



Hardware and Software
Engineered to Work Together



Custom CW: Oracle WebLogic Server 12c Administration

Student Guide - Volume 2

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 13: Lesson 16 - WebLogic Server Security	9
Objectives	10
Some Security Terms	11
Some Security Terms: Graphically	12
WebLogic Server Security Realm	13
What the Providers Do	14
Security Stores	17
Default Security Store Implementation	18
Default Security Configuration	19
Security Customization Approaches	20
Authentication Providers	21
Available Authentication Providers	22
Lightweight Directory Access Protocol (LDAP)	24
LDAP Structure	25
LDAP Search Operations	26
LDAP Query Basics	27
LDAP Authentication Providers	28
Available LDAP Authentication Providers	29
Creating a New LDAP Authentication Provider	30
Configuring the LDAP Provider: Connection	31
Configuring the LDAP Provider: Users	32
Configuring the LDAP Provider: Groups	33
Configuring the LDAP Provider: Subgroups	35
Configuring the LDAP Provider: Dynamic Groups	36
LDAP Failover	37
LDAP Caching	38
Multiple Authentication Providers	39
Control Flags	40
Administration Groups	42
Troubleshooting Authentication	43
Auditing Provider	44
Security Audit Events	45
Configuring the Auditing Provider	46
Security Realm Debug Flags	47
Common LDAP Issues	48
Quiz	49
Summary	52
Practice 16-1 Overview: Configuring an Authentication Provider	53
Chapter 14: Lesson 17 - Backing Up a Domain and Upgrading WebLogic Server	55
Objectives	56
Backup and Recovery	57
Backup Solution	58
Types of Backups	60
When to Back Up	62

Limitations and Restrictions for Online Backups	63
Performing Full Offline Backup	64
Performing Full Online Backup	66
Impact of Administration Server Failure	68
Automatically Backing Up a Domain Configuration	69
Recovery Operations	70
Directories to Restore	73
Recovery After Disaster	74
Recovery of Homes	75
Recovery of a Managed Server	76
Recovery of the Administration Server	77
Restarting the Administration Server on a New Computer	78
Managed Server Independence	80
Upgrading WebLogic Server 11g to 12c	81
Run the Reconfiguration Wizard	84
Upgrade the Managed Server Domains	85
Upgrading WebLogic Server 11g to 12c	86
Quiz	87
Summary	88
Practice 17-1 Overview: Backing Up and Restoring a Domain	89
Chapter 15: Lesson 5 - WebLogic Server Startup and Crash Recovery	91
Objectives	92
Agenda	93
Node Manager Architecture	94
Node Manager Default Behavior	95
Configure Java-Based Node Manager	96
Agenda	97
Starting the Node Manager at System Startup	98
Configuring the Node Manager as a Windows Service	99
Configuring the Node Manager on UNIX Systems	100
Configuring the Node Manager as an init.d Service	101
Registering and Managing init.d Services	104
Configuring the Node Manager as an xinetd Service	105
Solaris	107
Example SMF Manifest File	108
Set Up and Start Node Manager with SMF	111
Agenda	112
How the Node Manager Restarts an Administration Server	113
How the Node Manager Restarts a Managed Server	114
RestartInterval and RestartMax	115
Crash Recovery	116
CrashRecoveryEnabled	117
Quiz	118
Summary	119
Practice 5-1 Overview: Configuring Automatic Start and Restart of a System	120
Chapter 16: Lesson 6 - WebLogic Scripting Tool (WLST)	121
Objectives	122
Agenda	123

WebLogic Scripting Tool (WLST)	124
Agenda	125
Jython	126
Using Jython	127
Variable Declaration	128
Conditional Expressions	129
Loop Expressions	130
I/O Commands	131
Exception Handling	132
Quiz	133
Agenda	134
WLST Modes	135
WLST Example	136
Command-Line History and WLST	137
Running WLST Scripts	138
WLST Development Tools	139
configToScript()	140
Script Recording Using the Administration Console	141
Oracle Enterprise Pack for Eclipse	143
WLST Command Tips	144
General WLST Commands	145
Offline WLST Commands	146
Online WLST Commands	147
Quiz	148
Agenda	149
WebLogic JMX Overview	150
Configuration MBeans	151
Runtime MBeans	152
WebLogic Server MBean Examples	153
MBean Properties in the Administration Console	154
Browsing MBean Documentation	155
Referencing MBeans in WLST	157
Quiz	158
Agenda	159
Creating a Template and a Domain	160
Connecting to a Server	161
Password Management	162
WLST Variables	163
Password Management	164
Configuring a Server	165
Monitoring a Server	166
Adding a Server to a Cluster	167
Creating a Data Source	168
Monitoring a Data Source	170
Creating an LDAP Authentication Provider	171
Modifying a Domain Offline	172
Deploying an Application	173
Quiz	174

Agenda	175
Some FMW Commands	176
Summary	177
Practice 6-1 Overview: Creating and Modifying a Domain with WLST	178
Practice 6-2 Overview: Monitoring a Domain with WLST	179
Chapter 17: Lesson 13 - Working with the Security Realm	181
Objectives	182
Agenda	183
Security Realm: Review	184
Default Security Configuration	186
Agenda	187
How WebLogic Resources Are Protected	188
Examples of WebLogic Resources to Protect	189
Users and Groups	190
Group Membership	191
Roles	192
Policies	193
Configuring New Users	194
Configuring New Groups	195
Configuring Group Memberships	196
Configuring New Roles	197
What Is Role Mapping?	198
Configuring Role Mapping	199
Configuring Roles Using WLST	200
Configuring New Policies	201
Configuring Policies Using WLST	202
Security Configuration Sources	203
Configuring Sources Using WLST and weblogic.Deployer	204
Deployment Descriptor Security Example: weblogic.xml	205
Deployment Descriptor Security Example: web.xml	206
Embedded LDAP Server	207
Configuring the Embedded LDAP Server	208
Quiz	210
Practice 13-1 Overview: Creating Users	211
Agenda	212
Auditing	213
Sample Auditing Output	214
Security Audit Events	215
WebLogic Auditing Architecture	216
Custom Versus Default Auditing Provider	217
Creating the Default Auditing Provider	218
Configuring the Default Auditing Provider	219
Configuration Auditing	221
Quiz	223
Summary	224
Practice 13-2 Overview: Configuring WebLogic Auditing	225
Chapter 18: Lesson 15 - Diagnostic Framework	227
Objectives	228

Agenda	229
WebLogic Diagnostics Framework (WLDF)	230
WLDF Architecture	231
Diagnostic Archives	232
Configuring Server Diagnostic Archives	233
Diagnostic Modules	234
Dynamic Diagnostic Modules	236
Resource Descriptors	237
Creating a Diagnostic Module	238
WLST: Example	239
WLST Commands for WLDF	240
Quiz	241
Agenda	242
What Is a Diagnostic Image?	243
Capturing a Server Diagnostic Image	244
WLST: Example	245
Quiz	246
Agenda	247
What Is a Harvester?	248
Metric Collectors	249
Configuring a Metric Collector	250
WLST: Example	251
Quiz	252
Agenda	253
Watches and Notifications	254
Configuring a Watch	255
Quiz	257
Summary	258
Practice 15-1 Overview: Using a Built-in Diagnostic Module	259
Chapter 19: Lesson 16 - WebLogic and Coherence Integration	261
Objectives	262
Agenda	263
Oracle Coherence: Overview	264
Agenda	266
Types of Session Persistence	267
Coherence*Web	268
Coherence*Web and WebLogic Clusters	269
Coherence*Web Session Failover	270
Configuring Coherence*Web in WebLogic	271
Quiz	272
Practice 16-1 Overview: Configuring Coherence*Web	273
Agenda	274
Coherence and WebLogic Server	275
WebLogic Managed Coherence Servers Operations	276
What Is a Grid Archive (GAR)?	277
Coherence Application Deployment on WebLogic	278
Coherence Container: Benefits	279
Coherence Cluster	280

Managed Coherence Server	281
Quiz	283
Summary	284
Practice 16-2 Overview: Configuring Managed Coherence Servers	285
Appendix A: Monitor and Tune JVM Performance	287
Objectives	288
Assumptions and Expectations	289
Agenda	290
What Is Performance?	291
What Is Performance? (Notes Only)	292
Performance Focus for This Lesson	293
HotSpot JVM Monitoring Tools	294
Section Summary	295
Agenda	296
Garbage Collection Basics	297
Generational Garbage Collection	298
Generational Garbage Collection (Notes Only)	299
Garbage Collectors: Java Heap Options	300
Setting Heap Memory Size	301
Setting Heap Size Options	302
Setting Heap Size Options (Notes Only)	303
G1 Garbage Collector	304
G1 Garbage Collector (Notes Only)	305
GC Algorithms	306
GC Algorithms (Notes Only)	307
GC Performance Goals	308
Focusing on Throughput	309
Focusing on Responsiveness	310
Evaluating GC Algorithm	311
GC Tuning Tips	312
GC Tuning Tips (Notes Only)	315
Server-Class Machine Detection	316
Ergonomics: What It Does	317
Section Summary	318
Practice 4-1 Overview: Tuning JVM Garbage Collection	319
Agenda	320
Using jps	321
Using jps: Example	322
Using jcmd	323
Using jcmd: Examples	324
Using jcmd: Histogram	325
Using jinfo	326
Using jstat	327
Using jstack	328
jstack	329
Section Summary	330
Practice 4-2 Overview: Using JVM Command Line Tools	331
Agenda	332

Java Monitoring and Management Architecture	333
Java VisualVM	334
Java VisualVM Connections	335
Remote Monitoring	336
Java VisualVM Interface	337
Monitoring JVM	338
Monitoring Threads	339
Profiler Snapshot	340
Using VisualGC	341
Using jconsole	342
Using GCHisto	343
Practice 4-3 Overview: Using Java VisualVM	344
Practice 4-4 Overview: Using VisualGC	345
Mission Control	346
Java Discovery Protocol (JDP)	347
Mission Control: Architecture	348
JVM Browser	349
Mission Control: Management Console Overview Tab	350
General: Server Info	351
MBeans: Attributes	352
MBeans: Operations	353
MBeans: Notifications	354
MBeans: Metadata	355
MBeans: Triggers	356
Runtime: Server	357
Runtime: Memory	358
Runtime: Garbage Collection	359
Runtime: Memory Pools	360
Runtime: Threads	361
Best Practices	362
Practice 4-5 Overview: Using Mission Control	363
Flight Recorder	364
Flight Recorder: Benefits	365
Data Flow in Flight Recorder	366
Initiating Recording Using the Management Console	367
Tab Groups of Flight Recorder	368
Flight Recorder General Tab	369
Flight Recorder Memory: Overview Tab	370
Flight Recorder Memory: GC Collections Tab	371
Flight Recorder Memory: Object Statistics	372
Section Summary	373
Practice 4-6 Overview: Using Flight Recorder	374

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.

16

WebLogic Server Security

ORACLE

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

Objectives

After completing this lesson, you should be able to:

- Describe the basics of the WebLogic Server security architecture
- Describe basic LDAP concepts
- Configure an external LDAP authentication provider for WebLogic Server

Some Security Terms

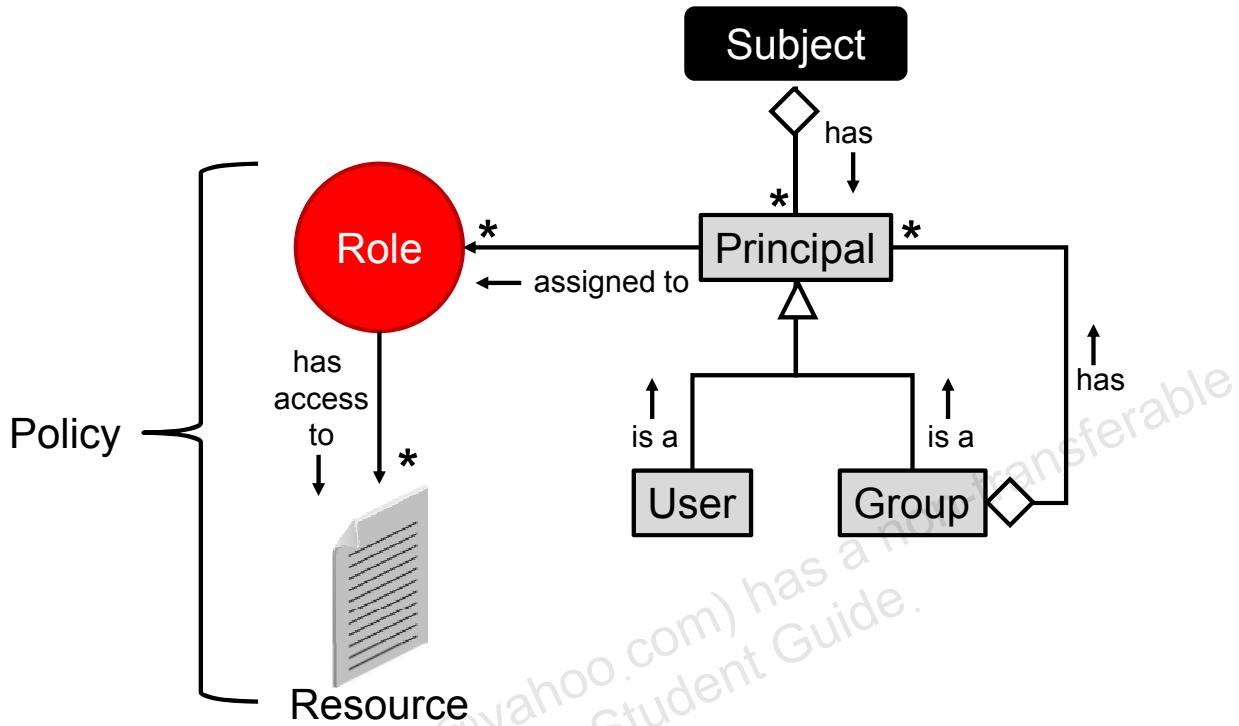
- **Subject:** The user (or service) accessing the system
 - A subject has one (or more) *principals*
- **Principal:** The unique identity of a subject, assigned after authentication
 - Usually a username or a group name
- **User:** An individual (or program) accessing the application
- **Credentials:** Usually username or password
- **Group:** A collection of users and/or other groups
- **Role:** A type of user
 - Principals can be assigned roles to say what kind of user they represent
- **Policy:** A security rule, usually an association of a resource to one or more roles



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

An example of a policy that is not an association of a resource to a role would be a “daytime access” rule: This particular resource (or set of resources) may only be accessed between the hours of 8:00 AM and 5:30 PM.

Some Security Terms: Graphically



ORACLE

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

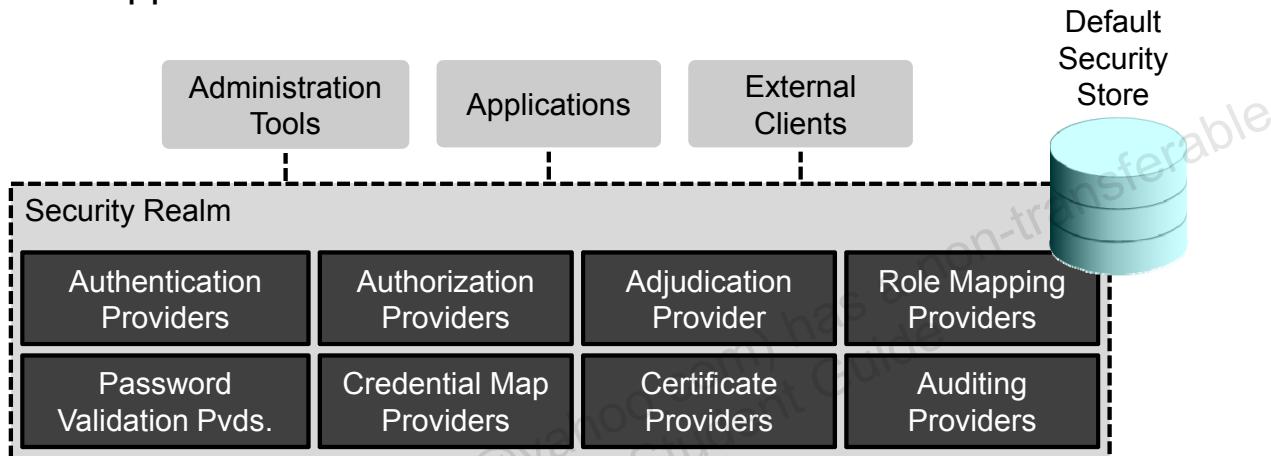
A User is a Principal. A Group is a Principal. A Subject, after authentication, is assigned one or more Principals (before authentication the Subject has zero Principals). A Group contains zero or more Principals (Users and other Groups). A Principal (a User or, more often, a Group) is assigned to zero or more Roles. A Policy states that a particular Role has access to a particular Resource (or set of Resources).

Note: The “*” means “zero or more.”

WebLogic Server Security Realm

A WebLogic Server security realm:

- Handles security logic and decisions for a domain
- Consists of a series of pluggable security providers
- Applies to all servers in a domain



ORACLE

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

The security service in WebLogic Server simplifies the configuration and management of security while offering robust capabilities for securing WebLogic Server. Security realms act as a scoping mechanism. Each security realm consists of a set of configured security providers, users, groups, security roles, and security policies. You can configure multiple security realms in a domain; however, only one security realm can be active.

A security policy is an association between a WebLogic resource and one or more security roles. Security policies protect the WebLogic resource against unauthorized access. A WebLogic resource has no protection until you create a security policy for it.

A security provider store contains the users, groups, security roles, security policies, and credentials used by some types of security providers to provide their services. For example, an authentication provider requires information about users and groups; an authorization provider requires information about security policies; a role mapping provider requires information about security roles, and a credential mapping provider requires information about credentials to be used to access remote applications. These security providers need this information to be available to function properly.

What the Providers Do

- **Authentication:** Who are you? Prove it.
 - Can optionally use an Identity Assertion Provider, which takes a token from outside of WebLogic Server, validates it, and, if valid, maps the token to a username.
- **Authorization:** Are you allowed to use this resource?
 - Uses the Role Mapping provider
- **Adjudication:** The multiple authorization providers do not agree. Can the user have the resource?
- **Role Mapping:** What type of user are you?
 - For example: manager, salesperson, administrator
- **Password Validation:** Does the new or modified password meet the password rules?

ORACLE

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

- **Authentication:** The authentication provider determines whether a user is legitimate. When a user logs in with a username and password, this provider validates that the username exists and that the password entered is correct. After successfully proving an identity, an authentication context is established (a subject containing one or more principals), which allows an identified user or system to be authenticated to other entities.
- **Identity Assertion:** Identity Assertion providers are used as part of a perimeter authentication process. When perimeter authentication is used, a token from outside of the WebLogic Server domain is passed to an active identity assertion provider in a security realm that is responsible for validating tokens of that type. If the token is successfully validated, the identity assertion provider maps the token to a WebLogic Server username, and sends that username back to WebLogic Server, which then continues the authentication process.

- **Authorization:** The authorization process is initiated when a user or system requests a WebLogic Server resource on which it will attempt to perform a given operation. The resource container that handles the type of resource receives the request (for example, the EJB container receives the request for an EJB resource). The resource container calls the WebLogic Security Framework and passes in the request parameters, including information such as the subject of the request and the WebLogic Server resource being requested. The WebLogic Security Framework calls the role mapping provider and passes in the request parameters in a format that the role mapping provider can use. The role mapping provider uses the request parameters to compute a list of roles to which the subject making the request is entitled, and passes the list of applicable roles back to the WebLogic Security Framework. The authorization provider determines whether the subject is entitled to perform the requested action on the WebLogic Server resource.
- **Adjudication:** The adjudication provider determines what to do if multiple authorization providers' access decisions do not agree. The adjudication provider resolves authorization conflicts by weighing each access decision and returning a final result.
- **Role Mapping:** The WebLogic Security Framework calls each role mapping provider that is configured as part of an authorization decision (see the explanation of the authorization provider). The role mapping provider returns the list of roles a particular user has. These roles are returned to the WebLogic Security Framework, where they can be used in an access decision. WebLogic Server resources can be configured so that certain roles can perform certain actions. (For example, in a web application, resources (given as URL patterns), can be protected so that only a user with the proper role is allowed access to them.)
- **Password Validation:** When the password validation provider is configured with an authentication provider, the authentication provider invokes the password validation provider whenever a password is created or updated. The password validation provider then performs a check to determine whether the password meets the criteria established by a set of configurable rules.

What the Providers Do

- **Credential Mapping:** Maps a user authenticated to WebLogic Server to a set of credentials for another system, so that the user can access that other system
- **Certificate Providers:** Keeps a list of trusted digital certificates and validates those certificates
- **Auditing:** For certain user tasks, tracks who did what and when

ORACLE

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

- **Credential Mapping:** The credential mapping provider is used to associate a WebLogic Server user to their credentials for some other system, to be used with a Resource Adapter. The provider maps a user's authentication credentials (username and password) to those required for some legacy application, so that the legacy application receives the necessary credential information and allows access.
- **Certificate Providers:** The certificate lookup and validation providers complete certificate chains and validate them. If multiple providers are configured, a certificate or certificate chain must pass validation with all of them in order for the certificate or certificate chain to be accepted. (A certificate chain, also known as the certification path, is a list of certificates used to authenticate an entity. The chain begins with the certificate of that entity, and each certificate in the chain is signed by the entity identified by the next certificate in the chain. The chain terminates with a root Certificate Authority (CA) certificate and is signed by the CA.)
- **Auditing:** An auditing provider collects, stores, and distributes information about operating requests and the outcome of those requests for the purposes of non-repudiation (users cannot later say that they did not perform some task). An auditing provider makes the decision about whether to audit a particular event based on specific audit criteria, including audit severity levels.

Security Stores

A persistent store is assigned to a security realm to persist assets such as:

- Users and groups
- Roles
- Policies
- Credential maps
- Certificates

Some providers use the default security store while others use an external system.



ORACLE

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

The default Auditing and Adjudication providers do not use the persistent security stores configured for their parent security realm.

If you have multiple security providers of the same type configured in the same security realm, these security providers may use the same security provider database. For example, if you configure two WebLogic authentication providers in the default security realm (called `myrealm`), both WebLogic authentication providers would use the same location in the embedded LDAP server as their security provider database and thus will use the same users and groups. Furthermore, if you or an administrator add a user or group to one of the WebLogic authentication providers, you will see that user or group appear for the other WebLogic authentication provider as well.

Default Security Store Implementation

- The WebLogic default:
 - An embedded LDAP server running on the admin server and replicated to the managed servers
- Or, you can configure the RDBMS security store:
 1. In the admin console, select the realm. Then select **Configuration > RDBMS Security Store**.
 2. Select **RDBMS Security Store Enabled** and fill in the required fields.
 - The schema files are located at <WL_HOME>/server/lib.



ORACLE

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

WebLogic Server uses its embedded LDAP server to store users, groups, security roles, and security policies for the WebLogic security providers. The embedded LDAP server is a complete LDAP server that is production quality for reasonably small environments (10,000 or fewer users). For applications that need to scale above this number, the embedded LDAP server can serve in the development and integration environments for future export to an external LDAP server for the test and production environments.

When the RDBMS security store is configured in a security realm, any of the following security providers that have been created in the security realm automatically uses only the RDBMS security store, and not the embedded LDAP server:

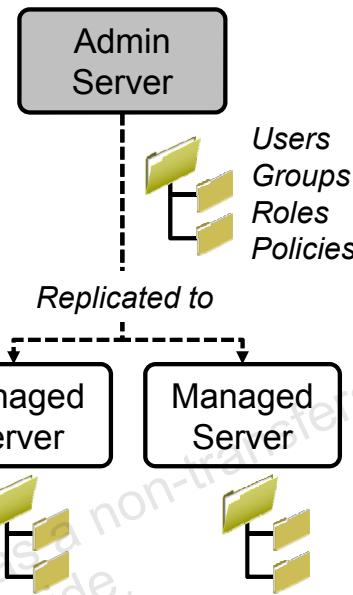
- XACML Role Mapping and Authorization
- Default, PKI, and SAML Credential Mapping
- SAML Identity Assertion

Other security providers continue to use their default security stores; for example, the WebLogic authentication provider continues to use the embedded LDAP server. Note that the use of the RDBMS security store is required to use SAML 2.0 services in two or more WebLogic Server instances in a domain, such as in a cluster.

Default Security Configuration

A new domain includes a default realm that:

- Includes default providers:
 - Default authenticator
 - Default identity asserter
 - XACML* role mapper
 - XACML* authorization provider
 - Default password validator
 - Default credential mapper
 - Default certificate path provider
 - Validates certificate chains
- Uses the embedded LDAP security store



* eXtensible Access Control Markup Language:
An XML-based security policy language



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

To simplify the configuration and management of security, WebLogic Server provides a default security configuration. In the default security configuration, a default security realm, `myrealm`, is set as the active security realm, and the WebLogic authentication, identity assertion, credential mapping, certification path, password validation, EXtensible Access Control Markup Language (XACML) authorization, and XACML role mapping providers are defined as the security providers in this security realm. (XACML is OASIS standard, XML-based, security policy, and access control language.)

The WebLogic Server embedded LDAP server for a domain consists of a master LDAP server, maintained in the domain's administration server, and a replicated LDAP server maintained in each managed server in the domain. When changes are made using a managed server, updates are sent to the embedded LDAP server on the administration server. The embedded LDAP server on the administration server maintains a log of all changes. The embedded LDAP server on the administration server also maintains a list of managed servers and the current change status for each one. The embedded LDAP server on the administration server sends appropriate changes to each managed server. This process occurs when an update is made to the embedded LDAP server on the administration server. However, depending on the number of updates, it may take several seconds or more for the changes to be replicated to the managed servers.

Security Customization Approaches

- Create an entirely new security realm and add (at least) the required providers.
 - After the new security realm is configured, make it the active security realm.
- Add, remove, and configure providers in the default realm, called `myrealm`.
- Have developers create custom security providers and add them to either the default realm or a custom security realm.



Most realm modifications require a domain restart.

ORACLE

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

The easiest way to customize the default security configuration is to add the security providers you want to the default security realm (`myrealm`). Many customers instead create an entirely new security realm and place in it the security providers they want. This preserves your ability to revert more easily to the default security configuration. You configure security providers for the new realm, migrate any security data, such as users and groups, from the existing default realm, and then set the new security realm as the default realm.

A valid security realm requires an authentication provider, an authorization provider, an adjudication provider, a credential mapping provider, a role mapping provider, and a certification path builder. Optionally, define identity assertion, auditing, and certificate registry providers. If you configured the default authentication, authorization, credential mapping, or role mapping provider or the default certificate registry in the new security realm, verify that the settings of the embedded LDAP server are appropriate.

Authentication Providers

Authentication providers are organized into two categories:

- *Authenticators*:
 - Establish the user's identity given some credentials (like username and password)
 - Can associate multiple principals with a single user, such as groups
- *Identity asserters*:
 - Validate tokens claiming a user has already been authenticated
 - Allow WebLogic Server to participate in single sign-on (SSO) solutions
 - Can map the token to a local user and use authenticators to look up that user's principals



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

Authentication providers are used to prove the identity of users or system processes. Authentication providers also remember, transport, and make that identity information available to various components of a system (via subjects) when needed.

Both users and groups can be used as principals by application servers like WebLogic Server. A principal is an identity assigned to a user or group as a result of authentication. The Java Authentication and Authorization Service (JAAS) requires that subjects be used as containers for authentication information, including principals. Each principal stored in the same subject represents a separate aspect of the same user's identity, much like credit cards in a person's wallet.

An *identity assertion provider* is a specific form of authentication provider that allows users or system processes to assert their identity using tokens supplied by clients. Typical token types include X509 certificates, SAML, and Kerberos. Identity assertion providers enable perimeter authentication and support single sign-on (SSO). For example, an identity assertion provider can generate a token from a digital certificate, and that token can be passed around the system so that users are not asked to sign on more than once.

Unlike in a simple authentication situation, identity assertion providers do not verify credentials such as usernames and passwords. They verify that the user exists.

Available Authentication Providers

- Available authenticators include:
 - Default (Internal LDAP)
 - LDAP (generic and vendor-specific)
 - Database (multiple DBMS providers)
 - Windows NT
 - SAML (Security Assertion Markup Language)
- Available identity asserters include:
 - Default
 - LDAP X509
 - SAML
 - Negotiate (SPNEGO)



ORACLE

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

The default WebLogic Server authentication provider manages users and groups in the embedded LDAP server.

LDAP authentication providers access external LDAP stores. WebLogic Server provides LDAP authentication providers that access Oracle Internet Directory, Oracle Virtual Directory, Open LDAP, iPlanet, Microsoft Active Directory, and Novell Directory Service stores.

A DBMS authentication provider is a username and password authentication provider that uses a relational database (rather than an LDAP server) as its data store for user, password, and group information.

The SAML authentication provider may be used with the SAML 1.1 or SAML 2.0 identity assertion provider to allow virtual users to log in via SAML. This provider creates an authenticated subject using the username and groups retrieved from a SAML assertion. (SAML is the Security Assertion Markup Language. It is an XML-based, OASIS standard data format for exchanging authentication and authorization data between parties, in particular, between an identity provider and a service provider.)

The Windows NT authentication provider uses account information defined for a Windows NT domain to authenticate users and groups and to permit Windows NT users and groups to be listed in the administration console.

The default identity assertion provider supports identity assertion with X509 certificates and CORBA Common Secure Interoperability version 2 (CSI v2). You can also map the tokens authenticated by the identity assertion provider to users in the security realm.

The LDAP X509 identity assertion provider receives an X509 certificate, looks up the LDAP object for the user associated with that certificate, ensures that the certificate in the LDAP object matches the presented certificate, and then retrieves the name of the user from the LDAP object.

The SAML identity assertion provider acts as a consumer of SAML security assertions, allowing WebLogic Server to participate in SSO solutions for web or web service applications. It validates assertions by checking the signature and validating the certificate for trust based on data configured for the associated partner. The provider then extracts the identity information contained in the assertion, and maps it to a local subject in the security realm.

The Negotiate identity assertion provider enables SSO with Microsoft clients. The provider decodes Simple and Protected Negotiate (SPNEGO) tokens to obtain Kerberos tokens, validates the Kerberos tokens, and maps Kerberos tokens to WebLogic users.

Lightweight Directory Access Protocol (LDAP)

- LDAP:
 - Is a TCP/IP protocol
 - Provides a hierarchical lookup and search service
 - Models information as a tree of entries, whose attributes are defined by a schema or “object class”
 - Defines default schemas for common entries like people and groups
 - Supports SSL
- Entries:
 - Identify their locations in the tree by using a *distinguished name* (DN)
 - Can be referrals that link to other LDAP servers



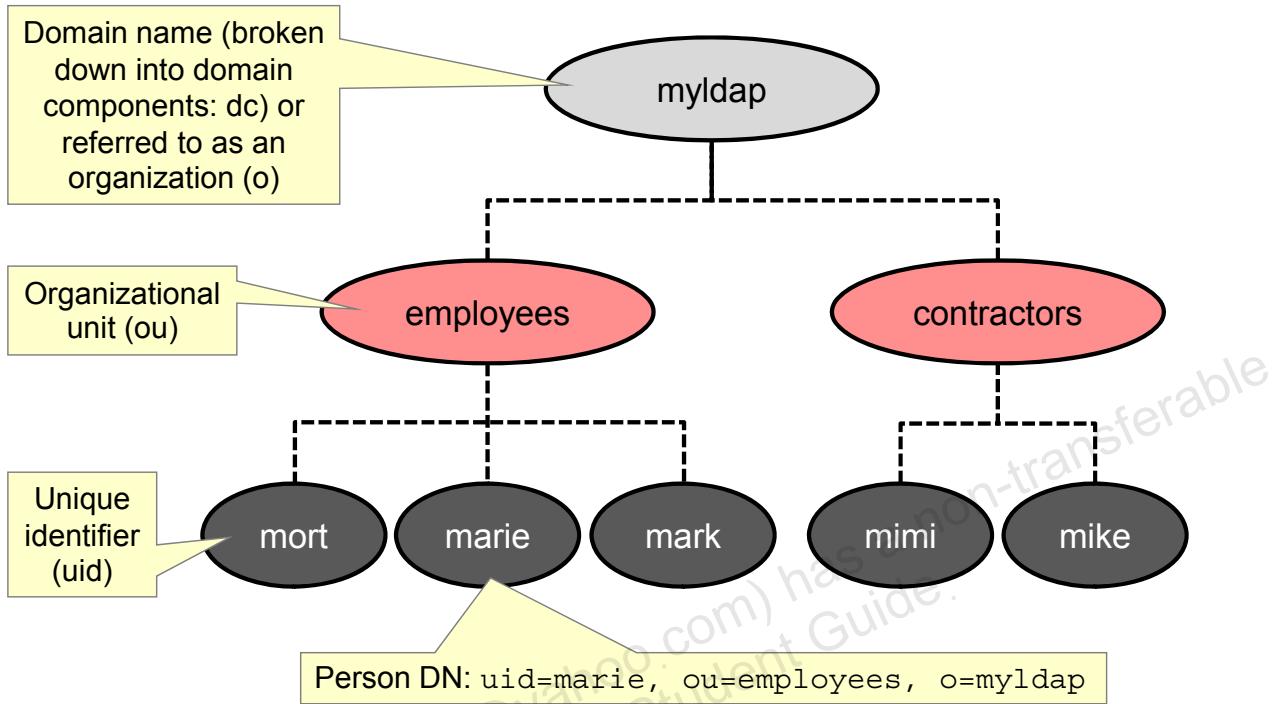
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Lightweight Directory Access Protocol, better known as LDAP, is a protocol that provides access to a compliant directory of information via TCP/IP (Transmission Control Protocol/Internet Protocol). The strengths of LDAP-compliant directories include speed, simplicity, and the ability to be replicated and distributed across several servers. An LDAP directory can be used to store a great deal of information from user login credentials to company telephone directories.

Unlike databases that are designed for processing hundreds or thousands of changes per minute, LDAP directories are heavily optimized for read performance. LDAP is intentionally designed for environments where search operations are much more common than modify operations.

LDAP Version 3 implements a referral mechanism that allows servers to return references to other servers as a result of a directory query. This makes it possible to distribute directories globally by partitioning a directory information tree (DIT) across multiple LDAP servers.

LDAP Structure



ORACLE

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

Directories are viewed as a tree, analogous to a computer's file system. Each entry in a directory is called an object. These objects are of two types: containers and leaves. A container is like a folder; it contains other containers or leaves. A leaf is simply an object at the end of a branch. A tree cannot contain any arbitrary set of containers and leaves. It must match the schema defined for the directory.

The top level of the LDAP directory tree is the base, referred to as the base DN. A base DN can be one of several forms. Here are some examples:

- A domain name, broken into components (`dc=Acme, dc=com`)
- An organization name (`o=Acme Corp`)
- An organization name along with a country (`o=Acme Corp, c=India`)

Organizational units are standard LDAP object classes that act as containers for other entries. The identifying attribute for an organizational unit is "ou." The standard LDAP schema also defines a `person` class and a `group` class, which is a collection of people.

The `person` type also includes other attributes such as Common Name (a person's full name: `cn`), Unique Identifier (`uid`), Surname (last name: `sn`), and Password (`userpassword`).

LDAP Search Operations

Searching for LDAP entries involves:

1. The base DN from which to start searching
2. A search filter that specifies the:
 - Search criteria in terms of attribute values
 - The type or “object class” of the desired entries
3. An indication whether or not the search should include any child entries



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

LDAP Query Basics

- = (equal)
 - Example: (uid=tjp)
- & (logical and)
 - Example: (&(uid=tjp)(sn=Parker))
- | (logical or)
 - Example: (|(uid=tjpark)(uid=tjp))
- ! (logical not)
 - Example: (! (sn=Parker))
- * (wildcard)

Here is an LDAP search filter that finds all person entries whose user ID begins with “t,” while ignoring those whose surname starts with “Th”:

```
(&(&(uid=t*)(!(sn=Th*))(objectclass=person))
```



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

The “&” represents a logical “and” when combining multiple expressions that have been grouped together in parentheses. Similarly, the “|” represents a logical “or,” and a “!” represents a logical “not.”

Examples:

- (uid=tjp) – All entries whose unique identifier is equal to tjp
- (&(uid=tjp)(sn=Parker)) – All entries whose unique identifier is equal to tjp and whose surname is equal to Parker
- (|(uid=tjpark)(uid=tjp)) – All entries whose unique identifier is equal to tjpark or equal to tjp
- (! (sn=Parker)) – All entries whose surname is not equal to Parker

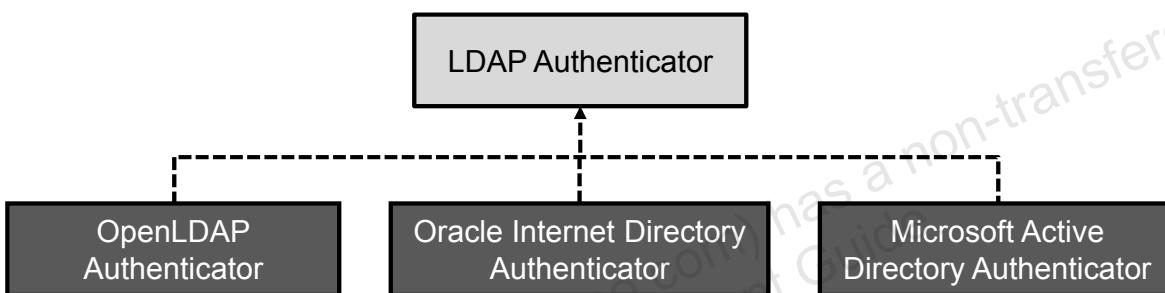
Search filters can specify one or more object classes. Here is an example:

```
(&(&(objectClass=person)(objectClass=organizationalPerson))
(objectClass=user))
```

LDAP Authentication Providers

WebLogic Server includes:

- A base LDAP authenticator that can be configured to support any compliant vendor
- Vendor-specific LDAP authenticators, whose attributes are set to vendor-specific defaults for convenience



ORACLE

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

Each LDAP authentication provider stores user and group information in an external LDAP server. The providers differ primarily in how they are configured by default to match typical directory schemas for their corresponding LDAP server. For example, the generic authenticator is configured to use a person's common name (`cn`) as a user ID, while by default Oracle Internet Directory uses the "`uid`" attribute for this purpose. Similarly, the names of object classes used to represent people or groups may vary from vendor to vendor. For example, the generic authenticator is configured to use the object class "`groupofuniqueNames`," while by default Oracle Internet Directory uses the object class "`groupofNames`."

WebLogic Server does not support or certify any particular LDAP servers. Any LDAP v2 or v3 compliant LDAP server should work with WebLogic Server.

If an LDAP authentication provider is the only configured authentication provider for a security realm, you *must* include the Admin role and assign that role to a user or group of users in the LDAP directory. If the LDAP user who boots WebLogic Server is not properly added to a group that is assigned to the Admin role, and the LDAP authentication provider is the only authentication provider with which the security realm is configured, WebLogic Server cannot be booted.

