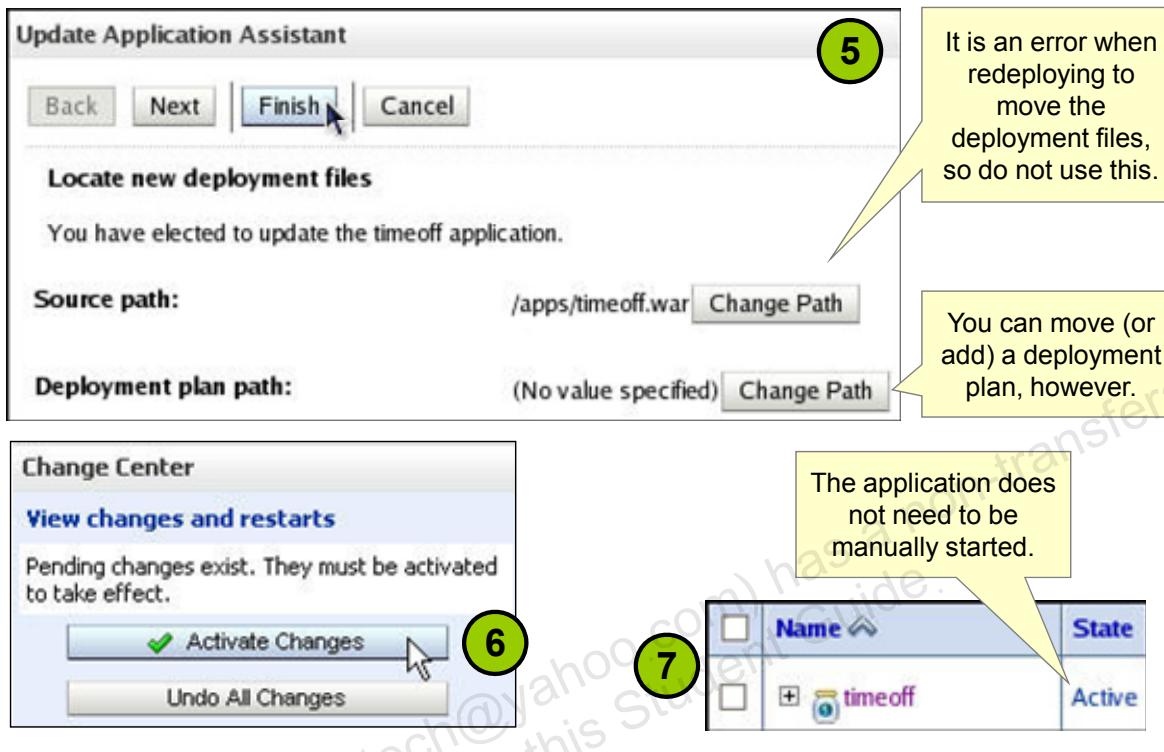
| | Name | State |
|-------------------------------------|---------|--------|
| <input checked="" type="checkbox"/> | timeoff | Active |

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

1. Copy over the current application with the new version.
2. Lock the configuration.
3. In the Domain Structure, select **Deployments**.
4. Select the deployment. Click the **Update** button.

Redeploying an Application



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

5. Click **Finish**. (The **Next** button takes you to a review screen with essentially the same information.)
6. In the Change Center, click **Activate Changes**.
7. The application should automatically be active. (You do not need to explicitly start it.)

Note: Redeploying an application like this could interrupt current users of the application. This in-place redeployment should be used for applications that are first taken offline. To redeploy a new version of an application, while leaving the old version available to current users, consider using Production Redeployment. For more information about Production Redeployment, see the chapter titled "Redeploying Applications in a Production Environment" in the *Deploying Applications to WebLogic Server* document.

Monitoring Deployed Applications: Admin Console

The administration console allows you to monitor applications:

1. Select the name of the deployed application in the Deployments table.
2. Select the **Monitoring** tab.
3. Select the subtab of interest.

| Context Root | Application | Server | Machine | State | Active Server Count | Source Information | Servlets | Sessions |
|--------------|-------------|---------|----------|--------|---------------------|--------------------|----------|----------|
| /timeoff | timeoff | server1 | machinel | Active | 1 | timeoff.war | 5 | 0 |

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The subtabs under Monitoring (for a web application) are:

- **Web Applications:** Information about web applications including the context root of the application, the server on which it is running, the application state, the number of Servlets, the number of active HTTP sessions, and so on
- **Servlets:** A list of the Servlets and the number of times each has been invoked, reloaded, and so on
- **Sessions:** Information about the HTTP sessions
- **PageFlows:** Information about page flows (A page flow is part of the retired Apache Beehive web application framework that acted as a front-end to the Apache Struts framework.)
- **Workload:** Information about work managers, threads, and so on
- **Web Service Clients:** Information about clients of web services that are part of this web application
- **JAX-RS Applications:** Information about the JAX-RS applications that are part of this application. JAX-RS is the Java API for RESTful web services. REST is Representational State Transfer, a software architecture that allows client/server calls based on URLs.

Monitoring Information Available from the Admin Console

| Application Type | Monitoring Information Available |
|------------------------|---|
| Web application | Targeted servers, context root, number of Servlets, the number of times each Servlet has been invoked, average execution time of each Servlet, the number of active HTTP sessions, creation time of each session, work manager and thread information, and more |
| EJB | Targeted servers, total number of EJBs that have been activated, current number of beans in use from the pool, total number of times a bean has been accessed from the cache, current number of beans in the cache, for a message-driven bean if it is connected to its destination, and more |
| Web service | Web service name, number of servers where the service is active, number of service errors, total number of times the service has been invoked, average service response time, and more |
| Enterprise application | Web application, web service, and EJB monitoring as described above, as well as JDBC and JMS module monitoring |

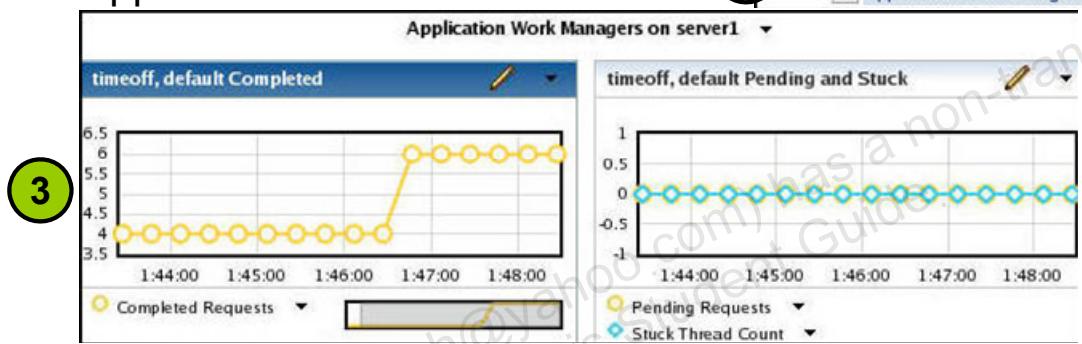
ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

See the admin console online Help for a full list of available information.

Monitoring Deployed Applications: Monitoring Dashboard

1. In the View List, expand the server and select the built-in view called **Application Work Managers on servername**.
2. Click the start button.
3. Find the charts for the application of interest.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The two charts for each application in this built-in view are the completed requests by the application's work manager, and the pending requests/stuck thread count for the application's work manager. In this example, the work manager is the default work manager.

What is a work manager? WebLogic Server uses a self-tuning thread pool. WebLogic Server does prioritize work and allocate threads based on an execution model that takes into account administrator-defined parameters, however. Administrators can configure a set of thread-scheduling guidelines by defining work managers. Work managers are optional, and if not created, the default work manager is used, as in the examples in the slide. Work managers are covered in the *Oracle WebLogic Server 12c: Administration II* course.

Application Errors

- WebLogic Server errors generally show up in the server log as messages from some troubled subsystem.
- Application errors often show in the server log as a Java Exception followed by a stack trace.
 - A stack trace shows all the methods that were interrupted when the error occurred.
 - Stack traces are used by developers to track down their faulty code.

The diagram illustrates a Java stack trace. At the top right is a yellow box labeled "Exception". Below it is a grey rectangular area containing text. The text starts with "####<...> <Error> <HTTP>...Servlet failed with an Exception" and then lists several lines of Java code. Below the grey area is a horizontal bar divided into five yellow boxes, each with a label: "Package", "Class", "Method", "Source file", and "Line number". Arrows point from the "Exception" box down to the stack trace, and from each of the five labels to its corresponding part in the bar below.

```
#####
<...> <Error> <HTTP>...Servlet failed with
an Exception
java.lang.ArrayIndexOutOfBoundsException: 3
at stcurr.BadClass.causeError (BadClass.java:9)
at jsr303_servlet._vision._jspService (/vision.java:7)
...

```

| | | | | | |
|-----|---------|-------|--------|-------------|-------------|
| ... | Package | Class | Method | Source file | Line number |
|-----|---------|-------|--------|-------------|-------------|

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A Java stack trace shows all the methods that were executing when an uncaught exception or error occurred. The stack trace is listed from the last method that was running to the first. Developers look through the trace looking for code they wrote that caused the problem. The error and stack trace are written to the server's log file, as well as system out.

Here is the code that caused the error above (see line 9):

```

01 package stcurr;
02
03 public class BadClass {
04
05     public void causeError() {
06         int[] intArray = {1,2,3};
07         // this loop goes beyond the last array element
08         for (int i=0; i <= intArray.length; i++) {
09             System.out.println(intArray[i]);
...

```

Application Testing

- The administration console displays test URLs based on the application configuration. Select the deployment from the Deployments table, and then click the **Testing** tab.
 - If a link works:
 - The application is deployed and started
 - Remember, however, that this is a minimal test, and it is possible that there are still issues with the application
 - If a link does not work:
 - It could indicate a deployment problem
 - It is also possible that the application is OK and accessed through some other URL

| Settings for benefits | | | |
|-----------------------|-----------------|---|----------|
| Overview | Deployment Plan | Configuration | Security |
| Testing | Monitoring | Notes | |
| Deployment Tests | | | |
| Name | | Test Point | |
| benefits | | http://host02.example.com:7011/benefits | |
| default | | http://host02.example.com:7011/benefits | |

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Performance Testing Methodology

1. Define the expected workload.
2. Define performance objectives.
3. Select the subsystems to study.
4. Create and perform a test to create an initial benchmark (baseline performance data).
5. Modify one system attribute.
6. Perform the test again.
7. Analyze the results.
8. Repeat steps 5–7.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Good system performance depends on good design, good implementation, defined performance objectives, and performance tuning. Performance tuning is an ongoing process. Use tools that provide performance metrics that you can compare against performance objectives. The goal is to meet the performance objectives, not to eliminate all bottlenecks. Resources within a system are finite; therefore, some resource (CPU, memory, or I/O) will be a bottleneck in the system. Tuning allows you to minimize the impact of bottlenecks on your performance objectives.

When testing, ensure that benchmarks are realistic; otherwise, there will be unexpected results when the application goes into production. If an application accesses a database, ensure that it accesses the database during the load or stress test.

Load and Stress Testing

- Load testing measures performance for a system at different levels of concurrent request loads.
- Stress testing measures a system's limits. For example:
 - Extremely high number of concurrent users
 - Extremely high data volume
- Performance testing a WebLogic Server application requires:
 - Measurable performance goals, often stated as Service Level Agreements (SLAs)
 - Example SLA: “The 90th percentile response time for the application under two times a normal load should be within 5 seconds.”
 - Load testing tools that can generate the data needed



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Functional testing verifies that an application demonstrates the correct behavior under certain inputs, whereas load testing determines whether an application can support a specified load (for example, 500 concurrent users) with specified response times. Load testing is used to create benchmarks.

Stress testing is load testing over an extended period of time. Stress testing determines whether an application can meet specified goals for stability and reliability, under a specified load, for a specified time period.

It is often possible to begin a stress-testing plan by taking the existing use cases for the application to be tested. Because use cases provide a description of how the application will typically be used when in production, they can often be translated directly into test scripts that can be run against the application.

Load Testing Tools

Many commercial and open-source load testing tools are available, including:

- The Grinder
- JMeter
- HP LoadRunner
- RadView WebLOAD
- Oracle Load Testing (part of Oracle Application Testing Suite)



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

JMeter is an Apache Jakarta project that can be used as a load-testing tool for analyzing and measuring the performance of a variety of services, with a focus on web applications. JMeter can be used as a unit test tool for JDBC database connections, web services, JMS, and HTTP. JMeter also supports assertions to ensure that the data received is correct.

LoadRunner is a performance and load testing product by Hewlett-Packard for examining system behavior and performance, while generating actual load. Working in LoadRunner involves using three different tools: Virtual User Generator (VuGen), Controller, and Analysis.

RadView WebLOAD is a software for performance testing Internet applications. It consists of three main parts: WebLOAD IDE (the authoring environment), WebLOAD Console (the execution environment), and WebLOAD Analytics (the analysis tool).

Oracle Load Testing is part of Oracle Application Testing Suite. It can load test web applications and web services by simulating thousands of virtual users accessing the application simultaneously. It is deployed to WebLogic Server and comes with a web-based interface that allows you to configure load tests, run tests, and view the results.

The Grinder

- The Grinder:
 - Is an open source load-testing tool based on Java and Python
 - Supports a distributed, agent-based load simulation model
 - Provides a graphical console to manage agents and view results
 - Supports HTTP/S (links, forms, cookies, and so on) but can be extended to support additional protocols
- A Grinder agent:
 - Uses a specific number of processes and worker threads
 - Executes a supplied test script with each thread



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

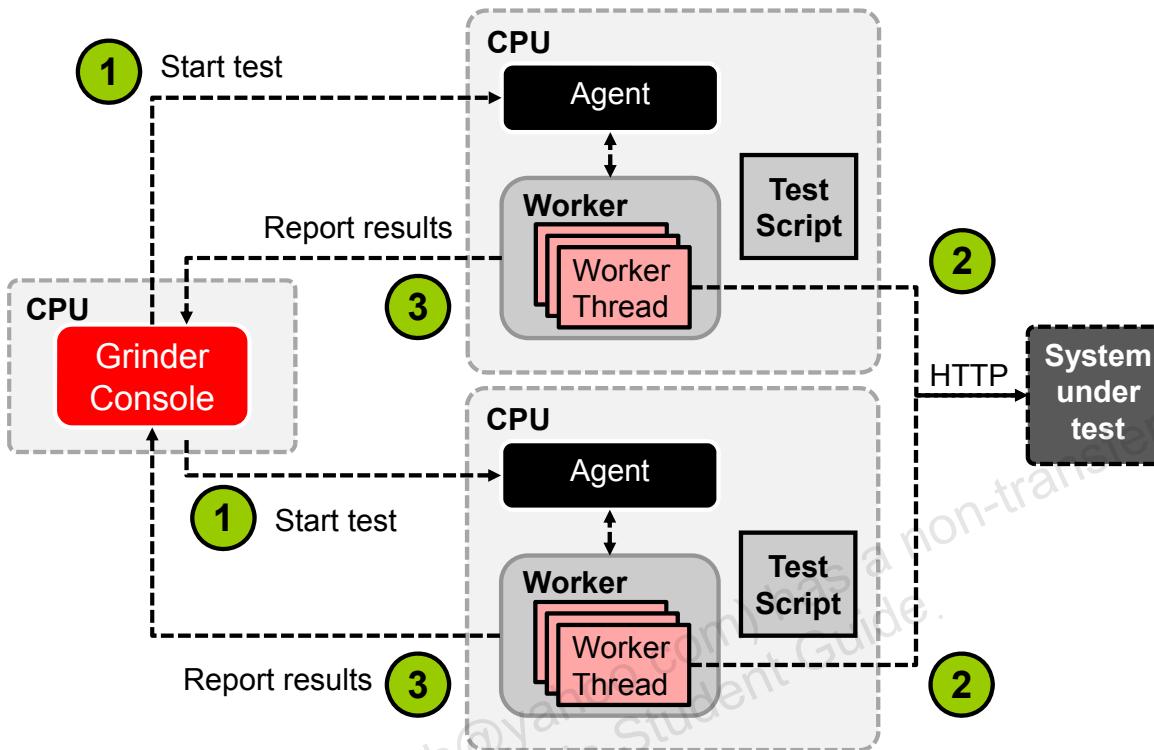
The Grinder is a Java load testing framework that makes it easy to run a distributed test using many “load injector” machines. It is freely available under a BSD-style open-source license, and is based on other open source technologies such as Jython, HttpClient, and XMLBeans.

Each test context runs in its own “worker” thread. The threads can be split over many processes depending on the requirements of the test and the capabilities of the load injector machine. The Grinder makes it easy to coordinate and monitor the activity of processes across a network of many load injector machines from a central console.

Scripts can be created by recording actions of a real user by using the TCPProxy utility. The script can then be customized by hand. Input data (for example, URL parameters or form fields) can be dynamically generated. The source of the data can be flat files, random generation, a database, or previously captured output.

The Grinder has special support for HTTP that automatically handles cookie and connection management for test contexts.

The Grinder Architecture



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

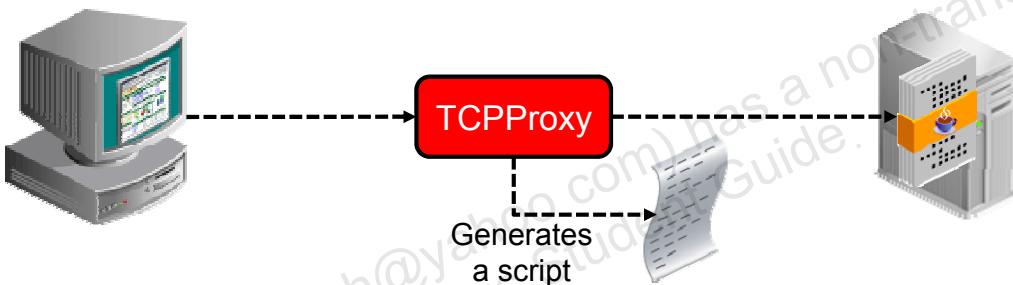
The Grinder is a framework for running test scripts across a number of machines. The framework has three types of processes (or programs): worker processes, agent processes, and the console. Worker processes interpret Jython test scripts and perform tests using a number of worker threads. Agent processes manage worker processes. The console coordinates the other processes, and collates and displays results. Because The Grinder is written in Java, each of these processes run in a JVM.

For heavy duty testing, you start an agent process on each of several “load injector” machines. The worker processes that they launch can be controlled and monitored using the console. There is no reason to run more than one agent on each load injector, but you can. Each worker process sets up a network connection to the console to report statistics. Each agent process sets up a connection to the console to receive commands, which it passes on to its worker processes. The console listens for both types of connections on a particular address and port.

A test is a unit of work against which statistics are recorded. Tests are uniquely defined by a test number and also have a description. Users specify which tests to run by using a Jython test script. The script is executed many times in a typical testing scenario. Each worker process has a number of worker threads, and each worker thread calls the script a number of times. A single execution of a test script is called a “run.”

The Grinder Proxy

- The Grinder tests are Python scripts, which can be:
 - Coded manually
 - Recorded using the TCPProxy
- To use TCPProxy:
 - Configure your web browser to proxy requests through the TCPProxy and then use your web application
 - TCPProxy creates a script that includes all GET and POST requests, cookies, and user “think times”



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The TCPProxy is a proxy process that you can place in a TCP stream, such as the HTTP connection between your web browser and a server. It filters the request and response streams, sending the results to the terminal window (`stdout`). You can control its behavior by specifying different filters. The TCPProxy's main purpose is to automatically generate HTTP test scripts that can be replayed with The Grinder's HTTP plug-in.

The TCPProxy appears to your web browser just like any other HTTP proxy server, and you can use your web browser as you normally would. If you open a web page with your web browser, it displays the page, and the TCPProxy outputs all the HTTP interactions between the web browser and the website. It is important to remember to remove any “bypass proxy server” or “No proxy for” settings that you might have, so that all the traffic flows through the TCPProxy and can be captured.

Having finished your web application run-through, click **Stop** on the TCPProxy console and the generated script is written to the `grinder.py` file. The `grinder.py` file contains headers, and requests. It groups the requests logically into “pages,” for the recorded test script. The script can also be edited manually to suit your needs.

Agent Properties

- Agents are configured by using the `grinder.properties` file, which has settings for:
 - The location of The Grinder console
 - The test script to run
 - The number of worker processes to start
 - The number of threads to start in each worker process
 - The number of times that each thread should run the test
 - Think time adjustments (speed up or slow down)
 - Output and logging levels
- If The Grinder console is not available when an agent is started, it starts running tests immediately.
- Test script files can also be distributed to agents by using the console.

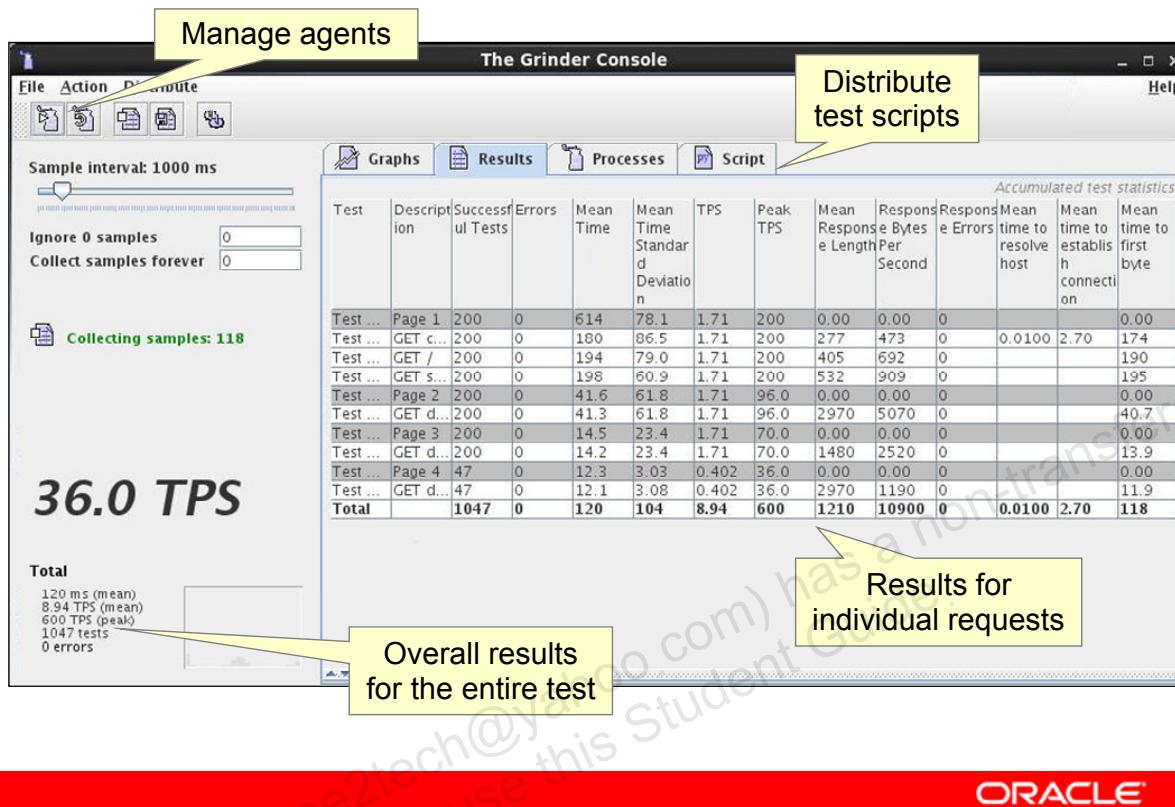


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Grinder worker and agent processes are controlled by setting properties in the `grinder.properties` file. All properties have default values. If you start a Grinder agent process without a `grinder.properties` file, the agent communicates with the console by using default addresses and uses one worker process, one thread, and makes one run through the test script found in the `grinder.py` file. The available properties include:

- `grinder.processes`: The number of worker processes that the agent should start
- `grinder.threads`: The number of worker threads that each worker process spawns
- `grinder.runs`: The number of runs of the test script that each thread performs. A value of 0 means “run forever.” Use this value when you are using the console to control your test runs.
- `grinder.processIncrement`: If set, the agent ramps up the number of worker processes, starting the number specified every `grinder.processesIncrementInterval` milliseconds. The upper limit is set by `grinder.processes`.
- `grinder.duration`: The maximum length of time, in milliseconds, that each worker process should run. The `grinder.duration` attribute can be specified in conjunction with `grinder.runs`, in which case the worker processes are terminated if either the duration time or the number of runs is exceeded.