Available LDAP Authentication Providers

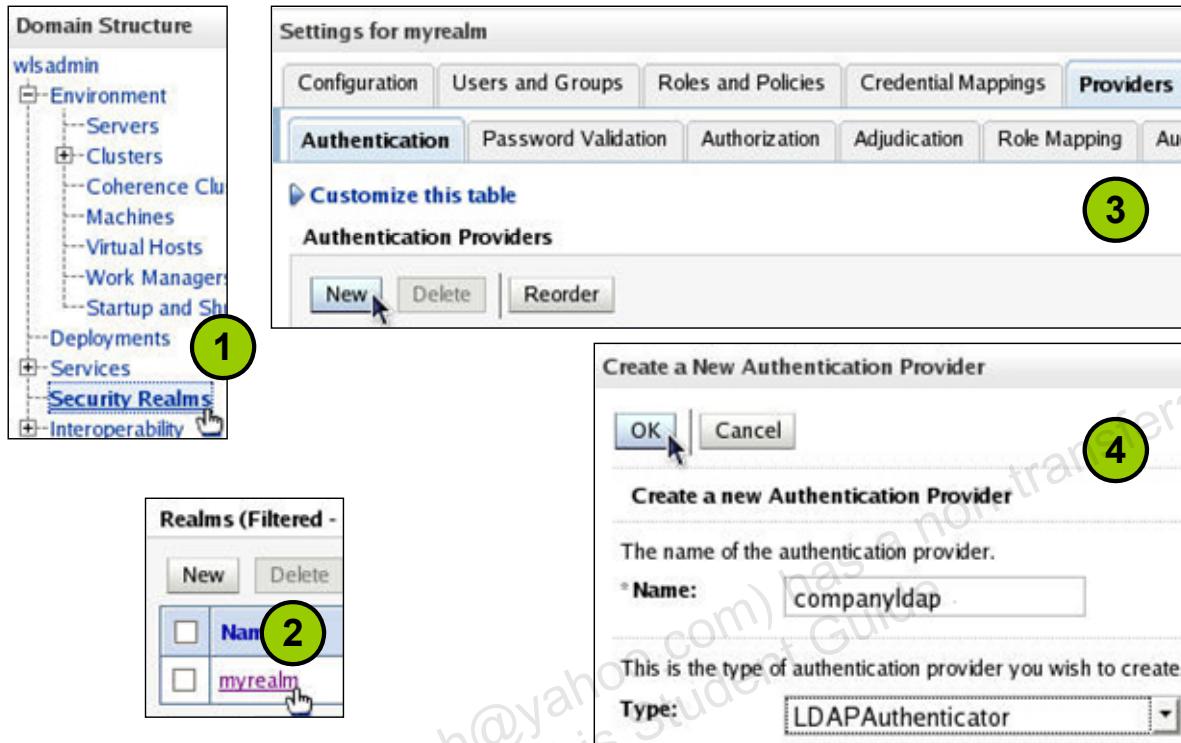
- The available LDAP authentication providers include:
 - LDAP Authenticator (generic)
 - Oracle Internet Directory Authenticator
 - Oracle Virtual Directory Authenticator
 - iPlanet Authenticator
 - Active Directory Authenticator
 - Novell Authenticator
 - OpenLDAP Authenticator
- These providers:
 - Can be used to change passwords of existing users
 - Cannot be used to create, update, or delete users and groups



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

The LDAP authentication providers in WebLogic Server are configured to work readily with the Oracle Internet Directory, Oracle Virtual Directory, iPlanet, Active Directory, Open LDAP, and Novell NDS LDAP servers. You can use an LDAP authentication provider to access other types of LDAP servers. Choose either the generic LDAP authenticator or an existing LDAP provider that most closely matches the new LDAP server and customize the existing configuration to match the directory schema and other attributes for your LDAP server.

Creating a New LDAP Authentication Provider



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

ORACLE

1. After locking the configuration, in the Domain Structure, select **Security Realms**.
2. In the Realms table, click the realm name.
3. Select **Providers > Authentication**. Click the **New** button.
4. Enter the **Name** of the provider and select the **Type** from the drop-down list. Then click **OK**.

Configuring the LDAP Provider: Connection

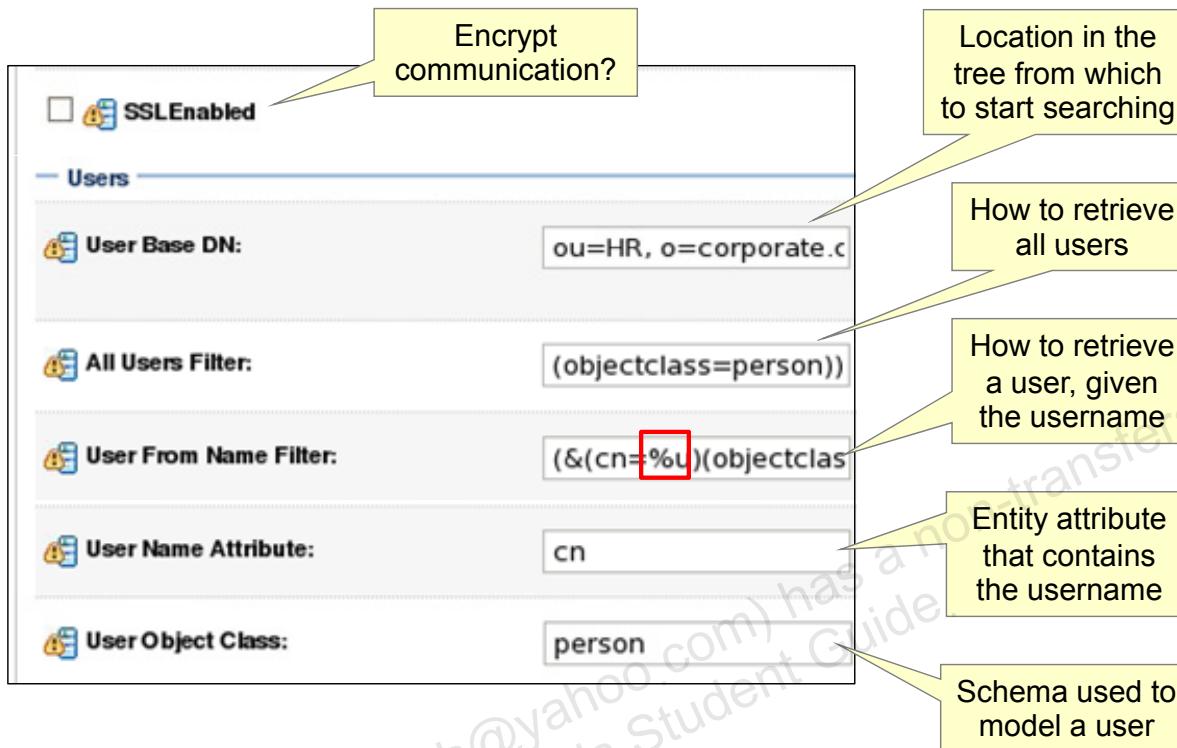
The screenshot shows the 'Settings for companyldap' configuration page. On the left, there's a table titled 'Authentication Providers' with columns 'Name', 'DefaultAuthenticator', 'DefaultIdentityAssertioner', and 'companyldap'. The 'companyldap' row is selected, indicated by a green circle with the number '5'. On the right, under the 'Provider Specific' tab, there are tabs for 'Configuration' and 'Performance'. The 'Configuration' tab is active. It contains sections for 'Connection' and 'Credential'. Under 'Connection', there are fields for 'Host' (ldap.corporate.com), 'Port' (389), and 'Principal' (user). Under 'Credential', there is a field with several dots (...). A green circle with the number '6' is over the 'Configuration' tab. A green circle with the number '7' is over the 'Principal' field. A green circle with the number '8' is over the 'Save' button, which has a cursor arrow pointing to it.

ORACLE

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

5. Click the name of the new provider in the Authentication Providers table.
6. Click **Configuration > Provider Specific**.
7. Enter values for the following connection attributes:
 - **Port:** The port of the LDAP server
 - **Principal:** The Distinguished Name (DN) of the LDAP user that WebLogic Server should use to connect to the LDAP server
 - **Credential:** The credential (usually a password) used to connect to the LDAP server
 - **SSL Enabled** (shown on next slide): Specifies whether the SSL protocol should be used when connecting to the LDAP server. For a more secure deployment, Oracle recommends using the SSL protocol to protect communication between the LDAP server and WebLogic Server.
8. Click **Save**. Then activate the changes.

Configuring the LDAP Provider: Users



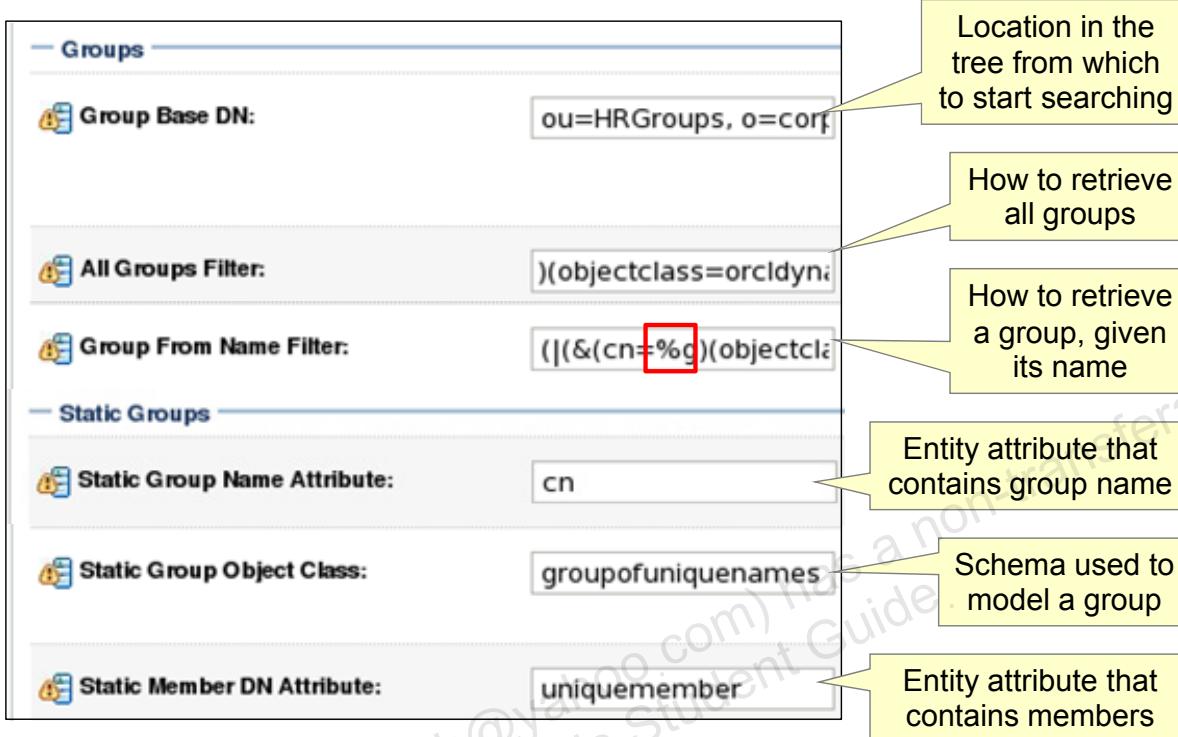
ORACLE

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

Enter values for any of the following user search attributes:

- **User Base DN:** The base distinguished name (DN) of the tree in the LDAP directory that contains users
- **All Users Filter:** The LDAP filter expression used to retrieve all users. If not specified, a simple default filter is generated based on the user object class.
- **User From Name Filter:** The LDAP filter expressions used to locate a user entry given its user ID. Use the token "%u" to indicate where the provider should insert the user ID before executing the search.
- **User Search Scope** (not shown): Specifies how deep in the LDAP directory tree the provider should search for users. Valid values are "subtree" and "onelevel."
- **User Name Attribute:** The attribute of an LDAP user object that specifies the user's login ID
- **User Object Class:** The LDAP object class that stores users
- **Use Retrieved User Name as Principal** (not shown): Specifies whether or not the username retrieved from the LDAP server should be used as the principal

Configuring the LDAP Provider: Groups



ORACLE

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

Enter values for any of the following group search attributes:

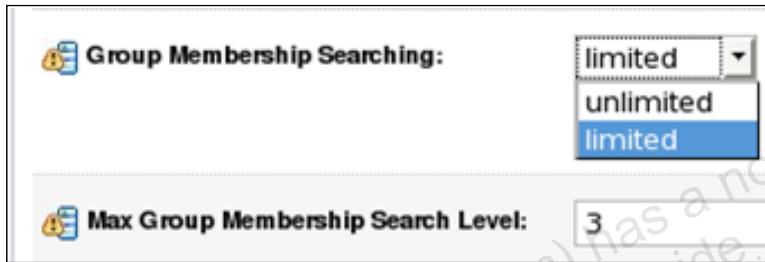
- **Group Base DN:** The base distinguished name (DN) of the tree in the LDAP directory that contains group definitions.
- **All Groups Filter:** An LDAP filter expression for finding all groups beneath the base group distinguished name (DN). If not specified, a simple default search filter is created based on the group object class.
- **Group From Name Filter:** An LDAP filter expression for finding a group given the name of the group. Use the "%g" token to indicate where the provider should insert the user ID before executing the search. If not specified, a simple default search filter is created based on the group schema.
- **Group Search Scope** (not shown): Specifies how deep in the LDAP directory tree to search for groups. Valid values are "subtree" and "onelevel."
- **Ignore Duplicate Membership** (not shown): Determines whether duplicate members are ignored when adding groups.

- **Static Group Name Attribute:** The attribute of a group object that specifies the name of the group
- **Static Group Object Class:** The name of the LDAP object class that stores groups
- **Static Member DN Attribute:** The attribute of a group object that specifies the distinguished names (DNs) of the members of the group
- **Static Group DNs from Member DN Filter** (not shown): Given the DN of a member of a group, returns the DNs of the groups that contain that member

Note: A static group contains a list of members that you explicitly administer. A dynamic group is one whose membership, rather than being maintained in a list, is computed, based on rules and assertions you specify.

Configuring the LDAP Provider: Subgroups

- Groups can include other groups.
- To improve performance, you can limit the depth that the provider will search for subgroups.



ORACLE

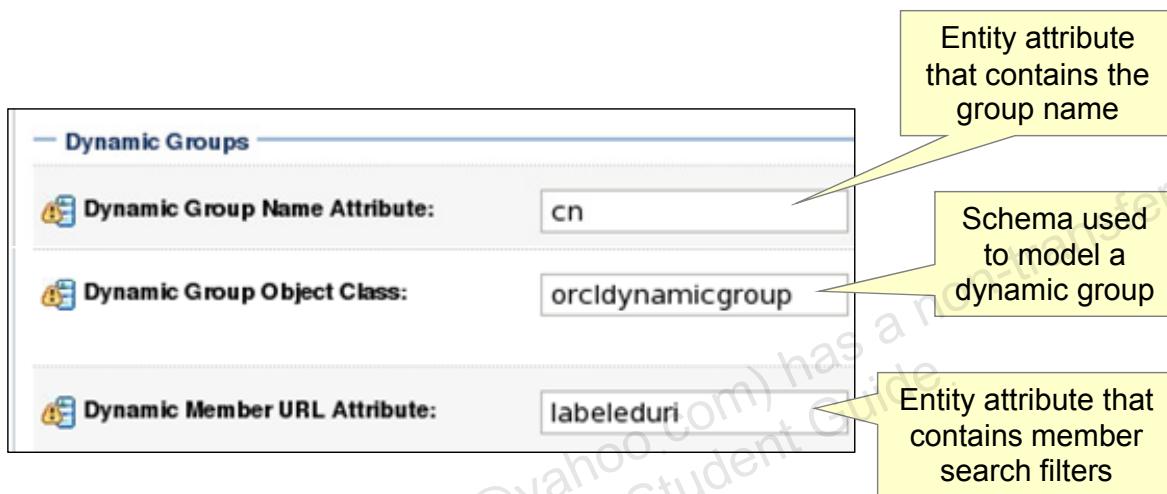
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Enter values for any of the following attributes that apply to subgroup searching:

- **Group Membership Searching:** Specifies whether recursive group searches into nested groups are unlimited or limited. For configurations that use only the first level of nested group hierarchy, this attribute allows improved performance during user searches by limiting the search to the first level of the group.
- **Max Group Membership Search Level:** Specifies how many levels of group membership can be searched. This setting is valid only if Group Membership Searching is set to "limited." A value of 0 indicates that only direct groups will be found. That is, when searching for membership in Group A, only direct members of Group A will be found. If Group B is a member of Group A, the members of Group B will not be found by this search. Any non-zero value indicates the number of levels to search. For example, if this attribute is set to 1, a search for membership in Group A will return direct members of Group A. If Group B is a member of Group A, the members of Group B will also be found by this search. However, if Group C is a member of Group B, the members of Group C will not be found by this search.

Configuring the LDAP Provider: Dynamic Groups

- Instead of a list of users, dynamic groups contain a list of search filters, each of which returns zero or more users.
- Member search filters are expressed as URLs.



ORACLE

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

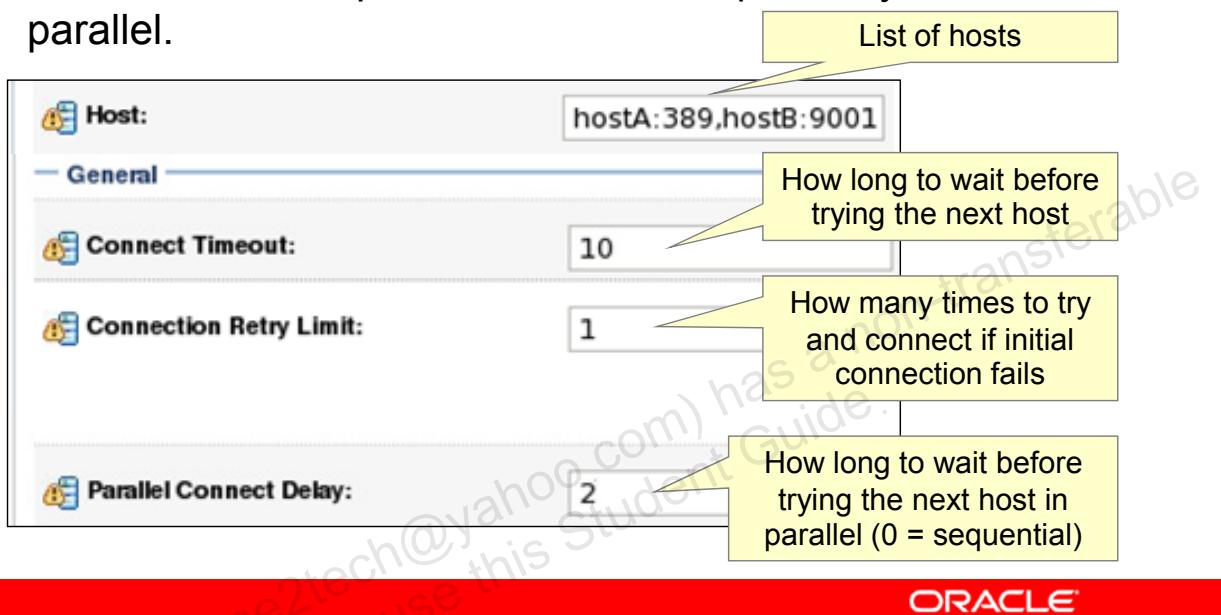
Many LDAP servers have a concept of dynamic groups or virtual groups. These are groups that, rather than consisting of a list of users and groups, contain some queries or code that define the set of users. The term “dynamic” describes the means of defining the group and not any runtime semantics of the group within WebLogic Server.

The provider attributes for dynamic groups are very similar to static ones, but the following additional attributes are also available:

- Dynamic Member URL Attribute:** The attribute of the dynamic LDAP group object that specifies the URLs of the members of the dynamic group. With a dynamic group, users are members if they match this URL “rule” (dynamic group members share an attribute or set of attributes). Here is an example URL that specifies a dynamic group that contains any users whose uid is in the tree (`o=myldap`) below the sales organization: `ldap://ou=sales,o=myldap??sub?(uid=*)`.
- User Dynamic Group DN Attribute:** A user attribute indicating its dynamic group membership. If such an attribute does not exist, the provider determines whether a user is a member of a group by evaluating the URLs on the dynamic group. If a group contains other groups, WebLogic Server evaluates the URLs on any of the descendants (subgroups) as well.

LDAP Failover

- The **Host** attribute supports a list of candidate servers for high availability.
- Connection attempts can be made sequentially or in parallel.



ORACLE

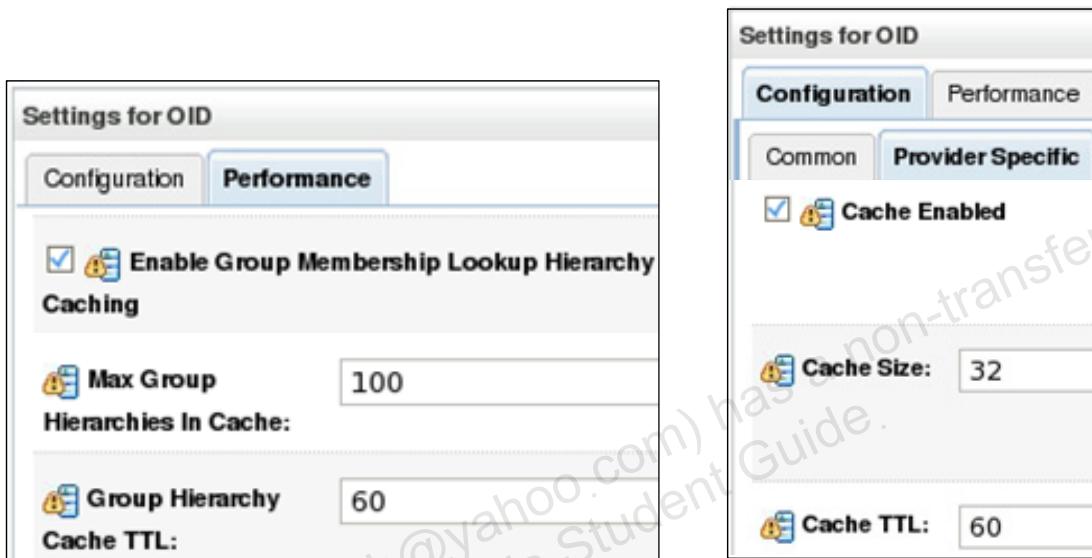
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can configure an LDAP provider to work with multiple LDAP servers and enable failover if one LDAP server is not available. Use the Host attribute to specify the names of the additional LDAP servers. Each host name may include a port number and a trailing comma. Also, configure the following additional attributes:

- Connect Timeout:** Specifies the maximum number of seconds to wait for the connection to the LDAP server to be established. If set to 0, there is no maximum time limit and WebLogic Server waits until the TCP/IP layer times out to return a connection failure.
- Connection Retry Limit:** Specifies the number of times to attempt to connect to the LDAP server if the initial connection failed
- Parallel Connect Delay:** Specifies the number of seconds to delay when making concurrent attempts to connect to multiple servers. An attempt is made to connect to the first server in the list. The next entry in the list is tried only if the attempt to connect to the current host fails. This setting might cause your application to block for an unacceptably long time if a host is down. If the value is greater than 0, another connection setup thread is started after the specified number of delay seconds has passed. If the value is 0, connection attempts are serialized.

LDAP Caching

- All authenticators can cache a group's member list.
- LDAP Authenticators can also cache individual entries.



ORACLE

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

Enabling a cache involves a trade-off of performance and accuracy. Using a cache means that data is retrieved faster, but it runs the risk that the data may not be the latest available.

The time-to-live (TTL) setting specifies how long you are willing to accept potentially stale data. What this value should be depends upon your particular business needs. If you frequently change group memberships for users, then a long TTL could mean that group-related changes will not show up for a while. If group memberships almost never change after a user is added, a longer TTL may be fine. TTL attributes are specified in seconds.

The cache size is related to the amount of memory you have available, as well as the cache TTL. Consider the number of entries that might be loaded in the span of the TTL, and size the cache in relation to that number. A longer TTL will tend to require a larger cache size. For group membership caching, specify the number of groups to cache. For basic entry caching, specify the maximum size of the cache in kilobytes.

Multiple Authentication Providers

- A single security realm can support multiple authentication providers.
- For authenticators, *control flags* determine the processing logic as each provider is executed.

The screenshot shows two windows from the Oracle WebLogic Administration Console. On the left is a table titled 'Authentication Providers' with columns for Name, Description, and Type. It lists three providers: 'CorporateLDAP', 'HRDB', and 'SAMLIdentityAsserter'. A yellow callout box points to the 'Reorder' button at the top of the table, with the text 'Change execution order.' Inside the table, there are checkboxes for selecting individual rows. On the right is a detailed configuration dialog for the 'HRDB' provider. It has tabs for 'Configuration' and 'Performance', and sub-tabs for 'Common' and 'Provider Specific'. Under the 'Common' tab, there is a section for 'Control Flag' with a dropdown menu set to 'OPTIONAL'.

ORACLE

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

Each security realm must have at least one authentication provider configured. The WebLogic Security Framework supports multiple authentication providers for multipart authentication. Therefore, you can use multiple authentication providers as well as multiple types of authentication providers in a security realm.

The order in which WebLogic Server calls multiple authentication providers can affect the overall outcome of the authentication process. The Authentication Providers table lists the authentication providers in the order in which they will be called. By default, authentication providers are called in the order in which they were configured. Use the **Reorder** button on the **Security Realms > Providers > Authentication** page in the administration console to change the order in which authentication providers are called by WebLogic Server and listed in the console.

When you configure multiple authentication providers, also use the Control Flag for each provider to control how the authentication providers are used in the login sequence. When additional authentication providers are added to an existing security realm, by default the Control Flag is set to OPTIONAL. If necessary, change the setting of the Control Flag and the order of authentication providers so that each authentication provider works properly in the authentication sequence.

Control Flags

Flag	Explanation	Success Action	Failure Action
REQUIRED	Must succeed	Execute next provider	Execute next provider, but outcome is: FAIL
REQUISITE	Must succeed	Execute next provider	Return control to application with: FAIL
SUFFICIENT	Not required to succeed	Return control to application with: SUCCESS	Execute next provider
OPTIONAL	Not required to succeed	Execute next provider	Execute next provider



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

- **REQUIRED:** The authenticator is required to succeed. If it succeeds or fails, authentication still continues to proceed down the list. If it has failed, the overall outcome of the authentication is a failure. Use case: This authenticator must succeed, but you still wish to call other authenticators. Perhaps another of the authenticator performs some auditing function.
- **REQUISITE:** The authenticator is required to succeed. If it succeeds, authentication continues down the list. If it fails, control immediately returns to the application (authentication does not proceed down the list). Use case: This authenticator must succeed. If it fails, there is no need to call any other authenticator.
- **SUFFICIENT:** The authenticator is not required to succeed. If it does succeed, control immediately returns to the application (authentication does not proceed down the list). If it fails, authentication continues down the list. Use case: This authenticator does not have to succeed, but if it does, it is sufficient to validate the user, so no other authenticators need to be called.
- **OPTIONAL:** The authenticator is not required to succeed. If it succeeds or fails, authentication still continues to proceed down the list. Use case: This authenticator does not have to succeed. No matter what it returns, the overall outcome can be success or failure.

The overall authentication succeeds only if all *Required* and *Requisite* authenticators succeed. If a *Sufficient* authenticator is configured and succeeds, then only the *Required* and *Requisite* authenticators prior to that *Sufficient* authenticator need to have succeeded for the overall authentication to succeed. If no *Required* or *Requisite* authenticators are configured for an application, then at least one *Sufficient* or *Optional* authenticator must succeed.

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

Administration Groups

At least one authentication provider must exist that associates users with groups that have WebLogic Server administrative rights.

Group	Default Capability (via roles and policies)
Administrators	Full administrative access to the domain and its applications
Operators	View domain configuration, start and stop servers
Deployers	View domain configuration, deploy, undeploy, and update applications
Monitors	View domain configuration
AppTesters	Access applications running in admin mode (servicing administration requests) through the admin port

Often the default authentication provider retains the administrative users, groups, roles and policies; another authentication provider is added for “regular” users, groups, roles and policies.



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

For easier management of authentication, create groups and add users to them. The default Role Mapping and Authorization providers include policies that grant specific groups different administrative rights in the domain. You can change these policies and roles if desired, but be careful that you do not accidentally make your domain inaccessible. Also, having at least two WebLogic Server administrators at all times helps prevent a single user being locked out, which can make the domain configuration inaccessible until the lockout timeout expires.

The Administrators group contains the domain’s main administrative user and is assigned to the Admin role. Similarly, the OracleSystemGroup group contains a user named OracleSystemUser and is assigned to the OracleSystemRole role.

The remaining administrative groups have no users by default, but are mapped to these roles:

- Deployers group: Deployer role
- Operators group: Operator role
- Monitors group: Monitor role
- AppTesters group: AppTester role
- CrossDomainConnectors group: CrossDomainConnector role
- AdminChannelUsers group: AdminChannelUser role

Troubleshooting Authentication

- If you think users are doing things that they should not do, configure an auditing provider.
 - The default auditing provider can be quickly configured.
- Use the server logs.
 - Enable security realm debug flags for more detailed log messages.
- Check the external LDAP authentication provider configuration attributes.
- Use any debug capabilities of the external LDAP Server software.

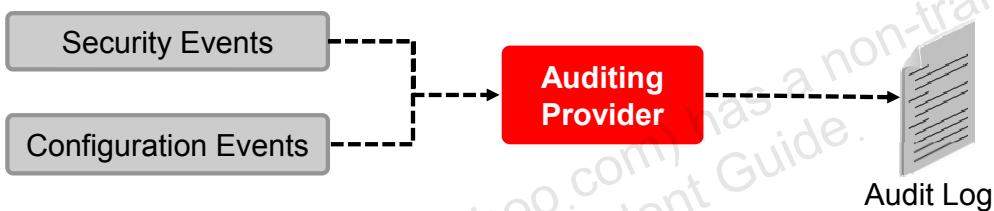
ORACLE

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

Auditing Provider

The WebLogic auditing provider:

- Creates a detailed record of all security changes and decisions within a domain in each server's logs directory to a file named DefaultAuditRecorder.log.
- Can also create a record of all domain configuration changes
- Is not enabled by default



ORACLE

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

Auditing is the process of collecting and storing information about requested operations and their outcome for the purposes of nonrepudiation. The goal of nonrepudiation is to be able to show that a particular operation is associated with a particular user, so later that user cannot claim someone else performed that operation. In WebLogic Server, an auditing provider performs this function. WebLogic Server includes a default auditing provider, but it is not activated for new security realms. The default auditing provider records information about a number of security requests, which are determined internally by the WebLogic Security Framework. The default auditing provider also records the event data associated with these security requests and the outcome of the requests.

You can also configure the administration server to create audit messages that enable the tracking of configuration changes in a domain. This provides a record of changes made to the configuration of any resource within a domain, as well as invocations of management operations on any resource within a domain. Configuration audit records can be saved to a log file, sent to an auditing provider in the security realm, or both.

All auditing information recorded by the default auditing provider is saved in <domainname>/servers/<servername>/logs/DefaultAuditRecorder.log by default. Although you configure the auditing provider at the domain security realm level, each server writes auditing data to its own audit log file in its logs directory.

Security Audit Events

- Typical security events:
 - An authentication or identity assertion is attempted.
 - A new role or policy is created.
 - A user account is locked out or is unlocked.
- Security events have the following characteristics:
 - Name
 - Severity (WARNING, ERROR, SUCCESS, and so on)
 - Zero or more “context attributes:”
 - Protocol, port, address
 - HTTP headers
 - EJB method parameters
 - SAML tokens



ORACLE

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

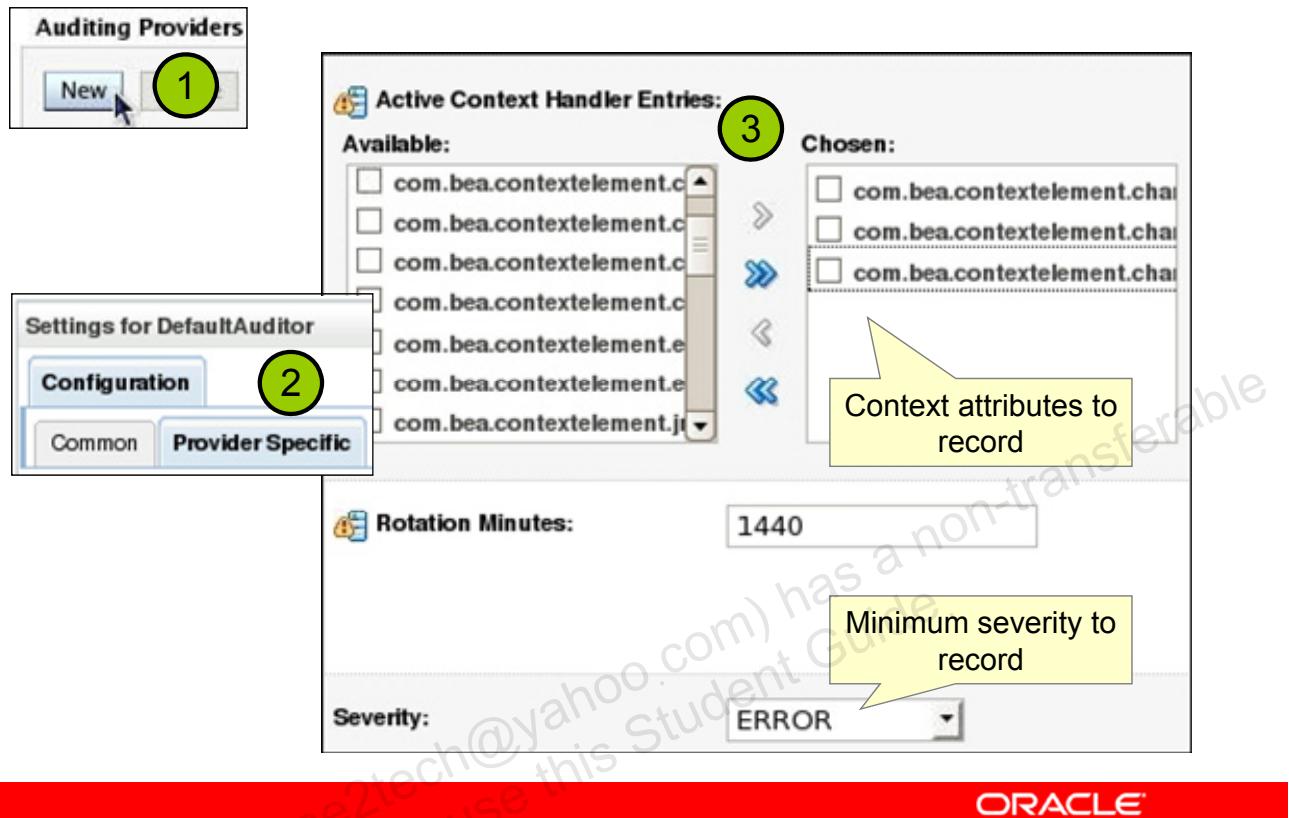
In addition to the events listed in the slide, the default WebLogic auditing provider records the following types of security events:

- When the lock-out on a user account expires.
- A security policy is used and an authorization decision is made.
- A role definition is used.
- A role or policy is removed or “undeployed.”

The WebLogic auditing provider audits security events of the specified severity and higher. The severity levels, in order from lowest to highest, are: INFORMATION, WARNING, ERROR, SUCCESS, FAILURE. You can also set the severity level to CUSTOM, and then enable the specific severity levels that you want to audit, such as ERROR and FAILURE events only.

An audit event includes a context object that can hold a variety of different attributes, depending on the type of event. When you configure an auditing provider, you specify which context attributes are recorded for each event. By default, no context attributes are audited.

Configuring the Auditing Provider



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

1. After locking the configuration, in the Domain Structure, select **Security Realms** and click the name of the realm you are configuring (for example, `myrealm`). Click the **Providers > Auditing** tabs. Click **New**. Give the provider a **Name** and keep the **Type of DefaultAuditor**.
2. Select the new provider in the table and click the **Configuration > Provider Specific** tabs.
3. Update the following fields, if desired:
 - **Active Context Handler Entries:** Specifies which context attributes are recorded by the auditing provider, if present within an event. Use the arrow buttons to move the Available entries to the Chosen list. The context attributes are things like `HttpServletRequest`, `HttpServletResponse`, `Port`, `Protocol`, `Address`, and so on.
 - **Rotation Minutes:** Specifies how many minutes to wait before creating a new audit file. At the specified time, the audit file is closed and a new one is created.
 - **Severity:** The minimum severity level an event must have to be recorded
 - There are more options if the Severity chosen is `CUSTOM`. There are also options to configure the format of audit log entries under “Advanced.”

Security Realm Debug Flags

Flag	Description
<code>DebugSecurityRealm</code>	Trace the initialization of the realm's providers and the loading of initial data from the default store.
<code>DebugSecurityAtn</code>	Trace the authentication and management of users and groups.
<code>DebugSecurityRoleMap</code>	Trace role policy evaluations and results.
<code>DebugSecurityAtz</code>	Trace authorization policy evaluations and access decisions.
<code>DebugSecurityAdjudicator</code>	Trace final authorization decisions.
<code>DebugSecurityUserLockout</code>	Trace the locking and unlocking of user accounts based on the number of invalid login attempts.
<code>DebugSecuritySAML*</code>	Trace the processing and/or generation of SAML tokens.

Multiple SAML security flags



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

The `DebugSecuritySAML*` flag is listed with an “*” to indicate that there are multiple SAML debug flags.

The flags with their scopes:

- `weblogic.security.realm.DebugSecurityRealm`
- `weblogic.security.atn.DebugSecurityAtn`
- `weblogic.security.rolemap.DebugSecurityRoleMap`
- `weblogic.security.atz.DebugSecurityAtz`
- `weblogic.security.adjudicator.DebugSecurityAdjudicator`
- `weblogic.security.userlockout.DebugSecurityUserLockout`

The `DebugSecuritySAML*` flags are found in the following scopes:

- `weblogic.security.saml.atn.*`
- `weblogic.security.saml.credmap.*`
- `weblogic.security.saml.lib.*`
- `weblogic.security.saml.service.*`
- And four more scopes, but replace `saml` with `saml2`

Common LDAP Issues

Typical causes include:

- The wrong base DN, object class, or attribute has been set for users or groups.
- A configured search filter is syntactically valid, but it is semantically incorrect.
 - So, it fails to retrieve the intended users or groups.
- An insufficient “maximum level for nested group memberships” has been set.
 - So, not all group members are found, which means some users are not mapped to their proper roles.
- WebLogic Server does not trust the LDAP server’s SSL certificate (and they are set to communicate over SSL).



ORACLE

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

After a user is authenticated, groups are searched to create a list of groups to which this user belongs. Then role mapping can occur between these groups and roles. If the user does not belong to any groups or if the search criteria are invalid, you will see debug messages:

```
<SecurityDebug> <search(...), base DN & below>
<SecurityDebug> <Result has more elements: false>
```

The **Max Group Membership Search Level** field, in the configuration of an LDAP authentication provider, specifies how many levels to search when looking for members of a group. (This setting is valid only if **Group Membership Searching** is set to limited.) A value of 0 indicates that only direct groups will be found. This means that, when searching for membership in Group A, only direct members of Group A are found. If Group B is a member of Group A, the members of Group B will not be found. A nonzero value indicates how many levels to search. For example, if it is set to 1, a search for membership in Group A will return direct members of Group A. If Group B is a member of Group A, the members of Group B will also be found. However, if Group C is a member of Group B, the members of Group C will not be found.

If WebLogic Server does not trust the LDAP server’s certificate and you have set them up to communicate over SSL, they will not be able to communicate. Perhaps the LDAP server’s certificate is from a CA that is not in WebLogic Server’s trust keystore.

Quiz

The WebLogic Server default security realm uses this as its security provider store by default:

- a. Oracle Database
- b. Embedded LDAP Server
- c. Derby Database
- d. OpenLDAP Server
- e. Any Database



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

Answer: b

Quiz

With LDAP, what does DN stand for?

- a. Directory Network
- b. Dynamic Name
- c. Distinguished Name
- d. Directory Name



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

Answer: c

Quiz

Which of the following is NOT an available authentication provider control flag?

- a. SUFFICIENT
- b. REQUISITE
- c. OPTIONAL
- d. ALWAYS
- e. REQUIRED



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

Answer: d

Summary

In this lesson, you should have learned how to:

- Describe the basics of the WebLogic Server security architecture
- Describe basic LDAP concepts
- Configure an external LDAP authentication provider for WebLogic Server



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

Practice 16-1 Overview: Configuring an Authentication Provider

This practice covers the following topics:

- Initializing Apache DS LDAP
- Setting DS LDAP as one of the authentication providers
- Setting the appropriate control flags
- Testing the new authentication provider

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.

11

Backing Up a Domain and Upgrading WebLogic Server

ORACLE®

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

Objectives

After completing this lesson, you should be able to:

- Back up a WebLogic Server domain
- Restore a WebLogic Server domain
- Describe the WebLogic Server upgrade process

Backup and Recovery

Backup

- Scheduled
- At least weekly
- Uses different tools for different components



Recovery

- Unscheduled (usually)
- At least annually (if only to test procedures)
- Not necessarily the reverse of backup; it may use other tools



Backup and recovery:

- Protect against failures of hardware or software, and accidental or malicious changes to the environment
- Guarantee a point of recovery and minimize loss of business availability
- May impact system availability (the system must be offline for an offline backup)
- May include hardware and software

ORACLE®

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

Commonly, the terms “backup” and “recovery” imply the use of secondary media to copy some data for later use. That kind of backup and recovery involves an offline or cold storage of the data such that if an outage occurs, then some process (human or automated) requires some time to get the system back up and running. Alternatively, “redundancy” and “failover” are additional means by which to back up and recover the data in more of an online or warm or hot storage mode, thus reducing, or even eliminating the switchover time. If an outage occurs with redundancy and failover implemented, it is often undetected by the user.

In addition to these features, a media backup plan is essential. The most common problems that require a backup and recovery are to overcome user error or media failure. A more serious problem is when there is a complete loss of the computer hosting the service.

Backup Solution

- Artifacts that need to be backed up include the database, installed products, configured WebLogic domains, WebLogic Server instances, and persistence stores for WebLogic Server TLogs (transaction logs) and JMS resources.
- Use Oracle Recovery Manager (RMAN) to back up database artifacts.
- Use file copy to back up product installations and configured domains, WebLogic Server instances, and persistent stores.
- You can also use the pack utility to back up managed servers.



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

Oracle Recovery Manager (RMAN) provides backup and recovery for the Oracle database. It provides both a command-line and Enterprise Manager interfaces.

In a Fusion Middleware environment, you should consider backing up the following directories and files:

- Static files (files or directories that do not change frequently)
 - Middleware home (`MW_HOME`): Middleware home contains the FMW products installations including WebLogic Server home, Oracle common home, and Oracle homes for other products. `MW_HOME` can also contain the `user_projects` directory, which contains Oracle WebLogic Server domains and Oracle instance homes, which are not static files.
 - `OraInventory`: In Linux and UNIX, the `oraInst.loc` file points to the directory where the FMW product inventory is located. The inventory is required for patching and upgrading installed components. The `oraInst.loc` file is in the `/etc` directory by default. You can specify a different location for this inventory pointer file by using the `oraInvPtrLoc` parameter during the installation of FMW products.

- Dynamic files (the runtime configuration files that change frequently)
 - Domain directories (admin server and managed servers): In most cases, you do not need to back up managed server directories separately because the domain where the administration server resides contains the configuration information for all of the managed servers in the domain, and the pack and unpack utilities can be used to re-create the domains for managed servers.
 - All Oracle instance homes that reside outside of the domain directory
 - Deployed applications, such as .ear or .war files that reside outside of the domain directory. You do not need to back up application artifacts in managed server directory structures because they can be retrieved from the Administration Server during managed server startup.
 - Database artifacts including any database-based metadata repositories used by Oracle Fusion Middleware. You can use Oracle Recovery Manager (RMAN) to back up an Oracle database.
 - Persistent stores, such as JMS resources and WebLogic Server TLogs (transaction logs)

Types of Backups

- Online:
 - Non-disruptive
 - Possibly inconsistent
 - If backing up takes one hour, the changes made during that hour will not be within the backup and must be tracked
- Offline:
 - Requires all processes to be stopped
 - Relatively easy
- Full:
 - Easier to recover, slower to create
- Incremental:
 - Harder to recover, faster to create



ORACLE

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

Online

If your environment requires 24x7 availability, you have no choice but to perform online backups. Different components require different tools to perform online backups (also known as hot or inconsistent backups). Inconsistent is not bad in itself; it just means that if the backup takes an hour to complete and you start at 1:00 AM, the files at 1:02 AM will be in a different state than those backed up at 1:59 AM. To accommodate this, there needs to be some kind of online log recording the changes occurring from 1:00 AM until 2:00 AM. This log needs to be incorporated into the recovery, and the logs themselves get backed up at a different time (usually, after they rotate).

Offline

If you can afford to shut down the entire middleware tier (application servers, database, web servers, and so on) for maintenance during some regularly scheduled time, an offline backup is fairly simple (also known as a cold or consistent backup). Using operating system tools such as TAR or ZIP, the backup is guaranteed to be consistent. Make sure you preserve file permissions on UNIX systems.

Full

After the initial installation, or after a major set of patches, a full backup should be performed. Often, this is done before going live, so the system is offline. It is very difficult (if not impossible) to perform a full backup online. If there is a complete loss of a host (for example, a disaster such as a fire or flood), recovery is simple; just copy the backup files to the new host and boot.

Name a backup so as to include the date, for example, `full_backup_2013_04_30.tar`, and keep several generations of them in case you accidentally capture “a problem” in the most recent backup.

Incremental

Considering that the executable files and the configuration files are usually backed up separately, most backups are incremental. Backing up “changes only” may require several sets of backups to perform a full recovery. RMAN can help automate this for databases, especially if the backups are kept online (on disk as opposed to tape).

You can make an incremental backup at the functional level. For example, you can make a WebLogic backup from `<WL_HOME>`, make an instance backup from `<ORACLE_INSTANCE>`, make a database backup from `<ORACLE_HOME>`, and so on. With WebLogic Server, make a backup of all domains and then make backups of individual domains. The disadvantage of doing this is that the backup process will take longer, but the advantage is that the recovery process can be simplified. Alternatively, if you do not make so many different kinds of incremental backups, the backup procedure will complete faster, but now you have complicated and lengthened your potential recovery time. It is a delicate tradeoff balancing storage space versus time versus complexity.

When to Back Up

- Offline backup after the initial domain is created
- Online backups at scheduled intervals
- Online backup after a component changes or the cluster configuration changes
- Online backup before application deployment
- Offline backup before and after patches or upgrades
- Online “database” backups for:
 - LDAP
 - Persistent stores
 - SOA repository



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

The initial software installation and most patches and upgrades require the servers to be offline anyway, so before and after the patches and upgrades is a good time to perform backups.

Many of the online configuration backups can be automatic by enabling the automatic backup of the domain configuration archive (discussed in the following slides).

The database should be in `archivelog` mode and then backed up with RMAN. In addition, the database should be configured with redundant critical files (for example, control files) and multiplexed critical logs (for example, redo logs). As an added high availability measure, the database can be made completely redundant by using Oracle Real Application Clusters (RAC).

Limitations and Restrictions for Online Backups

- Online backups of WebLogic Server persistent stores are likely to be inconsistent (changes can occur while you are backing up).
 - Database backups can more easily accommodate inconsistencies.
 - File-based stores and OS copies cannot easily accommodate online backup.



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

None of these restrictions apply to offline backups; they apply only to online backups. In many cases, WebLogic Server has the option to be configured to use either database or file storage for information. Choosing the database is a safer option, but you pay for it with greater complexity and slower performance. If your system uses a database, and the DBA is backing it up, then some additional WebLogic Server tables should not be any additional effort for them. For files such as the configuration XML, application JARs, WARs, or EARs, and properties files, database storage is not an option.

Performing Full Offline Backup

1. Shut down all the processes.
2. Back up <MW_HOME>.
3. Back up the domain.
4. Back up directories from which applications are deployed.
5. Back up the managed server domains on other machines or re-create their domains with the pack/unpack utilities.
6. Back up the instance home for configured system components (like OHS).
7. Back up the database using RMAN.
8. Back up Oracle Inventory.
9. Back up the oraInst.loc and oratab files (in /etc).



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

To perform a full offline backup:

1. Shut down all processes in Middleware home: 1) system components 2) managed servers 3) admin server 4) Node Managers 5) database 6) database listeners.
2. Back up the Middleware home directory, <MW_HOME>. For example:

```
tar -cpf mw_home_backup_04-12-2013.tar $MW_HOME/*
```

3. Back up the domain. For example:

```
tar -zcpf domain_backup_04-12-2013.tarz domain_dir/*
```

Note: It is also possible to use the pack utility.

4. Back up the directory from which applications are deployed. For example:

```
tar -zcpf app_backup_04-12-2013.tarz app_dir/*
```

Note

- If you deploy applications from a directory inside the domain, this is unnecessary.
- If you used the pack utility to back up the domain, the deployed applications are already in the JAR file in a directory called _apps_.

5. If the managed servers run on the same computer as the admin server, they use the same domain directories, so do nothing for this step. Managed server domains on other computers can be backed up in the same way as the admin server domain, or they can be recreated with the pack and unpack utilities.
6. Back up the Oracle instance home. The Oracle instance home contains configuration information about system components, such as Oracle HTTP Server or Oracle Internet Directory.
 - For example:

```
tar -cpf instance_home_backup_04-12-2013.tar
$ORACLE_INSTANCE/*
```
7. Back up the database repositories by using the Oracle Recovery Manager (RMAN). If you are doing a full offline backup, you do not need to use RMAN, instead you can just create a TAR file of the database.
8. Back up the OralInventory directory.
 - For example:

```
tar -cpf orainven_home_backup_04-12-2013
/u01/app/oracle/oraInventory
```
9. Back up the `oraInst.loc` and `oratab` files, which are usually located in the `/etc` directory.
10. Back up the `oraenv` utility found in the user `bin` directory, for example, `/usr/local/bin`.

Performing Full Online Backup

1. Lock the WebLogic Server configuration.
2. Back up the domain. For example:

```
$> tar -zcpf domain_backup_04-12-2013.tarz /domain_dir/*
```

3. Back up the application directories.

```
$> tar -zcpf app_backup_04-12-2013.tarz /app_dir/*
```

4. If the managed servers are in another location, back up those domain directories.
5. Back up the Oracle instance home.
6. Back up the database with RMAN.



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

You should perform a backup of runtime artifacts on a regular basis. To back them up:

1. To avoid an inconsistent backup, lock the WebLogic Server configuration, and do not make any configuration changes until the backup is completed.
2. Back up the domain. This backs up Java components such as Oracle SOA Suite and Oracle WebCenter. For example:

```
tar -zcpf domain_backup_04-15-2013.tarz  
domain_path/domain_name/*
```

You can also use the pack utility. The advantage of using pack is that it results in a portable JAR file and it also automatically saves all the deployed applications (in a directory in the JAR file called `_apps_`).

3. If the applications are deployed from a directory under the domain directory, you can skip this step. You can also skip this step if you used the pack utility.
4. Back up the managed server directories if they are on a different machine.
5. Back up the Oracle instance home, backing up system components, such as OHS. For example:

```
tar -cpf instance_home_backup_04-15-2013.tar  
      <ORACLE_INSTANCE>/*
```

Note: Inform administrators to refrain from configuration changes until after the backup.

6. Back up the database repositories by using the Oracle Recovery Manager (RMAN).

Impact of Administration Server Failure

- Failure of the administration server:
 - Prevents configuration changes in the domain
 - Prevents application deployments
 - Does not affect running managed servers
 - Prevents starting never-started-before managed servers
 - Allows starting previously-started managed servers if Managed Server Independence (MSI) mode is enabled
 - MSI is enabled by default
- Periodically, the managed servers attempt to synchronize configuration data with the administration server.
- When the administration server becomes available, the managed servers get the latest configuration data from the administration server.



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

When you first start a managed server, it must be able to connect to the administration server to retrieve a copy of the configuration. Subsequently, you can start a managed server even if the administration server is not running.

If a managed server cannot connect to the administration server during its start up, then it uses the locally cached configuration information. A managed server that starts without synchronizing its configuration with the administration server is running in Managed Server Independence (MSI) mode. By default, MSI mode is enabled. However a managed server cannot be started in MSI mode for the first time as the local configuration is not available.

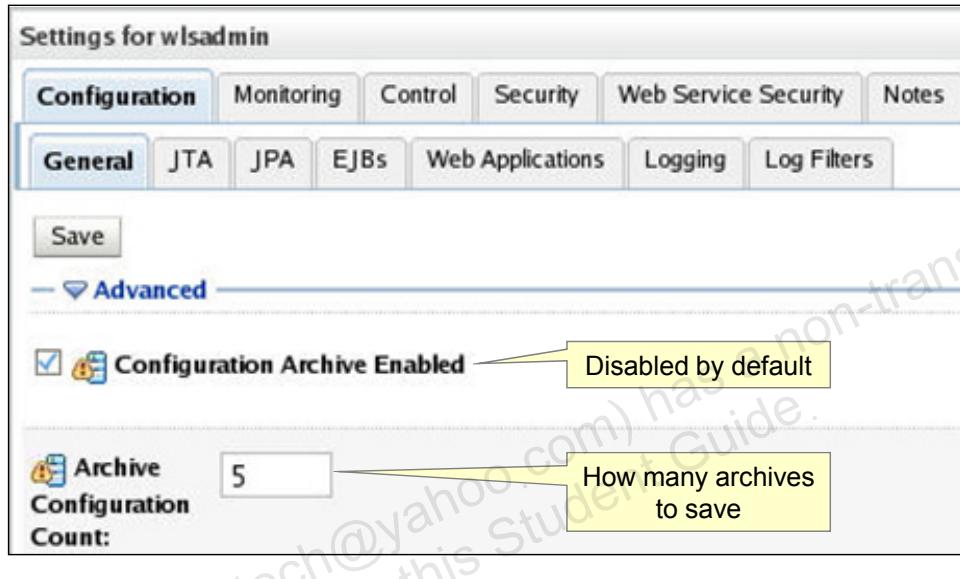
The failure of an administration server does not affect the operation of managed servers in the domain, but it does prevent you from changing the domain's configuration. If an administration server fails because of a hardware or software failure on its host computer, other server instances on the same computer may be similarly affected.

If an administration server becomes unavailable while the managed servers in the domain are running, those managed servers continue to run. Periodically, the managed servers attempt to reconnect to the administration server. When the connection is successful, the configuration state is synchronized with that of the administration server.

For clustered managed server instances, the load balancing and failover capabilities supported by the domain configuration continue to remain available.

Automatically Backing Up a Domain Configuration

Enabling this attribute causes a JAR file of the entire config directory to be created each time a configuration change is activated.



ORACLE

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

Under ***the_domain_name > Configuration > General > Advanced***, you can enable the automatic backup of the configuration at the domain level. Each startup of the administration server creates two files in the domain directory: config-booted.jar and config-original.jar. In addition, each activated change of the configuration makes a backup named configArchive/config-n.jar, where *n* is a sequential number. The Archive Configuration Count attribute limits the number of retained configuration JARs. In the example shown, there are never more than five archive files kept. After five backups, older backups are automatically deleted.

You may want to set a higher number such as 10 or 20 for the Archive Configuration Count depending on:

- The available disk space
- The need for backup and restoration
- The time taken for backup and restore activity

Recovery Operations

Some of the common recovery operations include restoring:

- A Middleware home
- An Oracle home
- An Oracle WebLogic Server domain
- The administration server configuration
- A managed server
- An Oracle instance
- Fusion Middleware system component configurations and data



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

To restore a Middleware home:

- Stop all relevant processes that are related to the domain, such as the administration server, Node Manager, and managed servers.
- Restore the Middleware home directory from a backup.
- Start all relevant processes that rely on the Middleware home.

To restore an Oracle WebLogic Server domain:

- Stop all processes that are related to the domain, such as the administration server and managed servers.
- Restore the domain directory from backup.
- Start all processes that are related to the domain.
 - If you cannot start the administration server, or managed server, you may need to perform recovery of those components.

If the administration server configuration has been lost because of file deletion or file system corruption, the administration server console continues to function if it was already running. The administration server directory is regenerated automatically. However, the security information is not generated. As a result, whenever you start the administration server, it prompts for a username and password. To prevent this, you can recover the configuration.

To recover a managed server :

- If the administration server is not reachable, recover the administration server, as previously described.
- If the managed server fails to start or if the file system is lost, perform the following steps:

- Recover the Middleware home from the backup, if required, as in the following example:

```
tar -xpf mw_home_backup_04-12-2013.tar
```

- Create a domain template JAR file by using the pack utility, as in the following example:

```
pack.sh -domain=path_to_doman/domain_name  
-template=/scratch/temp.jar -template_name=test_install  
-template_author=name -log=/scratch/logs/my.log -managed=true
```

- Copy the JAR file to the managed server computer. Unpack the domain template JAR file by using the unpack utility:

```
unpack.sh -template=/location_of_copy/temp.jar  
-domain=path_to_doman/domain_name  
-log=/scratch/logs/new.log -log_priority=info
```

- Ensure that the application artifacts are accessible from the managed server host. That is, if the application artifacts are not on the same server as the managed server, they must be in a shared location accessible by the managed server.
- Start the Node Manager on the machine, if it is not already running.
- Start the managed server using the Oracle WebLogic Server administration console or WLST.

Restoring components:

- You can restore a component's files if they are deleted or corrupted, or if an error occurred during configuration of a component. In these cases, you may want to revert to an earlier version. For Java components, perform the steps to restore a managed server. For System components such as OHS, Oracle Web cache, and so on:
 - Stop the component.
 - Restore the files from the appropriate backup.
 - Start the component.

Note the following key points about recovery:

- Your Fusion Middleware environment must be offline while you are performing recovery.
- Rename the important existing files and directories before you begin restoring files from backup, so that you do not unintentionally overwrite necessary files.
- Although, in some cases, it may appear that only one or two files are lost or corrupted, you should restore the directory structure for the entire element, such as an Oracle instance or a component, rather than just restoring one or two files. In this way, you are more likely to guarantee a successful recovery.
- Recover the database to the most current state, using point-in-time recovery (if the database is configured to be able to do this). This is typically a time right before the database failure occurred.

Directories to Restore

- Binaries (installation directories)
 - Be mindful of preserving group ownership and permissions.
 - These should be read-only for most users.
- Configurations
 - If the last configuration *caused* the problem, recover to a point in time prior to that.
- Log files are:
 - Not required for recovery
 - Created if they do not exist
- Data
 - Database restores data within tablespaces, not directories.
 - RMAN *restore* brings data up to the last backup, then *recover* brings data up to a later point in time.



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

In most cases, recovery is performed offline. If you think that only one or two files are missing, you may be tempted to recover only those individual files from the backups. However, instead, you should always recover whole directories because there may be other files that are related to these files.

If the directories were backed up from the root, you do not need to worry about the directory you are in when you recover them. The full path information is provided to the operating system, because it is contained in the backup. Restore them as the `root` user, from the root directory, and they will go back to their correct hierarchies. Do not forget the `-p` switch in the `tar` or `jar` command to get the original owner and group information correct.

Recovery After Disaster

- Possible causes of failure:
 - Data loss
 - User error
 - Malicious attack
 - Corruption of data
 - Media failure
 - Application failure
- Recovery depends on the cause:
 - Repair
 - Replace
 - Relocate



ORACLE

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

If the problem was caused by a minor configuration error, the administrator may be able to reverse the steps and remove the problem without a formal recovery. If the problem requires replacing hardware, restore using full backups. Recovery is complicated when you need to relocate some functions to an existing machine. According to the old configuration (and backups), the functions must be routed to the old name and address of “A,” but now according to the new configuration, the functions need to be routed to the new name and address of “B.”

Recovery of Homes

This applies to recovering a Middleware home, an Oracle home, or an Oracle instance home after data loss or corruption:

1. Stop all processes.
2. Make a new full offline backup as a checkpoint (which can be reused).
3. Change directory to the affected home.
4. Use the OS copy, tar -x, or unzip command for the directories affected.
5. Make a new full offline backup (especially if you have been performing incremental backups up until this point).
6. Restart all processes: A. Database listener B. Database C. Oracle instances (OHS, OID) D. Node Manager E. Administration server F. Managed servers



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

Ensure that all Fusion Middleware software is stopped so that this is an offline recovery. By performing the two extra backups, you guarantee that you can at least put everything back to the way it was before you tried the recovery.

Recovery of a Managed Server

- If the managed server fails, Node Manager will automatically restart it (if it started it).
- If the files are damaged, you can recover the files in their original places and restart the managed server.
- If the computer is damaged, perform either of the following:
 - Restore the files on a new host with the old computer name by using the OS commands, for example, copy, cp, tar, or unzip. (If you backed up by using pack, restore by using unpack.)
 - Restore the files on another host with a different host name by using pack and unpack.

If you used a virtual host name on the old computer, then even if the new computer has a different name, you can still use OS commands to restore the files. Just assign the new computer the same virtual host name.



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

The original pack command that created the remote managed server domain JAR file can be used to recreate the managed server domain in a recovery. The significant configuration and application files are stored at the administration server, so when the managed server is started, it first refreshes all its configuration information and redeploys all its applications from the administration server.

Recovery of the Administration Server

- If the administration server fails, and it was started by using Node Manager (through a WLST script), then Node Manager automatically restarts it.
- If the files are damaged, you can recover the files in their original places and restart the administration server.
- If the computer is damaged, restart the administration server on a new computer.

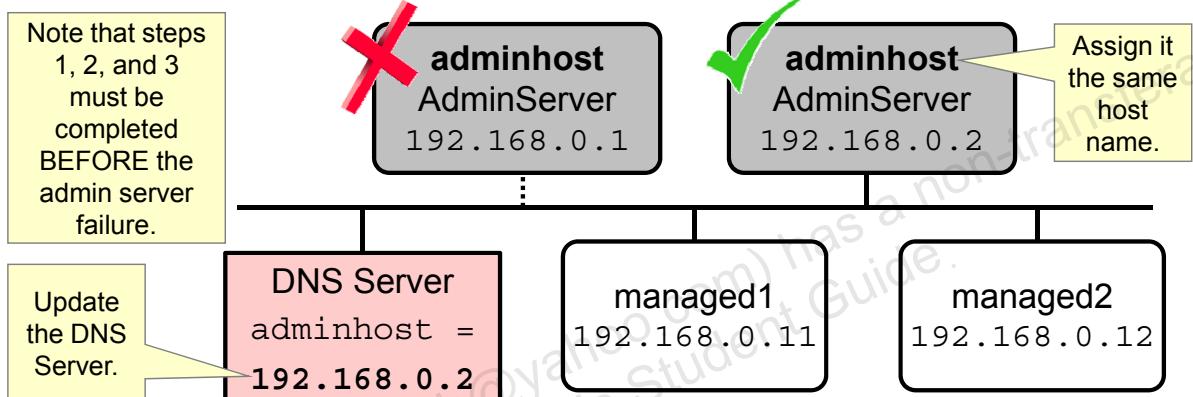
ORACLE

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

Restarting the Administration Server on a New Computer

To create a backup of the administration server:

1. Install Oracle WebLogic Server on the backup computer.
2. Copy the application files to the backup computer.
3. Copy the configuration files (or the domain) to the backup computer.
4. Restart the administration server on the backup computer.



ORACLE

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

If a hardware crash prevents you from restarting the administration server on the same computer, you can recover the management of the running managed servers as follows:

1. Install the Oracle WebLogic Server software on the new computer designated as the replacement administration server.
2. Make the application files available to the new administration server by copying them from backups or by using a shared disk. The files must be available in the same relative location on the new file system as on the file system of the original administration server.
3. Make the configuration and security files available to the new administration computer by copying them from backups or by using a shared disk. These files are located under the directory of the domain being managed by the administration server. An easier option is to copy the entire domain directory to the backup computer.
4. Restart the administration server on the new computer.

Note: In order for managed servers to reconnect after an administration server is restarted on a different IP address, you must have configured a DNS name for the administration server URL that maps to multiple IP addresses. Alternatively, you could have the administration server's listen address set to a virtual host name, and switch the virtual host to the new IP address. Or, if you are using floating IP addresses, assign the administration server's old IP address to the new hardware before restarting the administration server on that hardware.

Important: You cannot have two administration servers running at the same time, both claiming ownership of the same managed servers. Therefore, the administration server standby cannot be a warm standby; it must be a cold standby. The original administration server must be stopped or dead before starting the new administration server on the new hardware.

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

Managed Server Independence

Managed Server Independence (MSI) mode reduces the urgency to restart a failed admin server.

Settings for MedRecSvr1

Configuration Protocols Logging Debug Monitoring Control Deployments Services Security

General Cluster Services Keystores SSL Federation Services Deployment Migration **Tuning**

Save

Use this page to tune the performance and functionality of this server.

Advanced

Managed Server Independence Enabled

Specifies whether this Managed Server can be started when the Administration Server is unavailable. [More Info...](#)

Period Length: The time interval in milliseconds of the heartbeat period. A value of 0 indicates that heartbeats are turned off. [More](#)

Idle Periods Until Timeout: The number of idle periods until peer is considered unreachable. [More Info...](#)

Save

ORACLE

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

The administration server is required only for making changes to the active configuration. It is not required for the normal operation of the managed servers, as long as the managed servers have Managed Server Independence mode enabled, which is the default. This allows you time to recover the administration server without any service outages.

As shown in the screenshot, the heartbeat detected between the administration server and the managed servers is, by default, a one-minute time period. After four minutes of not hearing from the administration server, the managed servers become independent. After the administration server is fixed, the heartbeats start up again and the managed servers deactivate their independence, but MSI is still enabled for a future event. These default times can be changed to suit your particular environment.

Upgrading WebLogic Server 11g to 12c

1. Plan the upgrade.
 - A. Inventory the environment (admin server, managed servers, applications, external resources, scripts/templates).
 - B. Verify that all hardware and software components are supported.
 - C. Verify the compatibility of your applications.
 - D. Create an upgrade plan.
2. Prepare to upgrade
 - A. Undeploy incompatible applications.
 - B. Shut down servers.
 - C. Back up the environment.
 - D. Install new Oracle products.
 - E. Prepare remote managed server domains.
 - F. Set up environment variables.

ORACLE

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

1. Plan

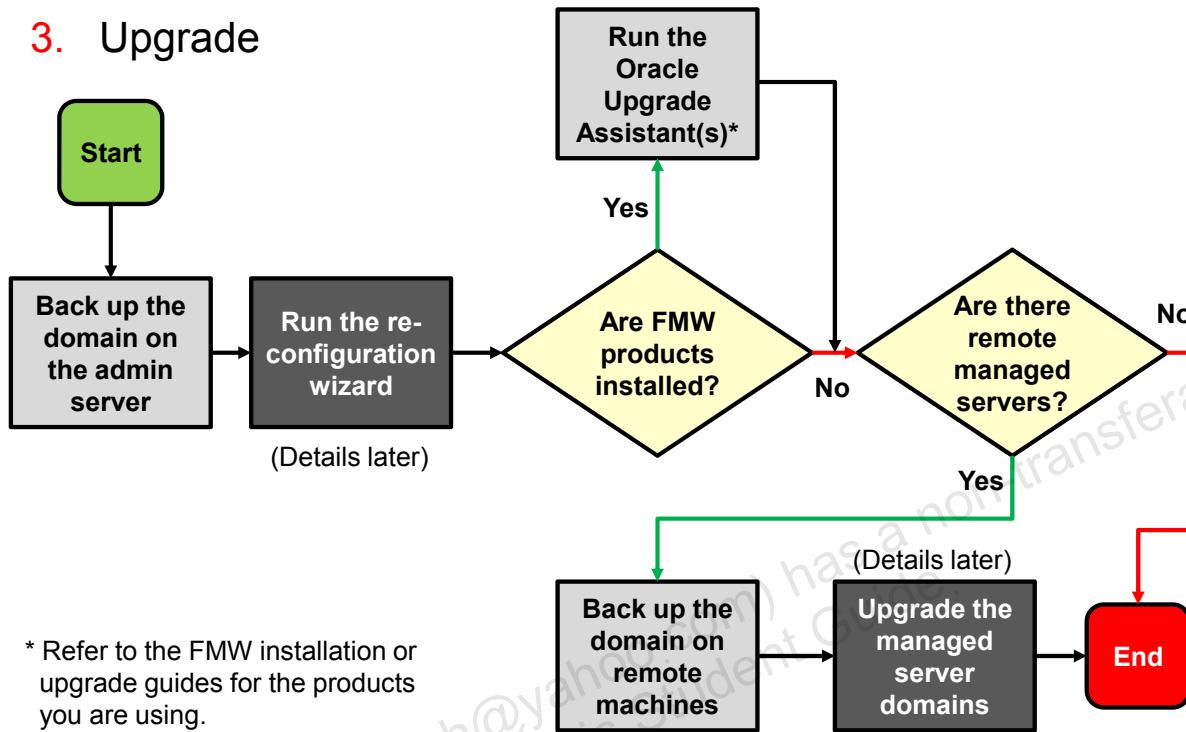
- A. The inventory lists all instances of WebLogic Server and the computers on which they reside, the location of all applications, external resources like databases, firewalls, load balancers, and scripts and templates.
- B. Use the *System Requirements and Supported Platforms* spreadsheet to determine if the hardware and software components in the application environment are supported.
- C. Use the *WebLogic Server Compatibility with Previous Releases* appendix to determine any changes that may affect your applications.
- D. Create an upgrade plan. Oracle recommends that you upgrade an application in development environments and use a standard QA, testing, and staging process to move upgraded applications to a production environment. If your environment is complex, you may want to upgrade components in stages.

2. Prepare

- A. In most cases, applications can be run without modification. Use the *WebLogic Server Compatibility with Previous Releases* appendix to see whether your applications use any deprecated or removed features. If so, you may need to modify or undeploy those applications.
- B. Shut down all servers in the environment before upgrading.
- C. Back up the environment:
 - i. Back up the domains on the computer where the admin server runs and the computers where the managed servers run. (Note that the Domain Upgrade wizard, which automatically backed up the domain being upgraded, is no longer provided with WebLogic Server.)
 - ii. Back up applications and data that reside outside of the domain directories.
 - iii. If you do not need a record of log files, you may want to delete them to conserve disk space.
- D. Install the new Oracle products on each computer in the domain.
- E. Prepare the domain directories on managed server computers by copying the following files from the pre-upgraded admin server domain directory to the managed server domain's "root directory:" /config/config.xml and /security/SerializedSystemIni.dat.
- F. In a terminal window run the <WL_HOME>/server/bin/setWLSEnv.sh script.

Upgrading WebLogic Server 11g to 12c

3. Upgrade



ORACLE

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

3. Upgrade: The steps in the flow chart that are dark gray are explained further in other slides. Note that if you need to upgrade from a version of WebLogic Server prior to version 10.3.1 (WebLogic Server 10g), you must first upgrade to version 10.3.6 (WebLogic Server 11g), then upgrade that to version 12.1.x (WebLogic Server 12c). You also must use the 11g Domain Upgrade wizard to upgrade the domain.

Run the Reconfiguration Wizard

- A. In the terminal window, run <MW_HOME>/oracle_common/common/bin/reconfig.sh.
- B. Go through the wizard screens.
- C. Manually finish the Node Manager configuration.



ORACLE

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

- C. To finish the Node Manager configuration:
 - Run <WL_HOME>/server/bin/startNodeManager.sh.
 - Copy the <WL_HOME>/common/nodemanager.properties file from the previous installation into the <MIDDLEWARE_HOME>/oracle_common/common/nodemanager/ directory of the new installation.
 - Shut down and restart Node Manager.
 - Verify that you can start servers through Node Manager.

Upgrade the Managed Server Domains

- A. Ensure that during the preparation phase, you copied these files from the pre-upgraded admin server domain directory to the managed server domain's "root directory:"
`/config/config.xml` and
`/security/SerializedSystemIni.dat`.
- B. Port the reconfigured domain from the admin server computer to the managed server computers with `pack` and `unpack`.



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

Upgrading WebLogic Server 11g to 12c

4. Complete post-upgrade procedures:
 - A. Re-apply any customizations you had in server start scripts.
 - B. Verify and reset file permissions (in Linux, file ownership goes to the user that did the upgrade).
 - C. Verify server start options (for example, JAVA_HOME and CLASSPATH may need to be updated for servers started via Node Manager).
 - D. After the environment has been tested, move it to production.

ORACLE

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

4. Post-upgrade

- A. The Upgrade Wizard does not carry forward any customizations that have been made to the default startup scripts. After the upgrade process is complete, you must customize the default scripts again.
- B. If you backed up the domain directory as part of the upgrade, you should make your backup files secure because they might contain confidential information. During the upgrade process, file permissions are not preserved. If non-default file permissions are set on files, they must be verified and reset. On a UNIX system, ownership and permissions for any new files created during the upgrade process are assigned to the user performing the upgrade.
- C. When you start the administration server, verify the remote server start options, such as JAVA_HOME, MW_HOME, BEA_HOME, and CLASSPATH, reference the WebLogic Server 12.1.x installation on the target managed server. This can be accomplished using the administration console on the **Configuration > Server Start** screen of the server.
- D. Test your applications in the new environment. If your applications use any deprecated or removed APIs, they can be modified and tested again. Once thoroughly tested, move the environment into production.

Quiz

The administration server of the domain has failed. Can a managed server currently not running be started?

- a. Yes, if Managed Server Independence Mode is enabled and the server has been started before.
- b. No, a managed server must always contact its admin server when it comes up.



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

Answer: a

Summary

In this lesson, you should have learned how to:

- Back up a WebLogic Server domain
- Restore a WebLogic Server domain
- Describe the WebLogic Server upgrade process

Practice 17-1 Overview: Backing Up and Restoring a Domain

This practice covers the following topics:

- Backing up a domain
- Restoring a domain

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 Startup and Crash Recovery

5

ORACLE

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

Objectives

After completing this lesson, you should be able to configure the Node Manager to start:

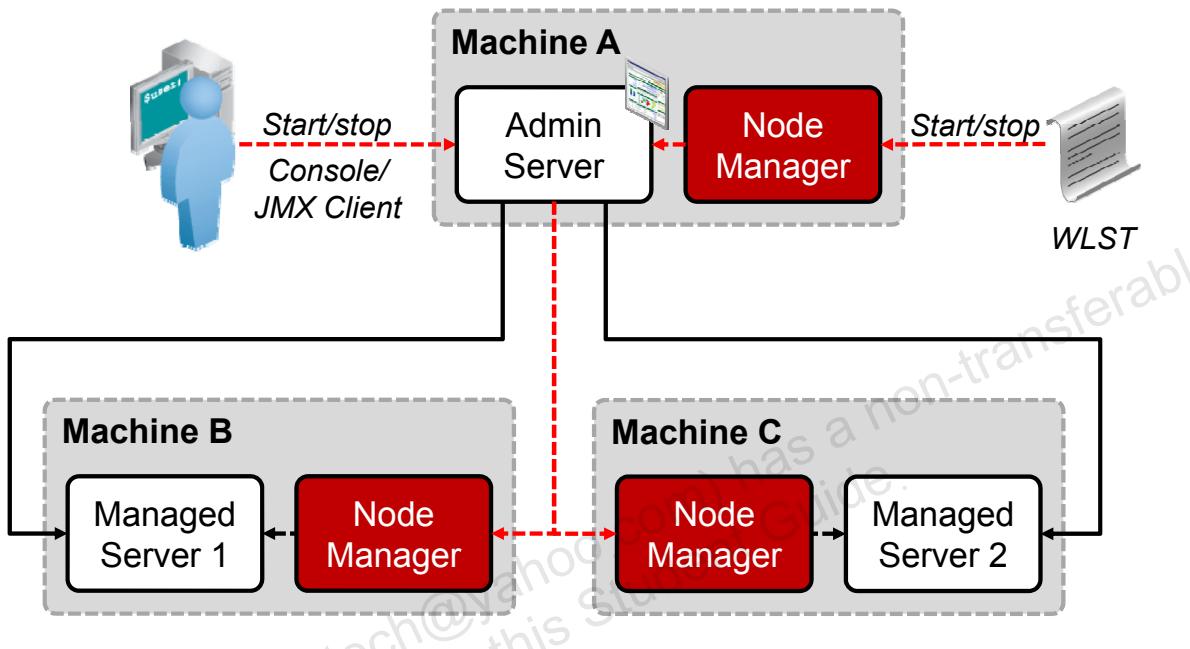
- Failed servers
- On system boot

Agenda

- Node Manager Review
 - Node Manager Architecture
 - Node Manager Default Behavior
 - Configure Java-Based Node Manager
- Configure the Node Manager to Start on System Boot
- Server Automatic Restart

Node Manager Architecture

Each Node Manager monitors and restarts servers it has started.



ORACLE

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

The graphic in the slide illustrates the relationship between a Node Manager, its clients, and the server instances that it controls. You can access the Node Manager using the following clients:

- **Administration server:** From the administration console, select Environments > Machines > Configuration > Node Manager page.
- **JMX utilities**
- **WebLogic Scripting Tool (WLST) commands and scripts:** WLST offline serves as the Node Manager command-line interface that can run in the absence of a running administration server. You can use the WLST commands to start, stop, and monitor a server instance without connecting to an administration server. Starting the administration server is the main purpose of the stand-alone client. However, you can also use it to:
 - Stop a server instance that was started by a Node Manager
 - Start a managed server
 - Access the contents of a Node Manager log file
 - Obtain the server status for a server that was started with a Node Manager
 - Retrieve the contents of the server output log

Node Manager Default Behavior

- After WebLogic is installed and a domain is created, the Node Manager is “ready-to-run”.
- By default, the following behaviors are configured:
 - The Node Manager can start servers using WLST.
 - The administration server uses Node Managers to start managed servers.
 - The Node Manager monitors all the servers that it started.
 - The automatic restart of all servers is enabled.

This is initiated by the administration console or WLST.



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

The Node Manager is ready to run after WebLogic Server is installed, a domain is created, and you use the demonstration secure sockets layer (SSL) configuration. The Node Manager restarts the server instances that it killed or that terminated unexpectedly.

Configure Java-Based Node Manager

- It is recommended that you configure the Node Manager to run as an operating system service.
- The configuration tasks for the Java-based Node Manager include:
 - Reconfiguring the startup service for a Windows installation
 - Running the Node Manager as a daemon process for UNIX systems
 - Configuring Java-based Node Manager security
 - Reviewing `nodemanager.properties`
 - Configuring the Node Manager on multiple machines



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

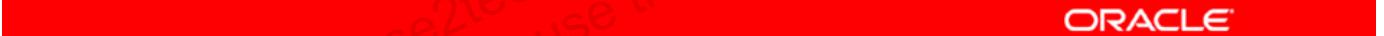
- `nodemanager.properties`: This is the configuration file that is used by the Java-based version of Node Manager. This file is located in `WL_HOME/common/nodemanager`.
- `nodemanager.domains`: This file contains mappings between the names of the domains managed by the Node Manager and their corresponding directories. This file is located in `WL_HOME/common/nodemanager`.
- `nm_data.properties`: This file stores the encryption data that the Node Manager uses as a symmetric encryption key. The data is stored in an encrypted form. This file is located in `WL_HOME/common/nodemanager`.