The Grinder Console



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

The “Start processes,” “Reset processes,” and “Stop processes” menu items send signals to The Grinder processes that are listening. These controls are disabled if no agents are connected to the console. On the **Processes** tab, you can check whether any agents are connected.

The “Start processes” control signals to worker processes that they should move into the running state. Processes that are already running ignore this signal. Processes that are in the finished state exit. The agent process then rereads the properties file, and launches new worker processes in the running state. The “Reset processes” control signals all the worker processes to exit. The agent process then rereads the properties file and launches new worker processes.

The “sample controls” determine how the console captures reports from the worker processes. It is important to understand that these control only the console behavior. They do not adjust the frequency at which the worker processes send reports. The slider controls how the console takes a sample. This involves adding up all the reports received over that sample interval and calculating the Tests per Second (TPS (the number of tests that occurred) / (interval length)). It is also the period at which the console statistics are updated.

Each time the worker processes run, they generate a new set of logs. Logs from previous runs are renamed. The number of logs to keep is set with `grinder.numberOfOldLogs`.

Finding Bottlenecks

- A CPU-bound system cannot process additional workload because the processor is too busy (at or near 100%).
 - Possible causes include:
 - Too frequent garbage collection
 - Excessive memory allocation—resulting in paging
- An I/O bound system's processor is not fully utilized (< 75%). The performance remains the same regardless of the client load.
 - Common culprits include:
 - Accessing remote disks too frequently
 - Database issues
 - Too few connections, poorly written queries
 - Insufficient network bandwidth



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

After the CPU is bound, the application server cannot do any more work. Possible causes of a CPU-bound system include too frequent garbage collection, excessive paging, and poorly designed applications. You can monitor the JVM garbage collection to see whether or not it is a problem. You can also use Java profilers to monitor your applications to see whether they have issues. Typically, profilers show method-level performance (execution time), showing the total execution time and number of invocations for each method. They can also analyze the memory usage of the application.

There are many possible causes of a database bottleneck. Sometimes, the number of connections is too low, and concurrent clients block, waiting for a connection. The solution there is to create and use more database connections. Sometimes, queries take too much time. In this case, the solution may be to create secondary indexes on certain fields. If your database server's machine is too slow, look for better hardware, or move the database to a dedicated machine. If these solutions do not resolve the problem, look into vendor-specific tuning options for your database. Many of the solutions to a database bottleneck reside with the developers. Perhaps they can make more efficient trips to the database by obtaining needed information in one query instead of multiple queries.

The network can be your bottleneck if it gets saturated. Monitor your network to determine how much bandwidth is being used. The easiest fix is to buy more bandwidth.

Correcting Bottlenecks

| Issue | Resolution |
|------------------------------------|--|
| Garbage collection | Try changing JVM garbage collection options, such as the type of collector or the size of the generations. |
| Memory | Try modifying JVM memory arguments. |
| Code performance | Use a Java profiler to find the methods that run most often and take the longest to run. Developers should make those methods more efficient. |
| Web application performance | Precompile JSPs. Ensure that Servlet Reload Check, Resource Reload Check, and JSP Page Check all have the value of -1. Session replication takes resources, developers should use the session sparingly. |
| Database performance | Developers should increase the efficiency of their queries. DBAs should tune the database. |



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Oracle HotSpot JVM has more than one collection algorithm and many options with which you can experiment. See the Oracle documentation as well as whitepapers in the Oracle Technology Network (<http://www.oracle.com/technetwork>).

The lesson titled “Starting Servers” covers setting JVM memory arguments.

The first time a JavaServer Page (JSP) is accessed, it is translated into a class and compiled. This translation and compilation can be done ahead of time, and makes JSPs load faster the first time they are accessed.

Java profilers can monitor applications. Typically, they show the total execution time and number of invocations for each method. They can also analyze the memory usage of the application. The methods that run the most should be made as efficient as possible. The methods that take the longest to run are sometimes poorly written and should be made more efficient, too. Java VisualVM is a profiler that comes with the Oracle HotSpot JDK (<JDK>/bin/jvisualvm).

During development, WebLogic Server periodically (how often is configurable) checks to see whether Servlets, JSPs, and other resources have been updated. If they have, it loads the new versions. Resources are needed to do these checks. The default in production is to set the values of how often to check to -1, which means to never check. Application deployment descriptors can override those defaults, so it is good to ensure these attributes have their production values.

Most web applications store objects in the `HttpSession`. To provide failover, session data is replicated. This takes resources. The larger the session, the more resources the session replication takes. Therefore, developers should store objects in the session sparingly, and try not to store very large objects in the session.

When a system is profiled, often the methods that run the longest are the ones that access the database. Developers should ensure the efficiency of their database queries. They should work with DBAs to ensure that the tables being used are properly indexed.

Quiz

An application must be stopped to be undeployed.

- a. True
- b. False



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a

Quiz

A redeployed application must be manually restarted before it can be used.

- a. True
- b. False



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: b

Summary

In this lesson, you should have learned how to:

- Deploy an application
- Test a deployed application
- Monitor a deployed application
- Load test an application

Practice 10-1 Overview: Deploying an Application

This practice covers the following topics:

- Deploying an application
- Redeploying an application
- Undeploying an application

Practice 10-2 Overview: Load Testing an Application

This practice covers the following topics:

- Using The Grinder to load test WebLogic Server
- Viewing the load test results in The Grinder console

12

Clusters – Overview, Creation, and Configuration

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

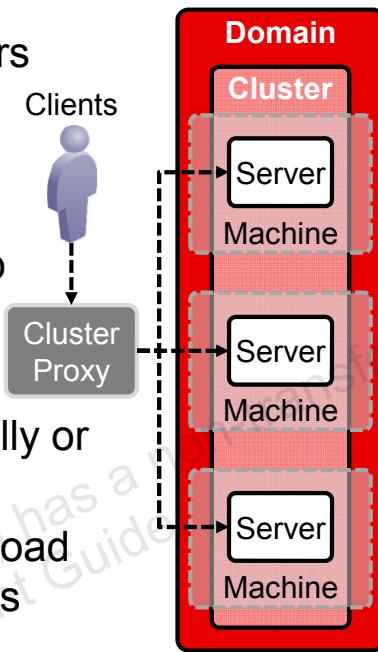
After completing this lesson, you should be able to:

- Describe two cluster architectures: basic and multi-tier
- Create and configure a cluster
- Create and configure a dynamic cluster

Cluster: Review

A cluster:

- Is a logical group of managed servers from the same domain that run cooperatively
- Supports features that provide high availability for web applications, web services, EJBs, and JMS
- Is transparent to its clients
- Can have servers added to it statically or dynamically
- Requires a cluster proxy to provide load balancing, if it hosts web applications



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A WebLogic Server cluster consists of one or more managed servers from the same domain running simultaneously and working together to provide increased reliability and scalability. A cluster appears to clients as one WebLogic Server instance. The server instances that constitute a cluster can run on one machine or on multiple machines.

A cluster achieves high availability through the replication of services. Because of this replication, failover is possible. When one server fails, a second server automatically can resume operation where the first server left off.

Load balancing, the distribution of work across the cluster, ensures that each server in the cluster helps carry the load.

Scalability is achieved because you can increase a cluster's capacity by adding server instances to the cluster, without making any architectural changes.

A cluster also assists in migration. After a system failure on one server, work can be continued by moving the services that server provided to another server in the cluster (service level migration), or by moving the entire server to a new hardware (whole server migration).

After a cluster is created, configured servers can be added to it. A dynamic cluster is based on a server template. A server template sets server attributes. After a server template is assigned to a cluster, servers based on the template are generated and added to the cluster.

Clusters support different types of applications as follows:

- Web applications: Load balancing is achieved through the cluster proxy. This proxy can be a web server using a WebLogic Server proxy plug-in or a hardware load balancer. Failover is achieved by replicating or storing the HTTP session state of clients.
- For Enterprise JavaBeans (EJBs), clustering uses the EJB's replica-aware stub for load balancing and failover. When a client makes a call through a replica-aware stub to a service that fails, the stub detects the failure and retries the call on another replica.
- For JMS applications, clustering supports transparent access to distributed destinations from any member of the cluster.

Benefits of Clustering

| Concept | Description |
|----------------|---|
| Scalability | More capacity for applications can be provided by adding servers, without interruption of service or making architectural changes. |
| Load balancing | Work (for example, client requests) is distributed across the members of a cluster. |
| Failover | When a server fails, another one can automatically take its place. Information on the failed server is replicated (or stored), so that the new server has access to it. |
| Migration | When a server fails, its “pinned” services can continue by moving them to another server in the cluster, or by moving the entire failed server to a new hardware. |

A “pinned” service is a service that must run only on a single instance of WebLogic Server at any given time.

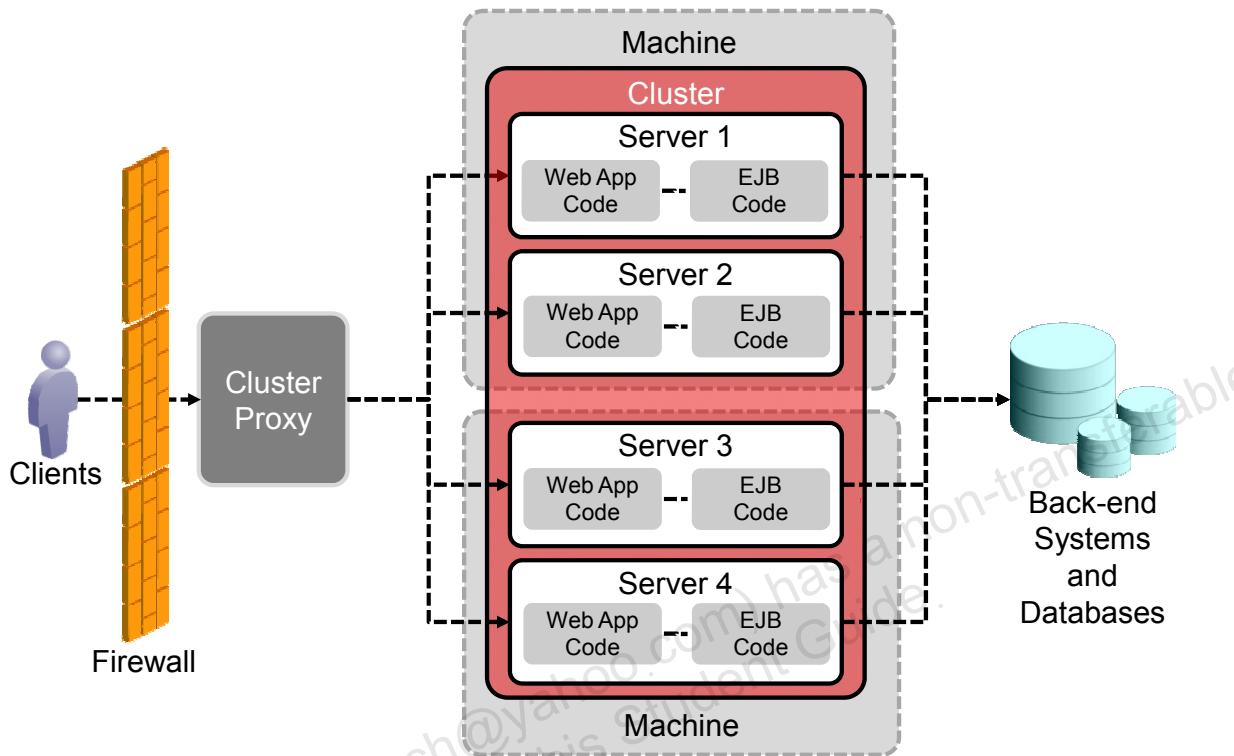


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A WebLogic Server cluster provides the following benefits:

- **Scalability:** The capacity of a cluster is not limited to one server or one machine. Servers can be added to the cluster dynamically to increase capacity. If more hardware is needed, a new server on a new machine can be added.
- **Load Balancing:** The distribution of jobs across the cluster members can be balanced, so no one server is overloaded.
- **Failover:** Distribution of applications and their objects on multiple servers enables easier failover of the session-enabled applications.
- **Availability:** A cluster uses the redundancy of multiple servers to insulate clients from failures. If one server fails, another can take over. With the replication (or storage) of server-specific information, the failover can be transparent to the client.
- **Migration:** This ensures uninterrupted availability of pinned services or components—those that must run only on a single server instance at any given time. An example of a pinned service is the use of the Java Transaction API (JTA).

Basic (Single-Tier) Cluster Architecture



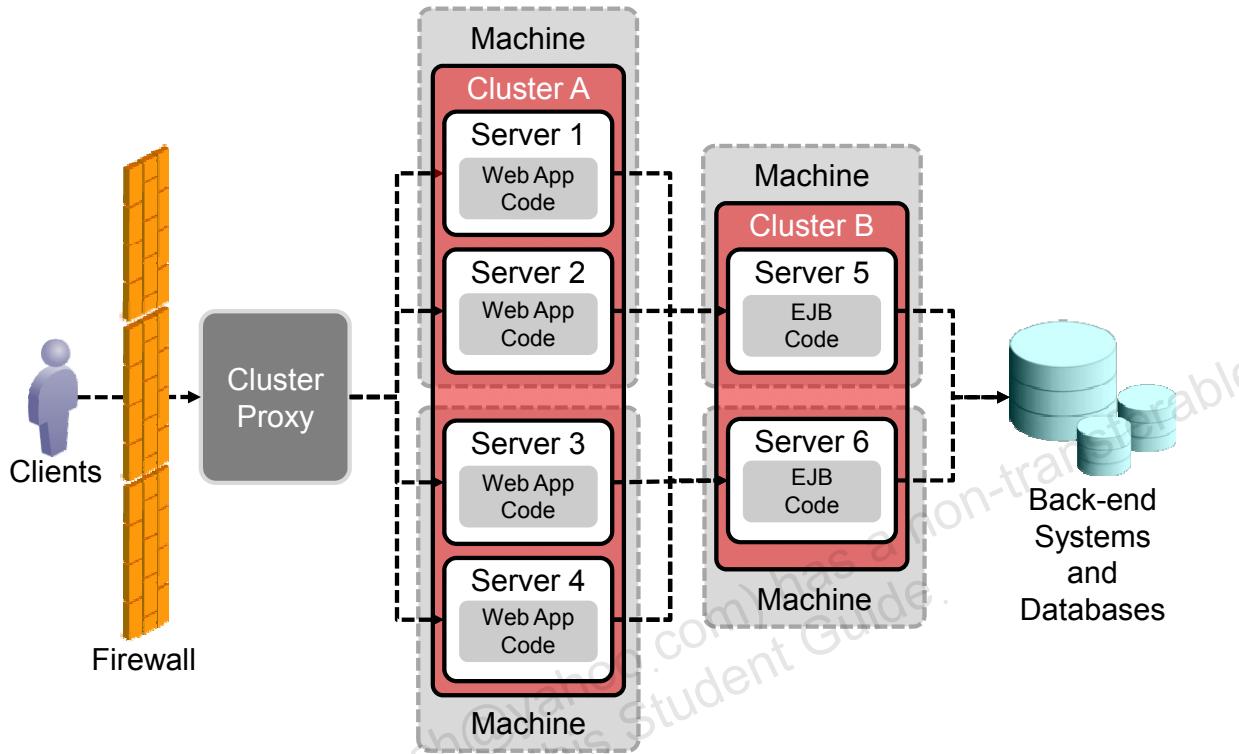
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The basic, single-tier cluster architecture has all WebLogic Server application code in a single tier. That single tier includes both web applications and Enterprise JavaBeans.

Remove the “EJB Code” box for systems that do not use EJBs.

Multi-Tier Cluster Architecture



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In the multi-tier cluster architecture, two separate WebLogic Server clusters are configured:

- Cluster A for the web application tier
- Cluster B to serve clustered EJBs

If your system does not use EJBs, you would not use the multi-tier cluster architecture in this way. You could have a second tier for JMS and use it to load balance JMS calls, however.

Architecture Advantages and Disadvantages

| Cluster Architecture | Advantages | Disadvantages |
|----------------------------|---|--|
| Basic (single-tier) | <ul style="list-style-type: none"> Easier to administer Less network traffic EJB calls are local (and therefore faster) | <ul style="list-style-type: none"> Cannot load balance EJB calls |
| Multi-tier | <ul style="list-style-type: none"> EJB calls are load balanced Scaling options (for example, you can shift (or add) hardware and WebLogic server instances to whichever tier is busier) More security options (for example, you could place a firewall in between the web application tier and the EJB tier) | <ul style="list-style-type: none"> Harder to administer Perhaps more hardware and licensing costs EJB calls are remote (and therefore slower) More network traffic |



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Basic (single-tier) advantages:

- Easier administration:** There is only one cluster to create, configure, and maintain. Also, because one cluster hosts web applications and EJBs, you can easily deploy enterprise applications to the cluster. The web application and EJBs are in the same archive.
- Less network traffic:** Clients access WebLogic Server through web applications. Calls from WebLogic Server to back-end systems and databases still occur, but all calls from the web application tier to EJBs are within the same JVM.
- Faster EJB performance:** Because the calls from the web applications to the EJBs are within the same instance of WebLogic Server, there is no network overhead. The calls to the EJBs are local, not remote. This is especially important if the web applications make frequent EJB calls.

Basic (single-tier) disadvantage:

- EJB calls cannot be load balanced. Each call from the web application tier to an EJB is always to the EJB running on that same instance of WebLogic Server. It is therefore possible that the server load becomes unbalanced. Let us say that 200 concurrent users are accessing your applications, and they have been load balanced so that 50 are accessing each of the four servers in the cluster. It just so happens that the 50 users on server 1 are performing tasks that call EJBs, while the other 150 users on the other servers are not. Server 1 will be exceptionally busy, while servers 2, 3, and 4 will not. If the EJB calls were load balanced to an EJB tier, however, then the “EJB load” would be spread across all of the servers in the EJB tier cluster.

Multi-tier advantages:

- EJB calls are load balanced: Each call to an EJB can be load balanced across all the servers in the EJB cluster. The “unbalanced” situation described above is no longer possible.
- More scaling options: Separating the web application and EJB tiers onto separate clusters provides you with more options for scaling the system and distributing the load. For example, if users spend most of their time using the web applications, and those applications make few EJB calls, you can use a larger number of WebLogic Server instances in the web application cluster. If things change, and your applications become more EJB-intensive, you can shift or add servers to the EJB cluster.
- More security options: With another layer there is an opportunity to add more security. For example, you could place a firewall in between the web application and EJB clusters.

Multi-tier disadvantages:

- More difficult administration: There is more than one cluster to create, configure, and maintain. Also, because one cluster hosts web applications and the other EJBs, deployment becomes more complicated.
- Perhaps higher costs: With two clusters you may have more instances of WebLogic Server and more hardware.
- Slower EJB performance: Because the calls from the web applications to the EJBs are always remote, you must pay the price of remote calls. The applications must be developed with this in mind so that calls to EJBs are infrequent and “course grained.”
- More network traffic: Because all EJB calls are remote, network traffic increases.

Cluster Communication

- Cluster members communicate with each other in two ways:
 - One-to-many messages:
 - For periodic “heartbeats” to indicate continued availability
 - To announce the availability of clustered services
 - **Note:** This communication can use either:
 - IP unicast: No additional configuration is required.
 - IP multicast: A multicast host and port must be configured.
 - Peer-to-peer messages:
 - For replicating HTTP session and stateful session EJB state
 - To access clustered objects that reside on a remote server (multi-tier architecture)
 - **Note:** This communication uses sockets.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

An instance of WebLogic Server uses one-to-many communication to send regular “heartbeat” messages that advertise its continued availability to other server instances in the cluster. The servers in a cluster listen for heartbeat messages to determine when a server has failed.

All servers use one-to-many messages to announce the availability of clustered objects that are deployed or removed locally. Servers monitor these announcements so that they can update their local JNDI tree to indicate the current deployment of clustered objects. This is the maintenance of the so-called “cluster-wide” JNDI tree.

IP multicast enables multiple applications to subscribe to an IP address and port number, and listen for messages. A multicast address is an IP address in the range 224.0.0.0 – 239.255.255.255. IP multicast does not guarantee that messages are received, so WebLogic Server allows for the possibility that some messages may be missed. If you use multicast, you must ensure your network propagates multicast messages to all clustered servers. The multicast time-to-live value can be increased if you find that messages are being missed. With multicast, you must ensure that no other applications share the multicast address and port, or servers will have to process extra messages, which introduces extra overhead.

Firewalls can break multicast transmissions. Although it might be possible to tunnel multicast transmissions through a firewall, this practice is not recommended. A final worry with multicast messaging is the possibility of a multicast “storm,” in which server instances do not process incoming messages in a timely fashion, which leads to retransmissions and increased network traffic.

IP unicast is configured by default because it does not have the network issues of multicast. You can set up a separate network channel for unicast communication, but it is not required. If no separate channel is defined, each server’s default channel is used (the default channel is the server’s configured host and port). There are scenarios where unicast group leaders can become overwhelmed with traffic and switching to multicast communication provides better performance.

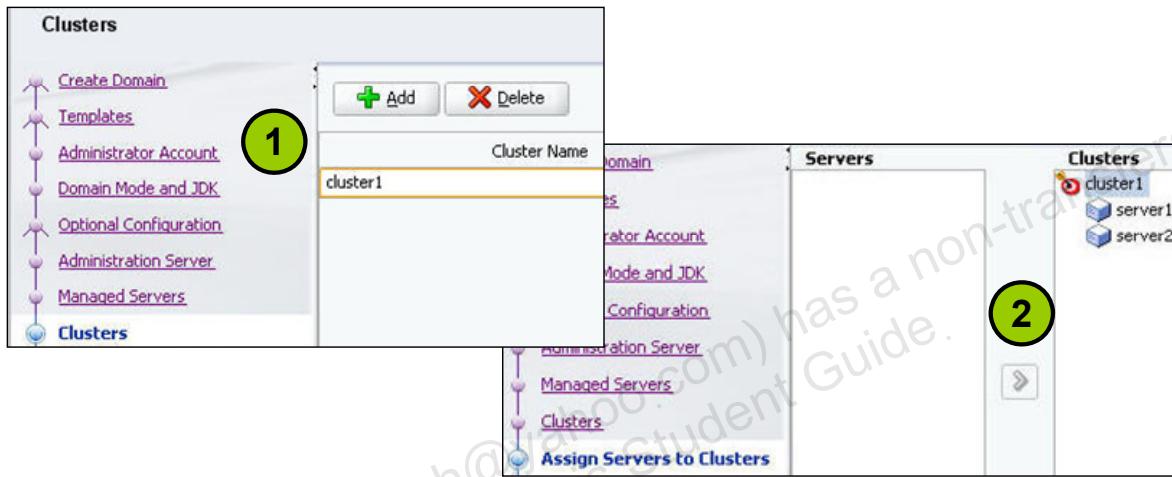
IP sockets provide a simple, high-performance mechanism for transferring messages and data between two applications. Clustered WebLogic Server instances use IP sockets for:

- Replicating HTTP session state and stateful session EJB state
- Accessing clustered objects that reside on a remote server instance (as in the multi-tier cluster architecture)

Creating a Cluster: Configuration Wizard

In the Configuration Wizard:

1. Add clusters.
2. Assign managed servers to them.

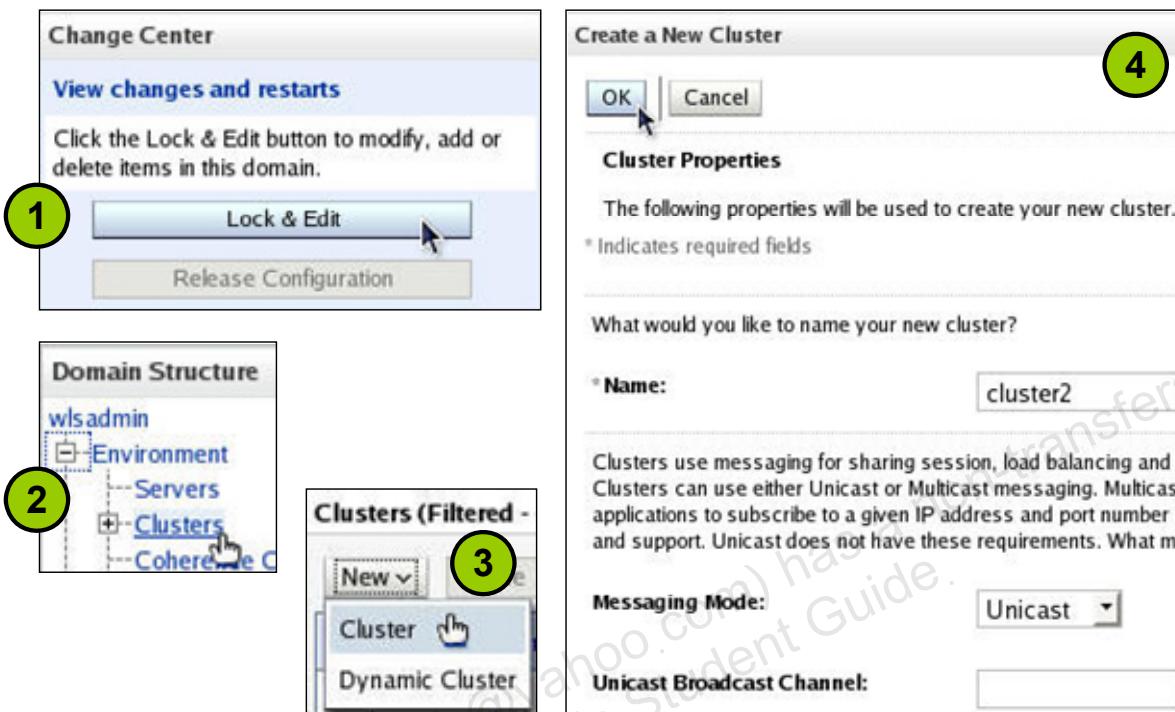


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

This is covered in the lesson titled “Creating Domains.”

Note that there is not a way to create dynamic clusters, server templates, or dynamic servers at the time of domain creation by using the Configuration Wizard. You can create a regular cluster by using the Configuration Wizard, and later create a server template and assign it to the cluster, which makes the cluster dynamic.

Creating a Cluster: Administration Console



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

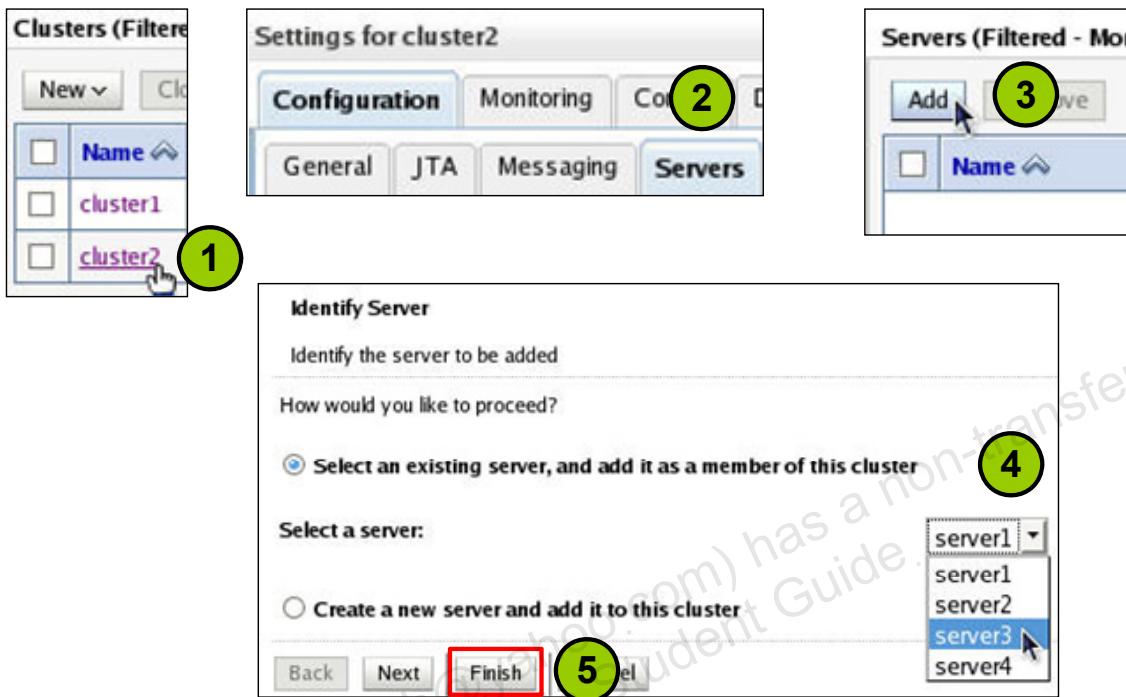
ORACLE

To create a new cluster, perform the following steps:

1. In the Change Center, click **Lock & Edit**.
2. Select **Clusters** under **Environment** in the Domain Structure.
3. Click the **New** button and select **Cluster**.
4. Give the cluster a unique name and select **Unicast** as the messaging mode. Click **OK**. If you have created a network channel for one-to-many cluster communication, enter it in the Unicast Broadcast Channel field. This field is optional. If left blank, each server's default network channel is used for this communication.

Note: The other messaging mode option is Multicast. If that is selected, you must also enter the Multicast Address and the Multicast Port.

Adding Servers to the Cluster: Administration Console



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

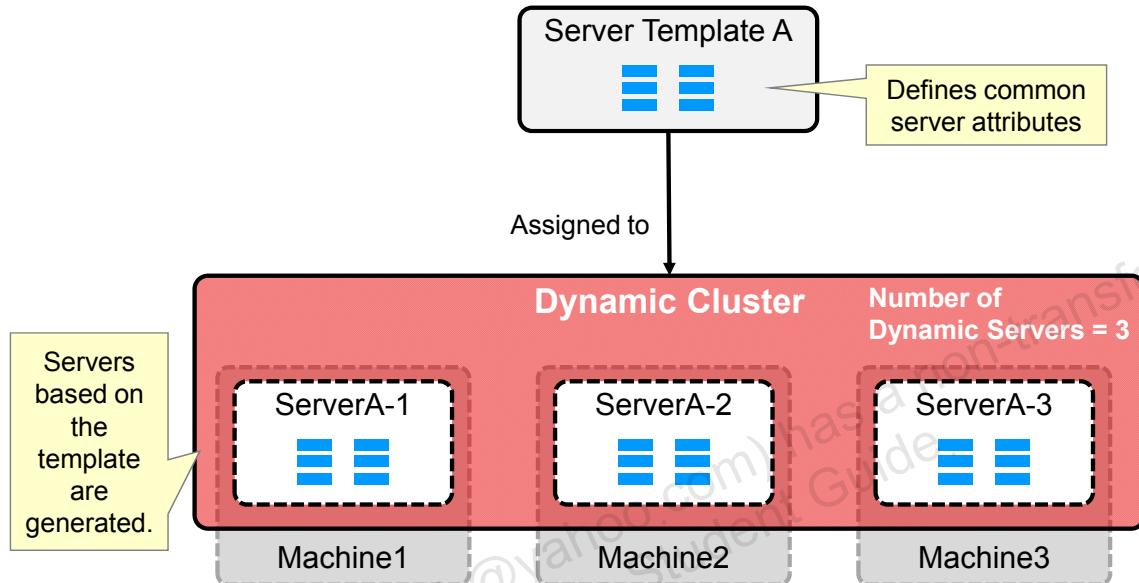
To add servers to the new cluster, perform the following steps (this assumes the configuration is still locked):

1. In the Clusters table, select the new cluster's name.
2. Select the **Configuration** tab and the **Servers** subtab.
3. Scroll down to the Servers table. Click the **Add** button.
4. To add an existing server to the cluster, choose **Select an existing server, and add it as a member of this cluster**. Then use the drop-down list to select a particular server. (The other option is to select **Create a new server and add it to this cluster**. You then click **Next** and are led through creating a new server.)
5. Click **Next** or **Finish**.
6. Repeat the process to add more servers to the cluster. (Not shown)
7. Finally, in the Change Center, click **Activate Changes**. (Not shown)

Note: You can also add a server to the cluster from the server's configuration. Lock the configuration. Select **Servers** under **Environment** in the Domain Structure. Click the name of a server in the Servers table. Select **Configuration > General**. Use the Cluster drop-down list to select a cluster. Click **Save**. Activate the changes. (The server cannot be running.)

Server Templates and Dynamic Clusters

A dynamic cluster is based on a server template.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Server Templates and Dynamic Clusters

- A server template defines server attributes.
 - Servers based on that template share those attributes.
 - If you change an attribute in the template, all of the servers based on that template change.
- A cluster can be associated with one server template. The cluster sets the number of dynamic servers needed.
 - That number of servers is generated and assigned to the cluster.
 - These servers show in the Servers table with the Type “Dynamic” (as opposed to “Configured”).
 - Attributes of dynamic servers that are server-specific are calculated when the servers are generated (for example, the server names).



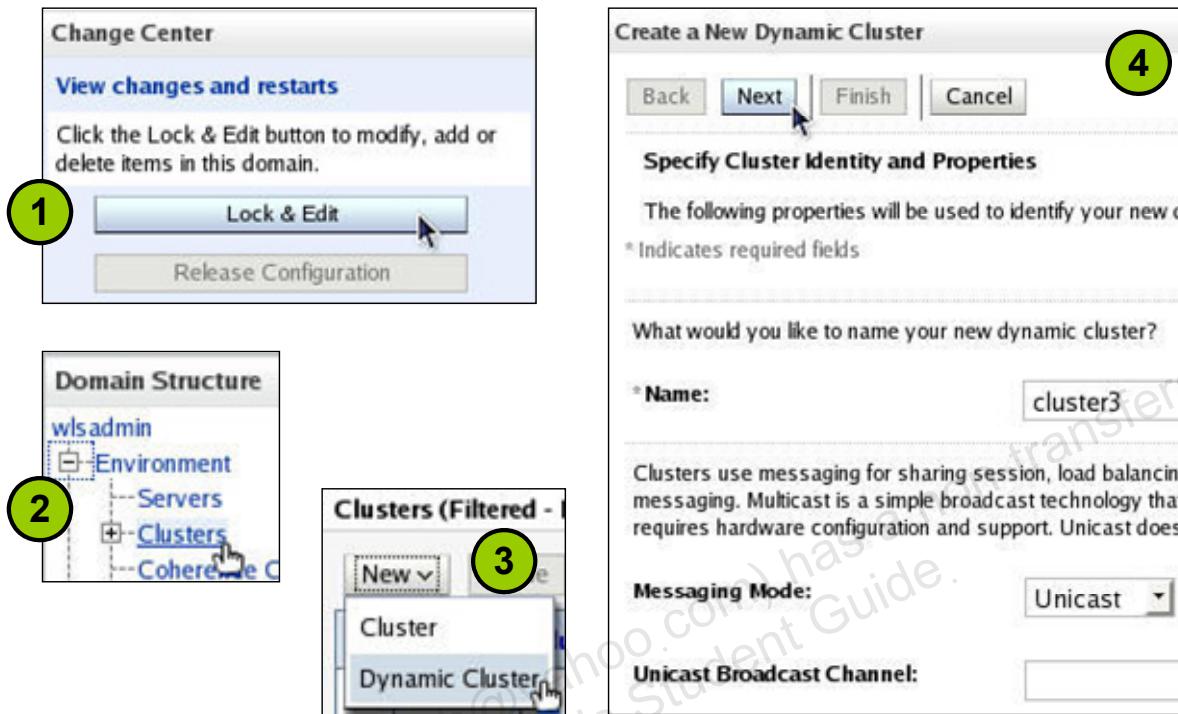
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Server templates allow you to define common attributes that are applied to a group of server instances. Because common attributes are contained in the template, you only need to change them in one place and they take effect in all of the servers that use the template. You use server templates in a heterogeneous server environment where server instances have a common set of attributes, but also have a set of unique, server-specific attributes. The primary use for server templates is in creating dynamic clusters.

Only one server template can be associated with a dynamic cluster. You use the server template to specify the configuration of the servers in the dynamic cluster, so that each server does not need to be manually created and configured for the cluster.

When configuring a dynamic cluster, you specify the number of server instances you anticipate needing at peak load. WebLogic Server creates the specified number of server instances and applies the calculated attribute values to each one. When you need additional capacity, you start one of the server instances not currently running, without having to first manually create and configure a new server and add it to the cluster.

Creating a Dynamic Cluster



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To create a new dynamic cluster, perform the following steps:

1. In the Change Center, click **Lock & Edit**.
2. Select **Clusters** under **Environment** in the Domain Structure.
3. Click the **New** button and select **Dynamic Cluster**.
4. Give the cluster a unique name and select **Unicast** as the messaging mode. Click **Next**. If you have created a network channel for one-to-many cluster communication, enter it in the Unicast Broadcast Channel field. This field is optional. If left blank, each server's default network channel is used for this communication.

Note: The other messaging mode option is Multicast. If that is selected, you must also enter the Multicast Address and the Multicast Port.

Creating a Dynamic Cluster

The screenshot shows two steps of a wizard:

Step 5: Specify Dynamic Server Properties

- Buttons: Back, **Next**, Finish, Cancel.
- Text: "The following properties will be used to specify the size and characteristics of your dynamic cluster." "How many dynamic servers will you need at peak load?"
- Input: Number of Dynamic Servers: 2.
- Text: "What naming convention would you like to use for new dynamic servers?" "Server Name Prefix: cluster3-server-".
- Text: "Server templates are used to configure the characteristics that are unique to a cluster and cannot be shared across this new cluster." "Server templates are used to configure the characteristics that are unique to a cluster and cannot be shared across this new cluster."
- Option: Create a new server template using domain defaults.

Step 6: Specify Machine Bindings

- Buttons: Back, **Next**, Finish, Cancel.
- Text: "Associating dynamic servers with machines is essential if you intend to start them from the Administration Console (or WLST) to start servers." "How do you want to distribute dynamic servers across machines?"
- Options:
 - Use any machine configured in this domain
 - Use a single machine for all dynamic servers
 - Use a subset of machines in this domain
- Text: "Selected Machine: machine1".
- Text: "Machine Name Match Expression: machine*".
- Text: "Use machines that have a name that starts with the string “machine”".

ORACLE

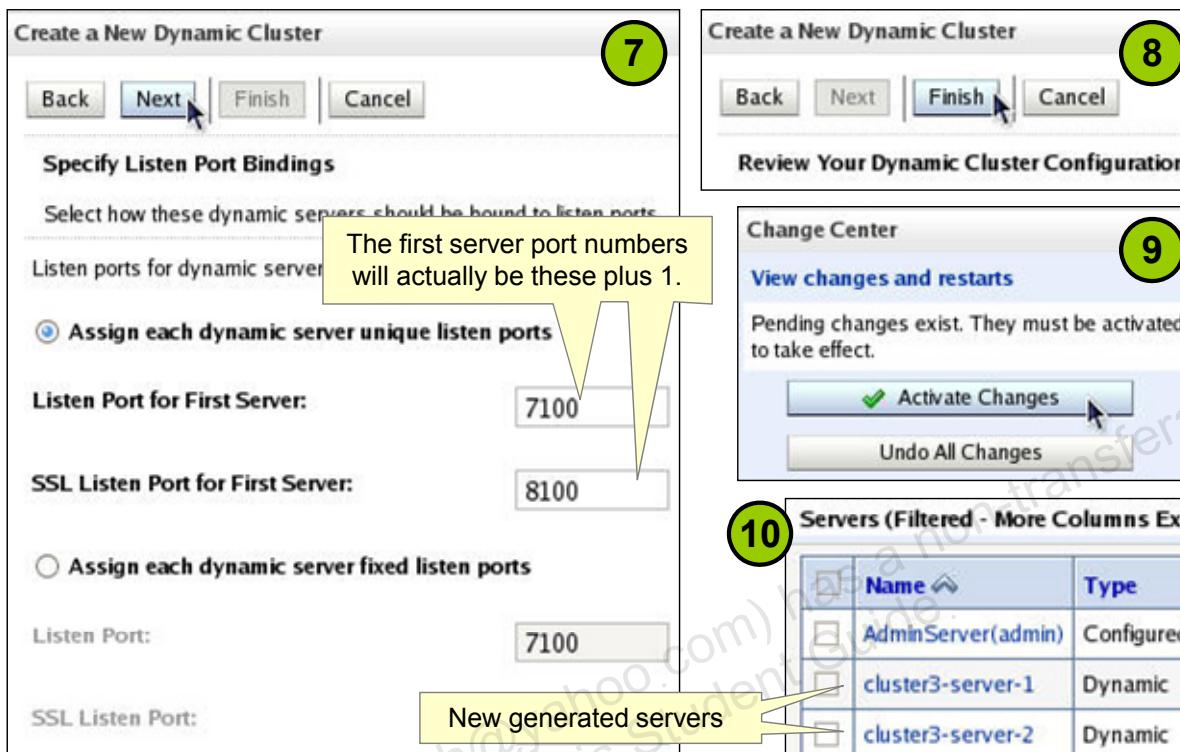
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

5. Enter the Number of Dynamic Servers. This is how many dynamic servers are created and placed in the cluster. This number should be the number of server instances you anticipate needing at peak load. Enter the Server Name Prefix. This is used to help generate the server names. Select **Create a new server template using domain defaults**. Click **Next**.

Note: The other option for the server template is to select **Clone an existing server template for this cluster** and use the drop-down list to select an existing server template to copy.

6. Select how to distribute the dynamic servers across machines, then click **Next**. The choices are:
 - **Use any machine configured in this domain:** The dynamic servers generated are assigned in a round-robin fashion to all machines defined in the domain.
 - **Use a single machine for all dynamic servers:** You want all dynamic servers on one machine. If this is chosen, you use the Selected Machine drop-down list to select that machine.
 - **Use a subset of machines in this domain:** The dynamic servers generated are assigned in a round-robin fashion to all machines defined in the domain that match the Machine Name Match Expression entered. An asterisk can be used as a wild card.

Creating a Dynamic Cluster

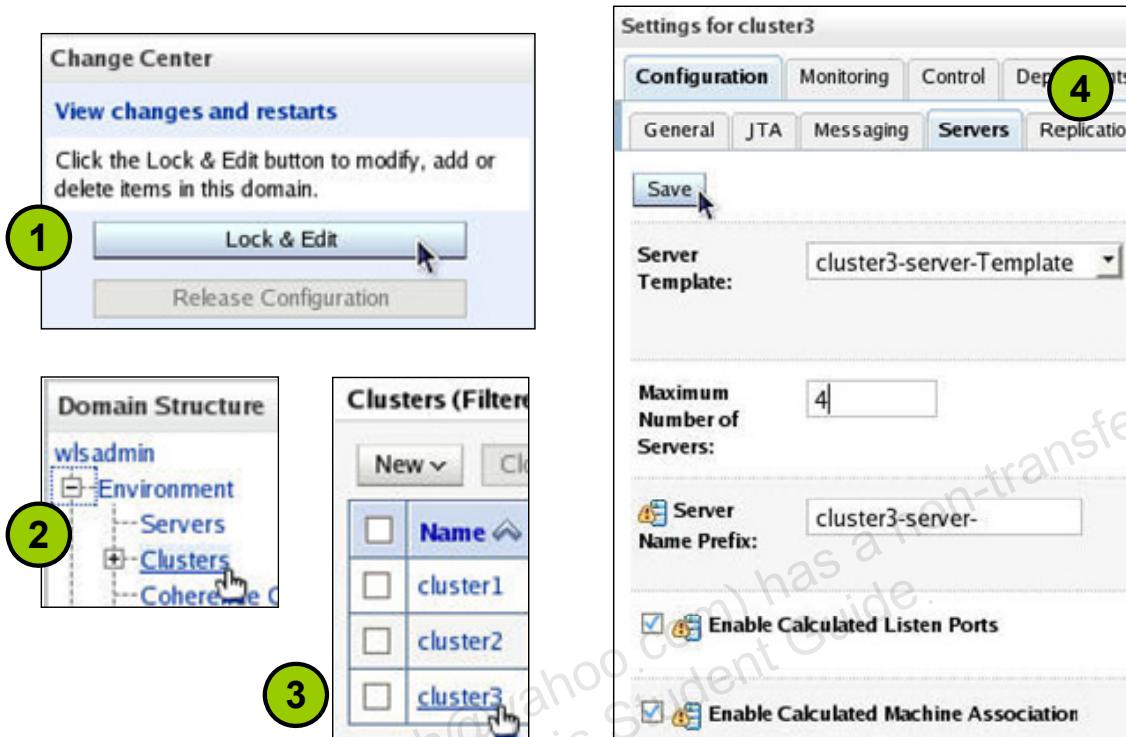


ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

7. Select how to assign listen ports to the dynamic servers and then click **Next**. The two choices for how to do the port assignments are:
 - **Assign each dynamic server unique listen ports:** The dynamic servers generated are assigned unique listen ports by using the numbers entered in the Listen Port for First Server and SSL Listen Port for First Server fields. The first server port numbers are the values entered plus 1. Each subsequent server will have 1 added to the port numbers of the server generated before it.
 - **Assign each dynamic server fixed listen ports:** The dynamic servers generated are assigned the same listen ports entered in the Listen Port and SSL Listen Port fields. Note that this only works if the servers have different listen addresses. (No two servers can share the same listen address and listen port.)
8. Review the dynamic cluster and click **Finish**. The new server template and dynamic cluster are created.
9. In the Change Center, click **Activate Changes**.
10. After the changes are activated, new servers are generated, as shown in the Servers table.