Agenda

- Node Manager Review
- Configure the Node Manager to Start on System Boot
 - Windows
 - Linux
 - Solaris
- Server Automatic Restart

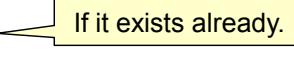
Starting the Node Manager at System Startup

- It is recommended that you run the Node Manager as:
 - A Windows service on Windows platforms
 - A daemon process on UNIX platforms
- Running the Node Manager during system startup allows it to restart automatically when the system is rebooted.
- You configure the Node Manager to start at boot time as:
 - A Windows service
 - A UNIX daemon process

ORACLE

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

Configuring the Node Manager as a Windows Service

1. Delete the Node Manager service by using `uninstallNodeMgrSvc.cmd`.

2. Edit `installNodeMgrSvc.cmd` to specify the Node Manager's listen address and listen port.
3. Run `installNodeMgrSvc.cmd` to reinstall the Node Manager as a service, listening on the updated address and port.

ORACLE

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

The `$WL_HOME/server/bin` directory (where `WL_HOME` is the top-level directory for the Oracle WebLogic Server installation) contains `uninstallNodeMgrSvc.cmd`, a script for uninstalling the Node Manager service, and `installNodeMgrSvc.cmd`, a script for installing the Node Manager as a service.

1. Delete the Node Manager service by using `uninstallNodeMgrSvc.cmd`.
2. Edit `installNodeMgrSvc.cmd` to specify Node Manager's listen address and listen port.

Note: By default, the service installer configures the Node Manager to run on `localhost:5556`. You edit the installation file to change the host address and port number. This slide refers to this reconfiguration.

Make the same edits to `uninstallNodeMgrSvc.cmd` as you make to `installNodeMgrSvc.cmd`, so that you can successfully uninstall the service in the future.

3. Run `installNodeMgrSvc.cmd` to reinstall the Node Manager as a service, listening on the updated address and port.

Configuring the Node Manager on UNIX Systems

UNIX and Linux operating system provide multiple different utilities for running services:

Utility	Description
init.d	A directory that contains system initialization and termination scripts that are used to configure and control system services.
inetd	Internet service daemon that manages Internet-based services.
xinetd	Extended Internet daemon that managers Internet-based services. The xinetd utility offers access control mechanisms including access control lists, logging capabilities, service availability by time, and limits on number of servers to start.
smf	The Solaris service management facility that is used to define and manage system services.



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

On UNIX, the Node Manager must be configured to run as a daemon manually. One of the key differentiators between init.d and smf and the *inetd utilities is that init.d and smf both start services immediately as part of the operating system initialization process. The *inetd utilities start services when a network request for the configured service is requested. Because of this difference, smf and init.d utilities are better choices for automatically starting WebLogic servers because processes will be started before requests are made for them.

Configuring the Node Manager as an init.d Service

```
#!/bin/sh
# nodemgr Oracle Weblogic NodeManager service
# chkconfig: 345 85 15
# description: Oracle Weblogic NodeManager service
# The script needs to be saved as /etc/init.d/nodemgr and
# then issue chkconfig --add nodemgr as root

### BEGIN INIT INFO
# Provides: nodemgr
# Required-Start: $network $local_fs
# Required-Stop:
# Should-Start:
# Should-Stop:
# Default-Start: 3 4 5
# Default-Stop: 0 1 2 6
# Short-Description: Oracle Weblogic NodeManager service.
# Description: Starts and stops Oracle Weblogic NodeManager.
### END INIT INFO
```

- Start this service for run levels 3, 4, and 5.
- The service start priority is 85.
- The service stop priority is 15.

Name of the service

Services this service depends on

Run levels to start this service

Run levels to stop this service

/etc/init.d/nodemgrsvc



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

Configuring the Node Manager as an init.d Service

```
. /etc/rc.d/init.d/functions  
  
export APP="/u01/app"  
export MW_HOME="/u01/app/fmw"  
export JAVA_HOME="$APP/jdk1.7.0_15"  
DAEMON_USER="oracle"  
PROCESS_STRING="^.*\u01/app/fmw/.*weblogic.NodeManager.*"  
  
source $MW_HOME/wlserver/server/bin/setWLSEnv.sh > /dev/null  
export NODEMGR_HOME="/u01/nodemanager"  
NodeManagerLockFile="$NODEMGR_HOME/nodemanager.log.lck"  
  
PROGRAM="$MW_HOME/wlserver/server/bin/startNodeManager.sh"  
SERVICE_NAME=`basename $0`  
LOCKFILE="/var/lock/subsys/$SERVICE_NAME"  
RETVAL=0
```

Source init.d internal functions

Set variables used by the script

/etc/init.d/nodemgrsvc

ORACLE

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

Configuring the Node Manager as an init.d Service

```
start() { <script to start Node Manager> }
stop() {<script to stop Node Manager> }
restart() {<script to restart Node Manager> }

case "$1" in
start)
    start      --> Call above start function
    ;;
stop)
    stop       --> Call above stop function
    ;;
restart|force-reload|reload)
    restart    --> Call above restart function
    ;;
. .
esac

exit $RETVAL
```

/etc/init.d/nodemgrsvc



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

The script in this slide is truncated for the `start`, `stop`, and `restart` functions to conserve space. Practice 5-1 provides a full example of this script, and an explanation of the variables and processing performed by the script.

Registering and Managing init.d Services

- You register a service with `init.d` using the `chkconfig` command as the `root` user.

```
$ chkconfig --add nodemgr
```

Command Line

- The following commands are used to manage the lifecycle of `init.d` services:

```
$ service nodemgr start  
$ service nodemgr stop  
$ service nodemgr restart  
$ service nodemgr status
```

Command Line

ORACLE

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

Configuring the Node Manager as an xinetd Service

- When configuring the Node Manager to run as an xinetd service, consider the following:
 - Ensure that `NodeManagerHome` and other system properties are defined.
 - If `xinetd` is configured with `libwrap`, you should add the `NOLIBWRAP` flag.
 - Ensure that the `hosts.deny` and `hosts.allow` files are configured correctly.
 - Depending on your network environment, additional configuration may be necessary.



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

NOLIBWRAP from `xinetd` man page: This disables internal calling of the `tcp-wrap` library to determine access to the service. This may be needed in order to use `libwrap` functionality not available to long-running processes such as `xinetd`; in this case, the `tcpd` program can be called explicitly (see also the `NAMEINARGS` flag). For RPC services using TCP transport, this flag is automatically turned on, because `xinetd` cannot get remote host address information for the RPC port.

`hosts.deny` and `hosts.allow` files: These provide access control lists of remote hosts that are granted or denied access to daemon processes running on this machine. The search for access stops at the first search in the following order:

- Access is granted when a client and daemon pair matches in `hosts.allow`.
- Access is denied when a client and daemon pair matches in `hosts.deny`.
- Access is granted if there are no matches in either file.

Configuring the Node Manager as an `xinetd` Service

```
# default: off
# description:nodemanager as a service
service nodemgrsvc
{
    type = UNLISTED
    disable = no
    socket_type = stream
    protocol = tcp
    wait = yes
    user = <username>
    port = 5556
    flags = NOLIBWRAP
    log_on_success += DURATION HOST USERID
    server = <path-to-java>/java
    env = CLASSPATH=<cp> LD_LIBRARY_PATH=<ldpath>
    server_args = -client -DNodeManagerHome=<NMHome> <java options>
                  <nodemanager options> weblogic.NodeManager -v
}
```

/etc/xinetd.d/nodemgrsvc



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

This slide shows an example of how the Node Manager can be configured within `xinetd`. The following commands are used to manage the lifecycle of `xinetd` and its services (run as root):

- `service xinetd start`
- `service xinetd stop`
- `service xinetd restart`
- `service xinetd status`

Solaris

Solaris provides an alternative to `init.d`, `inetd`, and `xinetd` service management. In addition to these services, it also provides the Service Management Facility (SMF).

- SMF provides:
 - Consistent interface
 - Dependency ordering
 - Delegation of tasks to non-root users
 - Parallel starting of services
 - Automatic service restart after failure



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

Solaris offers SMF in addition to the other service management utilities. Some say it is a more modern and complete offering. SMF provides the following:

- **Consistent interface:** SMF uses manifest files to manage services, rather than using specialized scripts and configuration files for each service.
- **Dependency order:** SMF allows you to make a service dependent on another service so services required by other services start first.
- **Delegation of tasks:** Allows services to run as users other than root. This protects the system by using the privileges of a user with more restrictions than the root user.
- **Parallel start:** Services are started in parallel, which speeds up boot time by running services simultaneously.
- **Automatic restart:** Works with the Solaris Fault Manager (SFM) to enable software recovery in the event of a failure.

Example SMF Manifest File

```
<?xml version='1.0'?>
<!DOCTYPE service_bundle SYSTEM
  '/>
<service_bundle type='manifest' name='nodemanager'>
<service name='application/management/nodemanager/oracle'
  type='service' version='1'>
  <single_instance/>
  <dependency
    name='multi-user-server' grouping='require_any'
    restart_on='error' type='service'>
    <service_fmri value='svc:/milestone/multi-user-server:default' />
  </dependency>
  <exec_method type='method' name='start'>
    exec='/u01/domains/wlsadmin/bin/startNodeManager.sh'
    timeout_seconds='120' >
    <method_context>
      <method_credential user='oracle' group='oinstall' />
    </method_context>
  </exec_method>
```

Start as oracle user

nodemanager.xml



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

The manifest file specifies the domain's `startNodeManager.sh` script as the script to run when the service is starting. This script is included with WebLogic.

Example SMF Manifest File

```
<exec_method type='method' name='stop'
  exec='/u01/domains/wlsadmin/bin/stopNodeManager.sh'
  timeout_seconds='120' >
<method_context>
  <method_credential user='oracle' group='oinstall' />
</method_context>
</exec_method>
<property_group name='start' type='method'>
  <propval name='action_authorization' type='astring'
    value='solaris.smf.manage.nodemanager/oracle'/>
  <propval name='modify_authorization' type='astring'
    value='solaris.smf.manage.nodemanager/oracle'/>
  <propval name='value_authorization' type='astring'
    value='solaris.smf.manage.nodemanager/oracle'/>
</property_group>
```

Stop as oracle user

Authorize oracle to start

nodemanager.xml

ORACLE

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

The manifest file also specifies the domain's `stopNodeManager.sh` script as the script to run when the service is stopping. This script must be created by you because it is not included with Weblogic by default. All the script should do is kill the Node Manager process.

Example SMF Manifest File

```
<property_group name='stop' type='method'>
    <propval name='action_authorization' type='astring'
        value='solaris.smf.manage.nodemanager/oracle'/>
    <propval name='modify_authorization' type='astring'
        value='solaris.smf.manage.nodemanager/oracle'/>
    <propval name='value_authorization' type='astring'
        value='solaris.smf.manage.nodemanager/oracle'/>
</property_group>
<property_group name='general' type='method'>
    <propval name='action_authorization' type='astring'
        value='solaris.smf.manage.nodemanager/oracle'/>
    <propval name='modify_authorization' type='astring'
        value='solaris.smf.manage.nodemanager/oracle'/>
    <propval name='value_authorization' type='astring'
        value='solaris.smf.manage.nodemanager/oracle'/>
</property_group>
.
.
</service>
</service_bundle>
```

Authorize oracle to stop

Authorize oracle for general commands

nodemanager.xml



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

Set Up and Start Node Manager with SMF

Validate manifest:

```
svccfg validate nodemanager.xml
```

Command Line

Give oracle user authorization (as root user):

```
solaris.smf.manage.nodemanager/oracle:::NodeManager Management::
```

/etc/security/auth_attr

```
usermod -A solaris.smf.manage.nodemanager/oracle oracle
```

Command Line

Import the manifest (as root user):

```
svccfg import nodemanager.xml
```

Command Line

Enable or disable the service as the oracle user:

```
svcadm enable application/management/nodemanager/oracle
```

```
svcadm disable application/management/nodemanager/oracle
```

Command Line

ORACLE

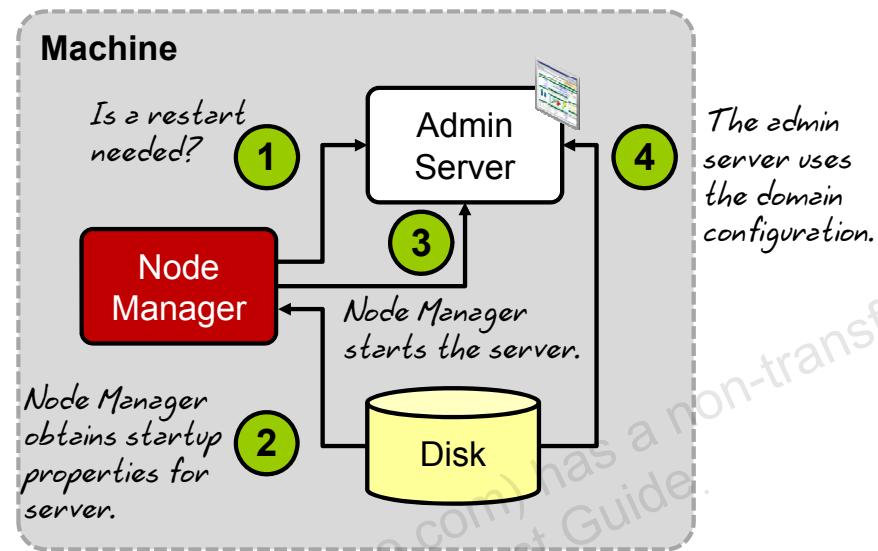
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

After you enable the service, Node Manager will now start automatically when the machine starts.

Agenda

- Node Manager Review
- Configure the Node Manager to Start on System Boot
- Server Automatic Restart
 - Administration Server
 - Managed Server
 - Crash Recovery
 - Configuring Node Manager Restart Parameters

How the Node Manager Restarts an Administration Server

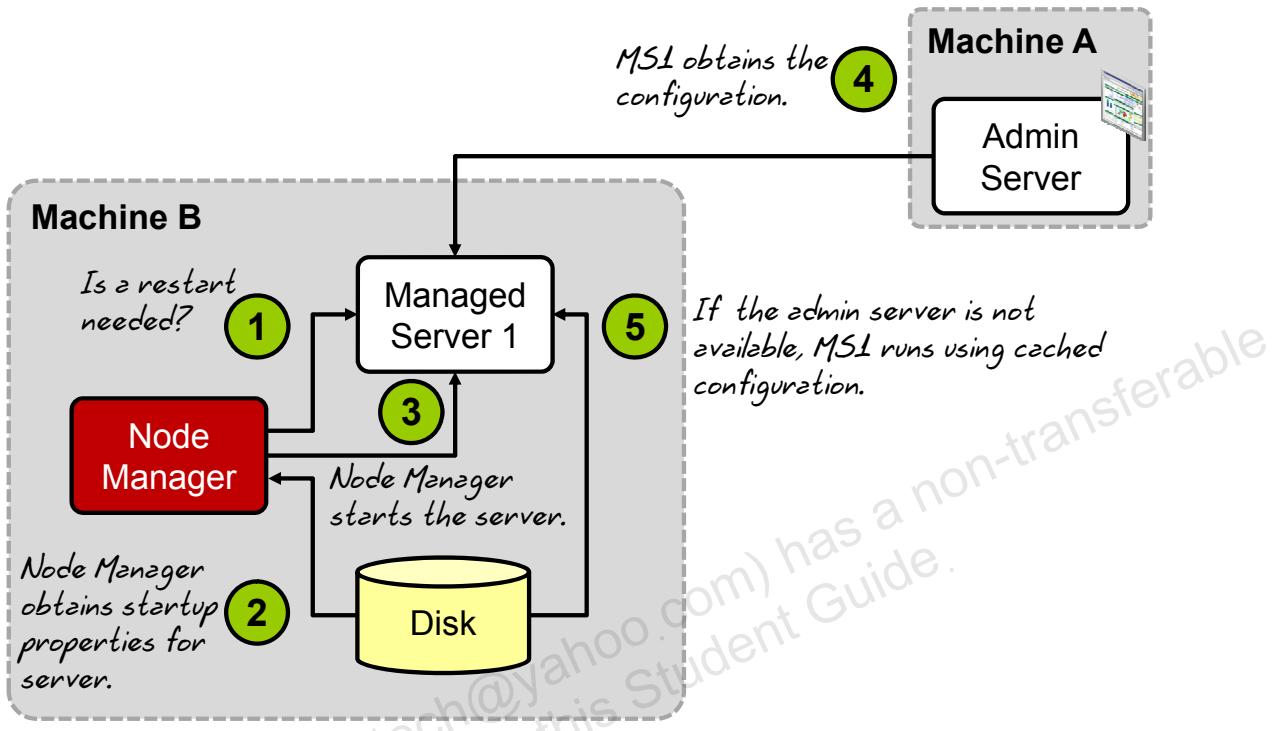


ORACLE

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

1. The Node Manager determines from the administration server process exit code that it requires a restart.
2. The Node Manager obtains the username and password for starting the administration server from the `boot.properties` file, and the server startup properties from the `server/security/startup.properties` file. These server-specific files are located in the `server` directory for the administration server.
3. The Node Manager starts the administration server.
4. The administration server reads its configuration data and starts up.

How the Node Manager Restarts a Managed Server



ORACLE

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

1. The Node Manager determines from Managed Server 1's last known state that it requires a restart.
2. The Node Manager obtains the username and password for starting Managed Server 1 from the `boot.properties` file, and the server startup properties from the `startup.properties` file. These server-specific files are located in the `server` directory for Managed Server 1.
3. The Node Manager starts Managed Server 1.
Note: The Node Manager waits `RestartDelaySeconds` after a server instance fails before attempting to restart it.
4. Managed Server 1 attempts to contact the administration server to check for updates to its configuration data. If it contacts the administration server and obtains updated configuration data, it updates its local cache in the configuration directory.
5. If Managed Server 1 fails to contact the administration server, and if Managed Server Independence (MSI) mode is enabled, Managed Server 1 uses its locally cached configuration data.

RestartInterval and RestartMax

Property	Description	Default Value
RestartMax	The number of times the Node Manager attempts to restart a failed server within the time defined by <code>RestartInterval</code>	2
RestartInterval	The time Node Manager waits before attempting to restart a failed server	3600

Example:

```
...
RestartMax=2
RestartInterval=3600
DOMAIN_HOME/servers/{server-name}/data/nodemanager/startup.properties
```



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

By default, the Node Manager is configured to restart servers immediately upon detecting a failure and can restart servers an unlimited number of times. You can manage how the Node Manager restarts servers by configuring the `RestartInterval` and `RestartMax` properties in the `startup.properties` file. These values are set in the administration console on a per-server basis in the server's Health Monitoring tab.

Crash Recovery

- Scenario:
 - Node Managers are configured to start as system processes.
 - Node Managers were used to start all servers in the domain.
 - All the machines experienced a power outage.

What happens by default when the machines start up again?

- a. Node Managers start and automatically restart all servers in the domain.
- b. Node Managers start and do not automatically restart all servers in the domain.
- c. Node Managers must be started manually.
- d. All servers in the domain must be started manually.



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

This is a trick question for people new to WebLogic. The correct answer is “b” because although Node Managers automatically restart failed servers by default, they do not restart any servers in the event of a complete server crash.

- Answer “a” is incorrect because the Node Managers do not restart servers automatically after a system crash.
- Answer “c” is incorrect because the Node Managers are configured as system processes and start with the system.
- Answer “d” is both correct and incorrect because if the term servers only refers to WebLogic servers, then by default it is correct. If this includes Node Managers, it is incorrect because they start automatically.

You set the CrashRecoveryEnabled property to true to allow the Node Manager to automatically restart all domain servers in the event of a system crash.

CrashRecoveryEnabled

Property	Description	Default Value
CrashRecoveryEnabled	Determines if the Node Manager will restart the servers under its control in the event of a machine failure	false

Example:

```
#Node manager properties  
...  
CrashRecoveryEnabled=true
```

nodemanager.properties

ORACLE

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

This slide shows the CrashRecoveryEnabled property found in the nodemanager.properties file.

Quiz

Which two of the following are reasons to start Node Manager as a system service?

- a. So Node Manager is automatically started when the machine is started
- b. So Node Manager can monitor WebLogic servers
- c. So Node Manager can automatically restart WebLogic servers
- d. So Node Manager can kill failing WebLogic servers



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

Answer: a, c

Summary

In this lesson, you should have learned how to configure the Node Manager to start:

- Failed servers
- On system boot

Practice 5-1 Overview: Configuring Automatic Start and Restart of a System

This practice covers the following topics:

- Setting CrashRecoveryEnabled to true in the nodemanager.properties file
- Using the Node Manager to start a domain with multiple servers
- Crashing and restarting machines to see the Node Manager start on boot and restart all servers

WebLogic Scripting Tool (WLST)

ORACLE

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

Objectives

After completing this lesson, you should be able to:

- Run commands in WLST interactive mode
- Write simple WLST scripts
- Run WLST scripts

Agenda

- WebLogic Scripting Tool (WLST)
- Jython concepts
- WLST concepts
- Java Management eXtension (JMX) concepts
- Common WLST tasks
- Fusion Middleware (FMW) commands

WebLogic Scripting Tool (WLST)

- The WLS command-line tools are useful:
 - For automating common administration activities
 - As an alternative to the Administration Console
 - When graphical tools are not supported
- WLST provides a command-line interface for:
 - Creating new WLS domains
 - Retrieving and updating WLS domain configurations
 - Deploying applications
 - Obtaining runtime server statistics



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

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

Agenda

- WebLogic Scripting Tool (WLST)
- Jython concepts
 - Variable declaration
 - Conditional and loop expressions
 - I/O commands
 - Exception handling
- WLST concepts
- Java Management eXtension (JMX) concepts
- Common WLST tasks
- Fusion Middleware (FMW) commands

ORACLE

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

Jython

Jython is a Java implementation of the popular Python scripting language:

- Simple and clear syntax
- Indentation to structure code (white space critical)
- Interactive command mode
- Custom commands
- Integration with any existing Java libraries

```
list = ['ab', 'cd', 'ef']

if len(list) >= 3:
    for x in list:
        print x, len(x)
print 'done'
```

```
from java.util import ArrayList

list = ArrayList()
list.add('ab')
```

ORACLE

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

The latest Jython release implements the same language as CPython and many of the CPython standard library modules.

Jython is an implementation of the high-level, dynamic, object-oriented language Python, seamlessly integrated with the Java platform. Jython is freely available for both commercial and noncommercial use, and is distributed with source code. Jython is complementary to Java and is especially suited for the following tasks:

- **Embedded scripting:** Java programmers can add Jython libraries to their system to allow end users to write simple or complicated scripts that add functionality to the application.
- **Interactive experimentation:** Jython provides an interactive interpreter that can be used to interact with Java packages or with running Java applications. This allows programmers to experiment and debug any Java system using Jython.
- **Rapid application development:** Python programs are typically two to ten times shorter than the equivalent Java program. This translates directly to increased programmer productivity. The seamless interaction between Python and Java enables developers to freely mix the two languages both during development and in shipping products.

Python (and therefore, Jython) uses leading white space to format structures such as conditions, loops, and functions, instead of braces ("{"}) like in Java.

Using Jython

Jython can interpret commands in two ways for administrative users:

- Interactive:** Supply commands one at a time from a command prompt. Enter a command in the WLST console and view the response immediately.
- Batch:** Provide a series of commands in a script file (.py). You create a text file with the .py extension that contains a series of WLST commands.



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

Interactive mode, in which you enter a command and view the response at a command-line prompt, is useful for learning the tool, prototyping the command syntax, and verifying configuration options before building a script. Using WLST interactively is particularly useful to get immediate feedback after making a critical configuration change.

The WLST scripting shell maintains a persistent connection with an instance of Oracle WebLogic Server. WLST can also write all the commands that you enter during a WLST session to a file. You can edit this file and run it as a WLST script.

Scripts invoke a sequence of WLST commands without requiring your input, much like a shell script. Scripts contain WLST commands in a text file that by convention ends with a .py file extension—for example, myscriptfile.py. Several sample scripts can be found in the WLST documentation online and at <WL_HOME>\samples\server.

Variable Declaration

```
i=0  
  
groups = ['AllUsers', 'Employees', 'Customers', 'HR']  
  
membership = [None, 'AllUsers', 'AllUsers', 'Employees']  
  
for group in groups:  
  
    if membership[i] is not None:  
        print 'Group ' + group + ' is a member of ' + membership[i]  
  
    i += 1
```

A simple declaration

Two static arrays

New variable **group** is defined as part of this **for** statement.

This array member is accessed using an index variable.

Variables used in a **print** statement

Simple increment of a variable by 1

ORACLE

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

Jython provides the ability to do everything with variables that are found in the most common programming languages. This slide shows only a few examples of how they are declared and how they are used.

Conditional Expressions

```
if x and y:  
    #do work  
    #do more work  
  
    If x is true and y is true  
    then do this work.  
  
if x > y:  
    print 'x is greater than y'  
elif x == y:  
    print 'x is equal to y'  
else:  
    print 'x is not greater than or equal to y'  
  
    If x is greater than y,  
    then print this message.  
  
    Otherwise, if x is equal to y  
    then print this message.  
  
    Otherwise, if none of the above is  
    true then print this message.  
  
if x <> y:    #do work  
if x != y:    #do work  
    If x is not equal to y, then do this.  
  
if x is y:    #do work  
    If x is the same object as y, then do this.  
  
result = x > y ? 'greater' : 'not greater'
```

ORACLE

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

This slide shows examples of conditional expressions that determine the flow of a script.

Loop Expressions

```
while x < 10:  
    x += 1  
#out of loop  
  
for i in range(10)  
    print i  
#out of loop  
  
names = ['Tom','Dick','Harry']  
for name in names:  
    print 'Name=' + name  
#out of loop  
  
while 1:  
    x += 1  
    if x % 2 != 0: continue  
    print 'Only print even numbers: ' + x  
    if x == 10: break  
#out of loop
```

Loop until x is equal to 10.

Loop until i is equal to 10 and print the value of i.

Loop through each value in the names array and print each name value.

Loop indefinitely.

Increment the value of x by 1.

Loop control: Start next iteration if x is not an even number.

Loop control: Stop loop when x is equal to 10.

ORACLE

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

This slide shows examples of different types of loops, which are sections of code that repeat until certain conditions are met.

I/O Commands

```
import sys  
  
f = open(sys.argv[1], "r")  
text = f.read()  
text = f.readlines()  
f.close()  
  
f = open(sys.argv[2], "w")  
f.write(text)  
f.writeLines(text)  
print >>f, 'This text is written to the file'  
f.close()
```

Import the Jython sys module for interacting with the operating system.

Open a file for reading.

Reads x bytes (all by default).

Reads all lines of the file into a list.

Closes a file

Open a file for writing.

Another way to write to a file

ORACLE

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

Exception Handling

Jython supports exception handling using a try block:

```
try:  
    connect('weblogic','Welcome1',t3://localhost:7001)  
    print '***Connected successfully***'  
except Exception, e:  
    print 'Domain must be running first'  
    print e  
    exit()
```



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

The commands within the try block are executed. If no exceptions are thrown by any commands, then the except block never executes. However, if an exception is thrown, processing within the try block halts and the code in the except block is executed.

Quiz

What is used to determine a block of code in Jython?

- a. A square bracket []
- b. A semicolon
- c. A whitespace
- d. The "begin ... end" words



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

Answer: c

Agenda

- WebLogic Scripting Tool (WLST)
- Jython concepts
- WLST concepts
 - WLST modes
 - WLST example
 - Running WLST scripts
 - WLST commands and requirements
- Java Management eXtension (JMX) concepts
- Common WLST tasks
- Fusion Middleware (FMW) commands

ORACLE

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

WLST Modes

- Online mode:
 - Connected to a running server
 - Access to all WLS configuration and runtime attributes
 - To create and activate change sessions similar to the WLS console
 - Manage resources, such as starting, stopping, suspending, and migrating
- Offline mode:
 - Domain not running
 - Access to only persisted domain configuration (config.xml)
 - To create or update domains similar to using the Configuration Wizard

ORACLE

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

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

You can use WLST to connect to a running administration server and manage the configuration of an active domain, view performance data about the resources in the domain, or manage security data (such as adding or removing users). You can also use WLST to connect to managed servers, but you cannot modify configuration data from managed servers. WLST online is a JMX client. It interacts with a server's in-memory collection of MBeans, which are Java objects that provide a management interface for an underlying resource.

Without connecting to a running Oracle WebLogic Server instance, you can use WLST to create domain templates, create a new domain based on existing templates, or extend an existing, inactive domain. However, you cannot use WLST offline to view performance data about the resources in a domain or modify security data (such as adding or removing users). WLST offline provides read and write access to the configuration data that is persisted in the domain's config directory or in a domain template JAR that is created using the Domain Template Builder.

WLST Example

```
[oracle@edvmr1p0 /]$ java weblogic.WLST

Initializing WebLogic Scripting Tool (WLST) ...
Welcome to WebLogic Server Administration Scripting Shell
Type help() for help on available commands

wls:/offline> connect('weblogic','password','t3://adminhost:7001')
Connecting to t3://adminhost:7001 with userid system ...
Successfully connected to Admin Server 'myAdmin' that belongs to domain 'mydomain'.

Warning: An insecure protocol was used to connect to the server. To
ensure on-the-wire security, the SSL port or Admin port should be used instead.

wls:/mydomain/serverConfig> cd('Servers')
wls:/mydomain/serverConfig/Servers> ls()
dr-- myAdmin
dr-- myserver1
dr-- myserver2

wls:/mydomain/serverConfig/Servers> cd('myserver1')
wls:/mydomain/serverConfig/Servers/myserver1> get('StartupMode')
'RUNNING'
wls:/mydomain/serverConfig/Servers/myserver1> exit()

Exiting WebLogic Scripting Tool.

[oracle@edvmr1p0 /]$
```

ORACLE

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

Command-Line History and WLST

- WLST interactive mode does not include command line history.
- Use a line reading utility, such as `rlwrap` or `jline` to provide this capability.

A screenshot of a terminal window with a black background and white text. It shows two commands being entered:
\$. setWLSEnv.sh
\$ rlwrap java weblogic.WLST

A yellow callout box with a black border and a black arrow points from the text "Place the rlwrap command just before the command normally used to start WLST." to the word "rlwrap" in the second line of the terminal output.

Place the `rlwrap` command just before the command normally used to start WLST.

Command Line

ORACLE

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

Command-line history is useful for command-line-based tools. Because using WLST in interactive mode is typing intensive, being able to use the arrow keys to recycle previous commands makes working with WLST easier. The `rlwrap` tool requires GNU readline to be installed prior to installation.

Running WLST Scripts

- Use the `setWLSEnv` script to initialize the PATH and CLASSPATH required for WLST.
 - If no script file is supplied, WLST runs in interactive mode.
- Use the `execfile()` command to run additional scripts.

```
$ . ./setWLSEnv.sh
$ java weblogic.WLST [scriptfile.py]

To support SSL connection to a server:
$ java -Dweblogic.security.SSL.ignoreHostnameVerification=true
-Dweblogic.security.TrustKeyStore=DemoTrust weblogic.WLST
```

Command Line

ORACLE

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

Add the Oracle WebLogic Server classes to the CLASSPATH environment variable and WEBLOGIC_HOME/server/bin to the PATH environment variable. You can use the WEBLOGIC_HOME/server/bin/setWLSEnv script to set both variables. (This script is also used indirectly by other domain scripts, such as `setDomainEnv`.)

For convenience, the following additional options are available for launching WLST in interactive mode:

- The `WEBLOGIC_HOME/common/bin/wlst` script
- On Windows, the Tools > WebLogic Scripting Tool from the Start menu

Use the following system properties if you plan to connect WLST to an Oracle WebLogic Server instance through an SSL listen port, and if the server instance is using the WebLogic demonstration SSL keys and certificates:

```
-Dweblogic.security.SSL.ignoreHostnameVerification=true
-Dweblogic.security.TrustKeyStore=keystorename
```

Use this option to run a WLST script, where `filePath.py` is an absolute or relative path name for the script: `[-i] filePath.py`

By default, WLST exits (stops the Java process) after it executes the script. Include `-i` to prevent WLST from exiting. Instead of using this command-line option, you can use the following command after you start WLST: `execfile('filePath.py')`

WLST Development Tools

There are some useful tools to help write WLST scripts:

Tool	Description
configToScript() (WLST)	WLST provides this command which parses an existing domain and generates the WLST script to recreate the domain.
Script recording (WLST)	WLST provides commands to capture interactive commands to a file: <code>startRecording(recordFilePath, [recordAll])</code> <code>stopRecording()</code>
Script recording (administration console)	Records administration console configuration actions and writes the equivalent WLST commands to a file
Oracle Enterprise Pack for Eclipse (OEPE)	Provides an integrated environment for developing, running, and debugging WLST scripts



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

WLST is challenging to new users. This is especially true of administrators who may not be used to coding. Sometimes, it is difficult to tell what is going wrong when you run your scripts.

configToScript()

- The configToScript WLST command:
 - Takes an existing domain as input
 - Generates a WLST script that connects to the domain and re-creates the entire configuration
 - Creates a corresponding property file for commonly changed parameters, such as the domain name, server name, port, and administrative password
 - Creates a separate encrypted file to store any encrypted password attributes
- Use WLST to generate a domain bootstrap script:

```
configToScript('/u01/domains/mydomain',
               'init_mydomain.py')
```

WLST Command Line

ORACLE

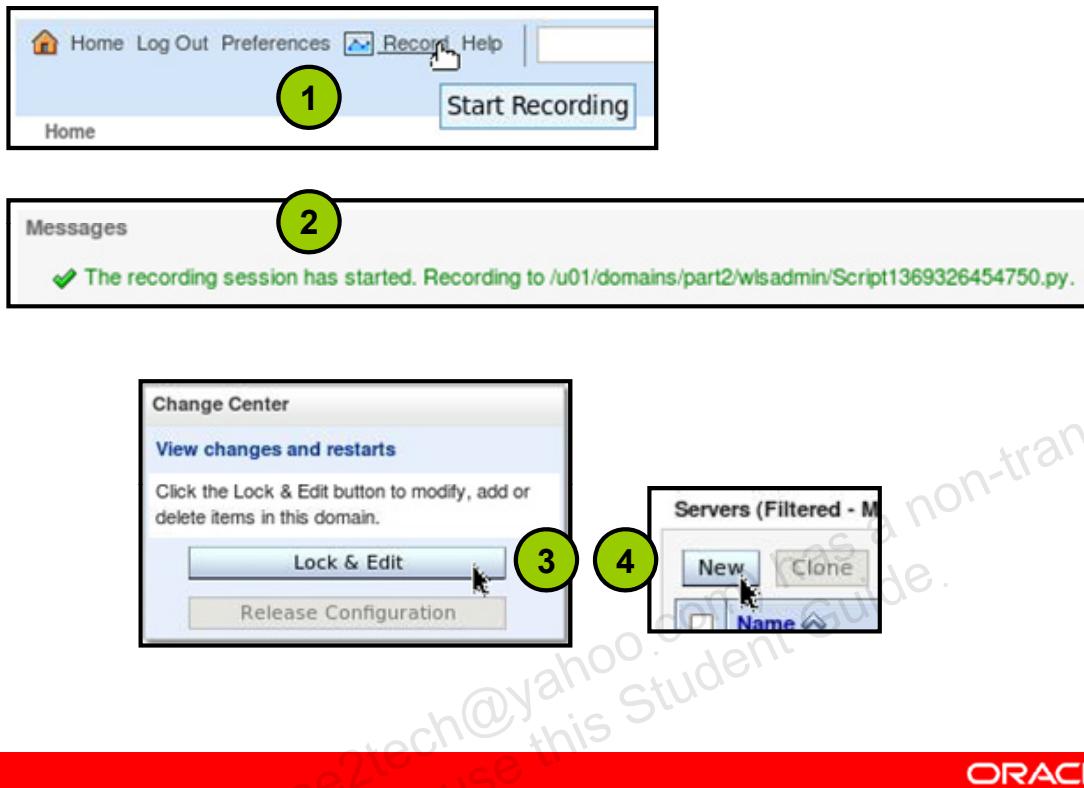
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

This WLST command converts an existing domain configuration (/config directory) into an executable WLST script. You can then use the resulting script to help re-create the domain resources on other machines, or restore a domain to a previously known state.

The command can be run while the domain is running or offline. It generates the following files:

- A WLST script that contains the commands needed to re-create the configuration
- A properties file that contains domain-specific values, including the location of the administration server and the required credentials to access it. You can update the values in this file to update another domain with the same configuration.
- A user configuration file and an associated key file to store encrypted attributes, such as passwords. The user configuration file contains the encrypted information. The key file contains a secret key that is used to encrypt and decrypt the encrypted information.

Script Recording Using the Administration Console



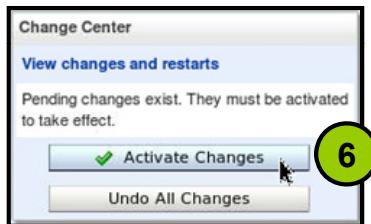
ORACLE

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

Perform the following steps to record administration console activities in a WLST script:

1. Click the Record button to turn on recording.
2. You should see a message that recording is started and where the file containing the script is located.
3. Click Lock & Edit to start an edit session.
Note: You can't click the Record button unless an edit session has already been started. However, if recording was started previously, then it stays active between edit sessions.
4. Configure something using the console. In this example, we create a new server.

Script Recording Using the Administration Console



```
startEdit()  
cd ('/')  
cmo.createServer('server3')  
  
cd ('/Servers/server3')  
cmo.setListenAddress('host01')  
cmo.setListenPort(7001)  
  
activate()
```

The screenshot shows a portion of a WLST script. The code is enclosed in a light gray box. A green circle containing the number '7' is positioned above the first line of code. The script performs several actions: it starts an edit session, creates a new server named 'server3' at the root level, moves into the '/Servers/server3' directory, sets the listen address to 'host01', sets the listen port to 7001, and finally activates the changes.

Script1369326454750.py

ORACLE

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

5. The server's properties are configured, server name, listen address, and listen port. Click Finish to create the server.
6. Click Activate Changes to realize the configuration change.
7. View the script recording file to find the WLST script used to perform the task.

Oracle Enterprise Pack for Eclipse

```
14
15 #Conditionally import wlstModule only when script is executed
16 if __name__ == '__main__':
17     from wlstModule import *#@UnusedWildImport
18
19 #----- Connect to server -----
20@def connectWLS(host):
21     #WLS Must be running for this script to work
22     try:
23         connect('weblogic','Welcome1',host)
24         print "***Connected successfully***"
25     except:
26         print 'Domain ' + host + ' must be running first'
27         exit()
28
29 #----- Create Roles -----
30@def createPolicy():
31     PyDev breakpoint()
32
33     #cmo.getSecurityConfiguration().getDefaultRealm().lookup()
```

Manage importing
WLST modules when
running in Jython.

Set breakpoints to stop
script execution so you
can see inside the script.

ORACLE

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

The Oracle Enterprise Pack for Eclipse provides a WLST scripting facet that leverages the PyDev extension that provides support for developing, executing, and debugging WLST scripts. Using the IDE, you gain the benefits of:

- Syntax and color highlighting and automatic syntax checking
- Auto-completion of methods
- An execution environment
- Real-time debugging with breakpoints and all the features you would expect from a debugger
- An MBean explorer that lets you see the paths to the MBeans that you need to configure
- A built-in interactive mode that includes command history and auto-completion, as well as automatic command line set up with only entering a partial command
- WLST integrated help
- WLST templates

WLST Command Tips

- Use case-sensitive names and arguments of commands.
- Use string literal arguments enclosed within single or double quotation marks.
- Precede the quoted string with **r** while specifying a backslash (\) in the string.
 - Example: `readTemplate (r'c:\mytemplate.jar')`
- Do not use the following invalid characters in object names:
 - Period (.)
 - Slash (/)
 - Backslash (\)
- Use the display help:
 - Example:
`wls:/mydomain/serverConfig> help('disconnect')`

ORACLE

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

Note: If you need to change directory (`cd`) to an object name that includes a slash (/) in its name, include the configuration object name in parentheses. For example:
`cd ('JMSQueue/ (jms/REGISTRATION_MDB_QUEUE) ')`

You cannot access security information through WLST while updating a domain. When you use WLST and a domain template, you can create and access security information only when you create a new domain.