Editing the New Dynamic Cluster



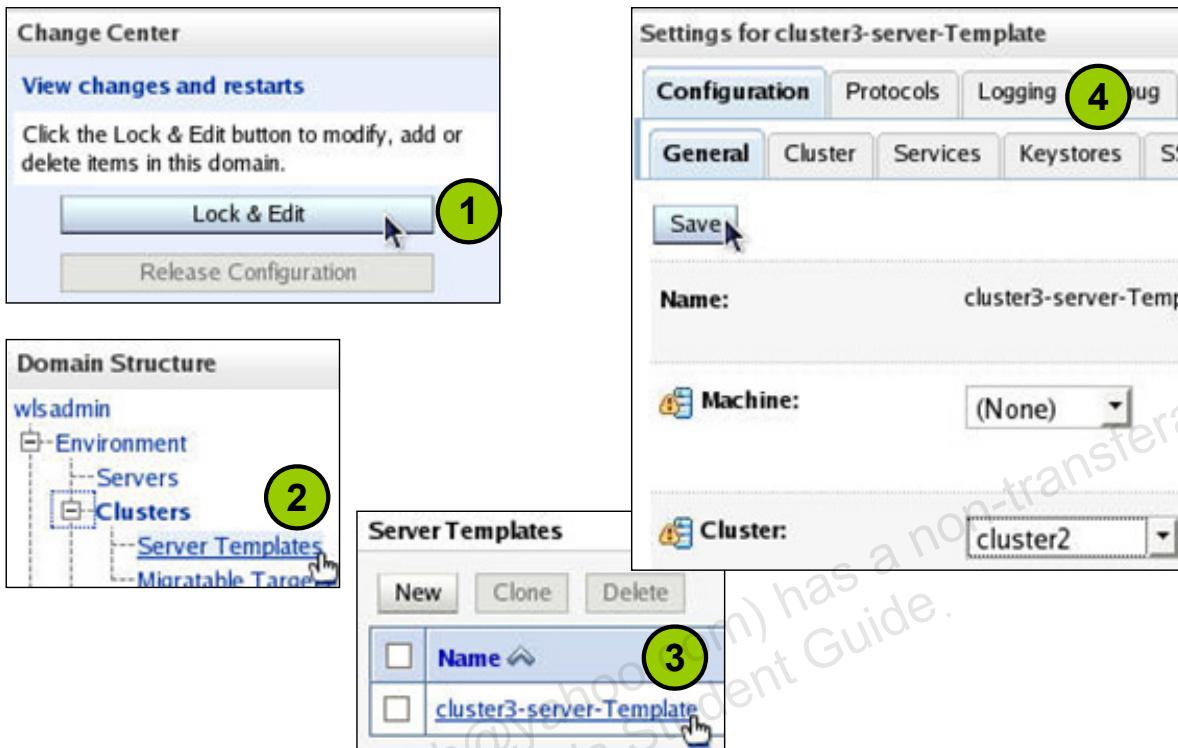
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To edit the new dynamic cluster, perform the following steps:

1. In the Change Center, click **Lock & Edit**.
2. Select **Clusters** under **Environment** in the Domain Structure.
3. Select the name of the new cluster in the Clusters table.
4. Select whichever tabs you want. Make changes to the cluster attributes. Click **Save**.
5. In the Change Center, click **Activate Changes**.

Editing the New Server Template



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To edit the new server template, perform the following steps:

1. In the Change Center, click **Lock & Edit**.
2. In the Domain Structure, expand **Environment**, expand **Clusters**, and select **Server Templates**.
3. Select the name of the new server template in the Server Templates table.
4. Select whichever tabs you want. Make changes to the server template attributes. Click **Save**.
5. In the Change Center, click **Activate Changes**.

Dynamic Server Calculated Attributes

Dynamic servers are generated for a dynamic cluster based on the server template. Server-specific attributes are calculated:

- Server name: The Server Name Prefix followed by indexes in order, starting with 1.
- Listen ports: Cluster has Enable Calculated Listen Ports selected
 - Dynamic: The port values entered in the template +1 for the first server, +2 for the second, and so on.
 - Static: Each server gets the same template port values
- Machine names: Cluster has Enable Calculated Machine Associations selected
 - No machine name match expression: All machines are rotated through as the servers are generated.
 - Machine name match expression: Only matching machines are rotated through as the servers are generated.



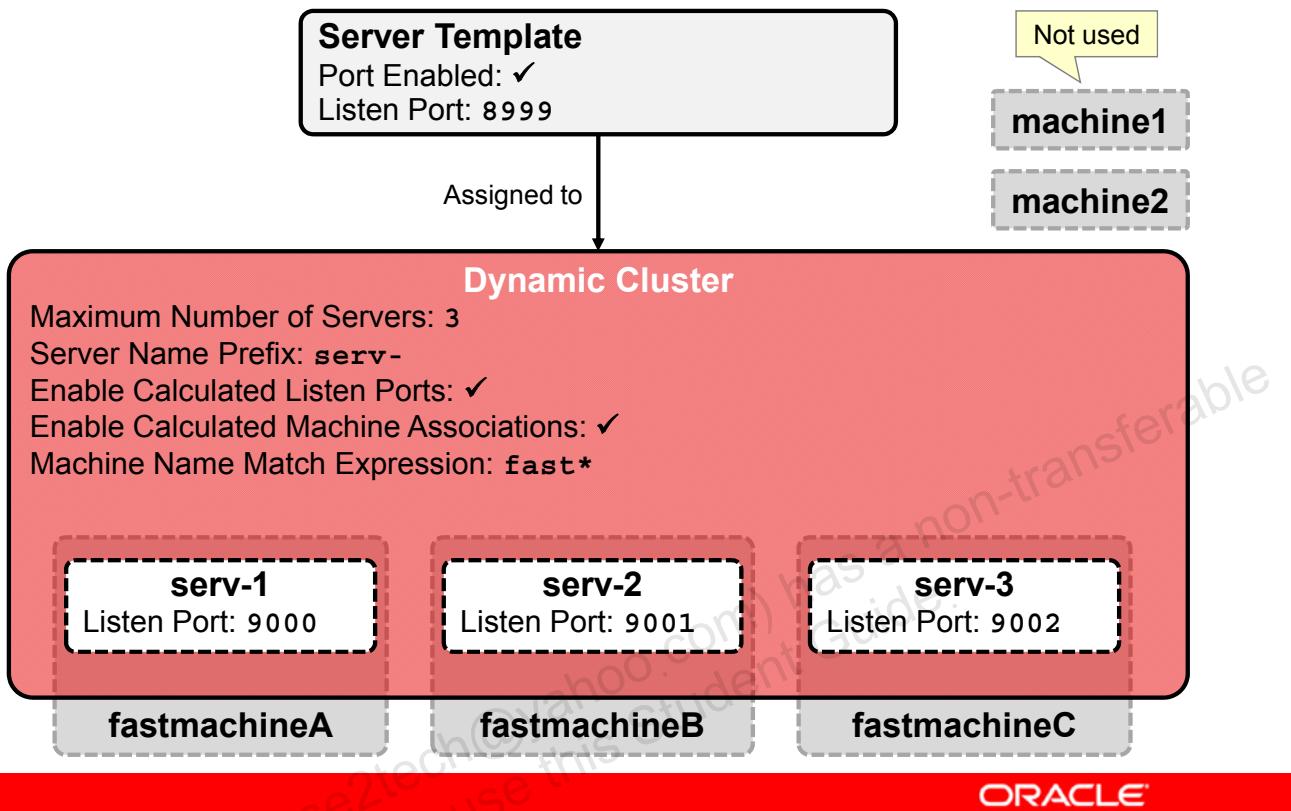
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

When you use a server template to create a dynamic cluster and specify the number of server instances you want, WebLogic Server uses calculated values for the following server-specific attributes:

- **Server name (always):** The calculated server name is controlled by the Server Name Prefix attribute. Server names are the specified prefix followed by an index. For example, if the prefix is set to myserver-, then the dynamic servers will have the names myserver-1, myserver-2, and so on.
- **Listen ports (optional—if the cluster has Enable Calculated Listen Ports selected):**
 - Dynamic:
 - Standard port: The first server is Listen Port for First Server + 1, the second server is Listen Port for First Server + 2, and so on.
 - SSL port: The first server is SSL Listen Port for First Server + 1, the second server is SSL Listen Port for First Server + 2, and so on.
 - Static:
 - Standard port: All server listen ports equal the Listen Port value.
 - SSL port: All server SSL ports equal the SSL Listen Port value.

- **Machines (optional—if the cluster has Enable Calculated Machine Associations selected):**
 - If no Machine Name Match Expression has been entered: All machines in the domain are used. Assignments are made in a round-robin fashion.
 - If a Machine Name Match Expression has been entered: Only those machines whose names match the expression are used. Assignments are made to matching machines in a round-robin fashion. Machine name match expressions can use asterisks for wildcards, and can list multiple expressions separated by commas. For example, say the domain has these machines defined: mach1, mach2, mach3, fastmach1, fastmach2, fastmach3, and fastmach4. And the Machine Name Match Expression is set to: mach1, fast*. The machines would be assigned in this order: mach1, fastmach1, fastmach2, fastmach3, fastmach4, mach1, fastmach1, and so on.
- **Network Channel (Access Point) listen ports (optional—if the server template has a Network Channel defined within it):** The first server is the network channel port + 1, the second server is the network channel port + 2, and so on.

Dynamic Server Calculated Attributes: Example



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The three servers generated for the dynamic cluster have a name that starts with `serv-` and ends in an index, starting with 1. The listen ports for the servers start with the template listen port +1. The servers are assigned to machines already defined, but only those machines with names that start with “fast.”

Comparing Configured and Dynamic Clusters

| Feature | Configured Cluster | Dynamic Cluster |
|--------------------------------------|--------------------|-----------------|
| Create with the Admin Console / WLST | Yes | Yes |
| Create with the Configuration Wizard | Yes | No |
| Edit individual server attributes | Yes | No |
| Servers generated automatically | No | Yes |
| Can contain configured servers | Yes | Yes |
| Can contain dynamic servers | No | Yes |
| Supports service-level migration | Yes | No |
| Supports whole-server migration | Yes | Yes |

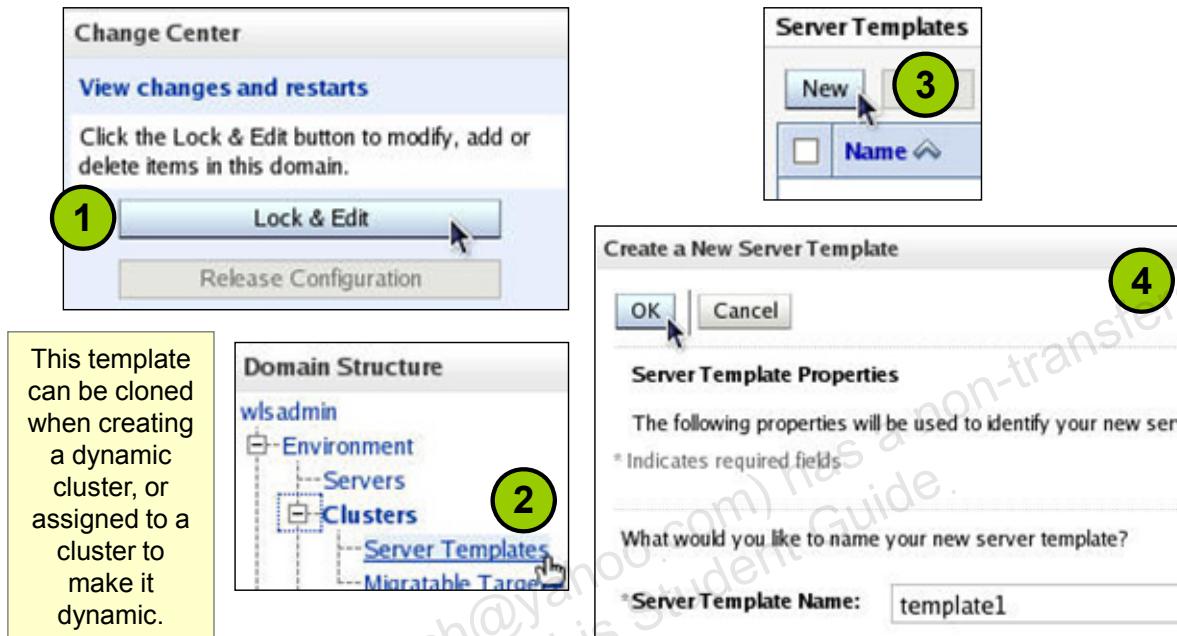


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A configured cluster that is assigned a server template and has dynamic servers generated for it becomes a dynamic cluster. A dynamic cluster can contain both configured and dynamic servers.

Creating a Server Template

You can create a server template independently from creating a dynamic cluster:



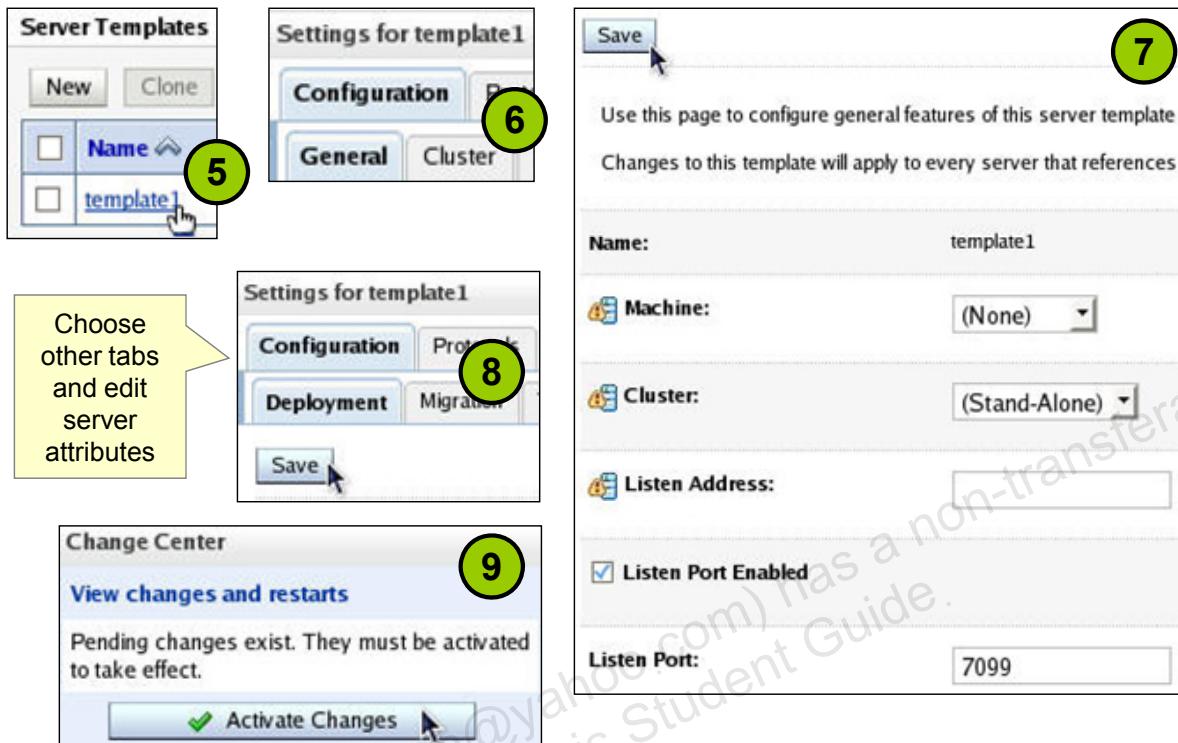
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To create a new server template, perform the following steps:

1. In the Change Center, click **Lock & Edit**.
2. In the Domain Structure, expand **Environment**, expand **Clusters**, and select **Server Templates**.
3. Click the **New** button.
4. Give the server template a unique name and click **OK**.

Creating a Server Template



ORACLE®

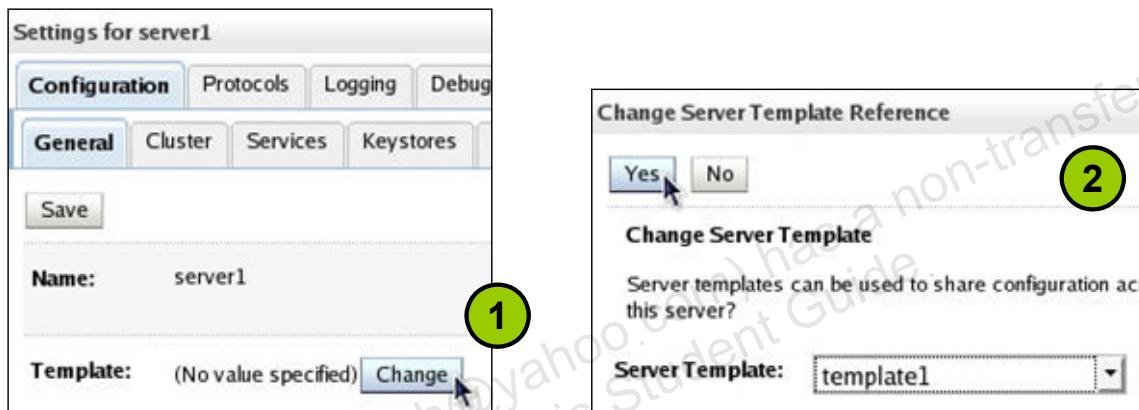
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

5. In the Server Templates table, select the name of the new template.
6. Select **Configuration > General**.
7. Enter any values for attributes that you want shared among servers created from this template. Some attribute values can be overridden when the template is assigned to a cluster. Other attribute values are used in calculations. For example, when a value for Listen Port is entered, it is used as the starting point for listen ports for the servers generated from the template. After entering the values, click **Save**.
8. Select any other tabs and enter whatever attribute values you want shared among these servers. Remember to click **Save** after entering values on a page.
9. In the Change Center, click **Activate Changes**.

Server Templates and Configured Servers

In addition to using server templates to define the servers in a dynamic cluster, a server template can be assigned to any number of configured servers, so those servers can share common, nondefault attributes.

- The attributes can be overridden by the individual servers.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

- After creating the server template and locking the configuration, in the Servers table, select the server. Ensure that **Configuration > General** is selected. Click the **Change** button for the Template field.
- Select the server template from the drop-down list and click **Yes**. Then activate the changes.

Quiz

The multi-tier cluster architecture allows you to load balance EJB calls. But, the basic (single-tier) architecture has an EJB-related advantage over multi-tier. The advantage is:

- a. It cannot use EJBs, which makes development simpler
- b. This is a trick question, because the single-tier architecture has no EJB-related advantages
- c. All EJB calls are local and, therefore, faster



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: c

Quiz

A dynamic cluster is based on:

- a. One server template
- b. Multiple server templates
- c. A cluster proxy
- d. A domain template



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a

Summary

In this lesson, you should have learned how to:

- Describe two cluster architectures: basic and multi-tier
- Create and configure a cluster
- Create and configure a dynamic cluster

Practice 12-1 Overview: Configuring a Cluster

This practice covers creating a cluster by using the administration console.

Practice 12-2 Overview: Configuring a Dynamic Cluster

This practice covers creating a dynamic cluster by using the administration console.



Clusters – Proxies and Sessions

13

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Install Oracle HTTP Server
- Configure Oracle HTTP Server as a cluster proxy
- Configure session failover
- Configure replication groups

A Cluster Proxy for a Web Application Cluster

- A cluster proxy provides load balancing and failover for a web application cluster. It gives the cluster its “single server” appearance.
- There are basically two kinds of cluster proxies:
 - A web server with the WebLogic proxy plug-in
 - A hardware load balancer

| Cluster Proxy | Advantages | Disadvantages |
|-------------------------|--|---|
| Web server with plug-in | <ul style="list-style-type: none"> • Low cost (or free) • You probably already have experience with the web server | <ul style="list-style-type: none"> • Only round-robin load balancing available • Must configure the plug-in |
| Hardware load balancer | <ul style="list-style-type: none"> • More sophisticated load balancing algorithms • No plug-in configuration | <ul style="list-style-type: none"> • Cost • Must be compatible with the WebLogic Server session cookie |



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

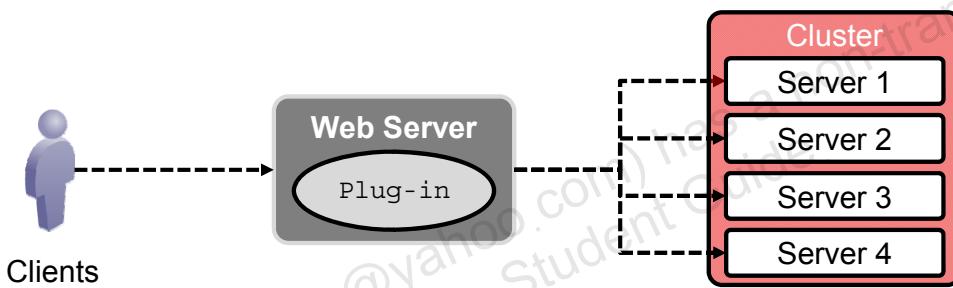
Cluster Proxies are how clients interact with a web application cluster, whether they are hardware or software based. You have two basic choices of cluster proxy: a web server using a plug-in or a hardware load balancer (such as F5 BIG-IP).

Hardware load balancers must support a compatible passive or active cookie persistence mechanism. Passive cookie persistence enables WebLogic Server to write a cookie containing session information through the load balancer to the client. You can use certain active cookie persistence mechanisms with WebLogic Server clusters, provided the load balancer does not modify the WebLogic Server session cookie. If the load balancer's active cookie persistence mechanism works by adding its own cookie to the client session, no additional configuration is required to use the load balancer with a WebLogic Server cluster.

A WebLogic Server proxy plug-in is available for Netscape Enterprise Server, Apache HTTP Server, Microsoft Internet Information Server (IIS), and Oracle HTTP Server (which is based on Apache). These plug-ins provide round-robin load balancing to the servers in the cluster. They use the WebLogic Server session cookie information to route requests to the server that has a client's session data.

Proxy Plug-Ins

- A proxy plug-in:
 - Load balances client requests to clustered WebLogic Server instances in a round-robin fashion
 - Avoids routing requests to failed servers in the cluster
 - Routes requests based on WebLogic Server session cookie information



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The web server with the proxy plug-in may do more than just load balance requests to WebLogic Server instances in a cluster. In some architectures, the web server serves up static files (HTML files and images, for example) and only passes through requests to WebLogic Server for “dynamic” pages (JSPs or calls to Servlets). To the web browser, the HTTP responses appear to come from one source—the web server. A variation of that architecture is to use a hardware load balancer in front of a bank of web servers (serving up static content), which are in front of a cluster of WebLogic Server instances (returning “dynamic” content).

Oracle WebLogic Server plug-ins can provide efficient performance by reusing connections from the plug-in to WebLogic Server (“keep-alive” connections).

Oracle HTTP Server (OHS)

OHS is a web server that is:

- A component of the Oracle Web Tier Suite
- Based on Apache HTTP Server
- Installed with the WebLogic Server plug-in module (`mod_wl_ohs`) by default
 - The plug-in must be configured by using the `mod_wl_ohs.conf` file.
- Managed and monitored as part of a WebLogic Server domain:
 - Configured as part of a system component template
 - Started and Managed by Node Manager with the `$DOMAIN_HOME/bin/startComponent.sh` script



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

OHS is based on the Apache web server. OHS supports single sign-on, clustered deployment and high availability, and Web Cache.

Configuration of Oracle HTTP Server is specified through directives in configuration files in the same manner as with Apache HTTP Server.

A `mod_wl_ohs` module is available in OHS. This module enables you to integrate your WebLogic Server environment with OHS immediately after the configuration of the OHS instance and the domains.

OHS directories are divided between the Oracle home and the Oracle instance. The Oracle home directories are read-only, and contain the Oracle Fusion Middleware binaries. The Oracle instance directories contain the modules, applications, and logs for OHS. Each OHS component has a root configuration directory found at

`$DOMAIN_HOME/config/fmwconfig/components/OHS/instances/<instance>`, which includes the WLS plug-in configuration file, `mod_wl_ohs.conf`. Similarly, each component's log files are found at `$DOMAIN_HOME/servers/<instance>/logs`.

Installing and Configuring OHS (Part of Oracle Web Tier): Overview

1. Download and unzip the Web Tier installer.
2. Run the downloaded executable.
 - A. Specify the Web Tier installation location.
 - B. Specify installation type: standalone or within an existing WebLogic Server installation

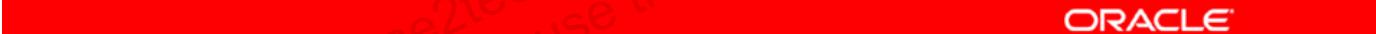
 ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

An “oracle instance” location contains one or more Fusion Middleware system components, such as Oracle Web Cache or Oracle HTTP Server. The oracle instance directory contains updatable files, such as configuration files, logs files, and temporary files.

Installing and Configuring OHS (Part of Oracle Web Tier): Overview

3. Configure an OHS instance by navigating to `<WEB_TIER>/oracle_common/common/bin` and run the Configuration Wizard: `config.sh`.
 - Enter domain location.
 - Specify Basic Standalone System Component Domain and Oracle HTTP Server templates.
 - Configure System Components:
 - Specify instance name and type.
 - Configure the Admin host and port (for FMWC domains only).
 - Enter the listen address, port, and SSL port.
 - Enter the server name.
 - Configure Node Manager type and credentials.
 - Click **Create**, and at 100% complete, click **Finish**.

The red bar contains the ORACLE logo.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Fusion Middleware Control (FMWC) can be used to manage OHS. If the domain is not enabled with FMWC, then you must use WLST to manage OHS.

Configuring OHS as the Cluster Proxy

Modules extend the functionality of OHS to enable it to integrate with other Fusion Middleware components.

- The proxy plug-in for WebLogic Server is called `mod_wl_ohs.so` and is found here:
`<DOMAIN_HOME>/ohs/modules`.
 - The plug-in is already installed, but must be configured.
- Configuration files for OHS are found here:
`<DOMAIN_HOME>/config/fmwconfig/components/OHS/instances/OHS_instance_name`.
 - The main configuration file is `httpd.conf`. It contains an include directive for the WebLogic plug-in configuration file:
 - `mod_wl_ohs.conf`. This is the file you edit to configure OHS to proxy a WebLogic Server cluster.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Modules extend the basic functionality of OHS, and support integration between OHS and other Oracle Fusion Middleware components. The `mod_wl_ohs.so` module is installed and loaded out-of-the-box with Oracle HTTP Server, but it is not configured by default. Therefore, you must configure it to specify the application requests that the module should handle. The `mod_wl_ohs` module enables requests to be proxied from an OHS to Oracle WebLogic Server. The configuration for this module is stored in the `mod_wl_ohs.conf` file, which can be edited manually with a text editor.

httpd.conf and mod_wl_ohs.conf

- The include directive in httpd.conf looks like :

```
include "mod_wl_ohs.conf"
```

- The mod_wl_ohs.conf file has various directives, but the WebLogicCluster directive is the most important.
 - It specifies the initial list of servers in the cluster, giving their host names and ports.
 - Remember, you do not need to update this list in the configuration file to add or remove servers from the cluster. This is the *initial* list of cluster members. Once the cluster is running, the plug-in uses the dynamic server list.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

mod_wl_ohs.conf

```
LoadModule weblogic_module "${PRODUCT_HOME}/modules/mod_wl_ohs.so" Load the proxy plug-in

<IfModule weblogic_module>
<Location /> Proxy to the cluster based on this URL path
    WLSRequest On
    WebLogicCluster host01.example.com:7011,host02.example.com:7012
</Location> Initial list of cluster members
</IfModule> Parameters for this specific location
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To proxy requests to a single server, use the `WebLogicHost` and `WebLogicPort` parameters. To proxy requests to a cluster of WebLogic Servers, use the `WebLogicCluster` parameter instead.

To proxy requests by path, use the `Location` block. To proxy requests by MIME type, add a `MatchExpression` line to the `IfModule` block. Note that if both MIME type and proxying by path are enabled, proxying by path takes precedence over proxying by MIME type. You can also use multiple `MatchExpression` lines.

Some Plug-in Parameters

| Parameter | Description |
|---|---|
| WebLogicHost , WebLogicPort | Proxy to a single server with this host and port |
| WebLogicCluster | Proxy to this initial list of clustered servers |
| MatchExpression | Proxy requests for files of this MIME type |
| PathTrim | Remove this text from the incoming URL path before forwarding a request. |
| PathPrepend | Add this text to the incoming URL path before forwarding a request. |
| ErrorPage | URL to direct users to if all servers are unavailable |
| WLExcludePathOrMimeType | Do not proxy for this specific URL path or MIME type. |
| WLProxySSL | Set to ON to establish an SSL connection to WebLogic if the incoming request also uses HTTPS. |
| MaxPostSize | Maximum allowable size of POST data, in bytes |
| Debug | Sets the type of logging performed |



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

- **WebLogicHost**, **WebLogicPort**: WebLogic Server host (or virtual host name as defined in WebLogic Server) to which HTTP requests should be forwarded. Port at which the WebLogic Server host is listening for connection requests from the plug-in.
- **WebLogicCluster**: Comma-separated list of host : port for each of the initial WebLogic Server instances in the cluster. The server list specified in this property is a starting point for the dynamic server list that the servers and plug-in maintain. WebLogic Server and the plug-in work together to update the server list automatically with new, failed, and recovered cluster members.
- **MatchExpression**: When proxying by MIME type, set the filename pattern inside of an IfModule block using the MatchExpression parameter.
- **PathTrim**: Specifies the string trimmed by the plug-in in the { PATH } / { FILENAME } portion of the original URL, before the request is forwarded to WebLogic Server.
- **ErrorPage**: You can create your own local error page that is displayed when your web server is unable to forward requests to WebLogic Server.
- **WLExcludePathOrMimeType**: This parameter allows you to exclude certain requests from proxying.

- **WLProxySSL**: Set this to **ON** to establish an SSL connection to WebLogic Server if the incoming request uses HTTPS.
- **MaxPostSize**: Maximum allowable size of POST data, in bytes. If the content-length exceeds this value, the plug-in returns an error message. If set to **-1**, the size of POST data is not checked. This is useful for preventing denial-of-service attacks that attempt to overload the server.
- **Debug**: Sets the type of logging performed for debugging operations. The debugging information is written to the `/tmp/wlproxy.log` file. Some of the possible values for this parameter are:
 - **ON**: The plug-in logs informational and error messages.
 - **OFF**: No debugging information is logged.
 - **ERR**: Only error messages are logged.
 - **ALL**: The plug-in logs headers sent to and from the client, headers sent to and from WebLogic Server, information messages, and error messages.

Starting and Stopping OHS

OHS is managed by Node Manager as part of a WebLogic domain:

- Start Node Manager:

```
$ cd $DOMAIN_HOME/bin  
$ ./startNodeManager.sh
```

- Start OHS server (via Node Manager):

```
$ ./startComponent.sh <instance_name>
```

- Reverse the order to stop OHS:

```
$ ./stopComponent.sh <instance_name>  
$ ./stopNodeManager.sh
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

<instance_name> is the name of the OHS instance that is configured in the domain.

Verifying that OHS Is Running

OHS is administered with Fusion Middleware Control (FMWC) if the domain is configured with the administration tool. Otherwise, you must use WLST.

```
wls:/offline> nmConnect('weblogic','Welcome1','localhost','5556')
Connecting to Node Manager ...
Successfully Connected to Node Manager.

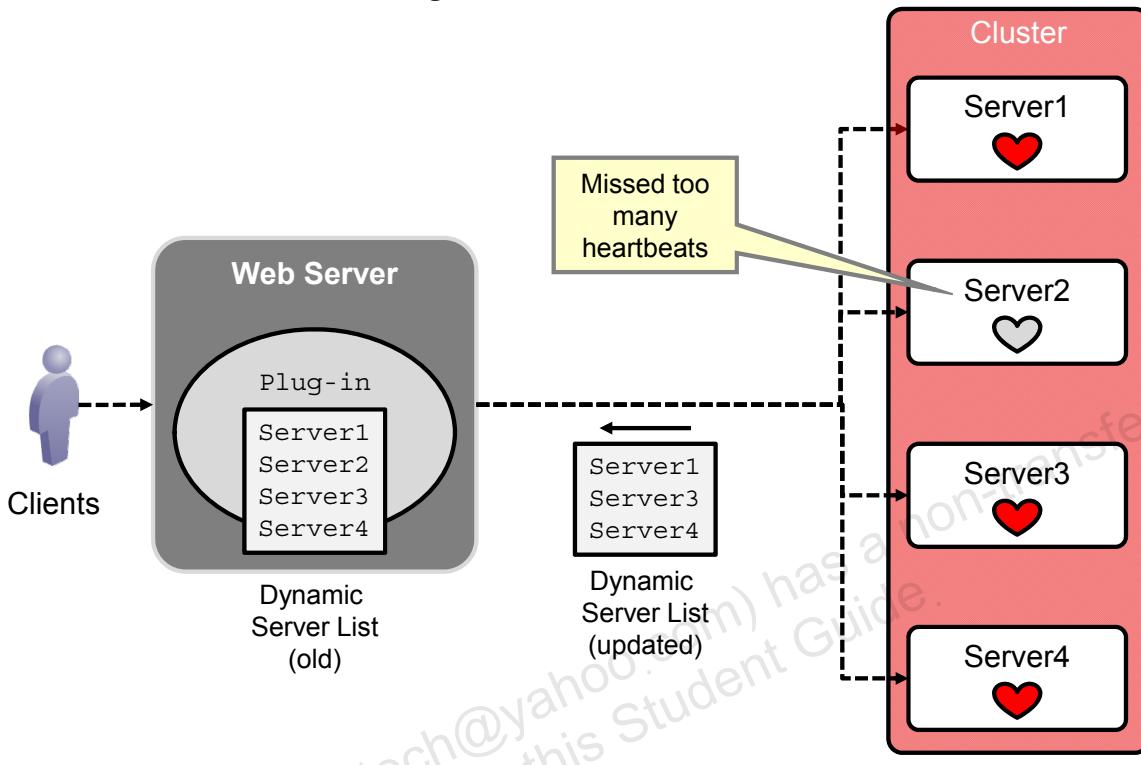
wls:/offline> nmServerStatus(serverName='ohs1', serverType='OHS')

RUNNING
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Failover: Detecting Failures and the Dynamic Server List



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The cluster and the proxy maintain a dynamic server list that has within it all viable servers in the cluster. When a server in the cluster fails it is detected and the list is updated. If a server misses too many heartbeats, it is marked as failed. Also, if a socket to a server in the cluster (from another cluster server) closes unexpectedly, the server is marked as failed.

The dynamic server list is also updated when new servers in the cluster are started.

Failover: Detecting Failures and the Dynamic Server List

- A clustered server detects the failure of another server in the cluster when:
 - A socket to that server unexpectedly closes
 - That server misses three* heartbeats
- In either case, that server is marked as “failed.”
- Responses from a clustered server to a cluster proxy include the “dynamic server list,” a list of all the current, viable servers in the cluster.
 - The list lets the proxy know which servers it can use.
 - The list is updated not only when servers fail, but also when new servers are added to the cluster.

* This number is configurable.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

When you specify the list of WebLogic Server instances for a cluster proxy, the plug-in uses that list as a starting point for load balancing among the members of the cluster. After the first request is routed to one of these servers, a dynamic server list is returned in the response that contains an updated list of servers in the cluster. The updated list adds any new servers in the cluster and deletes any that are no longer part of the cluster or that have failed. The dynamic server list is returned with each clustered server response so that the proxy always has an up-to-date list of viable servers.

To configure how many heartbeats can be missed: In the admin console, select the cluster, then select **Configuration > Messaging**. Under **Advanced**, change **Idle Periods Until Timeout**. The default is 3.

HTTP Session Failover

- Web applications store objects in HTTP sessions to track information for each client in memory.
- When an instance of WebLogic Server creates a session, it writes a cookie to the client's web browser indicating that it is the server for this client. Subsequent requests from that client are routed by the proxy to this server.
- If the server fails, its clients must be routed to other servers in the cluster, and session information is lost.
- WebLogic Server supports several strategies so that the session information is not lost when a server fails:
 - In-memory session replication
 - JDBC (database) session persistence
 - File session persistence

Recommended, as it is the fastest



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Java web application components, such as Servlets and JavaServer Pages (JSPs), can maintain data on behalf of clients by storing objects in the `HttpSession`. An `HttpSession` is available on a per-client basis. Once an instance of WebLogic Server creates a session for a client, it also writes a cookie to the client's web browser. This cookie indicates which server has this client's session. The cluster proxy checks this cookie on subsequent client requests, and routes the client to the instance of WebLogic Server that has the client's session.

If the server that has the client's session fails, when the client makes their next request, they cannot be routed to that server. The proxy chooses another server. That server does not have the client's session, so information about the client is lost.

To provide transparent failover for web applications, replication or shared access to the information in each `HttpSession` object must be provided. This is accomplished within WebLogic Server by using in-memory replication, file system persistence, or database persistence. A web application chooses which session failover option to use in the WebLogic Server deployment descriptor, `weblogic.xml`. Each option has its own configurable parameters that are also entered in `weblogic.xml`.

Note that in-memory replication has two options, synchronous and asynchronous. The asynchronous option replicates data in batches to improve cluster performance.

Configuring Web Application Session Failover: `weblogic.xml`

- Developers configure sessions in `weblogic.xml`, under the `<session-descriptor>` tag.
- Its subtag, `<persistent-store-type>`, configures session failover:

| <code><persistent-store-type></code> | Description |
|--|--|
| <code>memory</code> | No session replication or persistence |
| <code>replicated</code> | In-memory session replication |
| <code>replicated_if_clustered</code> | The same as <code>memory</code> if deployed to stand-alone servers, the same as <code>replicated</code> if deployed to a cluster |



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The value of the `<persistent-store-type>` tag determines session failover:

- `memory`: No session replication or persistence
- `replicated`: In-memory session replication. The syncing of sessions between the primary and secondary servers occurs synchronously. Note that it is an error to deploy a web application with this option to stand-alone servers.
- `replicated_if_clustered`: If the web application is deployed to stand-alone servers, this option is the same as `memory`. If the web application is deployed to a cluster, this is the same as `replicated`.

Configuring Web Application Session Failover: `weblogic.xml`

| <code><persistent-store-type></code> | Description |
|--|--|
| <code>async-replicated</code> | In-memory session replication with syncing done in batches |
| <code>async-replicated-if-clustered</code> | The same as <code>memory</code> if deployed to stand-alone servers, the same as <code>async-replicated</code> if deployed to a cluster |
| <code>file</code> | File-based persistence of sessions |
| <code>jdbc</code> | Database persistence of sessions |
| <code>async-jdbc</code> | Database persistence of sessions with updates done in batch |
| <code>cookie</code> | All session data stored in cookies |



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

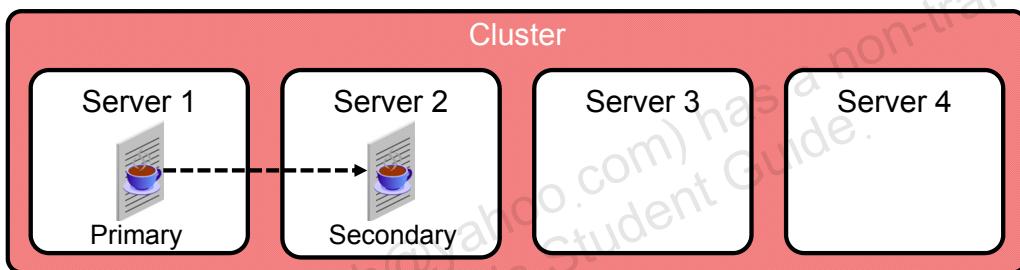
The value of the `<persistent-store-type>` tag determines session failover:

- **async-replicated:** In-memory session replication, but the syncing of sessions between the primary and secondary servers is done in batches, rather than synchronously. Note that it is an error to deploy a web application with this option to stand-alone servers.
- **async-replicated-if-clustered:** If the web application is deployed to stand-alone servers, this option is the same as `memory`. If the web application is deployed to a cluster, this is the same as `async-replicated`.
- **file:** File-based session persistence. This requires another subtag, `<persistent-store-dir>`, which specifies the directory where files containing session data are placed. This directory must be accessible to all servers in the cluster.

- **jdbc**: Session data is stored in a database. This requires another subtag, `<persistent-store-pool>`, which specifies the data source used to access the database. This data source must be deployed to all servers in the cluster. The database that is accessed through this data source must have a table named `WL_SERVLET_SESSIONS` with certain columns of particular data types. For more information, see the chapter titled “Using Sessions and Session Persistence” in the *Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server* document.
- **async-jdbc**: The same as the `jdbc` option, except the syncing of sessions to the database is done in batches, rather than synchronously.
- **cookie**: All session data is stored in cookies in the client’s web browser. If this option is chosen, only string data can be stored in the session.

In-Memory Session Replication

- Each client's session object exists on two servers:
 - Primary
 - Secondary
- The WebLogic Server session cookie stores both the client's primary and secondary servers.
- Each update to the primary session object is automatically replicated to the secondary server, either synchronously (the default) or asynchronously (batch).



ORACLE

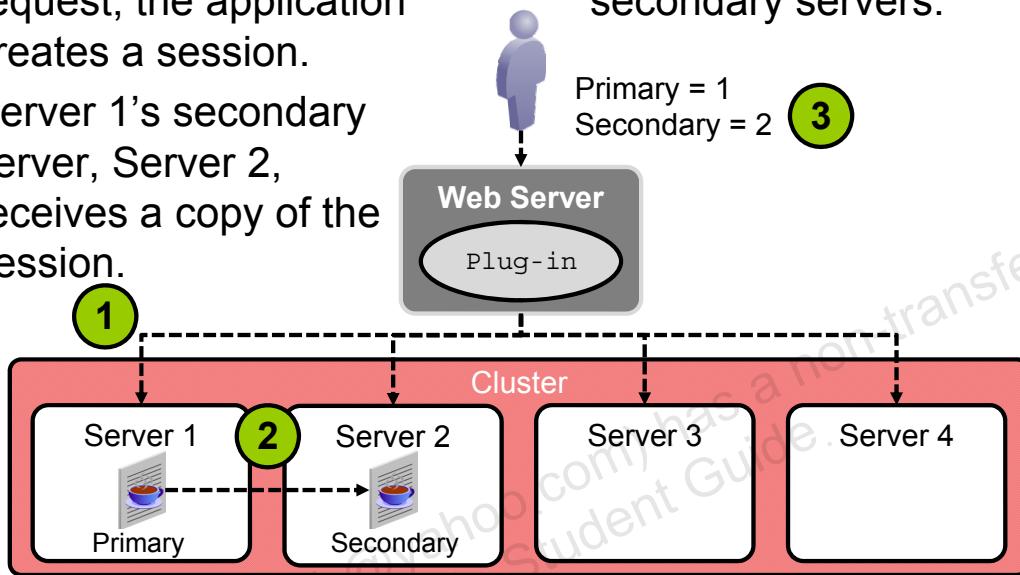
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Using in-memory replication, WebLogic Server copies session state from one server instance to another. The primary server stores the primary session state (the primary server is the server to which the client is connected when a session is first created). A replica of the session state is stored on another instance of WebLogic Server in the cluster (the secondary server). The replica is kept up-to-date so that the data can be used if the primary server fails.

The default session replication is synchronous. The asynchronous option replicates data in batches to improve cluster performance.

In-Memory Replication: Example

1. A client is load balanced to Server 1. On this request, the application creates a session.
2. Server 1's secondary server, Server 2, receives a copy of the session.
3. The cookie is written to track the primary and secondary servers.



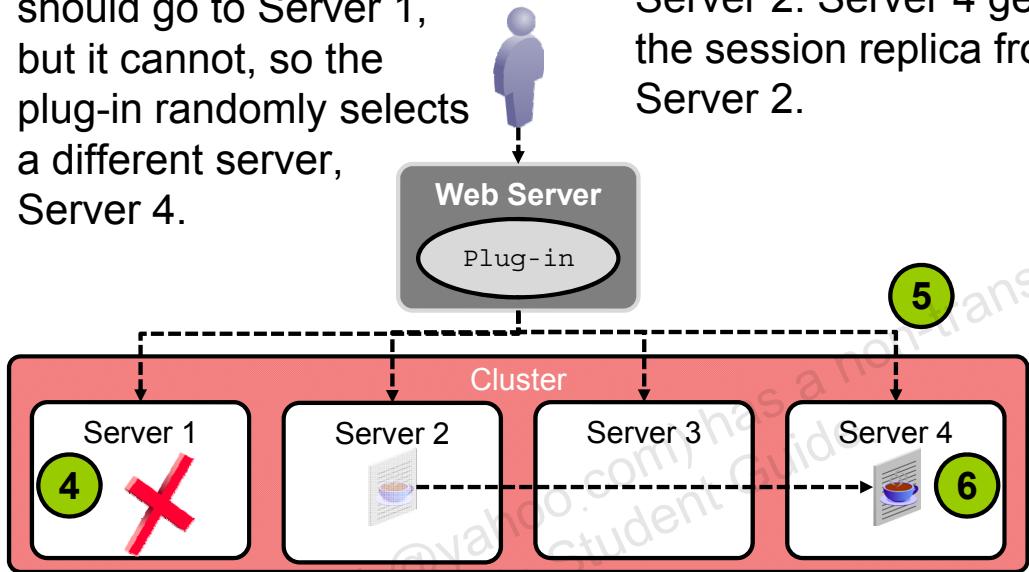
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

1. When a client first accesses the web application through the proxy plug-in, the plug-in load balances the request to one of the servers in the cluster (in the example, Server 1). The web application running on Server 1, because the application wishes to remember something about the client, creates a session for the client and stores something in it.
2. Server 1 is this client's primary server. To provide failover services for the web application, the primary server replicates the client's session state to a secondary server in the cluster. This ensures that a replica of the session state exists even if the primary server fails (for example, due to a network failure). In the example, Server 2 is the secondary server for Server 1, and gets a replica of the client's session object.
3. When WebLogic Server responds to the client, it writes a cookie to the client's browser. This cookie contains which server is the primary (Server 1, in the example) and which is the secondary (Server 2, in the example). The cookie also contains the client's session ID. Subsequent requests from this client are routed (if possible) to the primary server, Server 1, which has the client's session information.

In-Memory Replication: Example

4. Server 1 fails.
5. The client's next request should go to Server 1, but it cannot, so the plug-in randomly selects a different server, Server 4.
6. The client's cookie stores the secondary server, Server 2. Server 4 gets the session replica from Server 2.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

4. Server 1 fails.
5. When a subsequent request from the Server 1 client comes in, the proxy must use another server. It picks that server at random. In this example, Server 4 is chosen.
6. The proxy uses the client's cookie information to determine the location of the secondary server that holds the session replicas of the failed server. In this example, the secondary server is Server 2. The new primary server, Server 4, contacts the old secondary server, Server 2, and retrieves the replicated session object from it.

In-Memory Replication: Example

7. Server 4 is now the primary server for the client, and responds to the request. Server 4's secondary is Server 3.
8. The client's cookie is updated with the new primary/secondary information.



9. Server 3 stores the session replica of Server 4.

Primary = 4
Secondary = 3

8



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

7. Server 4 responds to the request. The session information for the client is available to Server 4, so the client does not realize that a different server is responding. Server 4 is the new primary server for this client.
8. Server 4 has a secondary server, in this example, Server 3. Part of Server 4's response is to update the client's cookie information. Server 4 is now the primary, with Server 3 as the secondary.
9. Server 3, as the secondary server, stores a replica of this client's session object for the new primary server, Server 4.

Configuring In-Memory Replication

- Configure in-memory replication in the `weblogic.xml` deployment descriptor.