General WLST Commands

Many of these commands take additional parameters.

Command	Description
<code>help()</code>	Get help for a given WLST command.
<code>exit()</code>	Quit WLST.
<code>dumpVariables()</code>	Display all variables used by WLST.
<code>dumpStack()</code>	Display the stack trace for the last error that occurred in WLST.
<code>redirect() / stopRedirect()</code>	Redirect all WLST output to a file.
<code>cd()</code>	Navigate from one MBean in the hierarchy to another MBean.



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

See the WLS documentation for a complete list of available WLST commands, with examples for each.

To display information about the WLST commands and variables, enter the `help` command. If you specify the `help` command without arguments, WLST summarizes the command categories. To display information about a particular command, variable, or command category, specify its name as an argument to the `help` command. To list a summary of all online or offline commands from the command line, use the following commands, respectively:

```
help('online')
help('offline')
```

The `help` command supports a query. For example, `help('get*')` displays the syntax and usage information for all commands that begin with `get`.

To redirect WLST information, error, and debug messages from standard out to a file, enter:

```
redirect(outputFile, [toStdOut])
stopRedirect()
```

This command also redirects the output of the `dumpStack()` and `dumpVariables()` commands.

Offline WLST Commands

Command	Description
<code>createDomain()</code>	Create a domain by using a given template.
<code>readDomain()</code>	Open an existing domain on the file system.
<code>readTemplate()</code>	Open an existing domain template.
<code>addTemplate()</code>	Apply a template file to the current domain.
<code>updateDomain()</code>	Save changes to the current domain.
<code>writeDomain()</code>	Save changes to the current domain to a specified directory.
<code>writeTemplate()</code>	Save the current domain to a template file.
<code>assign() / unassign()</code>	Target applications or services to servers.
<code>setOption()</code>	Configure domain creation options (domain name, Java home, start mode, and so on).

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

WLST offline provides read and write access to the configuration data that is persisted in the domain's config directory or in a domain template JAR that is created using the Domain Template Builder. This data is a collection of XML documents and expresses a hierarchy of management objects. The offline commands include:

- `createDomain(domainTemplate, domainDir, user, password)`
- `readDomain(domainDirName)`
- `readTemplate(templateFileName)`
- `addTemplate(templateFileName)`
- `writeTemplate(templateName)`
- `assign(sourceType, sourceName, destinationType, destinationName)`
- `setOption(optionName, value)`

WLST also includes the `configToScript` command that reads an existing domain and outputs a WLST script that can re-create the domain. Set the password for the default user, if it is not already set. The default username and password must be set before you can write the domain template. For example:

```
cd('/Security/domainname/User/username')
cmo.setPassword('password')
```

Online WLST Commands

Command	Description
<code>connect()</code>	Connect to a server by using supplied credentials.
<code>disconnect()</code>	Disconnect from the current server.
<code>shutdown()</code>	Shut down servers.
<code>start()</code>	Use the Node Manager to start servers.
<code>startEdit()</code>	Begin a new change session.
<code>stopEdit()</code>	Release the edit lock and discard any changes.
<code>activate()</code>	Commit all changes in the current session.
<code>showChanges()</code>	List all changes made in the current session.
<code>isRestartRequired()</code>	Determine if any changes require a server restart.
<code>deploy() / redeploy()</code>	Deploy an application to servers.
<code>undeploy()</code>	Shut down a running application on servers.



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

Learning WLST initially is sometimes overwhelming for somebody that is new to WLST and Jython. The administration console provides a script recording tool that works when the server is running. This allows an administrator to perform operations within the console and have the corresponding WLST commands written to a text file in the domain's root folder. This makes it easy to create new WLST scripts. The open source project, wlnav, is another way to work with WebLogic MBeans and learn WLST.

Quiz

Using WLST's _____ mode, you can supply commands one at a time and get immediate feedback.

- a. Management
- b. Operational
- c. Sequential
- d. Template
- e. Interactive



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

Answer: e

In Interactive mode, you can quickly prototype or troubleshoot some WLST commands.

Agenda

- WebLogic Scripting Tool (WLST)
- Jython concepts
- WLST concepts
- Java Management eXtension (JMX) concepts
 - JMX overview
 - Configuration and runtime mbeans
 - WebLogic Server MBean examples
 - Browsing MBean documentation
 - Referencing MBeans in WLST
- Common WLST tasks
- Fusion Middleware (FMW) commands

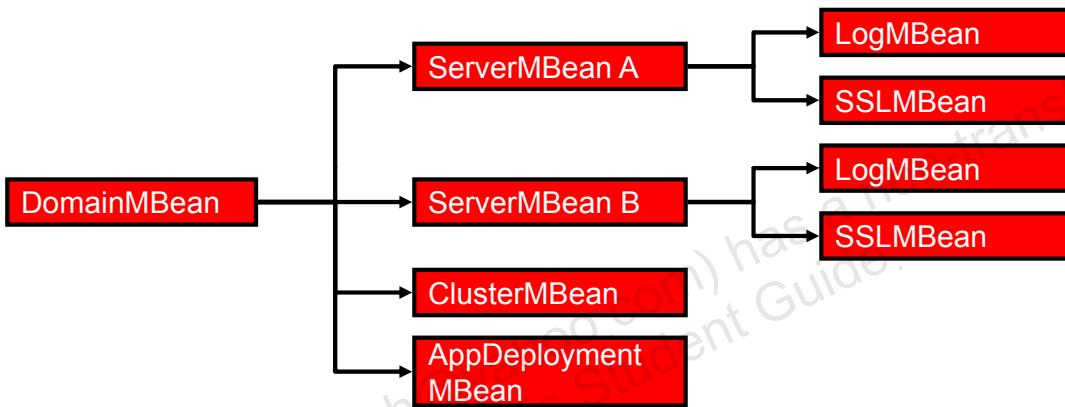
ORACLE

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

WebLogic JMX Overview

JMX MBeans:

- Are hierarchical Java objects found on the server
- Have attributes and operations
- Support the configuration, management, and monitoring of all types of server resources



ORACLE

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

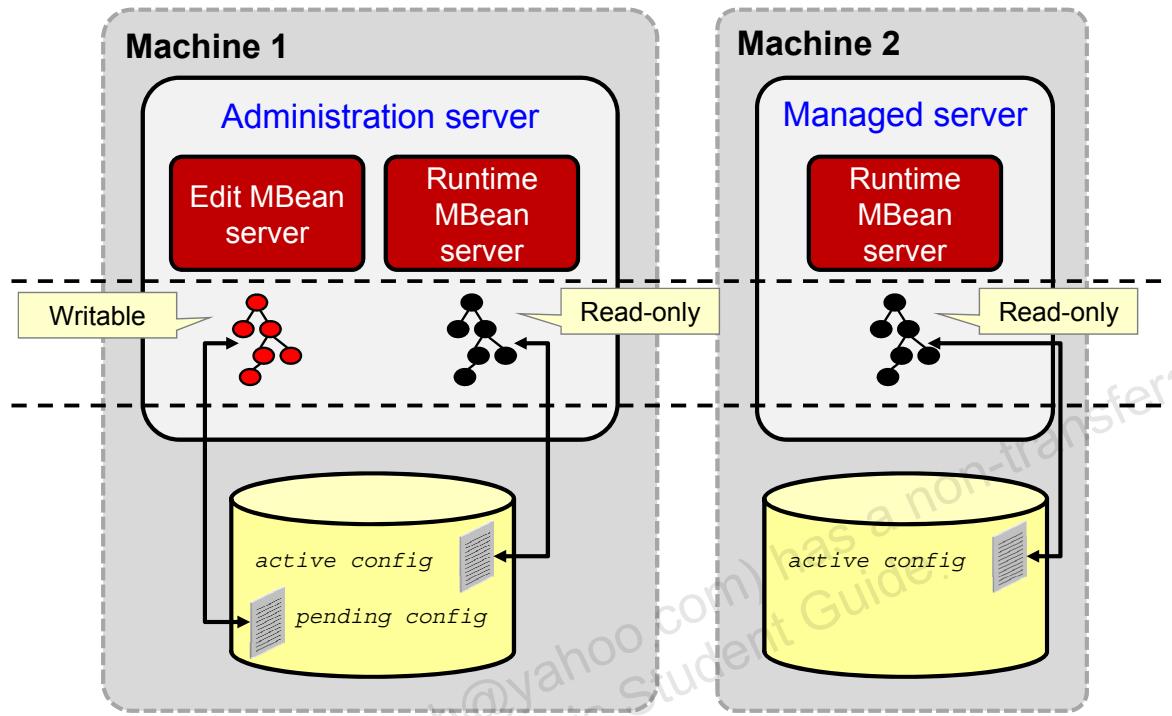
This diagram shows the hierarchical structure of MBeans.

The JMX specification does not impose a model for organizing MBeans. However, because the configuration of an Oracle WebLogic Server domain is specified in an XML document, Oracle WebLogic Server organizes its MBeans into a hierarchical model that reflects the XML document structure. For example, the root of a domain's configuration document is `<domain>` and below the root are child elements such as `<server>` and `<cluster>`. Each domain maintains a single MBean of the `DomainMBean` type to represent the `<domain>` root element. Within `DomainMBean`, the JMX attributes provide access to the MBeans that represent child elements such as `<server>` and `<cluster>`.

All Oracle WebLogic Server MBeans can be organized into one of the following general types:

- Runtime MBeans contain information about the runtime state of a server and its resources. They generally contain data only 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 lost.
- Configuration MBeans contain information about the configuration of servers and resources. They represent the information that is stored in the domain's XML configuration documents.

Configuration MBeans



ORACLE

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

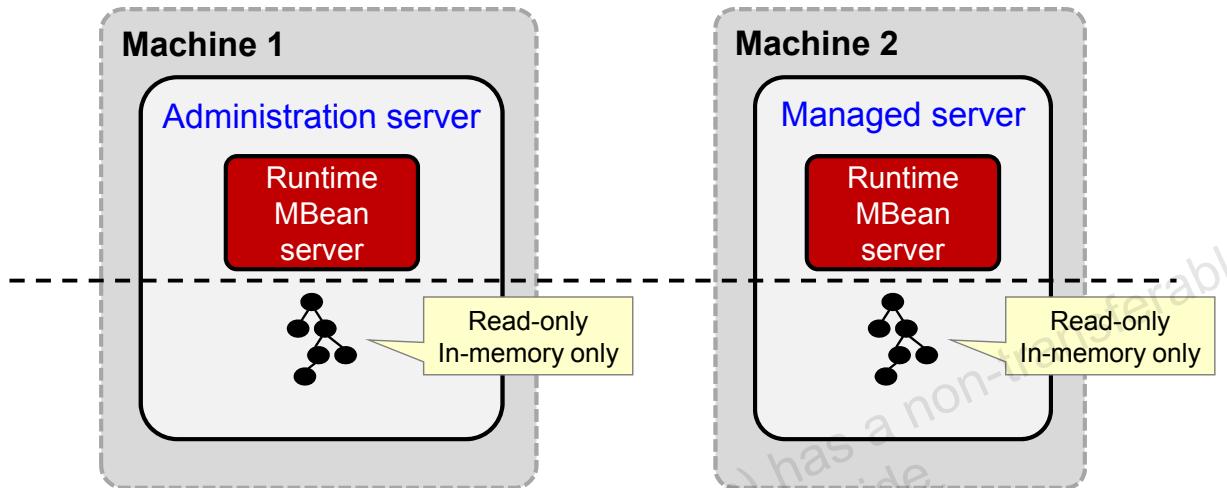
This graphic depicts MBeans reading and writing to the `config.xml` files. Each domain describes its configuration in an XML document that is located in the domain's configuration directory. At run time, each Oracle WebLogic Server instance in a given domain creates an in-memory representation of the configuration described in this document. The in-memory representation of a domain's configuration is a collection of read-only managed beans (MBeans) called Configuration MBeans.

In addition to the read-only Configuration MBeans, the administration server maintains another collection of Configuration MBeans that you can edit. To edit these Configuration MBeans, you use a JMX client (either the Administration Console, WLST, or a client that you create) to obtain a lock.

While you have the lock on the editable Configuration MBeans, you can save your in-memory changes, which causes the administration server to write the changes to a set of pending configuration documents in the domain directory. The Oracle WebLogic Server instances do not consume (commit) the changes until you activate them.

When you activate the changes, each server in the domain determines whether it can accept the change. If all the servers can accept the change, they update their copy of the domain's configuration document. Then they update their working copy of the Configuration MBeans and the change is completed.

Runtime MBeans



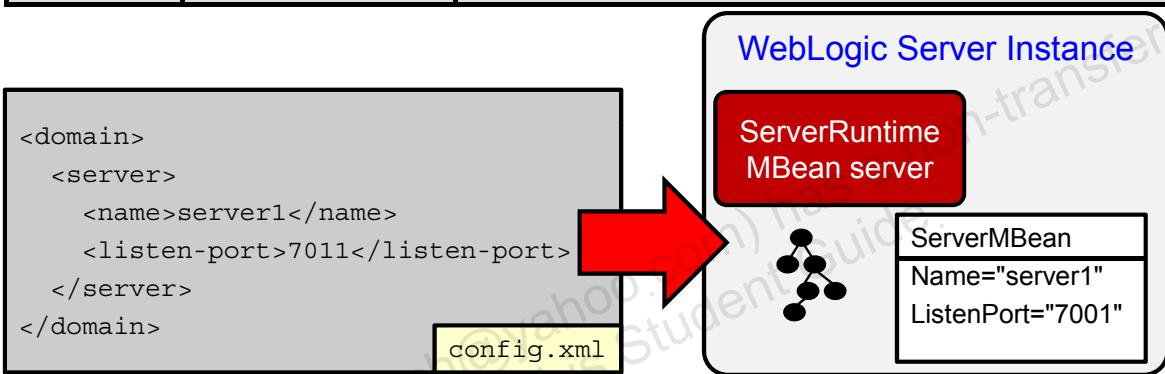
ORACLE

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

Just as Configuration MBeans represent the configuration of a WebLogic domain, Runtime MBeans represent the running state of all the running components in a WebLogic domain. Unlike Configuration MBeans, Runtime MBeans are not persisted to disk and are only kept in memory for monitoring purposes. If a server shuts down, then any Runtime MBean data that resided on the server is no longer available. Runtime MBeans are used to capture and monitor the historic and current state of the running server. This information is used to tune and troubleshoot your running application.

WebLogic Server MBean Examples

	Properties	Description
Name	<code>ServerMBean</code>	Represents the configuration of a WebLogic Server
Attributes	ListenAddress	Represents the listening host address of this server
	ListenPort	Represents the listening port of this server
Operations	<code>isSet (property)</code>	Returns <code>true</code> if the specified attribute is set



ORACLE

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

This slide shows how configuration properties in the `config.xml` file are realized as MBean properties within a running server.

MBean Properties in the Administration Console

The screenshot shows the 'Settings for server1' page in the WebLogic Administration Console. The 'Listen Port' section is highlighted with a red box around the input field. A callout box points to the 'More Info...' link with a green circle labeled '1'. Another callout box points to the MBean Attribute name 'ServerMBean.ListenPort' with a green circle labeled '2'.

Settings for server1

Listen Port The default TCP port that this server uses to listen for regular (non-SSL) incoming connections.

Administrators must have the right privileges before binding to a port or else this operation will not be successful and it will render the console un-reachable.

If this port is disabled, the SSL port must be enabled. Additional ports can be configured using network channels. The cluster (multicast) port is configured separately.

MBean Attribute: ServerMBean.ListenPort 2

Minimum value: 1

Maximum value: 65534

Name: server

Cluster: cluster1

Listen Address: 192.0.2.11

Listen Port Enabled

Listen Port: 7011 The default TCP port that this server uses to listen for regular (non-SSL) incoming connections. [More Info...](#) 1

ORACLE

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

This slide shows the same settings in the WebLogic administration console and how to find the corresponding MBean attribute using the administration console help:

1. Next to the property field in the administration console, there is a short description of the field. There is a *More Info...* link that opens the administration console help to a more detailed description of that field.
2. This is the administration console help page for Listen Port. It shows the associated MBean attribute name, `ServerMBean.ListenPort`, which is used to reference this field from within WLST.

Browsing MBean Documentation

The screenshot shows the Oracle Fusion Middleware Documentation page for Oracle WebLogic Server. At the top, the Oracle logo is visible. Below it, the title "Oracle Fusion Middleware Documentation" and the version "12c (12.1.2)" are displayed. A navigation bar below the title includes links for "Home", "12c Release 1 (12.1.2)", "Oracle WebLogic Server", and several tabs labeled "View by: Get Started", "Develop and Deploy", "Administer", "Secure", "Monitor and Tune", and "Reference". The "Reference" tab is highlighted with a mouse cursor. An arrow points from this cursor to a box containing a list of links under the heading "Programming APIs and MBeans".

Programming APIs and MBeans

- Java Platform EE 6 API specification
- Java SE 6 API specification
- Java API Reference for WebLogic Server
- JMS C API Reference for WebLogic Server
- WebLogic Messaging .NET API Reference for WebLogic Server
- **MBean Reference for WebLogic Server**
- WebLogic Web Services Reference for

ORACLE

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

Browsing MBean Documentation

The screenshot shows the 'WEBLOGIC SERVER MBEAN REFERENCE' interface. On the left, a tree view lists various MBeans under 'The WebLogic Server MBean Reference'. A yellow callout points to the 'BasicRealmMBean' node in the tree, with the text 'Organized list of all WebLogic MBeans'. On the right, the 'BasicRealmMBean' page is displayed. At the top, there are tabs for 'Overview', 'Related MBeans', 'Attributes', and 'Operations'. A yellow callout points to the 'Overview' tab with the text 'Select an MBean from the left panel and see its details in this panel.' Below the tabs, the 'Overview' section contains a detailed description of the MBean. Further down, there are tables for 'Fully Qualified Interface Name', 'Factory Methods', and 'Subtypes'. The 'Attributes' section lists two attributes: 'CachingDisabled' and 'ObjectName'. The Oracle logo is at the bottom right of the page.

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

ORACLE

Referencing MBeans in WLST

Use the `getMBean()` API to assign an MBean to a variable and reuse that variable to manage the MBean.

```
#Create channel on server1
server=getMBean('/Servers/server1')
server.createNetworkAccessPoint('RepChannel')
channel=
getMBean('/Servers/server1/NetworkAccessPoints/RepChannel')
channel.setProtocol('t3')
channel.setListenAddress('host01')
channel.setListenPort(5000)
channel.setEnabled(true)
```

CreateReplication.py

ORACLE

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

Quiz

WLST communicates with Oracle WebLogic Server's _____ to retrieve and update resources on a running server.

- a. Templates
- b. Logs
- c. MBeans
- d. Scripts

ORACLE

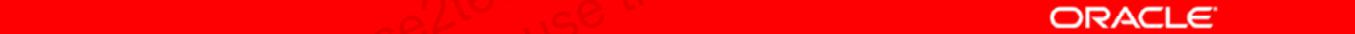
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: c

WLST is based on JMX, which supports remote server management through MBean objects.

Agenda

- WebLogic Scripting Tool (WLST)
- Jython concepts
- WLST concepts
- Java Management eXtension (JMX) concepts
- Common WLST tasks
 - Creating and modifying templates and domains
 - Connecting to, configuring, and monitoring a server
 - Adding a server to a cluster
 - Creating and monitoring a data source
 - Creating an LDAP authentication provider
 - Deploying an application
- Fusion Middleware (FMW) commands

 ORACLE

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

Creating a Template and a Domain

```
readTemplate('mybasetemplate.jar')           Reads in a template jar file

setOption('DomainName','mydomain')
setOption('JavaHome','/home/myjdk')
setOption('ServerStartMode','prod')          Sets some options

writeDomain('/home/mydomains')               Writes a domain based on the template
closeTemplate()                            and the changes and closes the template

readDomain('/home/mydomains/mydomain')
addTemplate('myjms.jar')
addTemplate('myapps.jar')
updateDomain()
closeDomain()
exit()
```

Reads in the domain, extends it with some templates, and updates the domain



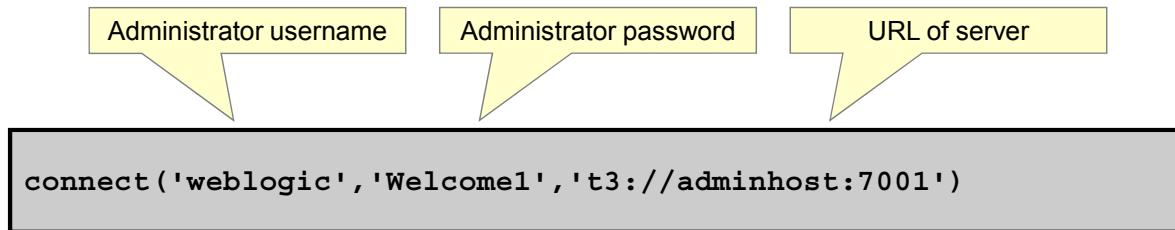
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

See the WLS documentation for the `setOption` command to view a list of available domain creation options.

The following example writes a domain configuration to a domain template:

```
readDomain('/home/mydomains/mydomain')
writeTemplate('myTemplate.jar')
```

Connecting to a Server



ORACLE

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

The connect WLST command is used to connect to a running WebLogic Server.

Password Management

To avoid using plain text credentials in your scripts, do one of the following:

- Require them as command-line arguments.
- Prompt the user to enter them.
- Create and use an encrypted password file.

Create encrypted password file for previously used credentials:

```
connect(username, password, myurl)  
storeUserConfig()
```

Stored in OS user's home directory by default

Connect to a server using the current password file:

```
connect(url=myurl)  
...
```

File retrieved from the home directory by default

ORACLE

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

The `storeUserConfig()` command creates a user configuration file and an associated key file for the identity of the current user you are connected to WLS with. The user configuration file contains an encrypted username and password. The key file contains a secret key that is used to encrypt and decrypt the username and password. Only the key file that originally encrypted the username and password can be used to decrypt the values. If you lose the key file, you must create a new user configuration and key file pair. If you do not specify file names or locations, the command stores the files in your home directory as determined by your Java Virtual Machine (JVM). The location of the home directory depends on the SDK and type of operating system on which WLST is running. The default file names are `<username>-WebLogicConfig.properties` and `<username>-WebLogicKey.properties`.

If you do not specify credentials as part of the `connect` command, and a user configuration and a default key file exist in your home directory, then the command uses the credentials found in those files. This option is recommended if you use WLST in script mode, because it prevents you from storing unencrypted user credentials in your scripts. Alternatively, you can provide the specific locations and names of these files by using the `userConfigFile` and `userKeyFile` command arguments.

WLST Variables

Some common WLST variables:

Variable	Description
cmo	Current Management Object. This variable is automatically set to the configuration MBean instance to which you navigate. <code>cmo.create('myServer', 'Server')</code>
domainName	The name of the domain to which WLST is connected.
domainRuntimeService	Represents the DomainRuntimeServiceMBean, which is only available when connected to the administration server. <code>domainRuntimeService.getServerRuntimes()</code>
runtimeService	Represents the RuntimeServiceMBean. <code>runtimeService.getServerRuntime()</code>
editService	Represents the EditServiceMBean, which is only available when connected to the administration server. <code>editService.getDomainConfiguration()</code>
exitonerror	A Boolean value that indicates whether WLST terminates script execution when it encounters an exception.



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

Password Management

Password change using WLST:

```
connect('weblogic','Welcome1','t3://host01.example.com:7001')

atnr=cmo.getSecurityConfiguration().getDefaultRealm().
    lookupAuthenticationProvider("DefaultAuthenticator")

atnr.changeUserPassword('weblogic','Welcome1','Welcome2')

disconnect()

connect('weblogic','Welcome2','t3://host01.example.com:7001')

exit()
```

The `cmo` built-in WLST variable is used to obtain a reference to the Default Authentication Provider.

The `changeUserPassword` method is invoked to change the password for the `weblogic` user.

A connection is attempted, using the password that was just reset.

ORACLE

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

CMO is the WLST built-in variable. CMO stands for Current Management Object. While programming in WLST you can use `cmo` to point to the current MBean (object) instance you are navigating into.

The example shows how to obtain the Authentication Provider Object by referencing CMO. With a handle to the Default Authentication Provider it is possible to call the `changeUserPassword()` function, which requires username, current password and new password as parameters, to change the password for the `weblogic` user.

Configuring a Server

```
newServerName = 'server1'           Variable declarations
newServerPort = 7011
machineName = 'machine1'

edit()
startEdit()                         Start a configuration edit session.

server = create(newServerName, 'Server')
server.setListenPort(newServerPort)
server.setMachine(getMBean('/Machines/' + machineName))

save()                                Save and activate all changes.
activate(block='true')
print 'Server created successfully.'
exit()
```

Create a new server
and then set its port
and machine.

ORACLE

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

This script shows the basics on how to create and configure a WebLogic server.

Monitoring a Server

```
serverRuntime()
threadPool = getMBean('/ThreadPoolRuntime/ThreadPoolRuntime')
webModule = getMBean('/ApplicationRuntimes/' + appName +
                     '/ComponentRuntimes/' + serverName +
                     '/' + appWebRoot)

while 1:                                     Loop indefinitely.
    throughput = threadPool.getThroughput()      Gather runtime information for
    appStatus = webModule.getStatus()            the server's current statistics.
    appCurrSessions = webModule.getOpenSessionsCurrentCount()

    print '%.1f\t' % (throughput) + appStatus + '\t' +
          str(appCurrSessions)                  Print statistics.

    java.lang.Thread.sleep(1000)                 Sleep for one second and then loop again.
```

ORACLE

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

This script shows an example of monitoring some server statistics and displaying them to standard out.

Adding a Server to a Cluster

```
edit()  
startEdit()           Start a configuration edit session.  
  
cluster = create('cluster1', 'Cluster')  
cluster.setClusterMessagingMode('unicast')    Create a new cluster and  
                                                set its broadcast protocol.  
  
server = getMBean('/Servers/server1')  
server.setCluster(cluster)  
server = getMBean('/Servers/server2')  
server.setCluster(cluster)          Add two configured  
                                                servers to the cluster.  
  
save()  
activate(block='true')           Save and activate all changes.  
print 'Cluster created successfully.'  
exit()
```



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

This script shows how to create a cluster and add two existing servers to it.

Creating a Data Source

```
edit()
startEdit()           Start a configuration edit session.

targetServer = getMBean('/Servers/server1')           Get MBean object
                                                       for server1.

jdbcSysRes = create('MyXADatasource', 'JDBCSystenResource')
jdbcResource = jdbcSysRes.getJDBCResource()
jdbcResource.setName('MyXADatasource')               Create data source
                                                       and set its name.

jdbcResParams = jdbcResource.getJDBCDataSourceParams()
jdbcResParams.setJNDINames(['jdbc/MyXADatasource']) Set the JNDI name and data source type.

jdbcResParams.setGlobalTransactionsProtocol('TwoPhaseCommit')

connectionPool = jdbcResource.getJDBCConnectionPoolParams()
connectionPool.setInitialCapacity(5)
connectionPool.setMaxCapacity(10)
connectionPool.setMinCapacity(1)                      Set connection pool
                                                       properties.
```

ORACLE

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

This script shows how to create a data source, set some of the most important properties, and configure its associated connection pool.

Creating a Data Source

```
dr = jdbcResource.getJDBCDriverParams()
dr.setDriverName('oracle.jdbc.xa.client.OracleXADataSource')
dr.setUrl('jdbc:oracle:thin:@localhost:1521:orcl')
dr.setPassword('Welcome1')
driverProperties = dr.getProperties()
userProperty = driverProperties.createProperty('user')
userProperty.setValue('wlsdata')

jdbcSystemResource.addTarget(targetServer)           Configure driver
                                                    properties.

save()                                              Target the data source to a server.

activate(block='true')
print 'Data Source created successfully.'
exit()                                              Save and activate all changes.
```

This script continues to show how to configure the JDBC driver for the data source, and target it for a server.

Monitoring a Data Source

```
serverRuntime()          Navigate to the server runtime MBean tree.  
ds = getMBean('/JDBCServiceRuntime/server1' +  
             '/JDBCDataSourceRuntimeMBeans/MyXADataSource')  
  
print 'CAP' + '\t' + 'TOTAL' + '\t' + 'ACTIVE' + '\t' +  
      'AVAIL' + '\t' + 'LEAK'  
  
while 1:                Loop indefinitely.  
    cap = ds.getCurCapacity()  
    total = ds.getConnectionsTotalCount()  
    active = ds.getActiveConnectionsCurrentCount()  
    avail = ds.getNumAvailable()  
    leak = ds.getLeakedConnectionCount()  
    print str(cap) + '\t' + str(total) + '\t' + str(active) +  
          '\t' + str(avail) + '\t' + str(leak)  
    java.lang.Thread.sleep(2000)  Sleep for two seconds and then loop again.
```

Gather runtime information for the server's current statistics.

Print statistics.

ORACLE

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

This script shows one way to monitor a data source within a loop.

Creating an LDAP Authentication Provider

```
realm = cmo.getSecurityConfiguration().getDefaultRealm()

provider = realm.createAuthenticationProvider('MyLDAPProvider',
'weblogic.security.providers.authentication.LDAPAuthenticator')

provider.setControlFlag('SUFFICIENT')
provider.setHost('localhost')
provider.setPort('389')
provider.setPrincipal('cn=Directory Manager')
provider.setCredential('Welcome1')
provider.setUserBaseDN('dc=example,dc=com')
provider.setUserNameAttribute('uid')
provider.setGroupBaseDN('dc=example,dc=com')
provider.setStaticGroupNameAttribute('cn')
```

Get the default security realm MBean.

Create an LDAP authentication provider named MyLDAPProvider.

Sets some basic settings required for the provider

ORACLE

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

This script shows how to create and configure an LDAP authentication provider. Note that the change management commands are not included in this script, but are still required for starting an edit session and activating your changes.

Modifying a Domain Offline

```
readDomain('/u01/app/domains/wlsadmin')  
  
currentMachine = create('machine1','UnixMachine')  
cd('/Machines/machine1')  
nodeManager = cmo.create('machine1','NodeManager')  
nodeManager.setListenAddress('myHostName')  
nodeManager.setListenPort(5556)  
  
for currentIndex in range(1, 5):  
    currentServerName = 'server' + str(currentIndex)  
    cd('/Servers/' + currentServerName)  
    cmo.setMachine(currentMachine)  
  
updateDomain()  
print ''  
print 'Domain updated successfully.'
```

Read in the original domain.

Create a machine and set some properties.

Loop through server1 through server5 and set each on the new machine.

Write out the domain with a new configuration.



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

This script shows you how to read in an existing domain in offline mode, modify it, and update the original domain by writing it back to disk.

Deploying an Application

```
Deploy application named MyApp  
deploy appName='MyApp',  
        path='/apps/MyApp.ear',  
        targets='server1')  
Found in this location  
To this server
```

This script shows an example of deploying an Enterprise Application to a WebLogic Server instance.

Quiz

The WSLT built-in variable which identifies the current JMX MBean is called:

- a. cto (Current Transaction Object)
- b. cdo (Current Data Object)
- c. cmo (Current Management Object)
- d. cwo (Current WebLogic Object)

ORACLE

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

Answer: c

Agenda

- WebLogic Scripting Tool (WLST)
- Jython concepts
- WLST concepts
- Java Management eXtension (JMX) concepts
- Common WLST tasks
- Fusion Middleware (FMW) commands

Some FMW Commands

FMW Product	Command	Description
SOA Suite	<code>sca_deployComposite()</code>	Deploy a SOA composite application.
WebCenter	<code>listJCRPortalConnections()</code>	List all Oracle Portal connections configured for WebCenter applications.
ADF	<code>adf_createHttpURLConnection()</code>	Create a new ADF URL connection.
JRF	<code>applyJRF()</code>	Configure a managed server or cluster with Java Required Files applications and services.
OAM	<code>createOAMIdentityAsserter()</code>	Create a new identity asserter.



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

Some FMW products have custom WLST scripts that are installed when the respective product is installed. This slide shows some examples of these commands.

Note: There are far too many custom FMW scripts to mention in this course. A reference of the commands is found in the *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference 11g Release 1*. FMW is not released as version 12c yet so all commands are currently listed in the 11g guide as of the time of this writing. Remember that these commands are subject to change when FMW 12c is released. FMW scripts require running within an environment that is specialized for that FMW product area. Each FMW product has its own version of `wlst.sh` that sets the environment properly to work with that product.

Summary

In this lesson, you should have learned how to:

- Run commands in WLST interactive mode
- Write simple WLST scripts
- Run WLST scripts

Practice 6-1 Overview: Creating and Modifying a Domain with WLST

This practice covers the following topics:

- Creating a new WebLogic domain using WLST
- Starting a domain using WLST
- Connecting to the Administration Server with WLST
- Modifying the domain configuration using WLST

Practice 6-2 Overview: Monitoring a Domain with WLST

This practice covers the following topics:

- Connecting to a domain with WLST
- Monitoring different subsystems with WLST

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.

Working with the Security Realm

13

ORACLE

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

Objectives

After completing this lesson, you should be able to:

- Define users, groups, roles, and policies for the embedded LDAP server
- Configure auditing and role mapping



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

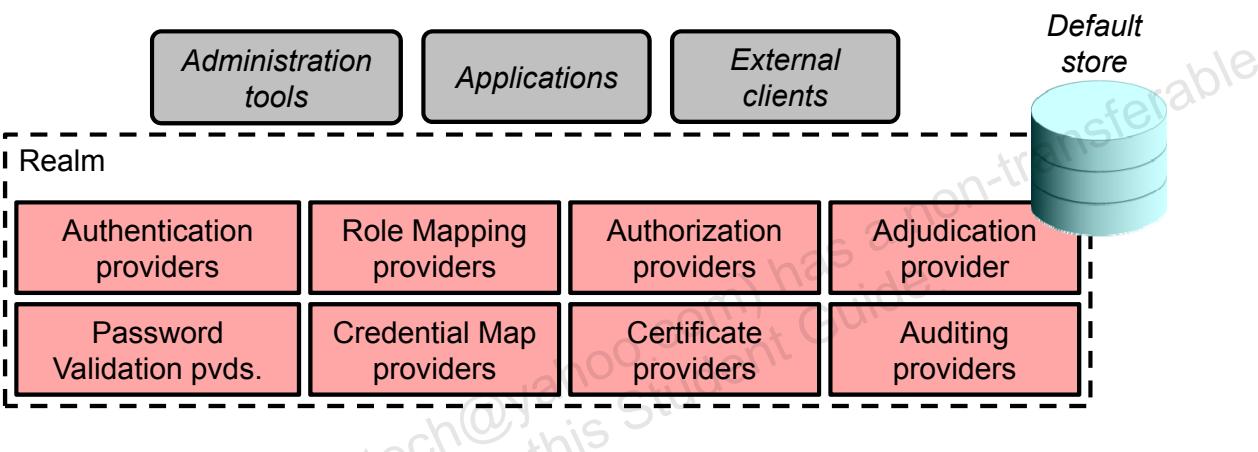
Agenda

- Security Review
 - Security Realm
 - Default Security Configuration
- The Embedded LDAP Authentication System
- Configure Auditing

Security Realm: Review

A security realm:

- Handles security logic and decisions for a domain
- Consists of a series of pluggable providers
- Is scoped to all servers in a domain



ORACLE

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

The security service in WebLogic Server simplifies the configuration and management of security while offering robust capabilities for securing your WebLogic Server deployment. Security realms act as a scoping mechanism. Each security realm consists of a set of configured security providers, users, groups, security roles, and security policies. You can configure multiple security realms in a domain; however, only one can be the active security realm.

A security policy is an association between a WebLogic resource and one or more users, groups, or security roles. Security policies protect the WebLogic resource against unauthorized access. A WebLogic resource has no protection until you create a security policy for it.

A security provider store contains the users, groups, security roles, security policies, and credentials used by some types of security providers to provide their services. For example, an authentication provider requires information about users and groups; an Authorization provider requires information about security policies; a Role Mapping provider requires information about security roles; and a Credential Mapping provider requires information about credentials to use for remote applications. These security providers need this information to be available in a database to function properly.

An Adjudication provider determines what to do if access decisions of multiple Authorization providers do not agree on an answer. The Adjudication provider resolves authorization conflicts by weighing each access decision and returning a final result.

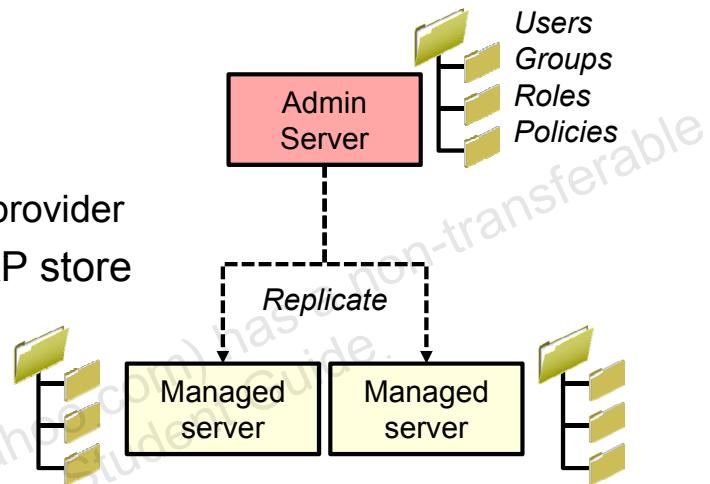
The Certificate Lookup and Validation providers complete certificate paths and validate X509 certificate chains, respectively. If multiple providers are configured, a certificate or certificate chain must pass validation with all of them in order for the certificate or certificate chain to be accepted.

An Auditing provider collects, stores, and distributes information about operating requests and the outcome of those requests for the purposes of nonrepudiation. An Auditing provider makes the decision about whether to audit a particular event based on specific audit criteria, including audit severity levels.

Default Security Configuration

A new domain includes a default realm that:

- Includes some basic providers such as:
 - Default authenticator
 - Default identity asserter
 - XACML role mapper
 - XACML authorizer
 - Default adjudicator
 - Default certificate path provider
- Uses the embedded LDAP store



ORACLE

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

To simplify the configuration and management of security, WebLogic Server provides a default security configuration. In the default security configuration, myrealm is set as the default (active) security realm, and the WebLogic Adjudication, Authentication, Identity Assertion, Credential Mapping, CertPath, EXtensible Access Control Markup Language (XACML) Authorization, and XACML Role Mapping providers are defined as the security providers in the security realm.

The WebLogic Server–embedded LDAP server for a domain consists of a master LDAP server (maintained in the domain’s Administration Server) and a replicated LDAP server (maintained in each managed server in the domain). When changes are made using a managed server, updates are sent to the embedded LDAP server on the Administration Server. The embedded LDAP server on the Administration Server maintains a log of all changes. It also maintains a list of managed servers and the current change status for each one. The embedded LDAP server on the Administration Server sends appropriate changes to each managed server and updates the change status for each server. This process occurs when an update is made to the embedded LDAP server on the Administration Server. However, depending on the number of updates, it may take several seconds or more for the change to be replicated to the managed server. The default interval is every thirty seconds.