```
...  
<session-descriptor>  
  <persistent-store-type>replicated_if_clustered  
  </persistent-store-type>  
</session-descriptor>  
...
```

`weblogic.xml`

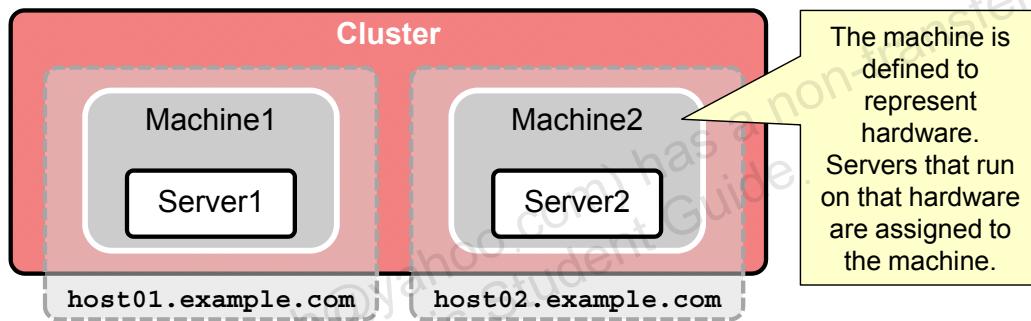


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In the `weblogic.xml` deployment descriptor file, set the `persistent-store-type` parameter in the `session-descriptor` element to `replicated`, `replicated_if_clustered`, `async_replicated`, or `async_replicated_if_clustered`.

Machines

- WebLogic Server uses machine definitions and the servers assigned to them to indicate which managed servers run on what hardware.
- WebLogic Server takes machine definitions into account when it chooses a secondary server as a backup for session information.
 - It prefers one on a different machine than the primary server.



Secondary Server and Replication Groups

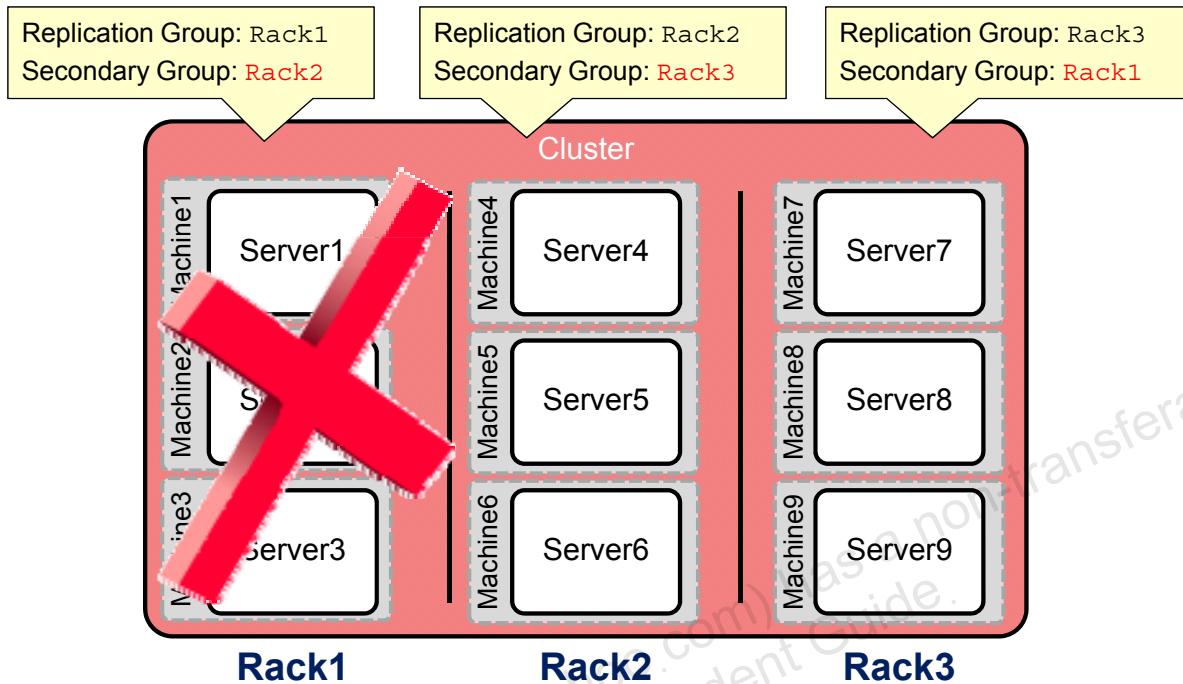
- A replication group is a logical grouping of servers in a cluster.
- WebLogic Server allows you to influence how secondary servers are chosen by configuring replication groups and configuring a server's "preferred secondary group."
- When choosing a secondary server, WebLogic Server attempts to:
 - Choose one in the primary server's preferred secondary group, if it is configured
 - Choose a server on a different machine
 - Avoid choosing a server in the same replication group



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In addition to taking into account machine definitions, WebLogic Server allows you to further control how secondary servers are chosen by using replication groups. A replication group is a preferred list of clustered instances to use for storing session replicas. When you configure a server instance that participates in a cluster, you can assign the server instance membership in a replication group. You can also assign a preferred secondary replication group to be considered for replicas of the session states that reside on the server. When a web client attaches to a cluster and a session is created, the WebLogic Server instance that is now the primary server ranks other servers in the cluster to determine which server should be the secondary. Server ranks are assigned using the server's machine and participation in a replication group.

Replication Groups: Example



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

This slide shows an example of using replication groups to persuade WebLogic Server into placing the primary sessions on servers running on a group of machines that run on the same physical rack. The cluster spans multiple machines that run on three different physical racks. In this example, all servers are configured with a replication group name that matches the rack name where they are running. So servers running on Rack1 have a replication group setting of Rack1. All servers are configured with a secondary replication group name that matches a rack name that is on a different rack. The configured secondary group for servers running on Rack1 is Rack2. This means that primary sessions in-memory on servers running on Rack1 have their secondary sessions replicated to one of the servers running on Rack2. Each server in this cluster is configured in this way to ensure that the primary session is always on a server within one rack while the secondary is located on a server in another rack. If somehow Rack1 becomes totally unavailable, client requests will fail over to other servers in the cluster and are guaranteed to recover their session state because the replication group configuration ensured that secondary sessions were located on another rack.

Configuring Replication Groups

The figure consists of three screenshots illustrating the configuration process:

- Screenshot 1:** Shows the "Domain Structure" page under "wlsadmin". The "Servers" node is highlighted with a green circle labeled "1".
- Screenshot 2:** Shows a list of servers: "AdminServer(admin)", "server1", and "server2". The "server1" entry is highlighted with a green circle labeled "2".
- Screenshot 3:** Shows the "Settings for server1" configuration page. The "Cluster" tab is selected and highlighted with a green circle labeled "3".

Below these screenshots is a configuration dialog box:

| Save | |
|--|------------------------------------|
| Use this page to define a cluster configuration for this server. A WebLogic Server cluster is a group of servers that share a common application platform. | |
| Replication Group: | <input type="text" value="Rack1"/> |
| Preferred Secondary Group: | <input type="text" value="Rack2"/> |

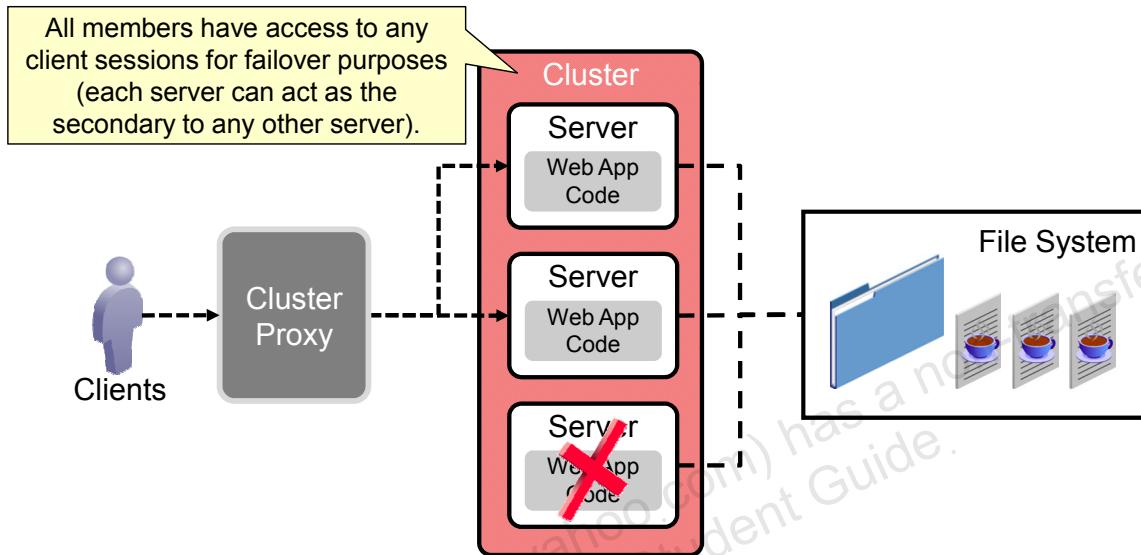
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

1. In the Domain Structure, expand **Environment** and select **Servers**.
2. Select the server for which you want to configure a replication group.
3. Select the **Configuration > Cluster** tabs.
4. Enter the name of the replication group that this server belongs to and the preferred secondary group name. Click **Save**.

File Session Persistence

File persistence stores session information in files to a highly available file system.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Session state may also be stored in the file system.

For file-based persistence:

- You must create the directory in which to store the files.
- The servers must have the appropriate access privileges.

Any server can act as the secondary server to back up any primary server. Therefore, the session cookie does not keep track of a secondary server.

Configuring File Persistence

1. Create a folder shared by all servers on the cluster on a highly available file system.
2. Assign read/write privileges to the folder.
3. Configure file session persistence in the `weblogic.xml` deployment descriptor.

```
...
<session-descriptor>
    <persistent-store-type>file</persistent-store-type>
    <persistent-store-dir>/mnt/wls_share</persistent-store-dir>
</session-descriptor>
...

```

weblogic.xml



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

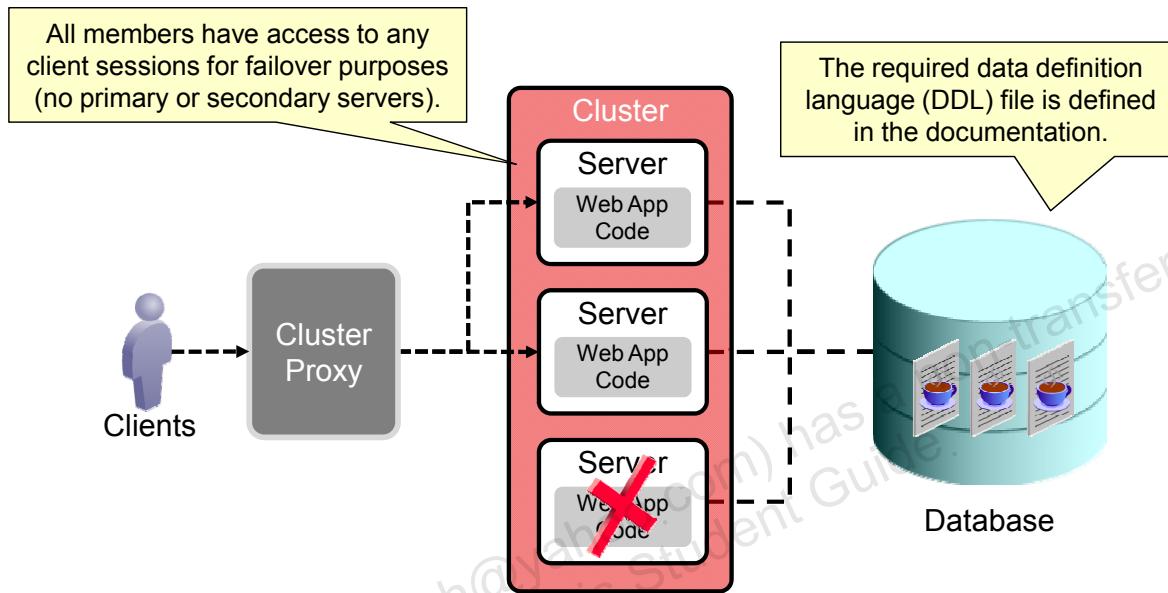
In the `weblogic.xml` deployment descriptor file, set the `persistent-store-type` parameter in the `session-descriptor` element to `file`.

Set the directory where WebLogic stores the sessions using the `persistent-store-dir` parameter. You must create this directory and make sure that appropriate access privileges are assigned to the directory.

Ensure that you have enough disk space to store the number of valid sessions multiplied by the size of each session. You can find the size of a session by looking at the files created in the location indicated by the `persistent-store-dir` parameter. Note that the size of each session can vary as the size of serialized session data changes.

JDBC Session Persistence

HTTP sessions are persisted to a database using a common JDBC data source.



ORACLE

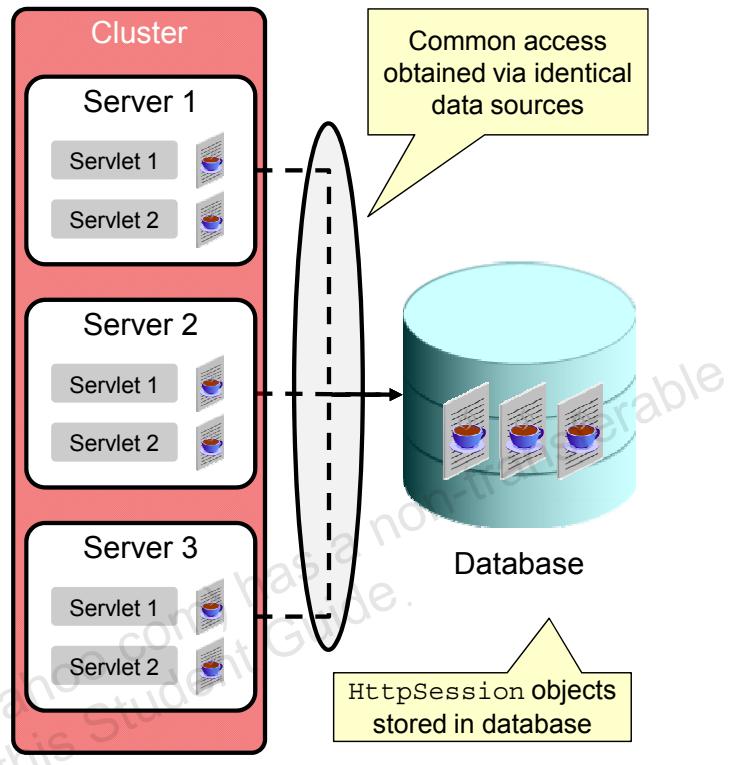
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

With Java Database Connectivity (JDBC) session persistence, a database is configured for storing `HttpSession` objects. After the database is configured, each server instance in a cluster uses an identical connection pool to share access to the database.

Whenever a web application creates or uses a session object, the WebLogic web container stores the session data persistently in the database. When a subsequent client request enters the cluster, any server in the cluster can handle the request. Each server in the cluster has identical access to the persistent store where it can look up the information needed to satisfy the client's request. This technique provides good failover capability because any server in the cluster can resolve a client's request, but there is a significant performance reduction due to the many database synchronizations required in a large web-based system. Because any server can respond to any request, the session cookie does not keep track of primary or secondary servers.

JDBC Session Persistence Architecture

- All server instances have access to all sessions.
- Subsequent requests from the same client can be handled by any server.
 - ✓ Great failover capability
 - ✗ Significant performance reduction
- Changing session objects causes (slow) database synchronization.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Whenever a servlet creates or uses a session object, the servlet stores the session data persistently in the database. When a subsequent client request enters the cluster, any server in the cluster can handle the request. Each server in the cluster has identical access to the persistent store where it can look up the information needed to satisfy the client's request. This technique provides for good failover capability because any server in the cluster can resolve a client's request, but there is a significant performance reduction due to the many database synchronizations required in a large web-based system.

Session persistence is not used for storing long-term data between sessions. That is, you should not rely on a session still being active when a client returns to a site at some later date. Instead, your application should record long-term or important information in a database.

You should not attempt to store long-term or limited-term client data in a session. Instead, your application should create and set its own cookies on the browser. Examples of this include an auto-login feature where the cookie lives for a long period or an auto-logout feature where the cookie expires after a short period of time. Here, you should not attempt to use HTTP sessions; instead you should write your own application-specific logic.

Note that even though it is legal (according to the HTTP Servlet specification) to place any Java object in a session, only those objects that are serializable are stored persistently by WebLogic.

Configuring JDBC Session Persistence

1. Create the required table in the database.
2. Create a JDBC data source that has read/write privileges for your database.
3. Configure JDBC session persistence in the `weblogic.xml` deployment descriptor.

```
...
<session-descriptor>
  <persistent-store-type>jdbc</persistent-store-type>
  <persistent-store-pool>mysessions</persistent-store-pool>
</session-descriptor>
...
```

`weblogic.xml`



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Set up a database table named `wl_servlet_sessions` for JDBC-based persistence. The connection pool that connects to the database needs to have read/write access for this table. Create indexes on `wl_id` and `wl_context_path` if the database does not create them automatically. Some databases create indexes automatically for primary keys.

Set the `persistent-store-type` parameter in the `session-descriptor` element in the `weblogic.xml` deployment descriptor file to `jdbc`.

Set a JDBC connection pool to be used for persistence storage with the `persistent-store-pool` parameter in the `session-descriptor` element in the `weblogic.xml` deployment descriptor file. Use the name of a connection pool that is defined in the WebLogic administration console.

You can use the `jdbc-connection-timeout-secs` parameter to configure the maximum duration that the JDBC session persistence should wait for a JDBC connection from the connection pool, before failing to load the session data.

To prevent multiple database queries, WebLogic caches recently used sessions. Recently used sessions are not refreshed from the database for every request. The number of sessions in cache is governed by the `cache-size` parameter in the `session-descriptor` element of the WebLogic-specific deployment descriptor, `weblogic.xml`.

JDBC Persistent Table Configuration

The WL_SERVLET_SESSIONS table must exist with read/write access:

| | Column Name | Column Data Type |
|-----------|--------------------------|------------------|
| Prim. Key | WL_ID | VARCHAR2 (100) |
| | WL_CONTEXT_PATH | VARCHAR2 (100) |
| | WL_CREATE_TIME | NUMBER (20) |
| | WL_IS_VALID | CHAR (1) |
| | WL_SESSION_VALUES | LONG RAW |
| | WL_ACCESS_TIME | NUMBER (20) |
| | WL_IS_NEW | CHAR (1) |
| | WL_MAX_INACTIVE_INTERVAL | INTEGER |



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

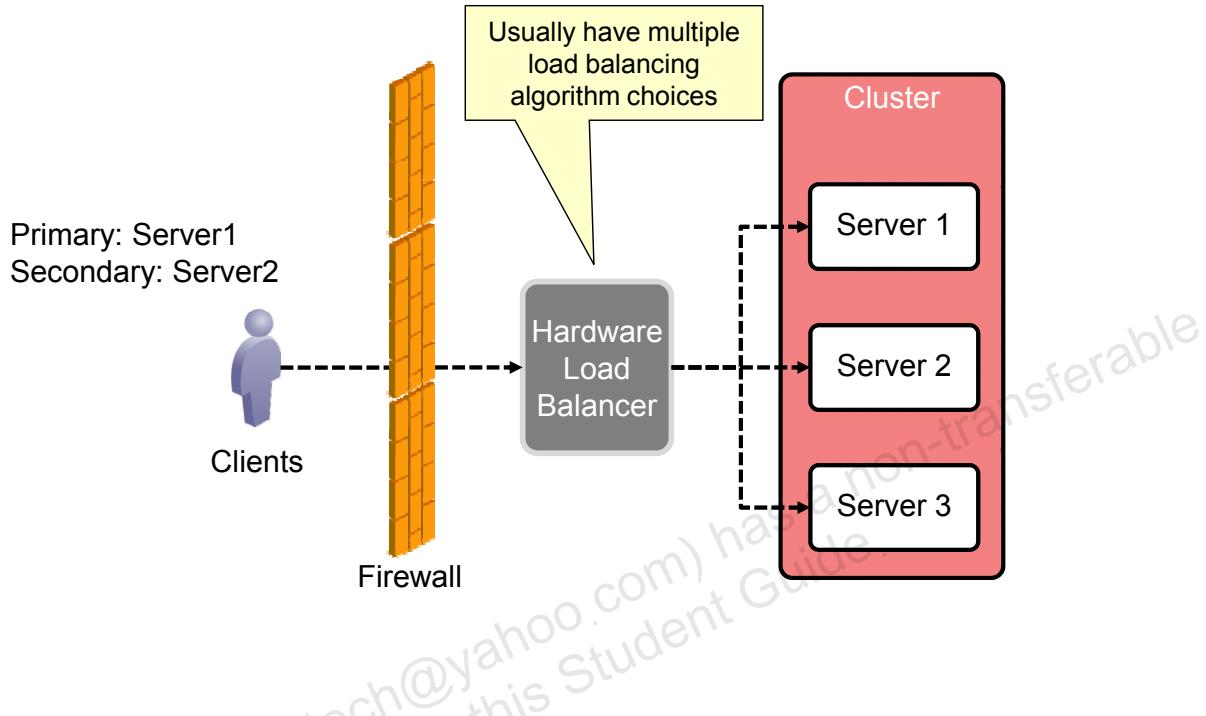
In the database that is referenced by the session persistence data source, you must configure a table, WL_SERVLET_SESSIONS, which will hold the session objects. The user specified with access to this table needs read/write/insert/delete access. The table columns are:

- WL_ID: The session ID
- WL_CONTEXT_PATH: This is the context. This column is used with WL_ID as the primary key. This is a variable-width alphanumeric data type of up to 100 characters.
- WL_IS_NEW: This value is true as long as the session is classified in the “new” state by the Servlet engine. This is a single char column.
- WL_CREATE_TIME: This is the time when the session was originally created. This is a numeric column, 20 digits.
- WL_IS_VALID: This parameter is true when the session is available to be accessed by a Servlet. It is used for concurrency purposes. This is a single char column.
- WL_SESSION_VALUES: This is the actual session data. It is a LONG RAW column.
- WL_ACCESS_TIME: This is the last time this session was accessed. This is a numeric column, 20 digits.
- WL_MAX_INACTIVE_INTERVAL: This is the number of seconds between client requests before the session is invalidated. It is an Integer. A negative value means the session should never time out.

The following is an example SQL statement to create this table, for Oracle Database:

```
CREATE TABLE WL_SERVLET_SESSIONS
(WL_ID VARCHAR2 (100) NOT NULL,
WL_CONTEXT_PATH VARCHAR2 (100) NOT NULL,
WL_IS_NEW CHAR (1),
WL_CREATE_TIME NUMBER (20),
WL_IS_VALID CHAR (1),
WL_SESSION_VALUES LONG RAW,
WL_ACCESS_TIME NUMBER (20) NOT NULL,
WL_MAX_INACTIVE_INTERVAL INTEGER,
PRIMARY KEY (WL_ID, WL_CONTEXT_PATH));
```

Configuring a Hardware Load Balancer



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Clusters that use a hardware load balancer can use any load balancing algorithm that is supported by the load balancer. If you choose to use load-balancing hardware instead of a proxy plug-in, you must use a hardware load balancer that supports secure sockets layer (SSL) persistence, passive cookie persistence, or active cookie persistence.

Hardware Load Balancer Session Persistence

- SSL Persistence
 - The load balancer performs all data encryption and decryption between clients and the WebLogic Server cluster.
 - The load balancer uses the plain text session cookie that WebLogic Server writes on the client to maintain an association between the client and the primary server
- Passive Cookie Persistence
 - The load balancer uses a string within the WebLogic Server session cookie to associate the client with the primary server. You must tell the load balancer where this string is.
- Active Cookie Persistence
 - If the load balancer creates its own cookie, and does not modify the WebLogic Server session cookie, this works without any additional configuration.

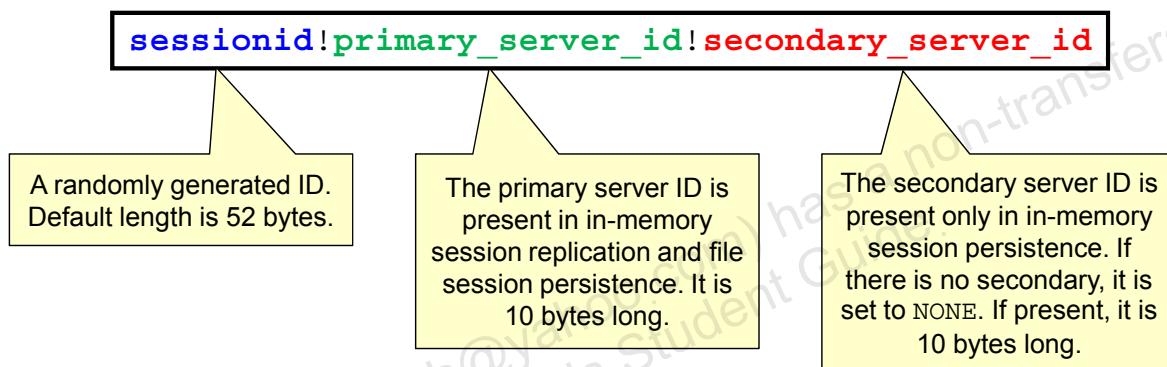


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

- If SSL persistence is used, then the load balancer performs all encryption and decryption of data between clients and the cluster. The load balancer then uses the plain-text cookie created by WebLogic Server to maintain the association between the client and the server in the cluster.
- Passive cookie persistence means the load balancer allows WebLogic Server to write its session cookie through the load balancer to the client. The load balancer, in turn, interprets an identifier in the client's cookie to maintain the relationship between the client and the primary WebLogic Server that hosts the HTTP session state.
- You can use certain active cookie persistence mechanisms with WebLogic Server clusters, provided the load balancer does not modify the WebLogic Server session cookie. If the load balancer's active cookie persistence mechanism works by adding its own cookie to the client session, no additional configuration is required to use the load balancer with a WebLogic Server cluster.

Passive Cookie Persistence and the WebLogic Server Session Cookie

- Configure a passive cookie load balancer:
 - Cookie name: JSESSIONID
 - Set the offset to 53 bytes (52 bytes for the session ID + 1 byte for the delimiter)
 - String length: 10 characters



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The hardware load balancer works with all session persistence types because the behavior is the same in each case. When the primary is not available, the load balancer uses its configured algorithm to select another server in the cluster. The secondary ID is not really used by the load balancer, but when in-memory persistence is configured the server that receives the request uses the cookie to fetch the session from the secondary session. In the other session failover types, the secondary is not used at all.

To configure a load balancer to work with your cluster, configure the load balancer to define the offset and length of the string constant. The default length of the WebLogic Session ID portion of the session cookie is 52 bytes. Configure the load balancer to set the following:

- String offset to 53 bytes: This is the default random session ID length plus one byte for the delimiter character.
- String length to 10 bytes: This is the length of the identifier for the primary server.

Quiz

In-memory session replication copies session data from one clustered instance of WebLogic Server to:

- a. All other instances of WebLogic Server in the cluster
- b. All instances of WebLogic Server in the Preferred Secondary Group
- c. All instances of WebLogic Server in the same Replication Group
- d. Another instance of WebLogic Server in the cluster



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: d

Summary

In this lesson, you should have learned how to:

- Install Oracle HTTP Server
- Configure Oracle HTTP Server as a cluster proxy
- Configure session failover
- Configure replication groups

Practice 13-1 Overview: Installing OHS (Optional)

This practice covers the following topics:

- Installing OHS from the Web Tier installer
- Creating an OHS instance



Practice 13-2 Overview: Configuring a Cluster Proxy

This practice covers the following topics:

- Configuring Oracle HTTP Server to act as a proxy to a WebLogic cluster
- Starting Oracle HTTP Server
- Testing in-memory session replication

Practice 13-3 Overview: Configuring Replication Groups

This practice covers configuring replication groups in a cluster.



Clusters – Communication, Planning, and Troubleshooting

14

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe the differences between unicast and multicast cluster communication
- Configure a replication channel for a cluster
- Describe planning for a cluster
- Monitor and troubleshoot a cluster

Review: Cluster Communication

- Cluster members communicate with each other in two ways:
 - One-to-many messages
 - For periodic “heartbeats” to indicate continued availability
 - To announce the availability of clustered services
 - **Note:** This communication can use either:
 - IP unicast: No additional configuration is required.
 - IP multicast: A multicast host and port must be configured.
 - Peer-to-peer messages
 - For replicating HTTP session and stateful session EJB state
 - To access clustered objects that reside on a remote server (multi-tier architecture)
- Note:** This communication uses sockets.



ORACLE

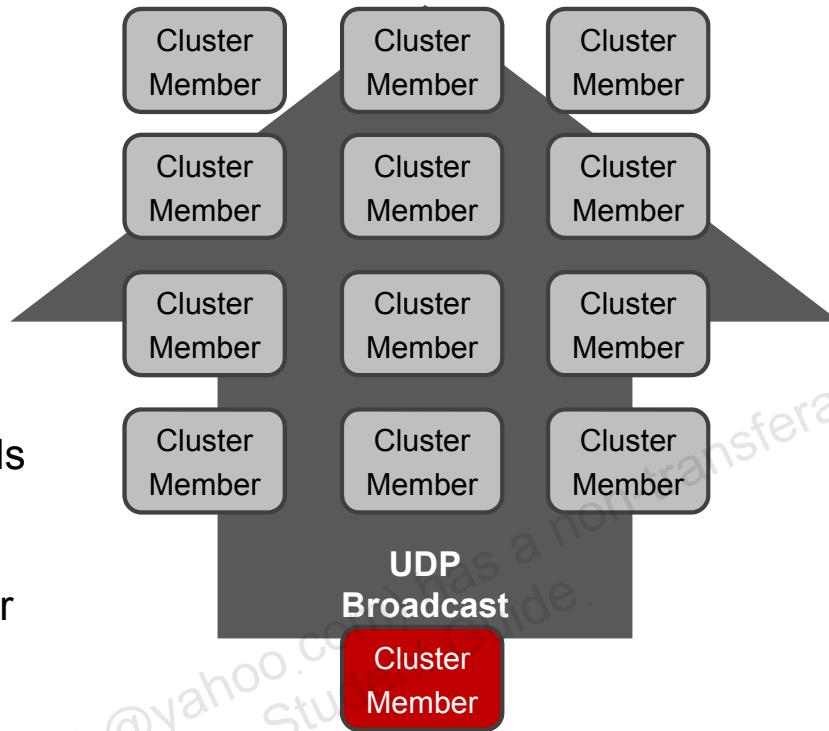
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

An instance of WebLogic Server uses one-to-many communication to send regular “heartbeat” messages that advertise its continued availability to other server instances in the cluster. The servers in a cluster listen for heartbeat messages to determine when a server has failed.

All servers use one-to-many messages to announce the availability of clustered objects that are deployed or removed locally. Servers monitor these announcements so that they can update their local JNDI tree to indicate the current deployment of clustered objects. This is the maintenance of the so-called “cluster-wide” JNDI tree.

How Multicast Works

Oracle recommends using multicast communication for certain large cluster scenarios.



ORACLE

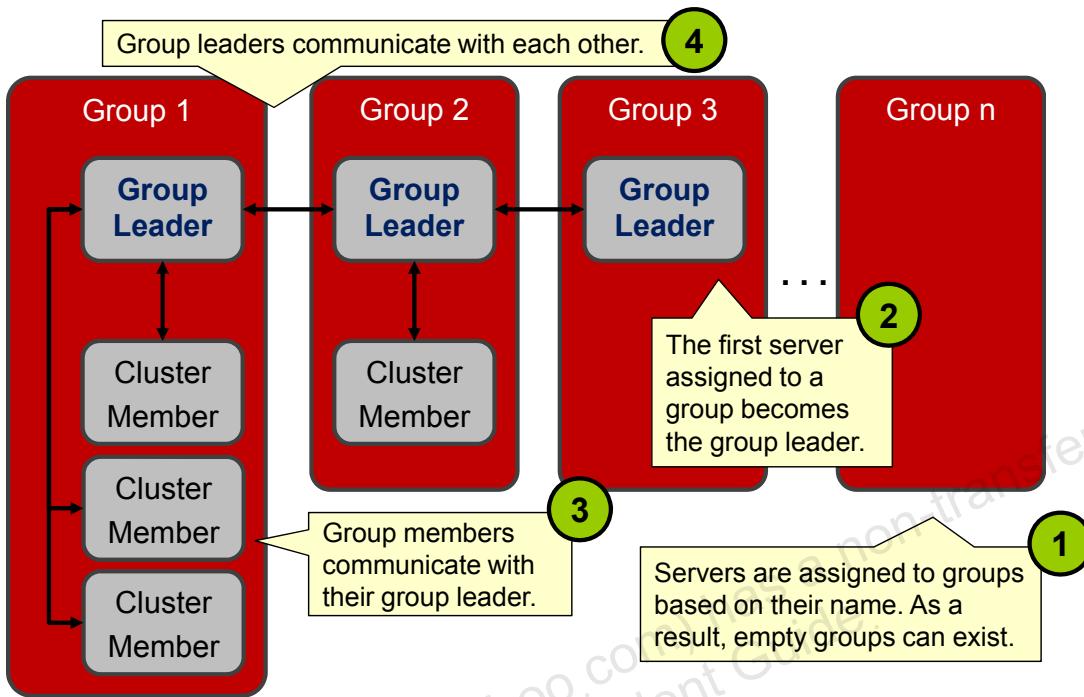
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

IP multicast enables multiple applications to subscribe to an IP address and port number, and listen for messages. A multicast address is an IP address in the range 224.0.0.0 - 239.255.255.255. IP multicast does not guarantee that messages are received, so WebLogic Server allows for the possibility that some messages may be missed. If you use multicast, you must ensure your network propagates multicast messages to all clustered servers. The multicast time-to-live value can be increased if you find that messages are being missed. With multicast, you must ensure that no other applications share the multicast address and port, or servers will have to process extra messages, which introduces extra overhead.

Firewalls can break multicast transmissions. Although it might be possible to tunnel multicast transmissions through a firewall, this practice is not recommended. A final worry with multicast messaging is the possibility of a multicast "storm," in which server instances do not process incoming messages in a timely fashion, which leads to retransmissions and increased network traffic.

Multicast messaging is also lightweight and works well for some clustered scenarios where unicast clustering is not ideal. Which protocol to use must be discerned on a case by case basis.

How Unicast Works



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The TCP protocol requires a point-to-point connection. Unlike multicast, which broadcasts to many servers simultaneously, unicast needs to create a connection for each server in the cluster. WLS would not scale well if every server in a cluster had to connect to every other server in the cluster. WebLogic implements unicast to scale well. When a cluster starts, the servers divide the cluster into up to ten groups of cluster members. One member in each group becomes the group leader. This creates a network topology that reduces the number of connections an individual member makes with other members in the cluster.

The picture in this slide shows a unicast cluster with several cluster members. Servers are assigned to one of up to ten server groups based on their name, which is comprised of a name and a numeric value typically. The first server assigned to a group becomes the group leader for that group. WebLogic allocates ten server groups, and because servers are assigned to groups based on their naming conventions there may be groups with no cluster members, as shown in group n.

Group members send and receive cluster messages through their group leader, and group leaders communicate with each other to make cluster traffic scalable. Group leaders act as simple network relays to their group members, and to other group leaders. Group leaders and members can receive multiple messages because they do not store any state data, so there is no risk of data corruption.

If a group leader is not available, another group member becomes the new group leader. If the original group leader becomes available again, the old group leader becomes the group leader again, and the cluster demotes the acting group leader back to a regular group member.

Considerations for Choosing Unicast or Multicast

| Multicast | Unicast |
|--|---|
| Uses UDP multicast | Uses TCP/IP |
| Requires more router configuration, time-to-live (TTL) when crossing subnets | Requires no additional configuration |
| Requires configured listen address and port | Uses WebLogic default channel or custom unicast channel |
| Each message sent and received directly from the network | Each message is delivered via a group leader (1-3 network hops) |
| Every server is visible to each other | Servers see only their group leader. Group leaders see their servers and other group leaders. |
| Three missed cluster messages for removal from the cluster | One missed cluster message for removal from the cluster |



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Unicast is the default protocol because it simplifies out of the box cluster configuration and because it is likely to meet the majority of user requirements. However, Oracle fully supports both protocols equally. Both protocols require that the cluster members get sufficient processing time to send and receive cluster messages in a timely fashion. This prevents unnecessary cluster membership changes and the inherent resynchronization costs associated with leaving and rejoining the cluster. It is recommended that you eliminate unnecessary cluster membership changes due to over-utilization of available resources.

When using unicast in particular, make sure that the group leaders are not resource constrained since they act as the message relay to deliver a cluster message to the rest of the cluster. Any slowness on their part can impact multiple cluster members and even result in the group electing a new group leader. You can set up a separate network channel for unicast communication, but it is not required. If no separate channel is defined, each server's default channel is used.

Contrast this with multicast, where a slow member can only really impact its own membership to the cluster. Multicast clusters are generally more efficient in terms of cluster message propagation, and therefore tend to be more resilient to oversubscription of resources. For these reasons, multicast may be a better option for very large clusters with high throughput requirements, provided the network environment supports WebLogic Server cluster UDP requirements.

Configure Multicast

First, you should test if the multicast address you want to use is working using the MulticastTest tool.

```
./setDomainEnv.sh
java utils.MulticastTest -n hello -a 237.0.0.1 -p 30000
```

Command Line Host 1

```
./setDomainEnv.sh
java utils.MulticastTest -n world -a 237.0.0.1 -p 30000
.

Using multicast address 237.0.0.1:30000
Will send messages under the name server1 every 2 seconds
Will print warning every 600 seconds if no messages are received
    New Neighbor hello found on message number 2
        I (world) sent message num 1
Received message 3 from hello
        I (world) sent message num 2
Received message 2 from world
Received message 4 from hello
        I (world) sent message num 3
```

Command Line Host 2

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can verify that multicast is working by running `utils.MulticastTest` from one of the Managed Servers.

The `MulticastTest` utility helps you to debug multicast problems when configuring an Oracle WebLogic Server cluster. The utility sends out multicast packets and returns information about how effectively the multicast is working on your network. Specifically, `MulticastTest` displays the following types of information via standard output:

1. A confirmation and sequence ID for each message sent out by the current server
2. The sequence and sender ID of each message received from any clustered server, including the current server
3. A missed-sequenced warning when a message is received out of sequence
4. A missed-message warning when an expected message is not received

To use `MulticastTest`, start one copy of the utility on each node on which you want to test the multicast traffic.

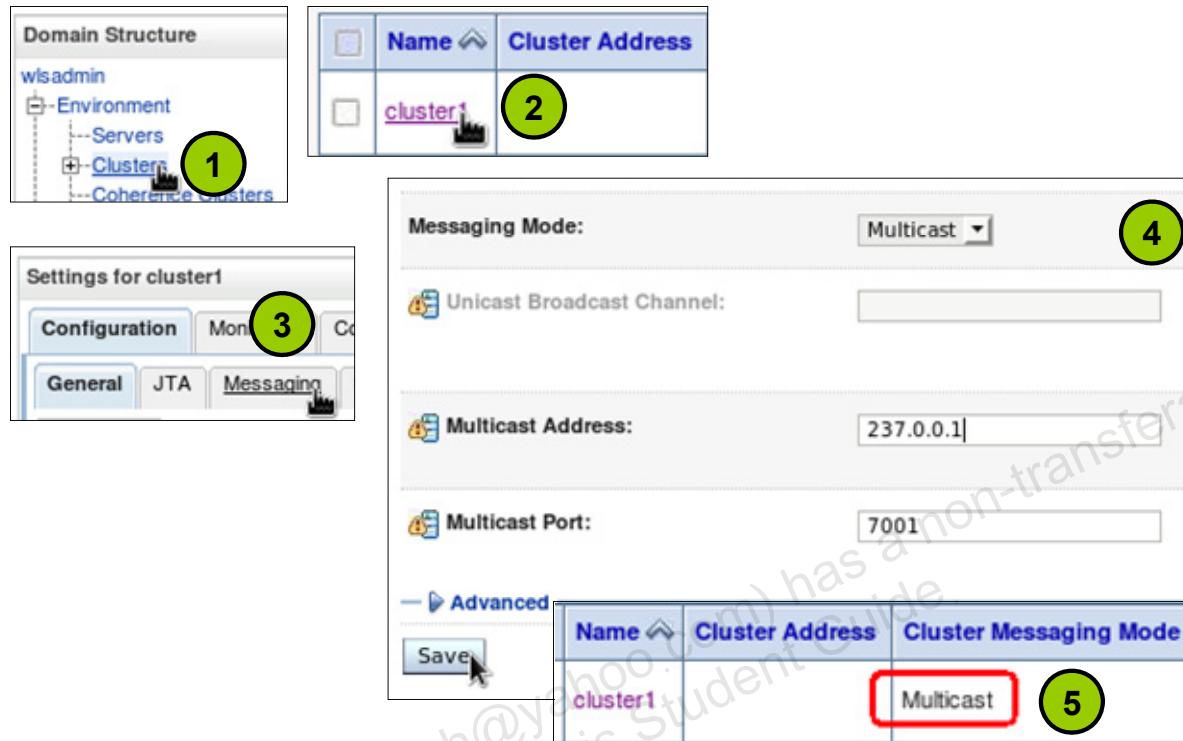
Warning: Do not run the `MulticastTest` utility by specifying the same multicast address (the `-a` parameter) as that of a currently running Oracle WebLogic Server cluster. The utility is intended to verify that the multicast is functioning properly before your clustered Oracle WebLogic Servers are started.

Syntax

```
$ java utils.MulticastTest -n name -a address [-p portnumber]  
[-t timeout] [-s send]
```

- **-n name** (required): A name that identifies the sender of the sequenced messages. Use a different name for each test process that you start.
- **-a address**: The multicast address on which: (a) the sequenced messages should be broadcast; and (b) the servers in the clusters are communicating with each other. (The default is 237.0.0.1.)
- **-p portnumber** (optional): The multicast port on which all the servers in the cluster are communicating. (The multicast port is the same as the listen port that is set for Oracle WebLogic Server, which defaults to 7001 if unset.)
- **-t timeout** (optional): Idle timeout, in seconds, if no multicast messages are received. If unset, the default is 600 seconds (10 minutes). If a timeout is exceeded, a positive confirmation of the timeout is sent to stdout.
- **-s send** (optional): Interval, in seconds, between sends. If unset, the default is 2 seconds. A positive confirmation of each message that is sent out is sent to stdout.

Configure Multicast



ORACLE

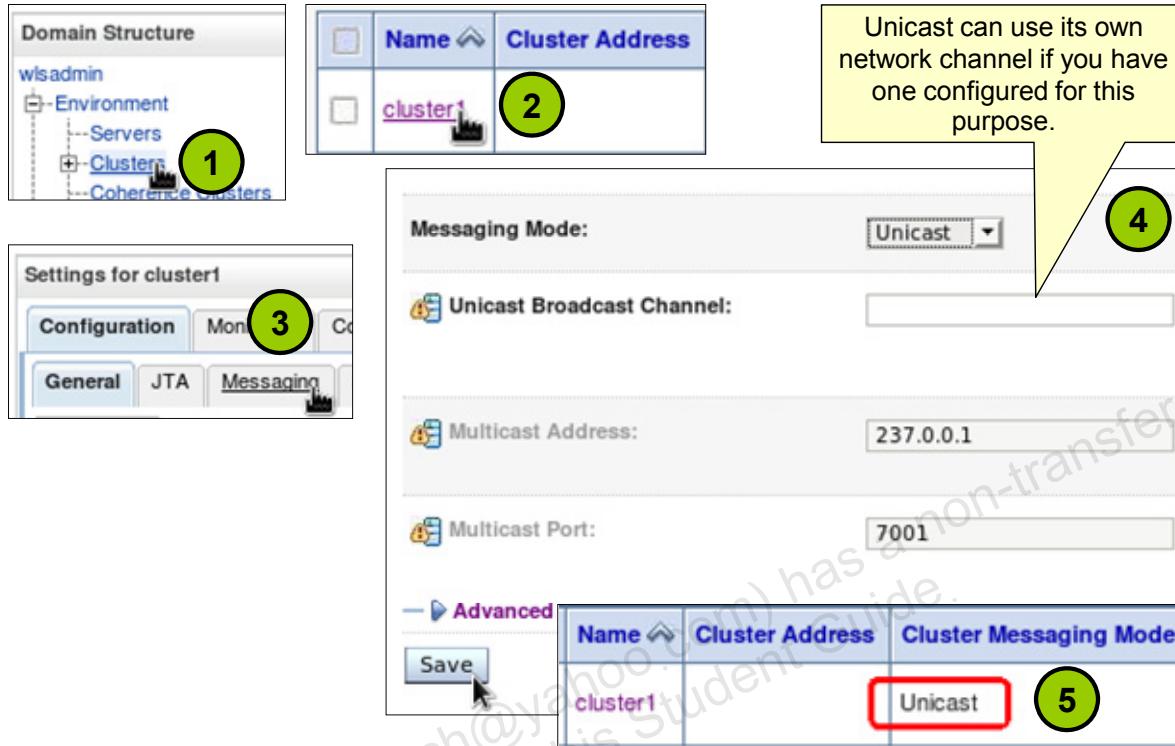
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You configure clustering using the following steps:

1. In the administration console, expand **Environment** and select **Clusters**.
2. Select the cluster you want to configure to use multicast communication.
3. Select the **Configuration > Messaging** tabs to display the cluster's broadcast communication settings page.
4. Set the Messaging Mode to **Multicast**, set the Multicast IP address, and Multicast Port. In this case, **237.0.0.1** is configured as the IP address and **7001** as the port. Save your changes.
5. Review your cluster and see that its Cluster Messaging Mode is now Multicast.

Note: This change requires restarting the affected servers of the cluster. Note that when you select the multicast messaging mode, the unicast settings are unavailable.

Configure Unicast



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

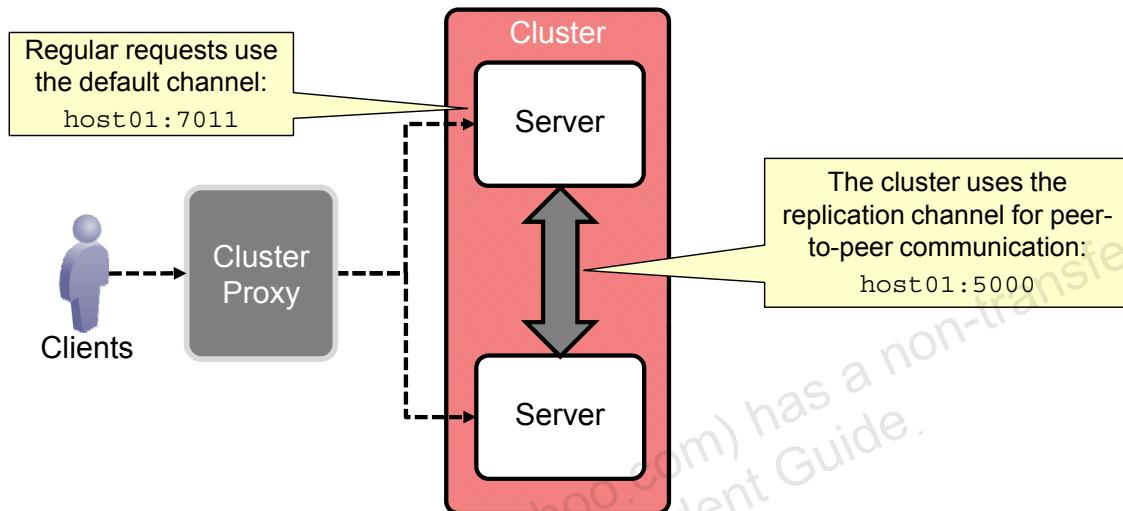
You configure clustering using the following steps:

1. In the administration console, expand **Environment** and select **Clusters**.
2. Select the cluster you want to configure to use unicast communication.
3. Select the **Configuration > Messaging** tabs to display the cluster's broadcast communication settings page.
4. Set the **Messaging Mode** to **Unicast**. There is also a field for entering the name of a network channel if you want to have unicast traffic on its own channel. In this case, we are just allowing traffic to use the default channel. Save your changes.
5. Review your cluster and see that its Cluster Messaging Mode is now **Unicast**.

Note: This change requires restarting the affected servers of the cluster. Note that when you select the unicast messaging mode, the multicast settings are unavailable.

Replication Channel

WebLogic Server allows you to configure a separate network channel for peer-to-peer cluster communication (replication).



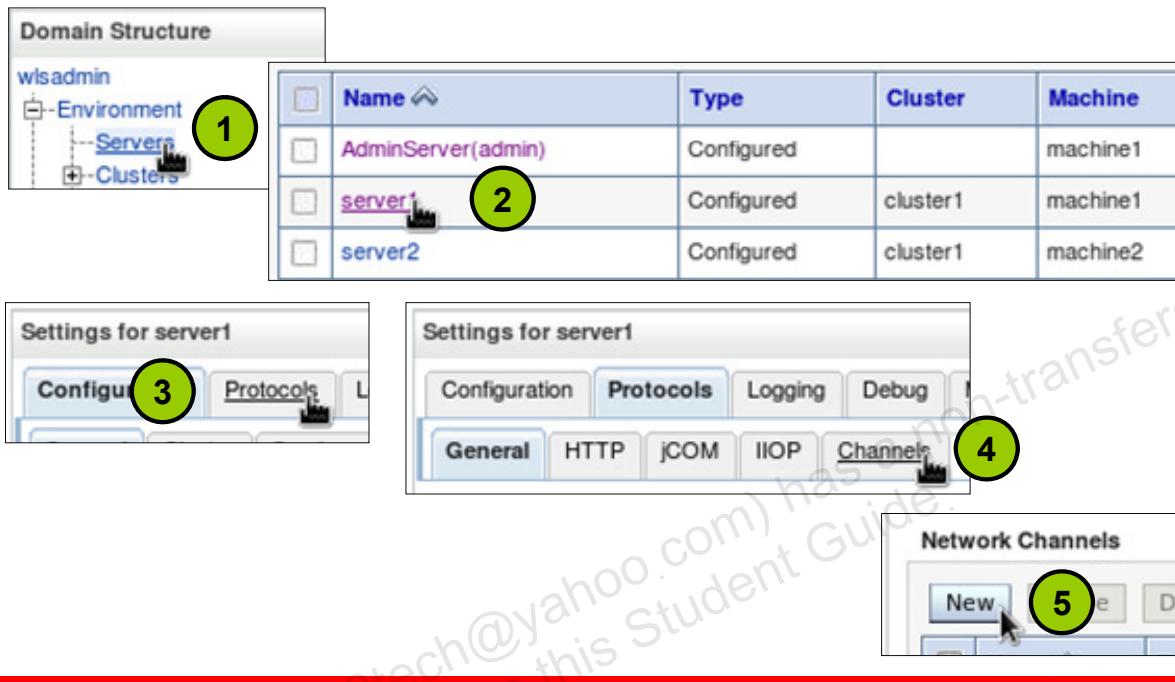
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In a corporate enterprise environment, network volumes can reach tremendous levels that may saturate a network. If this network is the same one that the WebLogic default channel uses for its servers, this can hinder high-priority or internal traffic. One example of this is session replication. Some applications may be more replication-intensive than others and can benefit from separating replication traffic from other traffic in the server. In other scenarios, such as when using Exalogic InfiniBand, some network stacks offer much higher performance than standard TCP-IP networks. WebLogic applications can benefit from using a faster network for replication if there is a lot of replication traffic. This allows client traffic and replication traffic to operate on different networks to avoid saturating the network.

Configure Replication Channels: Servers

First, configure each server with a network channel:

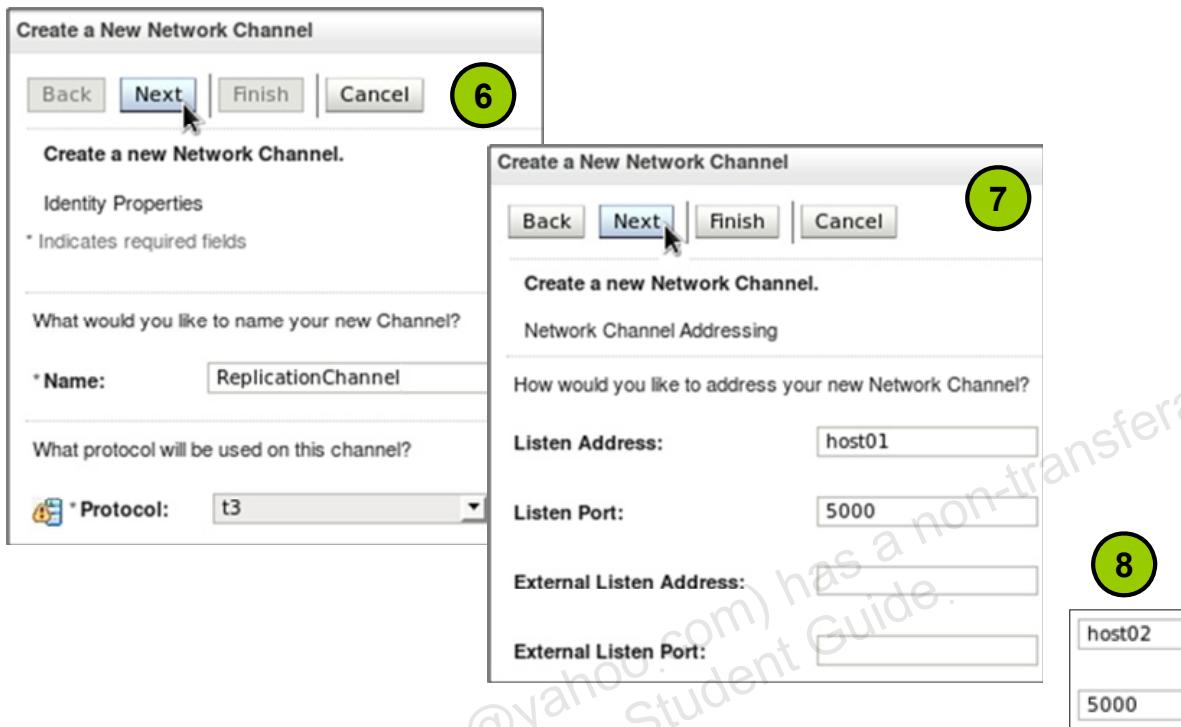


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

1. Within the administration console, expand **Environment** and select **Servers**.
2. Select the server for which you want to create a channel.
3. Select the **Protocols** tab.
4. Select the **Channels** subtab to display the list of configured channels for this server.
5. Click **New** to create a new channel.

Configure Replication Channels: Servers



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

6. Enter a name and select a protocol for your channel. In this case, the name is `ReplicationChannel` and the protocol is `t3`. Click **Next**.
7. Configure the network addressing for your channel. In this case, the listen address is `host01` and the listen port is `5000`. Click **Next**.
8. This procedure is repeated for each server in the cluster. Ensure that the network channel has the same name for each server. Assuming a two-node cluster for this example, this procedure is repeated for the second server with a listen address and port of `host02` and `5000`, respectively.

Configure Replication Channels: Servers

Create a New Network Channel

Back | **Next** | Finish | Cancel

Create a new Network Channel.

Network Channel Properties

Set any additional properties for this channel. The following checkboxes are available:

- Enabled
- Tunneling Enabled
- HTTP Enabled for This Protocol
- Outbound Enabled

Create a New Network Channel

Back | Next | **Finish** | Cancel

Create a new Network Channel.

Secure Network Channel

Define the security configuration of this network.

This server is configured to use Demo Identity A.

- Two Way SSL Enabled
- Client Certificate Enforced

| Name | Protocol | Enabled | Listen Address | Listen Port |
|--------------------|----------|---------|----------------|-------------|
| ReplicationChannel | t3 | true | host01 | 5000 |

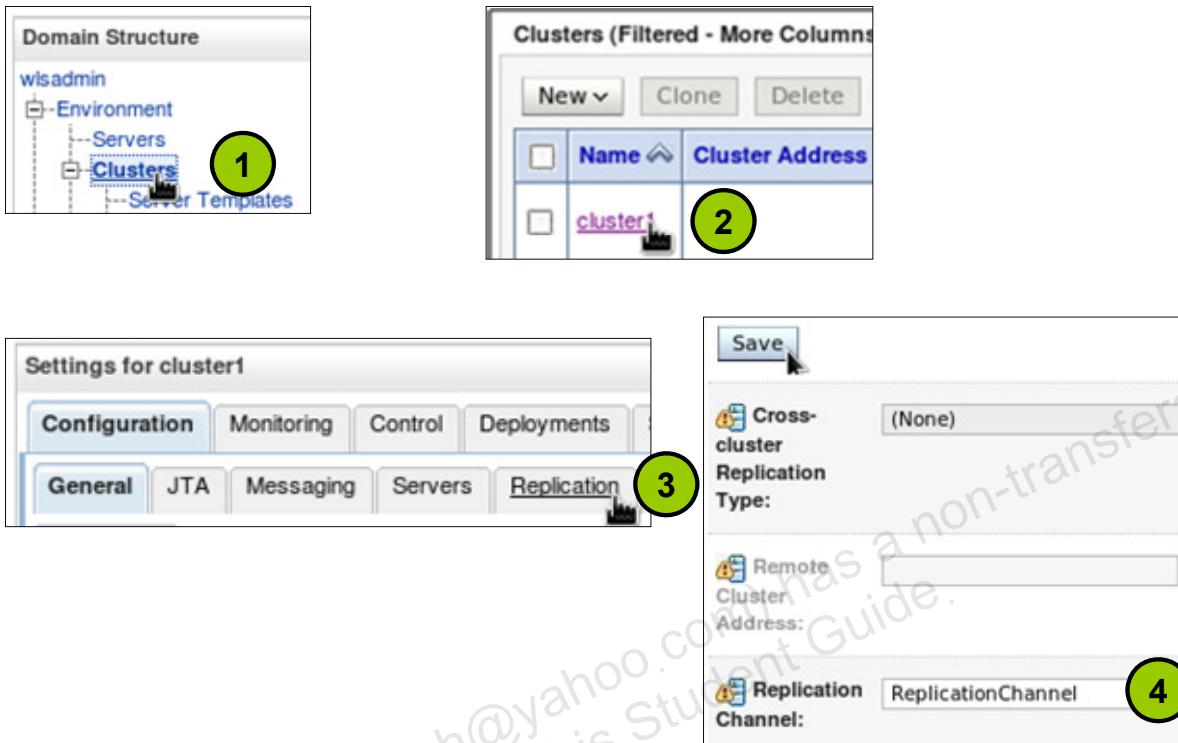
| Name | Protocol | Enabled | Listen Address | Listen Port |
|--------------------|----------|---------|----------------|-------------|
| ReplicationChannel | t3 | true | host02 | 5000 |

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

9. Next, you configure the properties for your channels. In this example, the channel is enabled, HTTP is enabled, and it allows for outbound communication. A replication channel must allow for outbound communication so it can both send and receive replication messages. Click **Next**. Do the same thing for each of the cluster server channels.
10. If there are any SSL requirements, you configure them on this page. In this example, you are not using SSL so you just click **Finish**.
11. After your network channels are created for all the servers in your cluster, you can view them in the console. Now you have to configure your cluster to use this channel for replication.

Configure Replication Channels: Cluster



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

1. Within the administration console, expand **Environment** and select **Clusters**.
2. Select the cluster you want to configure to use the new replication channel.
3. Select the **Replication** subtab under the **Configuration** tab.
4. Enter the name of the replication channel for this cluster to use. In this example, the same name that was used when creating the channels on each server in the cluster, **ReplicationChannel** is used. This tells the cluster to use the channel named **ReplicationChannel** for all replication traffic.

Note: You can optionally use SSL to secure your replication channel; however, doing so can potentially cause a slowdown in performance because most replication is done synchronously. This means that when a client updates its session state, that client waits for WebLogic to finish updating the state of the secondary session before getting control back.

Configure Replication Channels

You can verify that your replication channel is enabled by checking the system out of each server in the cluster.

```
<Notice> <Server> <BEA-002613> <Channel "ReplicationChannel">  
is now listening on 192.0.2.11:5000 for protocols t3, CLUSTER-  
BROADCAST, http.>
```

server1 output

```
<Notice> <Server> <BEA-002613> <Channel "ReplicationChannel">  
is now listening on 192.0.2.12:5000 for protocols t3, CLUSTER-  
BROADCAST, http.>
```

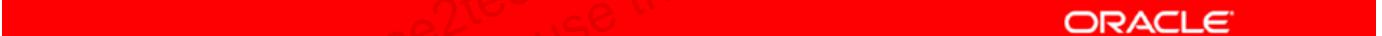
server2 output

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Planning for a Cluster

1. Determine your cluster architecture.
 - Basic
 - Multi-tier
2. Consider your network and security topologies.
 - A. Where to place firewalls
 - Do not place firewalls in between cluster members.
 - B. Decide on one-to-many cluster communication
 - Multicast
 - Unicast
3. Determine the type of cluster you will define.
 - Regular cluster
 - Dynamic cluster

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Planning for a Cluster

4. Choose hosts for the cluster.
 - A. Note each host's DNS name or IP address. DNS or virtual host names are recommended.
 - B. Choose the port number for each managed server*. Note the admin server host and port.
 - C. Decide on the names of servers*, machines, clusters, and so on (each WebLogic resource must have a unique name).
 - D. Start with one managed server per CPU core.
 - You can scale up later based on performance testing.
5. Choose your cluster proxy
 - Web server with a proxy plug-in
 - Hardware load balancer

* With Dynamic Clusters, some of these values are generated. For example, the server name prefix or starting port number is defined.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

For production environments, the use of DNS names is generally recommended. Virtual host names could also be used. The use of IP addresses can result in translation errors if:

- Clients connect to the cluster through a firewall, or
- You have a firewall between the web application and EJB tiers

You can avoid translation errors by binding the address of an individual server instance to a DNS name. Make sure that a server instance's DNS name is identical on each side of firewalls in your environment.

Oracle recommends that you start with one server per CPU core and then scale up based on the expected behavior. You should test the actual deployment with your system to determine the optimal number and distribution of server instances.

Planning for a Cluster

6. Decide how to handle HTTP session failover.
 - In-memory replication
 - File storage
 - JDBC storage
 - Coherence*Web
7. If using the multi-tier architecture with EJBs, decide on the EJB load balancing algorithm.
 - Round-robin
 - Random
 - Weight-based
8. Decide how pinned services will be handled.
 - Service-level migration
 - Whole server migration



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

EJB load balancing choices are:

- Round-robin (the default): The algorithm cycles through a list of WebLogic Server instances in order. The advantages of the round-robin algorithm are that it is simple, quick, and very predictable. The disadvantage is it treats all servers the same (even though you may have some that are faster than others).
- Random: The algorithm routes requests to servers at random. The disadvantages of random load balancing include the slight processing overhead incurred by generating a random number for each request, and the possibility that the load may not be evenly balanced over a small number of requests.
- Weight-based: Each server hosting EJBs can be given a weight. Select the server in the Servers table. Select **Configuration > Cluster**. Enter a number between 1 and 100 in the **Cluster Weight** field. This value determines what proportion of the load the server will bear relative to other servers in the EJB cluster.

A pinned service is one that is active on only one cluster host. JTA (transaction) recovery and JMS Servers are pinned services.

Service-level migration is migrating a pinned service from a failed cluster member to one that is active. It can be done manually or occur automatically.

Whole-server migration is an entire instance of WebLogic Server migrated to a different physical machine upon failure. It, too, can be done manually or happen automatically.

Managing a Cluster

Select the cluster in the Clusters table and click the **Control** tab. The **Start/Stop** subtab shows the servers in the cluster and allows you to start, stop, suspend, and resume them.

The screenshot shows the 'Settings for cluster3' interface. The 'Control' tab is active. Below it, the 'Start/Stop' subtab is selected. A table titled 'Managed Server Instances in this Cluster' lists three servers: 'cluster3-server-1' and 'cluster3-server-2' on 'machine1', and 'cluster3-server-2' on 'machine2'. The 'Start' button in the toolbar is highlighted with a mouse cursor. A yellow callout box contains the text: 'The same functions as under the Servers table Control tab.'

| Server | Machine | Listen Port | State |
|-------------------|----------|-------------|----------|
| cluster3-server-1 | machine1 | 7101 | SHUTDOWN |
| cluster3-server-2 | machine2 | 7102 | SHUTDOWN |

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Migration subtab allows you to manually migrate “singleton” services from one server in the cluster to another. Service-level and whole server migration are covered in the *Oracle WebLogic Server 12c: Administration II* course.

Troubleshooting a Cluster

When there are issues with a cluster, you have tools to help:

- WebLogic Server logs
- OHS logs
- Monitoring by using the administration console or the Monitoring Dashboard
- WLST

Common problems include:

- OHS to WebLogic Server connectivity issues
- Multicast communication issues (if using multicast)
- Cluster member uniformity problems
- Session failover issues



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Monitoring a Cluster: Admin Console

- In the administration console: Select the cluster from the Clusters table, use its Monitoring tab and its subtabs.

Settings for cluster3

Configuration Monitoring Control Deployments Services Notes

Summary Health Failover

Server Status (Filtered - More Columns Exist)

Showing 1 to 2 of 2 Previous Next

| Name | State | Drop-out Frequency | Heap Free Current | Heap Size Current | Open Sockets |
|-------------------|---------|--------------------|-------------------|-------------------|--------------|
| cluster3-server-1 | RUNNING | Never | 197064912 | 322437120 | 3 |
| cluster3-server-2 | RUNNING | Rarely | 207932504 | 322699264 | 3 |

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The table was customized to show each cluster server: name, state, how often it left the cluster (went down), free heap memory, total heap size, and number of open sockets.

WebLogic Server and OHS Logs

- The WebLogic Server logs contain cluster subsystem messages.
 - Set debug flags to generate more detailed log messages:
 - In the administration console, select the server from Servers table, click the **Debug** tab, expand scopes, and select flags.
- The OHS logs are found here:
`<DOMAIN_HOME>/servers/<instance>/logs`
 - The OHS error log: `<OHS_INSTANCE_NAME>.log`
 - Records OHS errors, but can be configured to record events.
 - The OHS access log: `access_log`
 - Records which components and applications are accessed and by whom



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

There are cluster debug flags in the `weblogic.core.cluster`, `weblogic.cluster`, and `weblogic.servlet.internal.session` scopes.

There are two types of logs for Oracle HTTP Server. Error logs record server problems, but can also be configured to record other server events. Access logs record which components and applications are being accessed and by whom. The location of the OHS logs is configurable, as are the names of the log files. The location and names given in the slide are their defaults.

Common OHS to WLS Connectivity Issues

- Connectivity problems can cause unnecessary failovers or HTTP errors to be sent to the client.
- Causes of unexpected connection failures include problems with these OHS parameters:
 - WebLogicCluster (the initial list of cluster members)
 - If this list is incorrect, the plug-in may not be able to proxy.
 - ConnectTimeoutSecs (how long the plug-in waits to establish a connection)
 - If this is set too low, the plug-in can give up on a server and not connect to it.
 - ConnectRetrySecs (pause time before retrying a connection)
 - If this is accidentally set higher than ConnectTimeoutSecs, the plug-in will always time out during a retry.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Common OHS to WLS Connectivity Issues

- Causes of unexpected request failures include problems with these OHS parameters:
 - `WLIOTimeoutSecs` (the amount of time the plug-in waits for a response from WebLogic Server)
 - If this is set too low, and WebLogic Server sometimes takes a long time to process a request, that server will be considered dead by OHS, even though it is not.
 - `MaxPostSize` (the size of a post)
 - If this is set too low on either the proxy or on the WebLogic Server instance, a request can fail because the request is too large.
 - `MaxSkipTime` (the wait time before the plug-in retries a server marked as “bad”)
 - If this is set too high, the proxy will be slow to use a restarted cluster member, affecting overall performance.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The `WLIOTimeoutSecs` parameter should typically be set to a large value (the default is five minutes). If the value is less than the time your application takes to process a request, then you may see unexpected results.

If the `MaxPostSize` parameter is greater than or equal to the same WLS setting, it will have no effect. The Max Post Size setting for an instance of WebLogic Server can be found in the admin console under a server's **Protocols > HTTP** tabs.

Multicast Communication Issues

- Problem with the multicast address
 - For each cluster on the network, the combination of the multicast address and port must be unique.
 - Ensure no other applications use that address and port.
- Missed multicast messages (heartbeats) can cause cluster members to be marked as “failed.”
 - Ensure the multicast time to live (TTL) value is large enough for the messages to get to all cluster members.
 - If multicast buffers fill up, messages are missed.
 - Increase the size of the multicast buffer.
 - Increase the multicast send delay, which helps avoid buffer overflow.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Multicast address and port configuration problems are among the most common reasons why a cluster does not start or a server fails to join a cluster. The following considerations apply to multicast addresses:

- The multicast address must be an IP address between 224.0.0.0 and 239.255.255.255 or a host name with an IP address in this range.
- Address conflicts within a network can disrupt multicast communications. Use the netstat utility to verify that no other network resources are using the cluster multicast address. Verify that each machine has a unique IP address.
- The value of the multicast time-to-live (TTL) parameter for the cluster must be high enough to ensure that routers do not discard multicast packets before they reach their final destination.
- Increasing the size of the multicast buffers can improve the rate at which announcements are transmitted and received, and prevent multicast storms. (A multicast storm is the repeated transmission of multicast packets on a network. Multicast storms can stress the network, potentially causing end-stations to hang or fail.)
- Multicast send delay specifies the amount of time the server waits to send message fragments through multicast. This delay helps to avoid OS-level buffer overflow.

Cluster Member Uniformity

- Every instance of WebLogic Server in a cluster should be like every other. All servers should:
 - Be the same version of WebLogic Server
 - Have the same CLASSPATH
 - Have the same deployments
 - Have the same services (like data sources)
- When cluster members are not the same, you have intermittent problems, which are very hard to debug.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

As an example, let us say five of the six cluster servers have the data source targeted to them. When clients are routed to five of the servers in the cluster, there are no problems when the application tries to use the database. When a client is routed to the sixth server, however, the application gives them errors, because the data source cannot be found.

Session Failover Issues

- Session replication or persistence problems often result in the loss of session data. This affects your clients:
 - A client must log in again.
 - The client's shopping cart items disappear.
- Typical culprits include:
 - Invalid session persistence settings
 - Session or cookie timeout settings are too low.
 - The developers of the web application did not use the HttpSession API appropriately.
 - The developers of the web application are storing non-serializable objects in the session.
 - Objects must be serializable so they can be streamed from the primary server to the secondary server (in-memory replication) or to files or the database (file or database persistence).



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To serialize an object means to convert its state to a byte stream so that the byte stream can be reverted into a copy of the object. A Java object is serializable if its class or any of its superclasses implements either the `java.io.Serializable` interface or its subinterface, `java.io.Externalizable`.

Quiz

A replication channel is:

- a. Another name for replication group
- b. Another name for the preferred secondary group
- c. A network channel used by cluster members for peer-to-peer communication
- d. The title of the tab in the Monitoring Dashboard that shows cluster charts



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: c

Summary

In this lesson, you should have learned how to:

- Describe the differences between unicast and multicast cluster communication
- Configure a replication channel for a cluster
- Describe planning for a cluster
- Monitor and troubleshoot a cluster

Practice 14-1 Overview: Configuring a Replication Channel

This practice covers configuring a replication channel for a cluster.

Unauthorized reproduction or distribution prohibited. Copyright© 2016, Oracle and/or its affiliates.

Tony Albrecht (base2tech@yahoo.com) has a non-transferable
license to use this Student Guide.