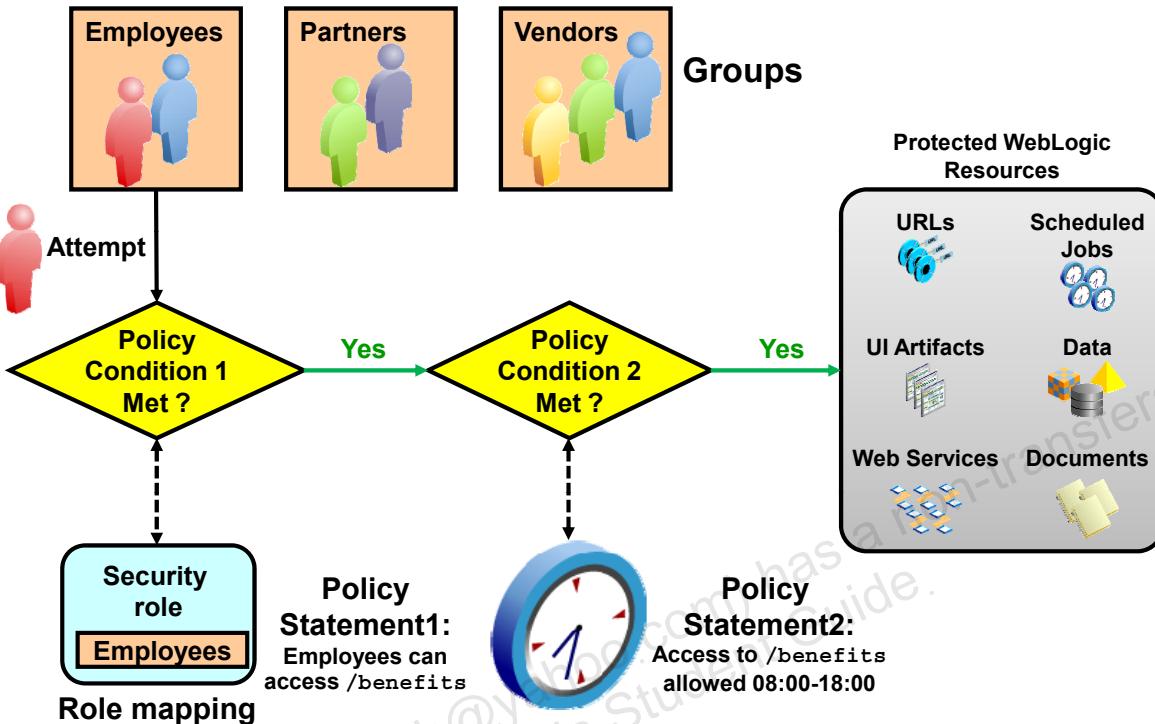
Agenda

- Security Review
- The Embedded LDAP Authentication System
 - How WebLogic Resources Are Protected
 - Users, Groups, Roles, and Policies
 - Configuring Users, Groups, Roles, and Policies
 - Configuring the Embedded LDAP server
- Configure Auditing

ORACLE

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

How WebLogic Resources Are Protected



ORACLE

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

This slide represents the big picture of how WebLogic security artifacts work together to protect WebLogic resources.

- First, you create users and groups and statically assign users to groups that represent organizational boundaries.
- Then you create a security role based on your established business procedures. The security role consists of one or more conditions that specify the circumstances under which a particular user, group, or other role should be granted the security role.
- At run time, the WebLogic Security Service compares groups against role conditions to determine whether users in the group should be granted a security role. This process of matching groups to roles is called role mapping.
- Next, you create a security policy based on your established business procedures. The security policy consists of one or more policy conditions that specify the circumstances under which a particular security role should be granted access to a WebLogic resource.
- At run time, the WebLogic Security Service uses the security policy to determine whether access to the protected WebLogic resource should be granted. Only users who are members of the group that is granted the security role can access the WebLogic resource.

Examples of WebLogic Resources to Protect

Policies that are more specific always override those that are more general.

Resource	Privileges
JDBC Data Source	<ul style="list-style-type: none">• admin• reserve• reset• shrink
JMS Destination	<ul style="list-style-type: none">• All operations• Send• Receive• Browse
JNDI Node	<ul style="list-style-type: none">• Lookup• List• Modify



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

This slide shows some examples of WebLogic resources and privileges that you can protect with security policies.

All the resources within a Java EE application or module that you deploy exist within a hierarchy, and policies on resources higher in the hierarchy act as default policies for resources lower in the same hierarchy. Policies lower in a hierarchy always override policies higher in the hierarchy.

Users and Groups

Users are entities that use WebLogic, such as:

- Application end users
- Client applications
- Other WebLogic Servers



Groups are:

- Logical sets of users
- More efficient for managing a large number of users



ORACLE

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

Users are entities that can be authenticated in a security realm. A user can be a person (such as an application end user), or a software entity (such as a client application or other instances of WebLogic Server). As a result of authentication, a user is assigned an identity or a principal. Each user is given a unique identity within the security realm.

Users can be placed into groups that are associated with security roles or be directly associated with security roles.

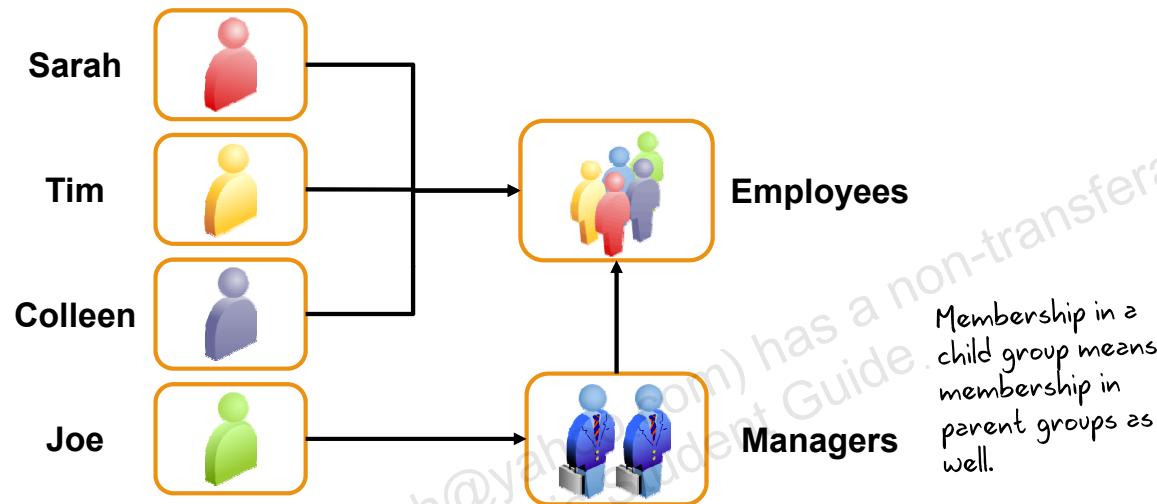
When users want to access WebLogic, they present proof material (such as a password or a digital certificate) to the authentication provider configured in the security realm. If WebLogic can verify the identity of the user based on that username and credential, then it associates the principal assigned to the user with a thread that executes code on behalf of the user. WebLogic then checks the security policy of the requested WebLogic resource to make sure that the user has the required permissions before the thread begins executing code.

A person can be defined as both an individual user and a group member. Individual-access permissions override any group member-access permissions. WebLogic evaluates each user by first looking for a group and testing whether the user is a member of the group and then looking for the user in the list of defined users.

Group Membership

WebLogic allows you to organize groups in various ways:

- Groups can contain users.
- Groups can contain other groups.



ORACLE

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

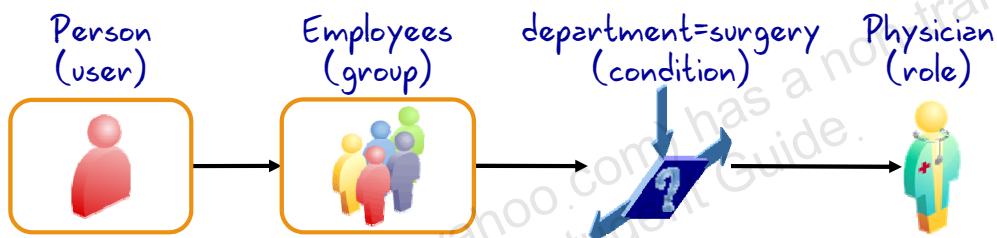
Groups can be organized in arbitrary ways, thereby providing greater flexibility. In the example in the slide, all the users (Sarah, Tim, Colleen, and Joe) are members of the Employees group. Joe is also a member of the Managers group. All Managers are also Employees.

Managing groups is more efficient than managing large numbers of users individually. For example, an administrator can specify permissions for 50 users at one time if those 50 users belong to the same group. Usually, group members have something in common. For example, a company may separate its sales staff into two groups: Sales Representatives and Sales Managers. This is because staff members have different levels of access to WebLogic resources depending on their job descriptions.

WebLogic supports assigning users to groups. Each group shares a common set of permissions that govern its member users' access to resources. You can mix group names and usernames whenever a list of users is permitted.

Roles

- Roles classify a set of users in an application or enterprise who have the same permissions.
- Users and groups can be granted multiple roles.
- The two types of roles are:
 - Global scoped
 - Resource scoped



ORACLE

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

A role definition is specific to a security realm. Roles are tightly coupled with the authorization provider. A role can be defined as resource scoped or global scoped. A resource-scoped role means that it is granted to a specific resource, such as a method of an EJB or a branch of the JNDI tree. Most roles are scoped. Global-scoped role means that a role is granted to all the resources in a security realm. WLS defines a set of default global roles for protecting the WebLogic resources. Roles are assigned statically by defining them in the deployment descriptor for specified users or groups, or dynamically through the administration console by defining a set of conditions.

The global roles that are available by default are the following: AppTester, Anonymous, Admin, Operator, Deployer, and Monitor.

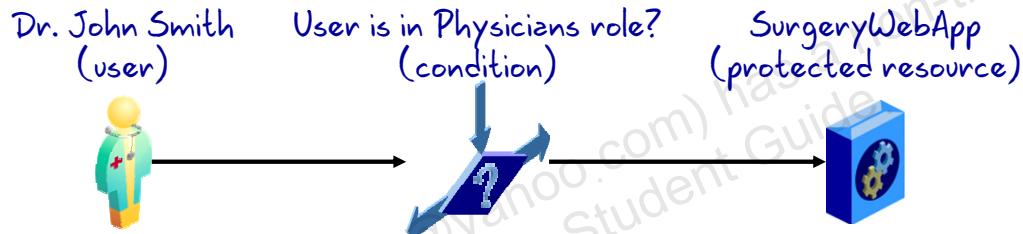
Roles that are defined in deployment descriptors can be inherited:

- During deployment of an application, security roles defined within an application may get migrated into the configured LDAP server.
- It is possible to configure or disable this feature.

You can manage role definitions and assignments without editing deployment descriptors or redeploying the application.

Policies

- Policies are a set of related data points that are evaluated to determine what privileges are granted or denied to a user.
- Policies are global scoped or resource scoped.
- Data points can include:
 - Users, groups, or roles
 - Access times and context field values

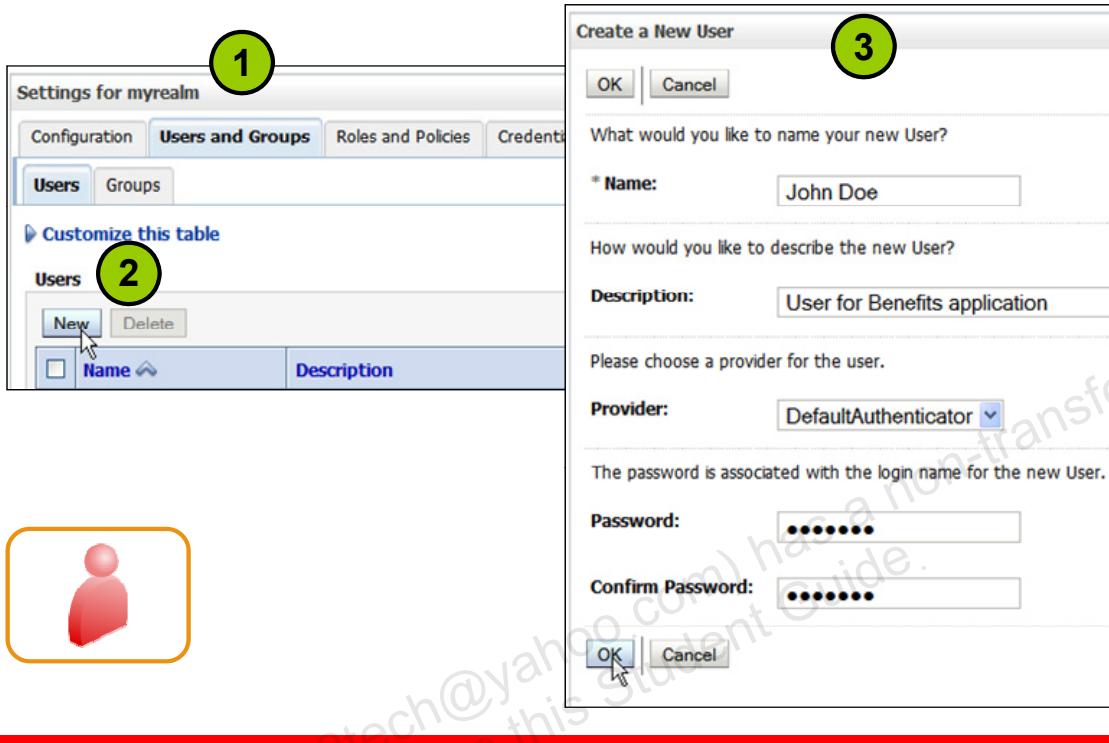


ORACLE

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

A policy is a rule that is used to determine if a user has privileges to access a WebLogic resource.

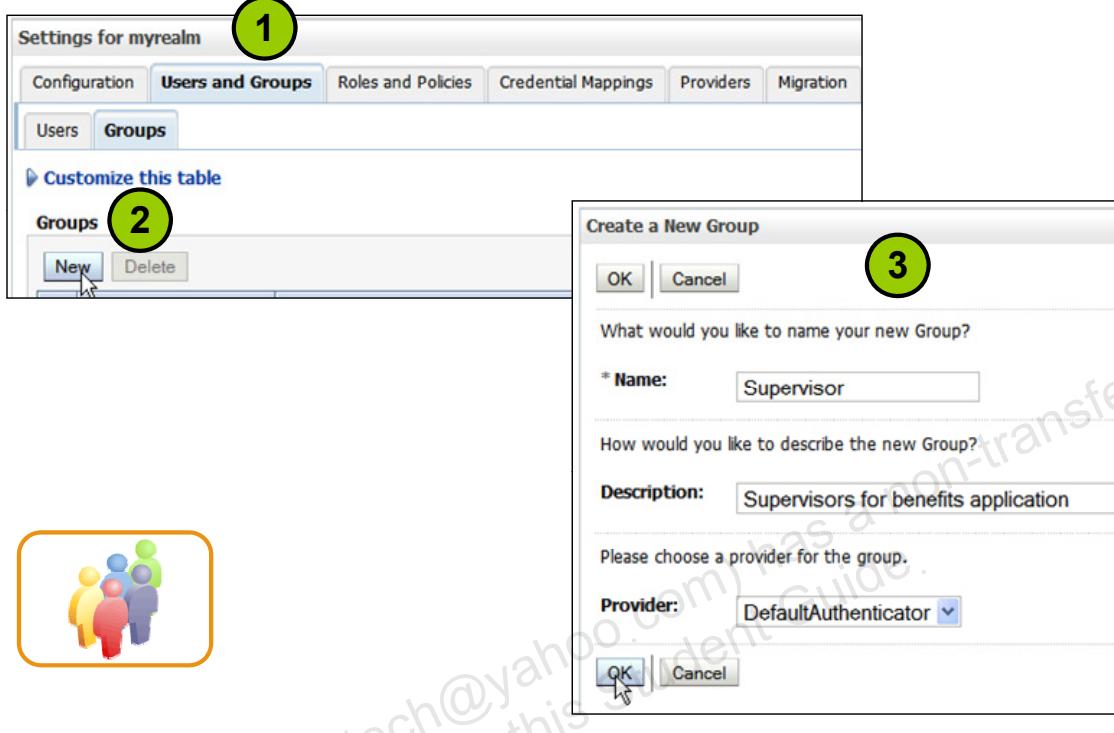
Configuring New Users



To configure a new user, perform the following steps:

1. Access Security Realms and select your security realm in the Realms Table on the *Summary of Security Realms* page. Click the *Users and Groups > Users* tab for your realm.
2. Click New in the Users table.
3. Enter the necessary details in the *Create a New User* dialog box and click OK. The name may contain spaces, but other systems may not allow spaces. Best practice is to reserve the use of spaces for the description and use underscores if needed for the name.

Configuring New Groups



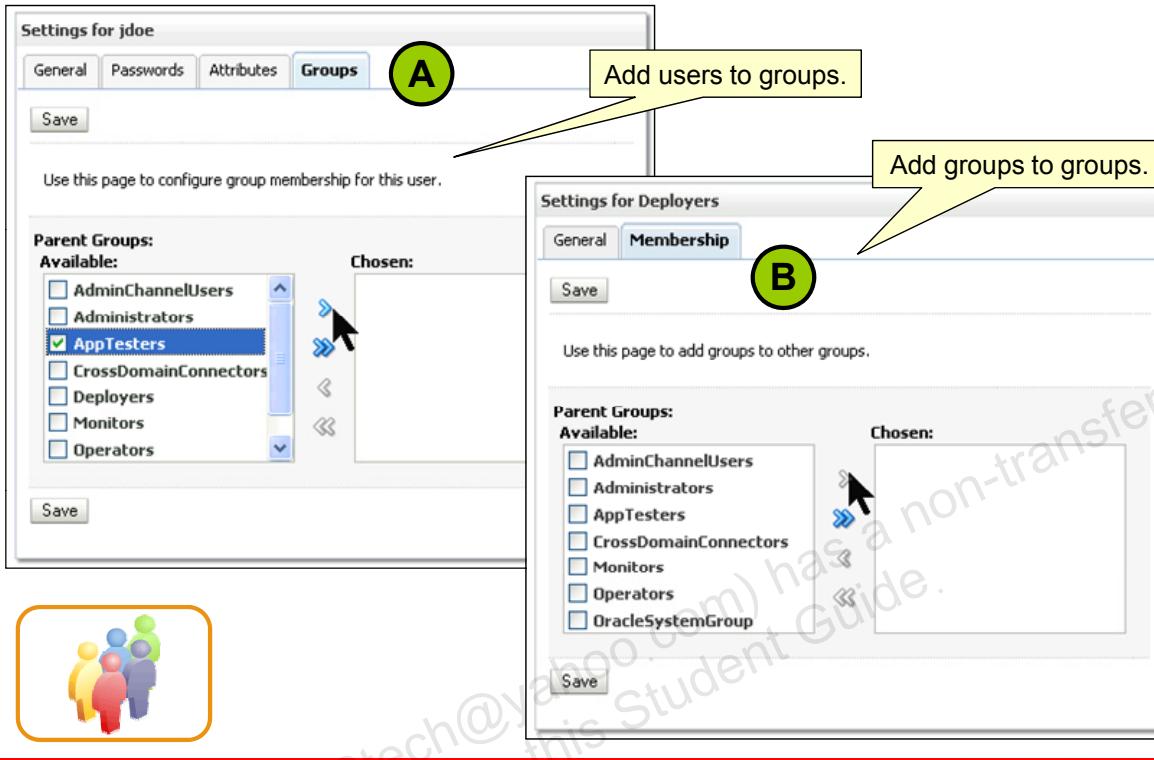
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

To configure a new group, perform the following steps:

1. Access Security Realms and select your security realm in the Realms Table on the *Summary of Security Realms* page. Click the *Users and Groups > Groups* tab for your realm.
2. Click New in the Groups table.
3. Enter the necessary details in the *Create a New Group* dialog box and click OK.

Configuring Group Memberships



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

ORACLE

Each group has two types of membership.

- A. You can configure a user to be a member of a group as follows:
 1. Navigate to the Users subtab under the *Users and Groups* tab of the security realm.
 2. Select the user for whom you want to configure the group membership.
 3. Click the Groups tab on the *Settings for <user>* page.
 4. Select the group from the Available list and click > to move it to the Chosen list. Then click Save.
- B. You can configure a group to be a member of another group as follows:
 1. Navigate to the Groups subtab under the *Users and Groups* tab of the security realm.
 2. Select the group that you want to configure as a child of another group.
 3. Click the Membership tab on the *Settings for <group>* page.
 4. Select the parent group from the Available list and click > to move it to the Chosen list. Then click Save.

Configuring New Roles



ORACLE

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

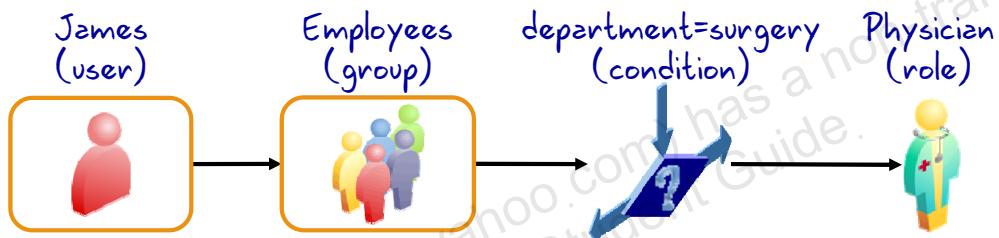
To create a global security role in WebLogic by using the administration console:

1. Navigate to the active security realm and click *Roles and Policies* > *Realm Roles* to display the list of configured roles.
2. Expand *Global Roles* and select *Roles* to display the list of configured global roles.
3. Click *New* to create a new role.
4. Enter a name for the role, select the role mapping provider (in this case, the default XACML RoleMapper provider), and click *OK*.

What Is Role Mapping?

Role mapping is the process of aligning a user into a security role which is used to determine access to resources.

- Static and dynamic role mapping are available.
- Users are potentially mapped to multiple roles.
- Mapping may include analyzing data points above and beyond simple group membership.



ORACLE

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

Configuring Role Mapping

The screenshot shows the Oracle WebLogic Administration Console interface. On the left, a list of 'Global Roles' is displayed, including 'Admin', 'AdminChannelUser', 'Anonymous', 'AppTester', 'CrossDomainConnector', 'Deployer', 'Monitor', 'Operator', 'OracleSystemRole', and 'Physicians'. A green circle labeled '1' highlights the 'Physicians' role. In the center, the 'Edit Global Role' page for 'Physicians' is shown. The 'Name:' field is set to 'Physicians'. Below it, a note states: 'These conditions determine membership in the role.' A 'Role Conditions:' section follows, containing an 'Add Conditions' button, a 'Combine' button, an 'Uncombine' button, 'Move Up', 'Move Down', 'Remove', and 'Negate' buttons, and a 'Save' button. A green circle labeled '2' is positioned over the 'Add Conditions' button. Another green circle labeled '3' is positioned over the 'Save' button. At the bottom of the page, there is a red bar with the 'ORACLE' logo and the copyright notice: 'Copyright © 2014, Oracle and/or its affiliates. All rights reserved.'

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

Continuing from where you left off after creating a new role using the administration console:

1. Select the Physician role you created to display the Edit Global Role page.
2. Click *Add Conditions* to specify the criteria used to determine if a user is granted the role.
3. Using the Role Conditions Wizard, you create the conditions that are used to map users to your role. In this case, if a user belongs to the Employees group and his or her department context variable is surgery, then the user is granted the role. The context element's value is based on HTTP Servlet Request attributes, HTTP Session Attributes, or EJB method parameters. During role evaluation, WebLogic retrieves this information from the ContextHandler object and compares it to the role mapping condition.

Configuring Roles Using WLST

Creating roles:

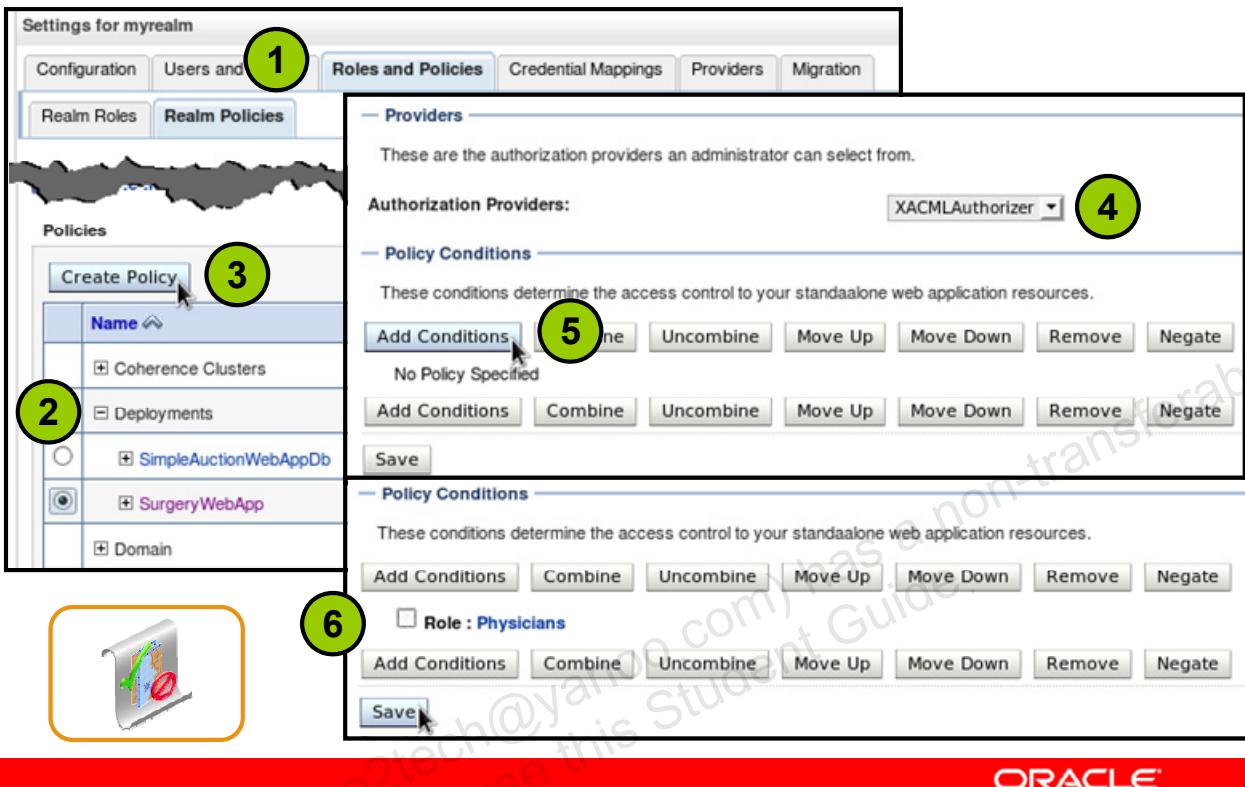
```
rm = cmo.getSecurityConfiguration().getDefaultRealm() +  
      'RoleMappers/XACMLRoleMapper')  
  
i=0  
print 'creating roles'  
roles = ['user', 'creator']  
groups = ['AuctionUsers', 'AuctionCreators']  
for role in roles:  
    try:  
        print 'creating role: ' + role  
        expression = 'Grp(' + groups[i] + ')' Build the role expression, setting role if user is a member of group[i]  
        rm.createRole(None, role, None) Create the role.  
        rm.setRoleExpression(None, role, expression) Add the expression to the role.  
        i += 1  
        print ''  
    except Exception, e:  
        print 'Role creation exception:'  
        print e  
        i += 1  
        print ''
```

Acquire the role mapping provider Mbean.
Loop through roles and match role to proper group[i]
CreateRole.py

ORACLE

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

Configuring New Policies



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

To create an authorization policy in WebLogic by using the administration console:

1. Navigate to the active security realm and click *Roles and Policies > Realm Policies* to display the list of configured policies.
2. Expand the appropriate scope to display the list of configured roles for that level. If applicable, select the entity (such as an application) that you want to protect with a policy.
3. Click *Create Policy* to create a new policy.
4. Select the authorization provider from a list of configured providers for the realm.
5. Use the buttons in the Policy Conditions section to create the conditions that comprise the policy itself.
6. The policy conditions are displayed as they are created. When the policy is complete, you save the changes and the policy is in effect. In this case, the policy protects access to the *SurgeryWebApp* application allowing access only to users in the *Physicians* role.

Configuring Policies Using WLST

Creating a policy:

```
authz=cmo.getSecurityConfiguration().getDefa
```

Acquire the authorization provider Mbean.

```
lDefaultRealm() +  
    '/Authorizers/XACMLAuthorizer')
```

Create the string version of the resource.

```
#URLResource(application, context_path, pattern, httpMethod, transport)  
r=weblogic.security.service.URLResource('SimpleAuctionWebAppDbSec',  
    '/SimpleAuctionWebAppDbSec',  
    '/createAuction.jsp', None, None)
```

Create a URL pattern-based resource to protect.

```
resourceID=r.toString()  
expression='Rol(creator)'  
Create the expression for the policy to check if the user is in the creator role.
```

```
try:  
    authz.createPolicy(resourceID, expression)  
    print ''  
except Exception, e:  
    print 'Policy creation exception:'  
    print e  
    print ''  
Create the policy using the resource and expression.
```

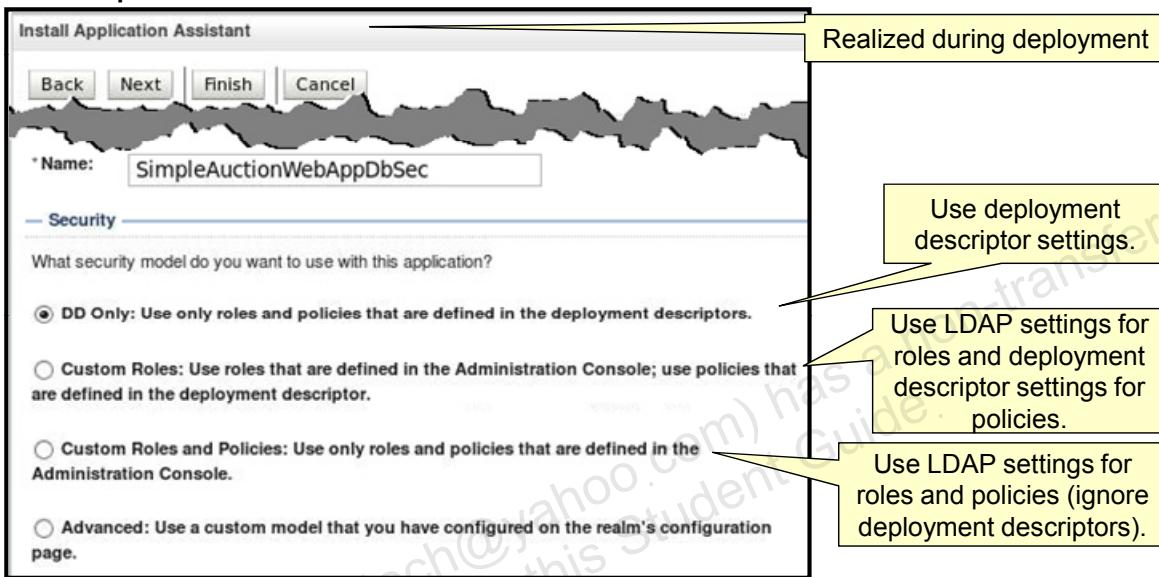
CreatePolicy.py

ORACLE

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

Security Configuration Sources

Security roles, role mapping, and policies are configured either in the embedded LDAP store or an application's deployment descriptor.



ORACLE

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

Configuring Sources Using WLST and weblogic.Deployer

WLST:

```
> deploy(appName='SimpleAuctionWebAppDbSec',
         path='/apps/SimpleAuctionWebAppDbSec.war',
         targets='cluster1',
         securityModel='[DDOnly | CustomRoles |
                         CustomRolesAndPolicies | Advanced')
```

WLST Command Line

weblogic.Deployer:

```
$ java weblogic.Deployer
    -adminurl t3://host01.example.com:7001 -username weblogic
    -password Welcome1 -deploy -targets cluster1
    -securityModel [ DDOnly | CustomRoles |
                      CustomRolesAndPolicy | Advanced]
    /deployments/SimpleAuctionWebAppDbSec.war
```

Command Line

ORACLE

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

Deployment Descriptor Security Example: `weblogic.xml`

Security roles are defined in the `weblogic.xml` file and policies are defined in the `web.xml` file.

```
<security-role-assignment>
    <role-name>user</role-name>
    <principal-name>AuctionUsers</principal-name>
</security-role-assignment>

<security-role-assignment>
    <role-name>auctionCreators</role-name>
    <principal-name>AuctionCreators</principal-name>
</security-role-assignment>
```

Users that are in the AuctionUsers group are mapped to the user role.

Users that are in the AuctionCreators group are mapped to the auctionCreators role.

weblogic.xml



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

Deployment Descriptor Security Example: web.xml

```
<security-constraint>
    <display-name>create auction</display-name>
    <web-resource-collection>
        <web-resource-name>createAuction</web-resource-name>
        <url-pattern>/createAuction.jsp</url-pattern>
        <url-pattern>/CreateAuctionServlet</url-pattern>
    </web-resource-collection>

    <auth-constraint>
        <role-name>auctionCreators</role-name>
    </auth-constraint>
</security-constraint>

<security-role>
    <role-name>user</role-name>
</security-role>
<security-role>
    <role-name>auctionCreators</role-name>
</security-role>
```

These URL resources
are accessed only by users in the auctionCreators role.
References security roles that are defined in the weblogic.xml file

web.xml

ORACLE

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

Embedded LDAP Server

- In WebLogic, users, groups, and authorization information can be stored in the embedded LDAP server.
- Several properties can be set to manage the LDAP server, including:
 - Credentials
 - Backup settings
 - Cache settings
 - Replication settings



ORACLE

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

The embedded LDAP server is used as a storage mechanism for WebLogic authentication, authorization, role mapping, and credential mapping providers.

Information from these providers is stored and updated in the administration server and replicated to all the managed servers in the domain. The read operations performed by WebLogic security providers (when running on a managed server) access the local, replicated embedded LDAP server. The write operations access the master embedded LDAP server on the administration server and any updates are replicated to all the managed servers in the domain. If the administration server is not running, operations executed by WebLogic security providers that write to the embedded LDAP server (for example, adding new users, groups, or roles, or adding resources) are not possible.

Configuring the Embedded LDAP Server

The screenshot shows two panels of the Oracle WebLogic Server Administration Console. The left panel is titled 'Settings for wlsadmin' and has tabs for Configuration, Monitoring, Control, Security, and Web Service Security. Under Security, it has tabs for General, Filter, Unlock User, Embedded LDAP, Roles, and Policies. The 'Embedded LDAP' tab is selected. It contains fields for Credential (password), Confirm Credential, Backup Hour (23), Backup Minute (5), and Backup Copies (7). A 'Save' button is at the bottom. The right panel is titled 'Cache Enabled' and contains settings for Cache Size (32), Cache TTL (60), Refresh Replica At Startup (unchecked), Master First (unchecked), Timeout (0), and Anonymous Bind Allowed (unchecked). A note at the bottom says: 'Back up copies are stored in zip files in \$DOMAIN_HOME/servers/AdminServer/data/ldap/backup'.

Back up copies are stored in zip files in
\$DOMAIN_HOME/servers/AdminServer/data/ldap/backup

ORACLE

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

- **Credential:** The credential (usually password) that is used to connect to the embedded LDAP server. This password is encrypted. The default is randomly generated and must be changed if you want to use an LDAP client to connect to the server.
- **Backup Hour:** The hour at which to back up the embedded LDAP server. Minimum is 0, Maximum is 23, and Default is 23.
- **Backup Minute:** The minute at which to back up the embedded LDAP server. This attribute is used with the `Backup_Hour` attribute to determine the time at which the embedded LDAP server is backed up. Minimum is 0, Maximum is 59, and Default is 05.
- **Backup Copies:** The number of backup copies of the embedded LDAP server. Minimum is 0, Maximum is 65534, and Default is 7.
- **Cache Enabled:** Whether or not a cache is used for the embedded LDAP server. The default is True.
- **Cache Size:** The size of the cache (in KB) that is used with the embedded LDAP server. Minimum is 0 and Default is 32.
- **Cache TTL:** The time-to-live (TTL) of the cache in seconds. Minimum is 0 and Maximum is 60.

- **Refresh Replica At Startup:** Whether or not a managed server should refresh all replicated data at boot time. This is useful if you made a large number of changes when the managed server was not active and you want to download the entire replica instead of having the administration server push each change to the managed server. The default is false.
- **Master First:** The connections to the master LDAP server should always be made instead of connections to the local replicated LDAP server. The default is false.
- **Timeout:** Specifies the maximum number of seconds to wait for results from the embedded LDAP server before timing out
- **Anonymous Bind Allowed:** Specifies whether the embedded LDAP server should allow anonymous connections

Quiz

A security policy is used to:

- a. Map a user to a role
- b. Map a group to a role
- c. Make a user a member of a group
- d. Authorize if a user has access to a resource



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

Answer: d

Practice 13-1 Overview: Creating Users, Groups, Roles, and Policies

This practice covers the following topics:

- Creating users and groups
- Adding users to groups
- Mapping users to roles
- Authorizing access to a resource with a policy

Agenda

- Security Review
- The Embedded LDAP Authentication System
- Configure Auditing
 - Auditing
 - Security Audit Events
 - WebLogic Auditing Architecture
 - Custom Versus Default Auditing Provider
 - Configuring the Default Auditing Provider
 - Configuration Auditing

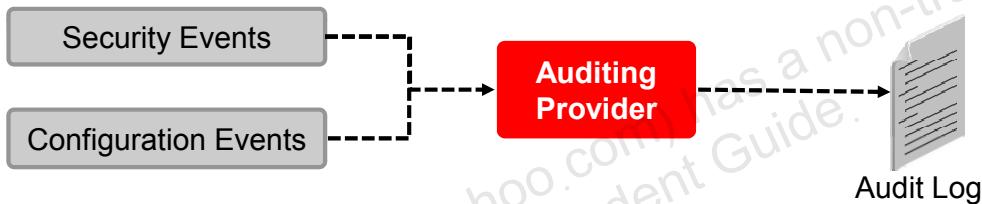


ORACLE

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

Auditing

- Auditing provides a trail of user activity related to security and configuration changes.
- The default auditing provider records event data that is associated with security requests and the outcome of the requests.
- Auditing can also create a record of all domain configuration changes.



ORACLE

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

WebLogic provides an auditing provider to collect, store, and distribute information about requests and the outcome of those requests for nonrepudiation. If you are not familiar with the term, nonrepudiation means that someone is not able to refute, or deny, that they performed an action by using a system. This is because their actions are recorded, or audited. An auditing provider decides whether to audit a particular event based on specific audit criteria, including audit severity levels. The audit information is written to output repositories, such as a simple file.

You can configure multiple auditing providers in a security realm, but none are required.

You can set the location of the audit log file by setting the `-Dweblogic.security.audit.auditLogDir` property on the WebLogic Server command line.

Sample Auditing Output

The default location of the audit log is:

DOMAIN_HOME/yourServer/logs/DefaultAuditRecorder.log

```
#### Audit Record Begin <Apr 3, 2013 5:51:28 PM><Severity =INFORMATION> <<<Event  
Type = RoleManager Audit Event ><Subject: 0  
><<jdbc>><type=<jdbc>, application=, module=, resourceType=ConnectionPool,  
resource=jdbc/AuctionDB, action=reserve>>> Audit Record End ####  
#### Audit Record Begin <Apr 3, 2013 5:51:28 PM> <Severity =SUCCESS> <<<Event  
Type = Authorization Audit Event V2 ><Subject: 0  
><ONCE><<jdbc>><type=<jdbc>, application=, module=, resourceType=ConnectionPool,  
resource=jdbc/AuctionDB, action=reserve>>> Audit Record End ####
```

DefaultAuditRecorder.log

ORACLE

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

Security Audit Events

- Typical security events include:
 - Attempting an authentication or identity assertion
 - Creation of a new role or policy
 - Locking out or unlocking a user account
- Security events have the following characteristics:
 - Name
 - Severity (WARNING, ERROR, SUCCESS, and so on)
 - Zero or more context attributes:
 - Protocol, port, address
 - HTTP headers
 - EJB method parameters
 - SAML tokens



ORACLE

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

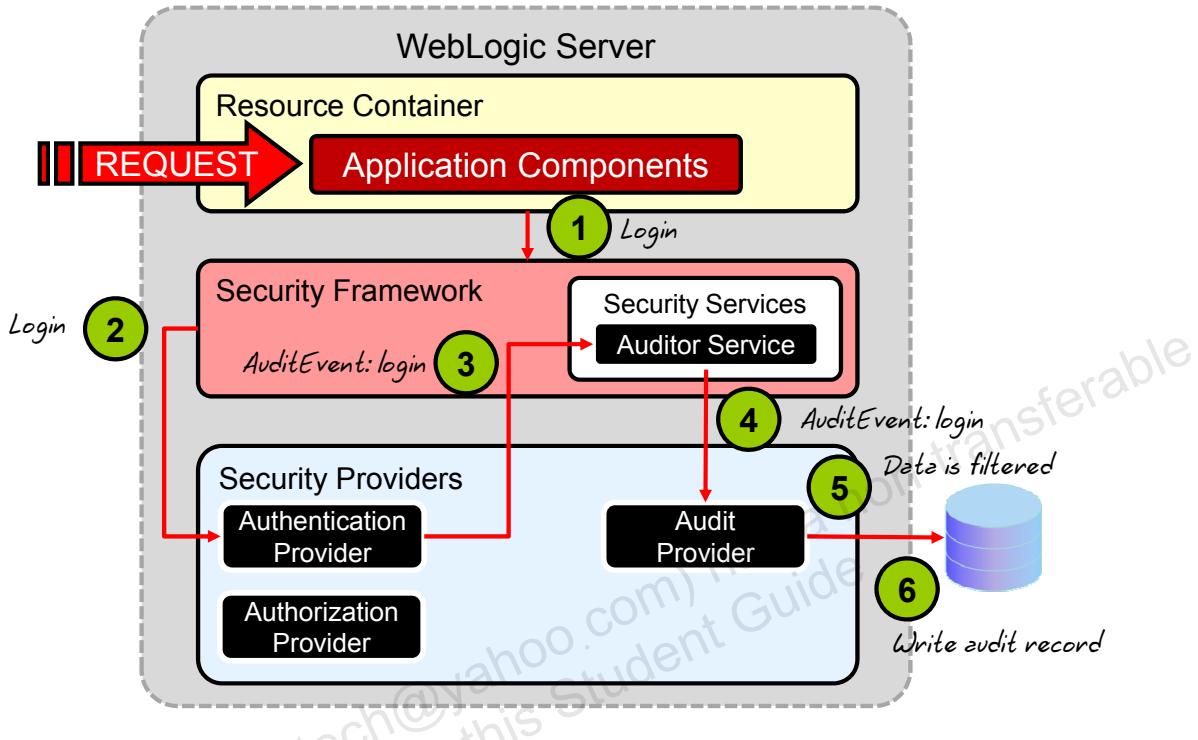
In addition to the events listed in the slide, the default WebLogic auditing provider records the following types of security events:

- When the lockout on a user account expires
- A security policy is used and an authorization decision is made.
- A role definition is used.
- A role or policy is removed or “undeployed.”

The WebLogic auditing provider audits security events of the specified severity and higher. The severity levels, in order from lowest to highest, are: INFORMATION, WARNING, ERROR, SUCCESS, FAILURE. You can also set the severity level to CUSTOM, and then enable the specific severity levels that you want to audit, such as ERROR and FAILURE events only.

An audit event includes a context object that can hold a variety of different attributes, depending on the type of event. When you configure an auditing provider, you specify which context attributes are recorded for each event. By default, no context attributes are audited.

WebLogic Auditing Architecture



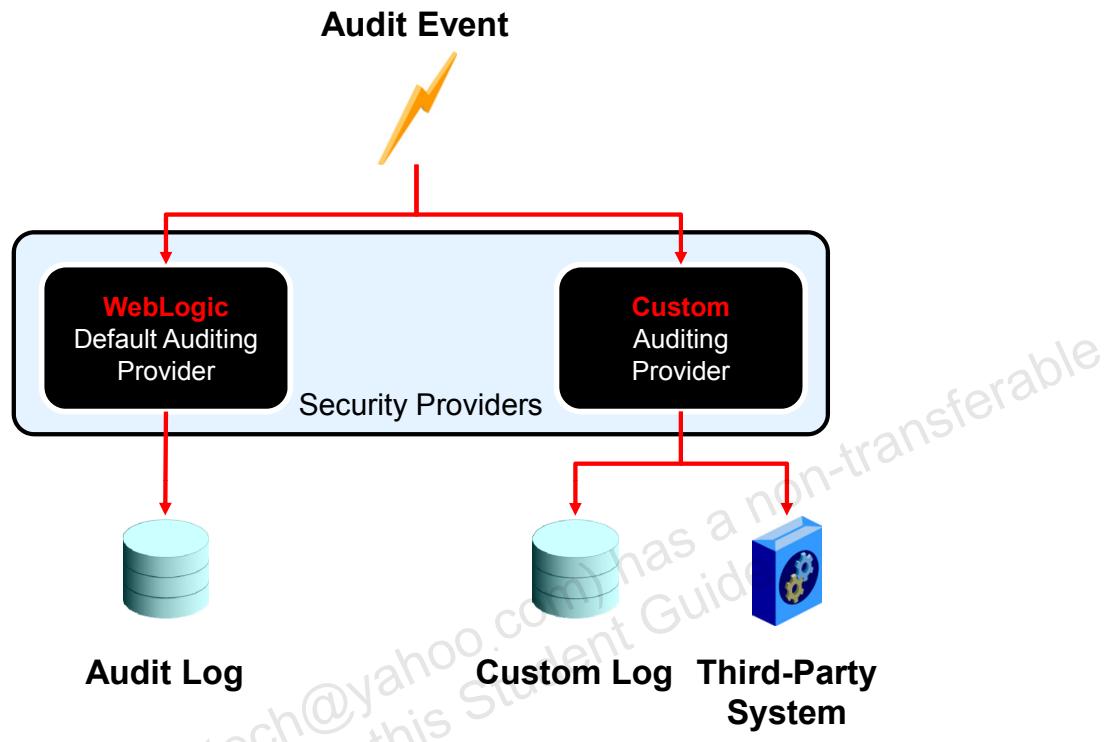
ORACLE

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

This slide shows an example of the process architecture of WebLogic auditing. Security providers are implemented to post audit events for certain activities that occur as part of the authentication and authorization process. Those events are handled by the Auditor Service, which passes them on to the Audit Provider for processing.

1. A request for an application comes into the server and the user is logging in.
2. The login request is handled by the authentication provider.
3. The authentication provider posts the login audit event as part of its processing.
4. The Auditor Service receives the login audit event and sends it to the Audit Provider for processing.
5. The Audit Provider receives the login audit event, compares it to the current settings that are configured for auditing, and applies all settings to determine whether or not an audit record is written.
6. When an audit event meets the configured criteria for auditing, the audit record is written to the log file.

Custom Versus Default Auditing Provider

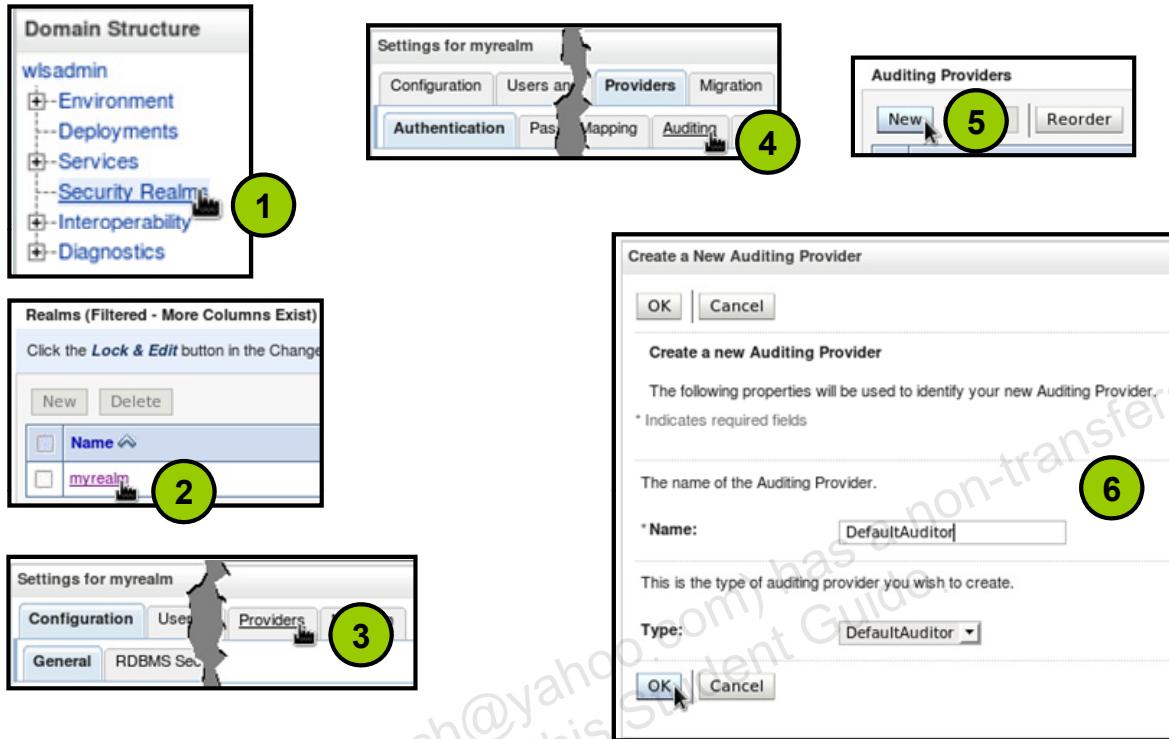


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

The default WebLogic auditing provider posts audit events and logs audit records to a file. If your application has different or enhanced auditing requirements, you can create a custom audit provider that provides for those requirements. You can configure multiple audit providers in WebLogic, thus allowing each provider an opportunity to audit events.

Creating the Default Auditing Provider



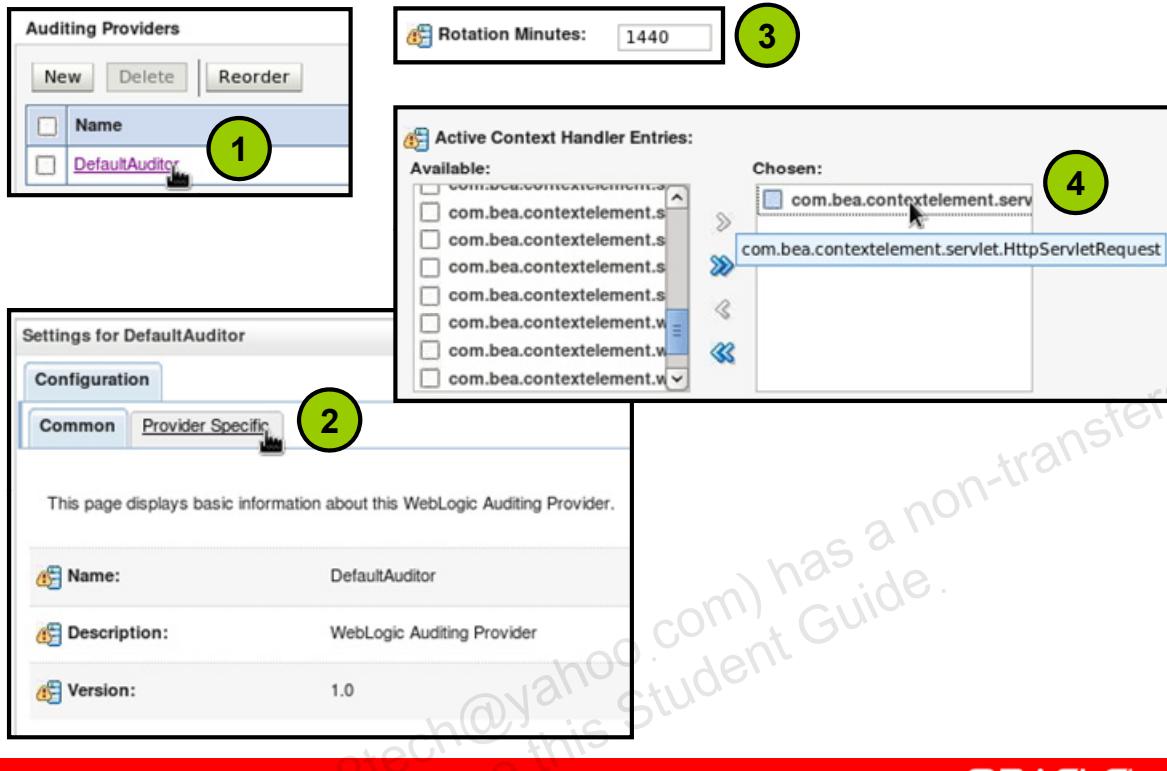
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

You create a WebLogic default audit provider by performing the following steps using the administration console:

1. Click Security Realms in the Domain Structure panel.
2. Select the active realm of the domain.
3. Click the Providers tab to display the configuration tabs for all security providers.
4. Click the Auditing tab to list the currently configured audit providers.
5. Click New to create a new audit provider.
6. The Create a New Auditing Provider page is displayed:
 - **Name:** Enter a unique name for your provider.
 - **Type:** The only type available is DefaultAuditor because no custom providers have been configured.
 - Click OK.

Configuring the Default Auditing Provider



ORACLE

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

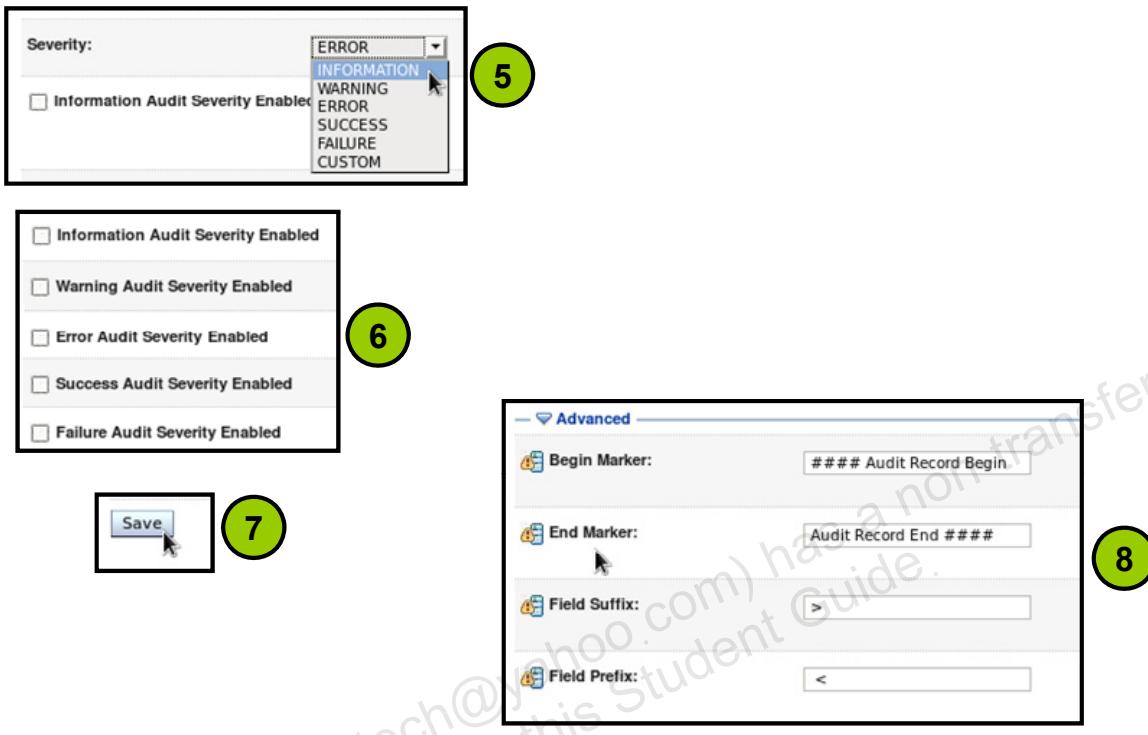
After the audit provider is created, you configure how it functions:

1. Select your newly created audit provider.
2. Click the Provider Specific tab to display the configuration options available for the provider.
3. Configure the number of minutes WebLogic uses to determine when to back up the current audit log file and create a new one for more audit records.
4. Optionally, select from several server context handlers available to include data related to the audited action in the audit record. Objects in the context handler are usually logged by using the `toString()` method of the object.

An audit event includes a ContextHandler that can hold information that is related to the audited request. Some examples include:

- `com.bea.contextelement.servlet.HttpServletRequest`: Includes the servlet request as part of the audit record
- `com.bea.contextelement.saml.SSLClientCertificateChain`: Includes an array of X.509 certificate objects that represent a certificate chain of a SAML client request

Configuring the Default Auditing Provider



ORACLE

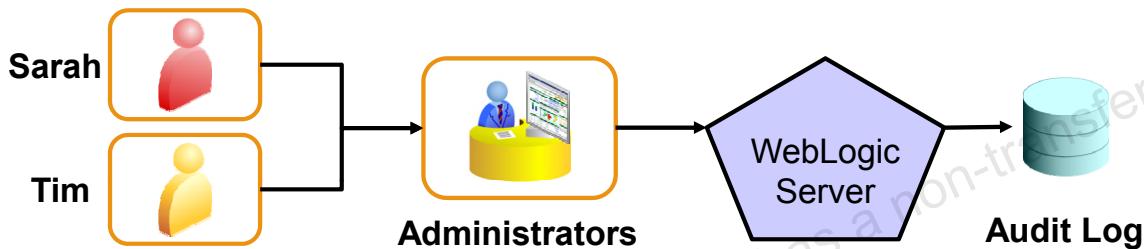
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

5. Optionally, select the severity of audit events to log including, INFORMATION, WARNING, ERROR, SUCCESS, FAILURE, and CUSTOM.
6. If the CUSTOM severity is selected, you can select a combination of multiple severities to include in audit records.
7. Save your changes.
8. You can optionally set some advanced settings to control the output of audit records.

Configuration Auditing

Configure auditing for configuration changes to a domain:

- Add users to the WebLogic Administrators group to make them administrators.
- Audit records contain the administration user's information.



ORACLE

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

WebLogic supports auditing changes made to a domain's configuration. Any user that is a member of the Administrators group is considered a WebLogic administrative account that can make changes to a domain. When individual users are also administrators, their identity is associated with the changes they make in the system. When configuration auditing and the WebLogic default auditor are configured, WebLogic either posts audit events, writes audit records to a log file, or both. When administrators make changes to the system, their identities are captured along with the change so there is a record of who made the change, what the change was, and the status of the change. You can set filters to set the severity of which configuration change audit records are generated including, SUCCESS, FAILURE, and ERROR.

Configuration Auditing

Using the administration console:



Using the command line:

```
-Dweblogic.domain.ConfigurationAuditType="audit"
```

Can be audit,
log, or
logaudit

```
startWebLogic.sh
```

ORACLE

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

You configure the type of configuration auditing either by using the administration console or by setting the `ConfigurationAuditType` property on the WebLogic Server command line.

Using the administration console, you:

1. Select the domain in the Domain Structure panel.
2. On the *Configuration > General tab*, click the Advanced link to display advanced settings.
3. Find the *Configuration Audit Type* field and set it to *None*, *Change Log*, *Change Audit*, or *Change Log and Audit*:
 - **None:** Disables configuration auditing
 - **Change Log:** Writes configuration audit records only to the server log file
 - **Change Audit:** Only emits configuration audit events for processing by the audit provider
 - **Change Log and Audit:** Emits configuration audit events and writes them to the server log file

Quiz

Identify two activities for which the default auditor provides auditing.

- a. Security and configuration use
- b. Security and application use
- c. Security, configuration, and application use
- d. Security and third-party use



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

Answer: a

Summary

In this lesson, you should have learned how to:

- Define users, groups, roles, and policies for the embedded LDAP server
- Configure auditing and role mapping



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

Practice 13-2 Overview: Configuring WebLogic Auditing

This practice covers the following topics:

- Configuring the default WebLogic audit provider
- Creating and viewing audit records

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.

15

Diagnostic Framework

ORACLE®

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

Objectives

After completing this lesson, you should be able to configure the WebLogic Diagnostic Framework (WLDF) to monitor a domain.

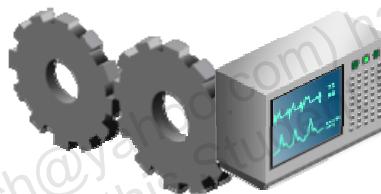


Agenda

- General Architecture
 - WebLogic Diagnostics Framework (WLDF)
 - Diagnostic Archives
 - Diagnostics Modules
- Diagnostic Images
- Harvesters
- Watches and Notifications

WebLogic Diagnostics Framework (WLDF)

- WLDF provides a generic framework to gather and analyze WebLogic runtime data for monitoring and troubleshooting.
- Use WLDF to:
 - Capture a snapshot of key server metrics for distribution to support personnel
 - Capture metrics at specific code points in WebLogic or your application
 - Periodically record selected MBean attributes
 - Send notifications when attributes meet certain conditions

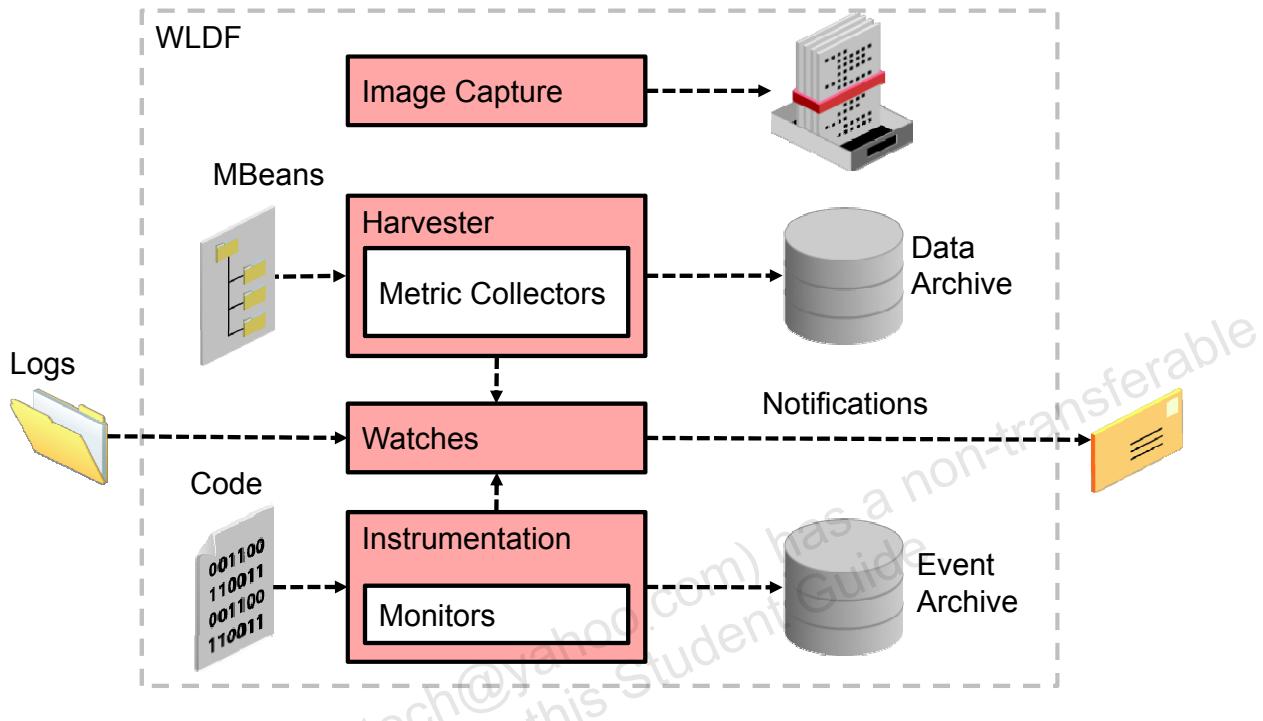


ORACLE®

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

WLDF consists of several components that work together to collect, archive, and access diagnostic information about a WebLogic Server instance and the applications it hosts.

WLDF Architecture



ORACLE

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

Data creators generate diagnostic data, which is consumed by the logger and the harvester. Those components coordinate with the archive to persist the data, and they coordinate with the watch and notification subsystem to provide automated monitoring. The data accessor interacts with the logger and the harvester to expose current diagnostic data and with the archive to present historical data. MBeans make themselves known as data providers by registering with the harvester. Collected data is then exposed to both the watch and notification system for automated monitoring and to the archive for persistence.

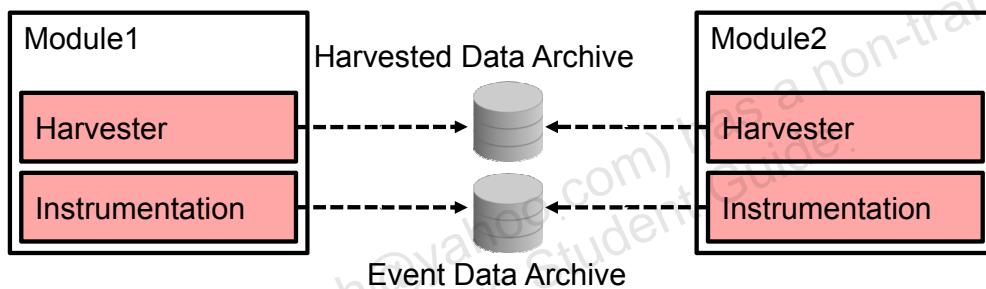
The instrumentation system creates monitors and inserts them at well-defined points in the flow of code execution within the JVM. These monitors trigger events and publish data directly to the archive. They can also take advantage of watches and notifications.

Diagnostic image capture support gathers the most common sources of the key server state used in diagnosing problems. It packages that state into a single artifact, which can be made available to support technicians.

The past state is often critical in diagnosing faults in a system. This requires that the state be captured and archived for future access, creating a historical archive. In WLDF, the archive meets this need with several persistence components. Both events and harvested metrics can be persisted and made available for historical review.

Diagnostic Archives

- Collected MBean metrics and events are recorded in the server's diagnostic archives:
 - WLDF file store (<server>/data/store/diagnostics, by default)
 - WLDF JDBC store
- Recorded data can be limited using size- or age-based retirement policies.



ORACLE

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

The Archive component of the WebLogic Diagnostic Framework (WLDF) captures and persists all data events, log records, and metrics collected by WLDF from server instances and applications running on them. You can access archived diagnostic data in online mode (that is, on a running server). You can also access archived data in offline mode using the WebLogic Scripting Tool (WLST). You configure the diagnostic archive on a per-server basis.

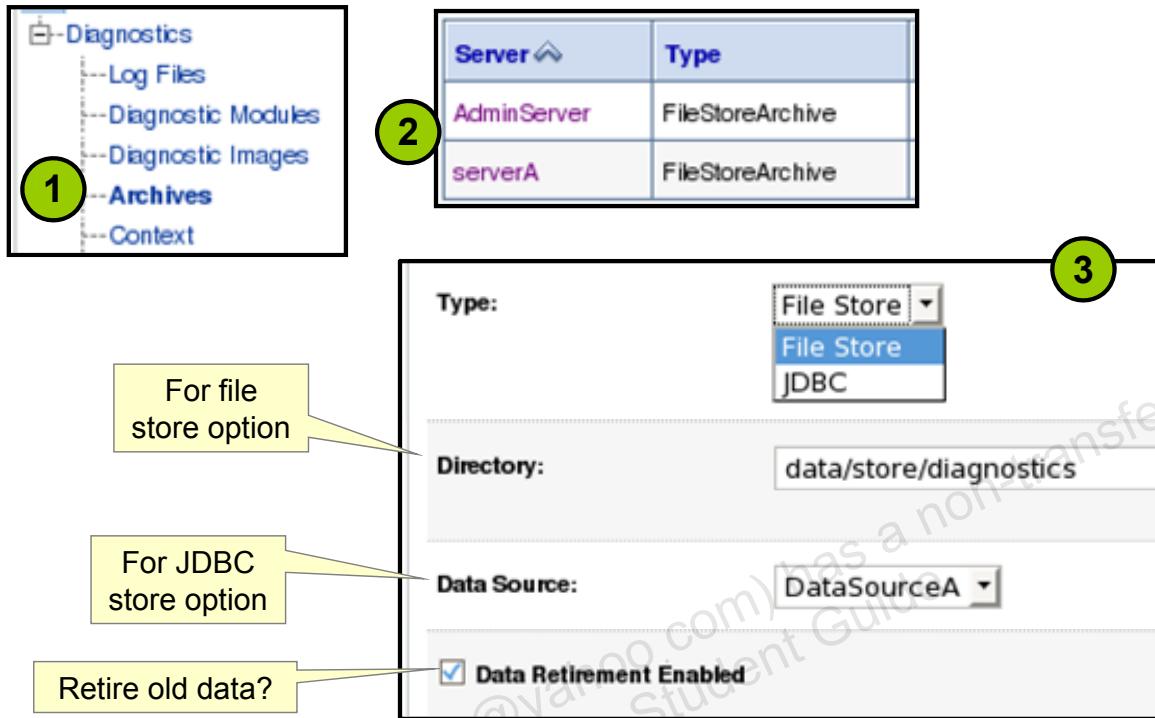
For a file-based store, WLDF creates a file to contain the archived information. The only configuration option for a WLDF file-based archive is the directory where the file will be created and maintained. The default directory is

<domain>/servers/<server>/data/store/diagnostics. When you save to a file-based store, WLDF uses the WebLogic Server persistent store subsystem.

To use a JDBC store, the appropriate tables must exist in a database, and JDBC must be configured to connect to that database. The `wls_events` table stores data generated from WLDF Instrumentation events. The `wls_hvst` table stores data generated from the WLDF Harvester component. Refer to the WebLogic *Configuring Diagnostic Archives* documentation for the required schema.

WLDF includes a configuration-based, data-retirement feature for periodically deleting old diagnostic data from the archives. You can configure size-based data retirement at the server level and age-based retirement at the individual archive level.

Configuring Server Diagnostic Archives



ORACLE

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

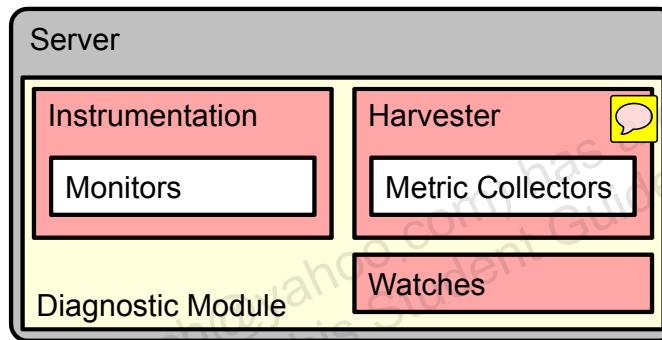
Log files are archived as human-readable files. Events and harvested data are archived in binary format, in a WebLogic persistent store or in a database.

1. In the left pane, expand Diagnostics and select Archives.
2. Click the name of the server for which you want to configure diagnostic archive settings.
3. Select one of the following archive types from the Type list:
 - Select File Store to persist data to the file system. If you choose this option, enter the directory in the Directory field.
 - Select JDBC to persist data to a database. If you choose this option, select the JDBC data source from the Data Source list. You must first configure JDBC connectivity to use this option.
4. Select or deselect Data Retirement Enabled to enable or disable data retirement for this server instance, respectively. In the Preferred Store Size field, enter a maximum data file size, in megabytes. When this size is exceeded, enough of the oldest records in the store are removed to reduce the size of the store below the maximum. In the Store Size Check Period field, enter the interval, in minutes, between the times when the store will be checked to see if it has exceeded the preferred store size.

Diagnostic Modules

A diagnostic module is used to contain the configuration of WLDF components and is used to target that configuration to servers and clusters.

- Built-in modules
- User-created system modules



ORACLE

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

To configure and use the Instrumentation, Harvester, and Watch and Notification components at the server level, you must first create a system resource called a “diagnostic system module.” System modules are globally available for targeting to servers and clusters configured in a domain. In WebLogic 12c, you can target multiple diagnostic system modules to any given server or cluster.

There are two types of diagnostic modules:

- **Built-in:** These are diagnostic modules that are part of the WebLogic product:
 - Production mode domains have a built-in module enabled by default.
 - Development mode domains have all built-in modules disabled by default.
 - Modules provide for low overhead historical server performance metrics.
 - There are three built-in modules:
 - **Low:** Captures the most important data from key WebLogic MBean attributes (default)
 - **Medium:** Captures additional attributes captured by Low and data from other MBeans
 - **High:** Captures more verbose data than Medium and from a larger number of MBeans
 - Built-in modules can be cloned to customize a new module based on a built-in module.

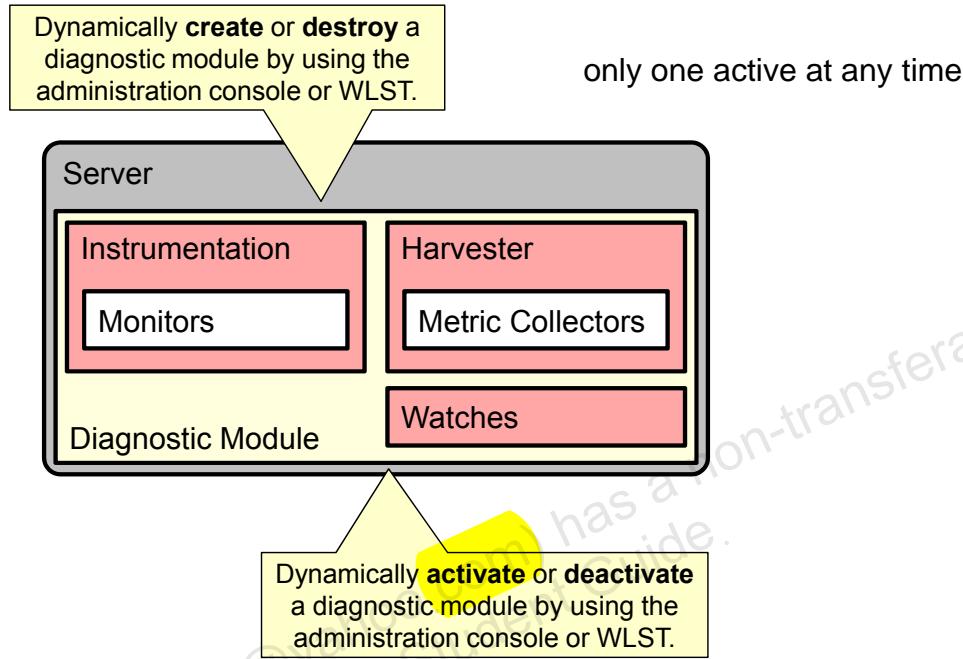
- **User created:** These are diagnostic modules that are created and deployed to WebLogic by using the administration console or WLST. The configuration contained within is configured by an administrator.

WLDF provides features for generating, gathering, analyzing, and persisting diagnostic data from WebLogic Server instances and from applications deployed to them. For server-scoped diagnostics, some WLDF features are configured as part of the configuration for a server in a domain. Other features are configured as system resource descriptors that can be targeted to servers (or clusters). For application-scoped diagnostics, diagnostic features are configured as resource descriptors for the application.

The harvester, watch, and instrumentation features are configured, packaged, and targeted as part of a diagnostic module, similar to a JMS module resource. Only instrumentation monitors can be defined in application-scoped modules, which are placed in the application's `weblogic-diagnostics.xml` file.

You create a diagnostic system module through the administration console or WLST. It is created as a `WLDFResourceBean`, and the configuration is persisted in a resource descriptor file called `<module>.xml`, where `<module>` is the name of the diagnostic module. The file is created by default in the domain's `config/diagnostics` directory with a `.xml` extension.

Dynamic Diagnostic Modules



ORACLE®

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

When a server hosting a dynamic diagnostic module shuts down, the module reverts to whatever is contained in the configuration when the server is restarted.

Resource Descriptors

A diagnostic system module has a corresponding resource descriptor:

```
<wldf-resource
    xmlns="http://xmlns.oracle.com/weblogic/weblogic-diagnostics"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.oracle.com/weblogic/weblogic-
        diagnostics/1.0/weblogic-diagnostics.xsd">
    <name>MyDiagnosticModule</name>
    <instrumentation>
        <!-- Configuration for harvesting metrics -->
    </instrumentation>
    <harvester>
        <!-- Configuration for harvesting metrics -->
    </harvester>
    <watch-notification>
        <!-- Configuration for watches and notifications -->
    </watch-notification>
</wldf-resource>
```

MyDiagnosticModule.xml

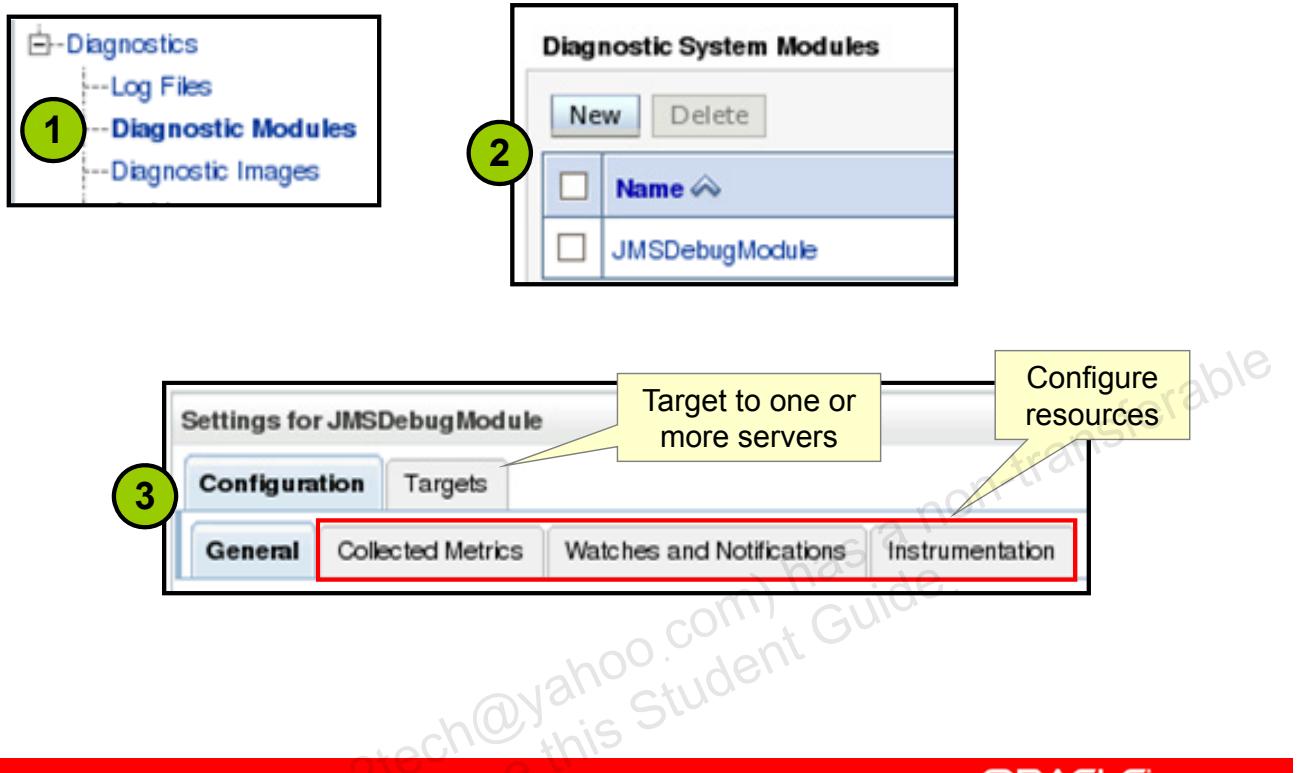


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A resource descriptor contains the actual configuration of WLDF components and features within an XML file. There are two types of resource descriptors:

- **Configured:**
 - Is part of the domain configuration stored in \$DOMAIN_HOME/config/diagnostics
 - Is referenced by the config.xml file
 - Can be deployed, activated, and deactivated dynamically
- **External:**
 - Is external to the domain configuration
 - Is not referenced by the config.xml file
 - Is only deployed, activated, and deactivated dynamically, and can be administered only by WLST
 - Resides in memory of the server until the server is shut down
 - Never gets persisted to the domain configuration

Creating a Diagnostic Module



ORACLE

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

Perform the following steps to create a diagnostic module:

1. In the left pane, expand Diagnostics and select Diagnostic Modules.
2. Click New. Enter a name for the module and, optionally, enter a description. Then click OK.
3. Use the various Configuration tabs to add diagnostic components to this module.
4. To target the module to a server or cluster, click the Targets tab.

WLST: Example

```
serverConfig> listSystemResourceControls()
External      Enabled          Name
-----
false         false           MyModule1
false         false           MyModule2

serverConfig> createSystemResourceControl('external-wldf', 'external-wldf.xml')
serverConfig> listSystemResourceControls()
External      Enabled          Name
-----
false         false           MyModule1
false         false           MyModule2
true          false           external-wldf

serverConfig> enableSystemResource('external-wldf')
serverConfig> listSystemResourceControls()
External      Enabled          Name
-----
false         false           MyModule1
false         false           MyModule2
true          true            external-wldf
```

Modules defined as part of the domain

Modules defined outside the domain in the external-wldf.xml resource descriptor

DiagnosticModuleExamples.py

WLST Commands for WLDF

Command	Description
listSystemResourceControls	Lists all available diagnostic system modules
enableSystemResource	Activates a diagnostic system module
disableSystemResource	Deactivates a diagnostic system module
createSystemResourceControl	Creates a diagnostics system module dynamically by using a specified descriptor file
destroySystemResourceControl	Destroys a previous dynamically created system module
dumpDiagnosticData	Dumps the diagnostics data from a harvester to a local file

ORACLE

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

Quiz

The WebLogic Diagnostic Framework (WLDF) enables administrators to analyze archived diagnostic data only in online mode. If the WebLogic cluster is down, no diagnostic data is available.

- a. True
- b. False



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

Answer: b

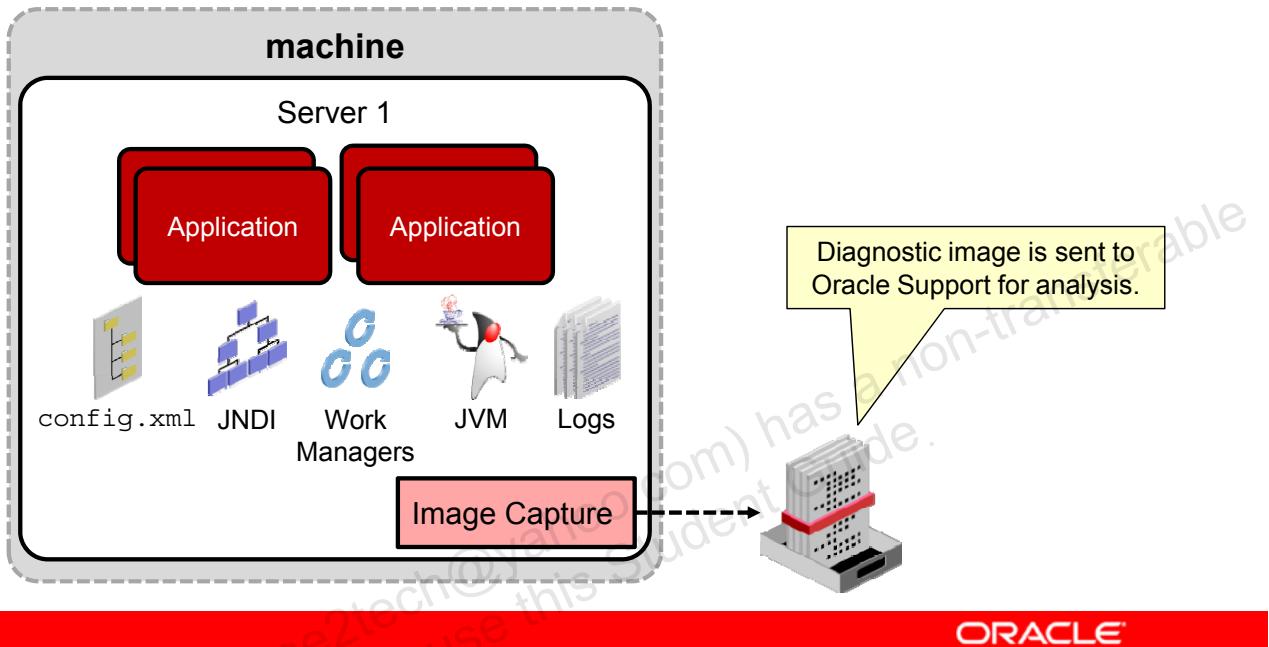
Archived data can also be accessed in offline mode by using WLST.

Agenda

- General Architecture
- Diagnostic Images
 - What Is a Diagnostic Image?
 - Capturing a Diagnostic Image
- Harvesters
- Watches and Notifications

What Is a Diagnostic Image?

A diagnostic image is a post-failure snapshot of the state of a running WebLogic Server instance:



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

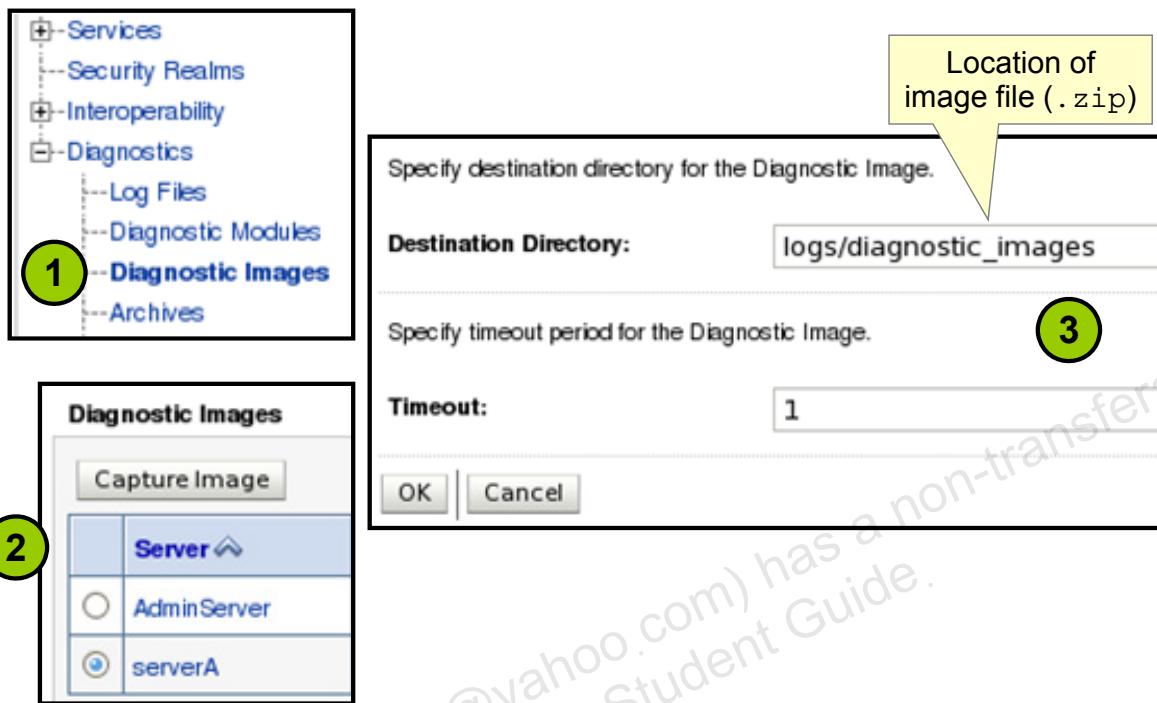
Diagnostic image capture support gathers the most common sources of the key server state used in diagnosing problems. It packages that state into a single artifact, which can be made available to support technicians.

The past state is often critical in diagnosing faults in a system. You use the diagnostic image capture component of WLDF to create a diagnostic snapshot, or dump, of a server's internal runtime state at the time of the capture. This information helps support personnel analyze the cause of a server failure. You can capture an image manually by using the console or WLST, or generate one automatically as part of a watch notification.

Because the diagnostic image capture is meant primarily as a post-failure analysis tool, there is little control over what information is captured. It includes the server's configuration, log cache, JVM state, work manager state, JNDI tree, and most recently harvested data. The image capture subsystem combines the data files produced by the different server subsystems into a single zip file.

If Java's Flight Recorder feature is enabled on your JVM, the diagnostic image will also contain the recording file, which you can view with Java Mission Control (JMC). If enabled, WebLogic can record its own events to this JVM file, which you can also browse with JMC.

Capturing a Server Diagnostic Image



ORACLE

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

Perform the following steps to capture a server image by using the console:

1. In the left pane, expand Diagnostics and select Diagnostic Images.
2. Select the server for which you want to capture a diagnostic image, and click Capture Image.
3. Enter a new directory in the Destination Directory field, or accept the default directory. If you change the directory for this image capture, it does not change the default directory for capturing images for this server when they are captured by other means, such as a watch notification. Then click OK.

WLST: Example

Capture a server diagnostic image:

```
serverRuntime()  
wldfCapture = getMBean('WLDFRuntime/WLDFRuntime/  
WLDFImageRuntime/Image')  
wldfCapture.captureImage('logs/diagnostic_images', 30)
```

CreateImage.py



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

Additional WLDF WLST examples can be found in the WLDF guide in the product documentation.

Refer to the following MBeans for more information about MBeans related to the diagnostic framework:

- WLDFResourceBean
- WLDFHarvesterBean (a subclass of WLDFResourceBean)
- WLDFHarvestedTypeBean (a component of WLDFHarvesterBean)
- WLDFInstrumentationBean (a subclass of WLDFResourceBean)
- WLDFInstrumentationMonitorBean (a component of WLDFInstrumentationBean)
- WLDFWatchNotificationBean (a subclass of WLDFResourceBean)
- WLDFWatchBean (a component of WLDFWatchNotificationBean)
- WLDFNotificationBean (abstract; subclasses exist for each notification type)
- WLDFRuntimeMBean
- WLDFImageRuntimeMBean (a component of WLDFRuntimeMBean)
- WLDFControlRuntimeMBean
- WLDFSystemResourceControlRuntimeMBean

Quiz

The MBean method used to obtain a post-failure snapshot of the state of a running WebLogic Server from a WLST script is:

- a. getSnapshot ()
- b. capturePic ()
- c. captureImage ()
- d. dumpState ()

ORACLE

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

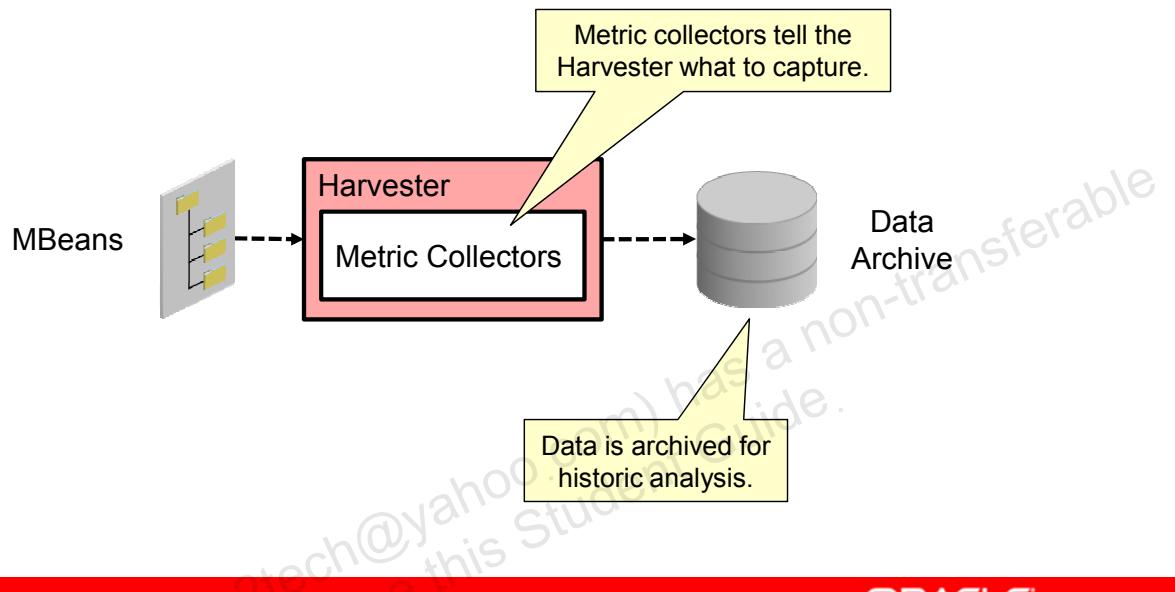
Answer: c

Agenda

- General Architecture
- Diagnostic Images
- Harvesters
 - What Is a Harvester?
 - Configuring Metric Collection
- Watches and Notifications

What Is a Harvester?

The Harvester is a WLDF component that you can configure to capture WebLogic runtime MBean data.

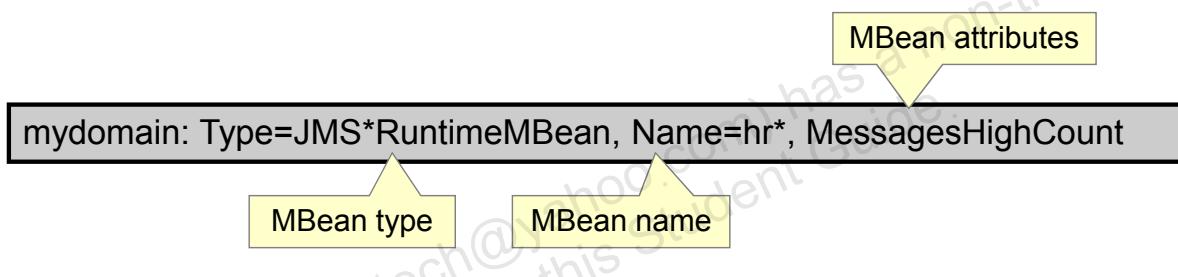


ORACLE

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

Metric Collectors

- A metric collector definition for the Harvester includes:
 - The runtime MBean type to query
 - The specific MBean instance names to query (all instances, by default)
 - The MBean attributes to collect (all attributes, by default)
 - How often to sample data
- Alternatively, use the MBean expression syntax, which also supports wildcards and complex attributes.



ORACLE

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

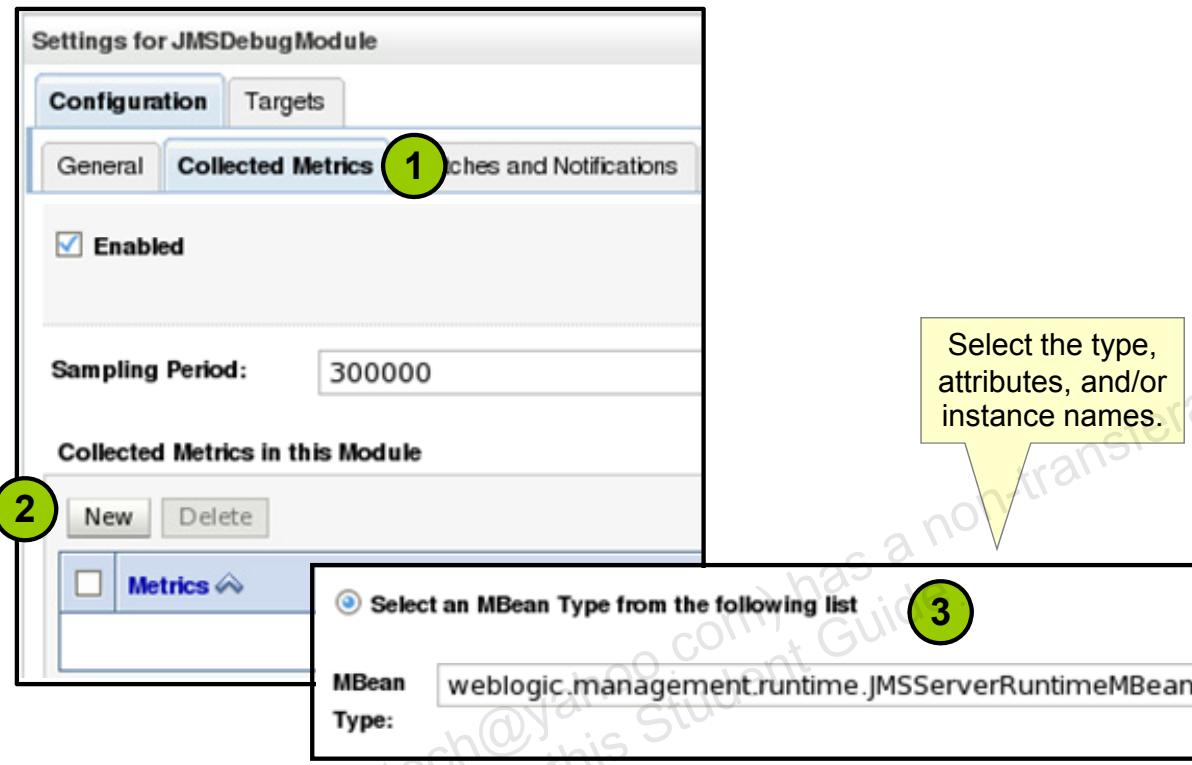
Harvesting metrics is the process of gathering data that is useful for monitoring the system state and performance. Metrics are exposed to WLDF as attributes on qualified MBeans. The Harvester gathers values from selected MBean attributes at a specified sampling rate. Therefore, you can track potentially fluctuating values over time. For custom MBeans, the MBean must be currently registered with the JMX server.

You can configure the Harvester to harvest data from named MBean types, instances, and attributes. If only a type is specified, data is collected from all attributes in all instances of the specified type. If only a type and attributes are specified, data is collected from all instances of the specified type.

The sample period specifies the time between each cycle. For example, if the Harvester begins execution at time T, and the sample period is I, the next harvest cycle begins at T+I. If a cycle takes A seconds to complete and if A exceeds I, then the next cycle begins at T+A. If this occurs, the Harvester tries to start the next cycle sooner, to ensure that the average interval is I.

WLDF allows for the use of wildcards (*) in type names, instance names, and attribute specifications. WLDF also supports nested attributes using a dot delimiter, as well as complex attributes such as arrays and maps. WLDF watch expressions also support similar capabilities.

Configuring a Metric Collector



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

ORACLE

Metrics are configured and collected in the scope of a diagnostic system module targeted to one or more server instances. Therefore, to collect metrics, you must first create a diagnostic system module.

1. Click the name of the module for which you want to configure metric collection. Then click Configuration > Collected Metrics.
2. To enable or disable all metric collection for this module, select or deselect the Enabled check box. To set the period between samples, enter the period (in milliseconds) in the Sampling Period field. To define a new collected metric, click New.
3. From the MBean Server Location drop-down list, select either DomainRuntime or ServerRuntime. Then click Next. Select an MBean that you want to monitor from the MBean Type list. Then click Next again.
4. In the Collected Attributes section, select one or more attributes from the Available list and move them to the Chosen list (default is all attributes). Click Next.
5. In the Collected Instances section, select one or more instances from the Available list and move them to the Chosen list (default is all instances). Click Finish.
6. Click Save.

WLST: Example

Create a diagnostic module and metric collector (harvester):

```
module = cmo.createWLDFSystemResource('JMSDebugModule')
module.addTarget(getMBean('/Servers/serverA'))
harvester = getMBean('/WLDFSystemResources/JMSDebugModule/
    WLDFResource/JMSDebugModule/Harvester/JMSDebugModule')
harvester.setSamplePeriod(300000)
harvester.setEnabled(true)
harvestType = harvester.createHarvestedType
    ('weblogic.management.runtime.JMSRuntimeMBean')
harvestType.setEnabled(true)
harvestType.setHarvestedAttributes(['MessagesHighCount'])
```

CreateWLDF.py



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

Additional WLDF WLST examples can be found in the WLDF guide in the product documentation.

Refer to the following MBeans for more information about MBeans related to the diagnostic framework:

- WLDFResourceBean
- WLDFHarvesterBean (a subclass of WLDFResourceBean)
- WLDFHarvestedTypeBean (a component of WLDFHarvesterBean)
- WLDFInstrumentationBean (a subclass of WLDFResourceBean)
- WLDFInstrumentationMonitorBean (a component of WLDFInstrumentationBean)
- WLDFWatchNotificationBean (a subclass of WLDFResourceBean)
- WLDFWatchBean (a component of WLDFWatchNotificationBean)
- WLDFNotificationBean (abstract; subclasses exist for each notification type)
- WLDFRuntimeMBean
- WLDFImageRuntimeMBean (a component of WLDFRuntimeMBean)

Quiz

The Harvester is a WLDF component that:

- a. Checkpoints performance metrics, received from the Collector, either to disk or database
- b. Captures WebLogic runtime MBean data
- c. Gathers statistics from the Oracle database
- d. Uses a two-phase commit protocol to synchronize the collection of performance metrics across the enterprise



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

Answer: b

Agenda

- General Architecture
- Diagnostic Images
- Harvesters
- Watches and Notifications

Watches and Notifications

A WLDF *watch*:

- Inspects data generated from metric collectors, events generated from monitors, or server log files
- Compares data to one or more conditions or “rules”
- Triggers one or more *notifications*

Available notification types include:

- JMS
- Email
- SNMP trap
- Diagnostic image capture



ORACLE

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

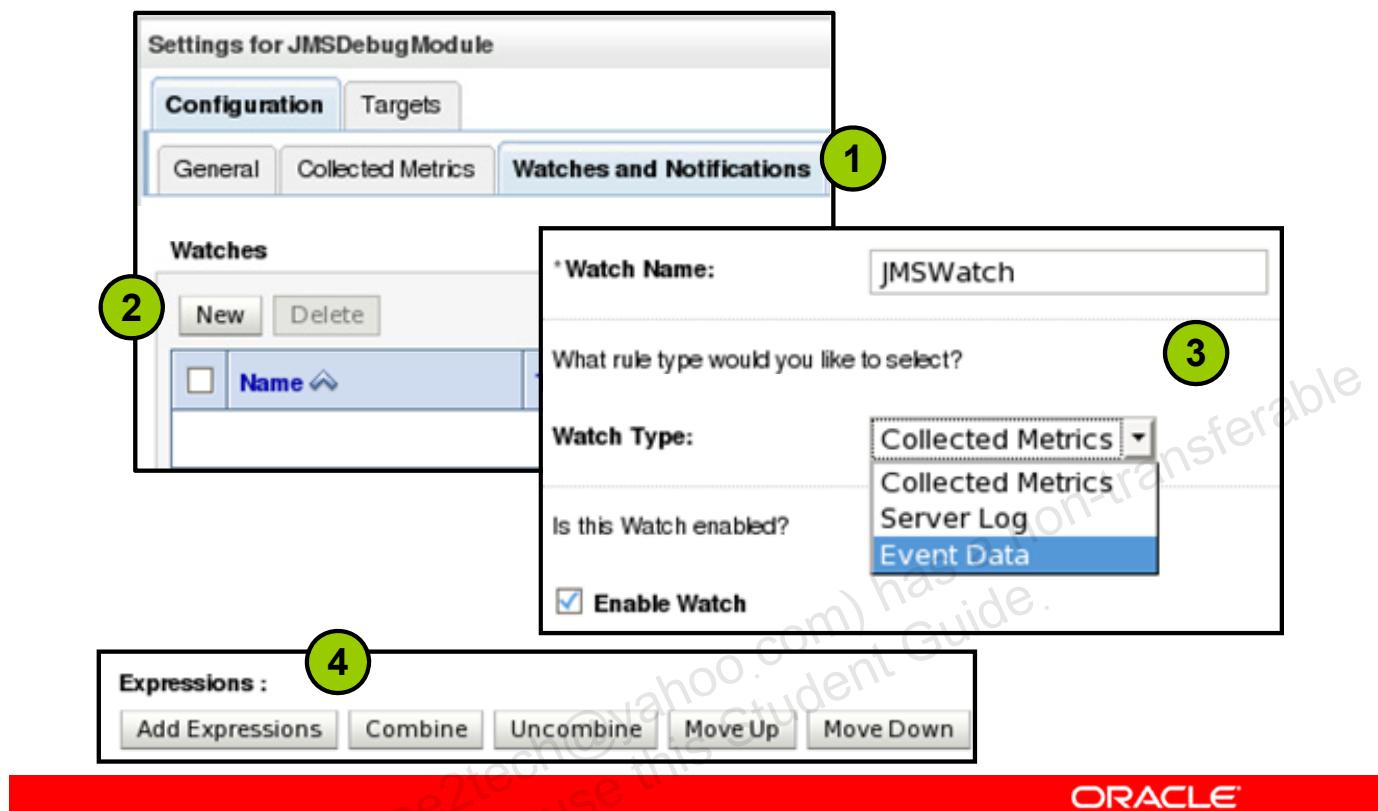
A watch identifies a situation that you want to trap for monitoring or diagnostic purposes. You can configure watches to analyze log records, data events, and harvested metrics. A watch is specified as a watch rule, which includes a rule expression, an alarm setting, and one or more notification handlers. A notification is an action that is taken when a watch rule expression evaluates to true. You must associate a watch with a notification for a useful diagnostic activity to occur, for example, to notify an administrator about specified states or activities in a running server.

Log and instrumentation watches are triggered in real time, whereas harvester watches are triggered only after the current harvest cycle completes.

Watches and notifications are configured separately from each other. A notification can be associated with multiple watches, and a watch can be associated with multiple notifications. This provides the flexibility to recombine and reuse watches and notifications, according to current needs. Each watch and notification can be individually enabled and disabled as well.

A complete watch and notification configuration includes settings for one or more watches, one or more notifications, and any underlying configurations required for the notification media, for example, the SNMP configuration required for an SNMP-based notification.

Configuring a Watch



ORACLE

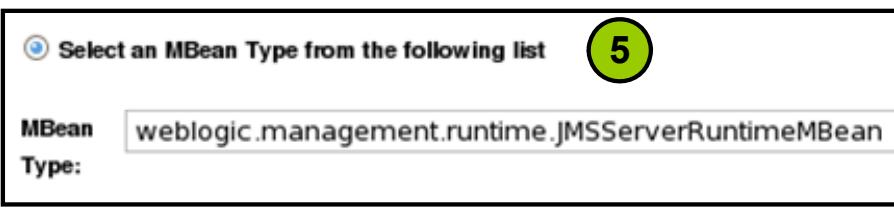
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A collected metric watch can monitor any runtime MBean in the local runtime MBean server. Log watches monitor the occurrence of specific messages and/or strings in the server log. Watches contain a severity value that is passed through to the recipients of notifications. Instrumentation watches are triggered as a result of the event being posted that matches some criteria.

The example in the slide above depicts a collected metric watch:

1. Click the name of the module for which you want to create a watch. Then click Configuration > Watches and Notifications.
2. In the Watches section, click New.
3. Enter a name for the watch in the Watch Name field. Then select a Watch Type. To enable or disable the watch, select or deselect the Enable Watch check box. Then click Next.
4. Click the Add Expressions button to construct one or more watch expressions. Expressions can be entered manually by using the WLDF expression language or constructed graphically. To group two or more expressions, select the check boxes adjacent to the expressions that you want to group and click Combine. To reorder expressions, select the check boxes adjacent to the expressions that you want to move and click Move Up or Move Down.

Configuring a Watch

5 

6
What is the value of your expression?

7
Notifications:
Available Chosen
EmailToAdmin

ORACLE

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

5. For each watch expression, select either DomainRuntime or ServerRuntime from the MBean Server Location drop-down list. Then click Next. Select an instance from the Instance drop-down list and click Next again.
6. Select an attribute from the Message Attribute list and an operator from the Operator list, and enter a value with which to compare the attribute by using the Value field. The value must be an appropriate value for the attribute chosen above. Click Next.
7. Select and move one or more existing Notifications from the Available list to the Chosen list. Then click Finish.

Quiz

A WLDF watch identifies:

- a. A specific harvester collecting performance metrics
- b. A situation that one wants to trap for monitoring or diagnostic purposes
- c. A notification mechanism used to alert administrators in case of sub-optimal performance
- d. A set of database tables or tables containing performance metrics



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

Answer: b

Summary

In this lesson, you should have learned how to configure the WebLogic Diagnostic Framework (WLDF) to monitor a domain.

Practice 15-1 Overview: Using a Built-in Diagnostic Module

This practice covers the following topics:

- Configuring a built-in diagnostic module
- Changing the settings of a built-in module
- Cloning and customizing a built-in module

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 and Coherence Integration

16

ORACLE

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

Objectives

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

- Coherence*Web
- Managed Coherence Servers

Agenda

- Coherence Overview
- Coherence*Web Session Replication
- Managed Coherence Servers

Oracle Coherence: Overview

Coherence:

- Provides a distributed, in-memory caching solution for Java
- Is based on a *grid* of cache servers or *nodes*
- Automatically distributes or partitions cached data across the grid based on the number of servers
- Implements replication for high availability
- Includes a decentralized management and monitoring model based on JMX
- Supports a proxy architecture to provide connectivity beyond the Coherence cluster
- Supports additional languages such as .NET and C++



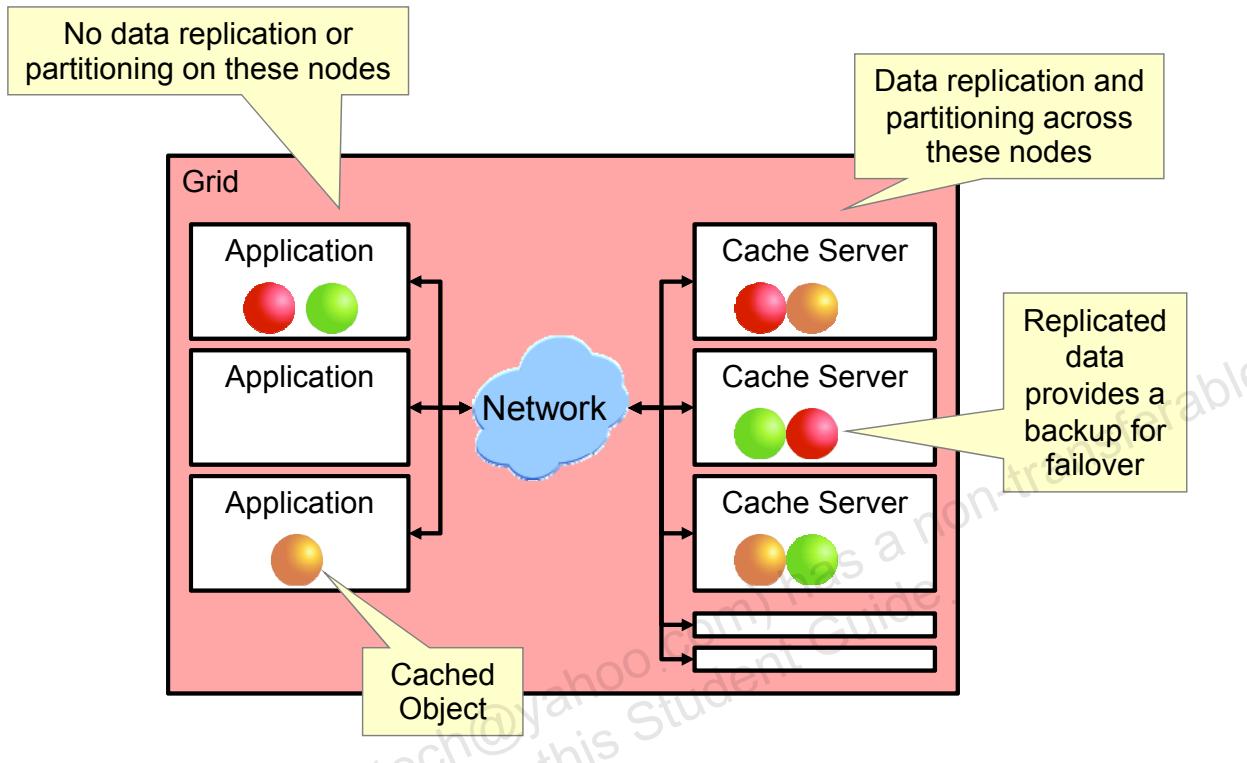
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

One of the primary uses of Oracle Coherence is to cluster an application's objects and data. In the simplest sense, this means that all the objects and data that an application delegates to Coherence are automatically available to and accessible by all servers in the application cluster. None of the objects or data will be lost in the event of server failure. By clustering the application's objects and data, Coherence solves many of the difficult problems related to achieving availability, reliability, scalability, performance, serviceability, and manageability of clustered applications.

Oracle Coherence is a JCache-compliant, in-memory caching and data management solution for clustered Java EE applications and application servers. Coherence makes sharing and managing data in a cluster as simple as on a single server. It accomplishes this by coordinating updates to the cached data by using clusterwide concurrency control, replicating and distributing data modifications across the cluster, and delivering notifications of data modifications to any servers that request them. Developers can easily take advantage of Coherence features by using the standard Java Collections API to access and modify data, and by using the standard JavaBean event model to receive data change notifications.

Coherence provides a clusterwide view of management information through the standard JMX API, so that the entire cluster can be managed from a single server. The information provided includes cache sizes along with hit and miss rates.

Oracle Coherence: Overview



ORACLE

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

Partitioning refers to the ability of Coherence to load-balance data storage, access, and management across all the servers in the cluster. For example, when using Coherence data partitioning, if there are four servers in a cluster, each will manage 25% of the data. And if another server is added, each server will dynamically adjust so that the five servers will manage 20% of the data. This data load balancing will occur without any application interruption and without any lost data or operations. Similarly, if one of those five servers were to fail, each of the remaining four servers would readjust to managing 25% of the data. Once again, there is no data loss, including the 20% of the data that was being managed on the failed server.

While the partitioning feature dynamically load balances data evenly across the entire server cluster, replication ensures that a desired set of data is always available and up-to-date at all times in the cluster. Replication allows operations 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, and in most applications the data access rate far outweighs the data modification rate. To eliminate the latency associated with partitioned data access, Coherence can employ "near caching." Frequently and recently used data from the partitioned cache are maintained on the specific servers that are accessing that data within a near cache, and this local data is kept up-to-date by using event-based invalidation.

Agenda

- Coherence Overview
- Coherence*Web Session Replication
 - Types of Session Persistence
 - Coherence*Web
 - Coherence*Web and WebLogic Clusters
 - Coherence*Web Session Failover
 - Configuring Coherence*Web in WebLogic
- Managed Coherence Servers

Types of Session Persistence

WebLogic Server supports several strategies so that the session information is not lost when a server fails:

- In-memory session persistence
- File session persistence
- JDBC (database) session persistence
- Coherence*Web session persistence



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 an `HttpSession`. An `HttpSession` is available on a per-client basis. After 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 persisted, shared access to the information in each `HttpSession` object must be provided. This is accomplished in WebLogic Server by using in-memory persistence, file system persistence, database persistence, or Coherence*Web persistence. Each of these options is configured in the web application's WebLogic deployment descriptor, `weblogic.xml`. Each option has its own configurable parameters that are also entered in `weblogic.xml`.

Note that in-memory and JDBC persistence have two options: synchronous and asynchronous. The asynchronous option persists data in batches to improve cluster performance.

Coherence*Web

- Is a plug-in that enhances WebLogic Server's session management implementation with a Coherence grid
- Provides a highly scalable session replication solution that can span WebLogic domains and clusters
- Reduces WebLogic Server's memory footprint
- Requires no changes to existing application code
- Allows session data to be shared across applications
- Can be enabled on an application basis



ORACLE

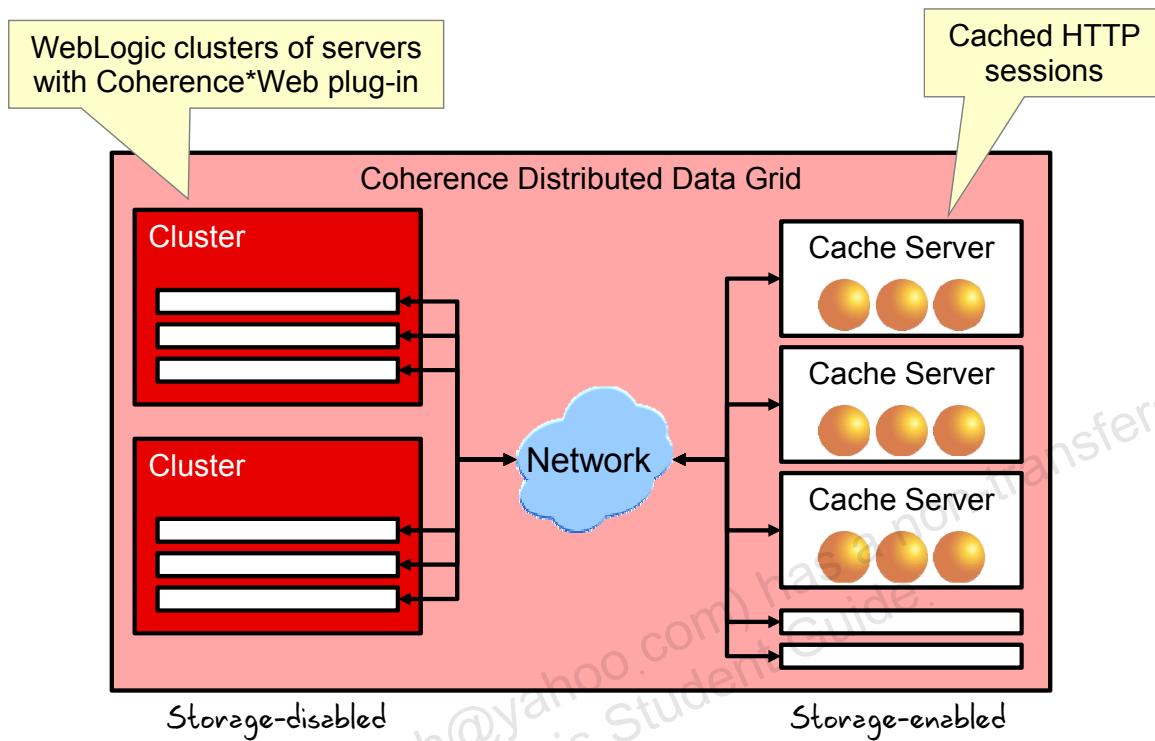
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Coherence*Web is an application server plug-in dedicated to managing session state in clustered environments. It brings the scalability, availability, reliability, and performance characteristics of a Coherence data grid to in-memory session management and storage. Also, because it is not constrained by the deployment topologies of the application server, it enables session sharing and management across different web applications, domains, and even different application server products. Sometimes you may want to explicitly prevent session data from being shared by different Java EE applications that participate in the same Coherence cluster, so Coherence*Web supports this approach as well.

If sessions are shared across web applications, you may want to scope individual session attributes so that they are either globally visible (that is, all web applications can see and modify these attributes) or scoped to an individual web application (that is, not visible to any instance of another application). The latter approach may be desired to help avoid namespace collisions when multiple applications use the same session attribute names.

With Coherence*Web, session data is stored outside of the application server, thereby freeing server heap space. This architecture also allows you to individually tune and scale the sizes of your application server clusters and session data grids.

Coherence*Web and WebLogic Clusters



ORACLE

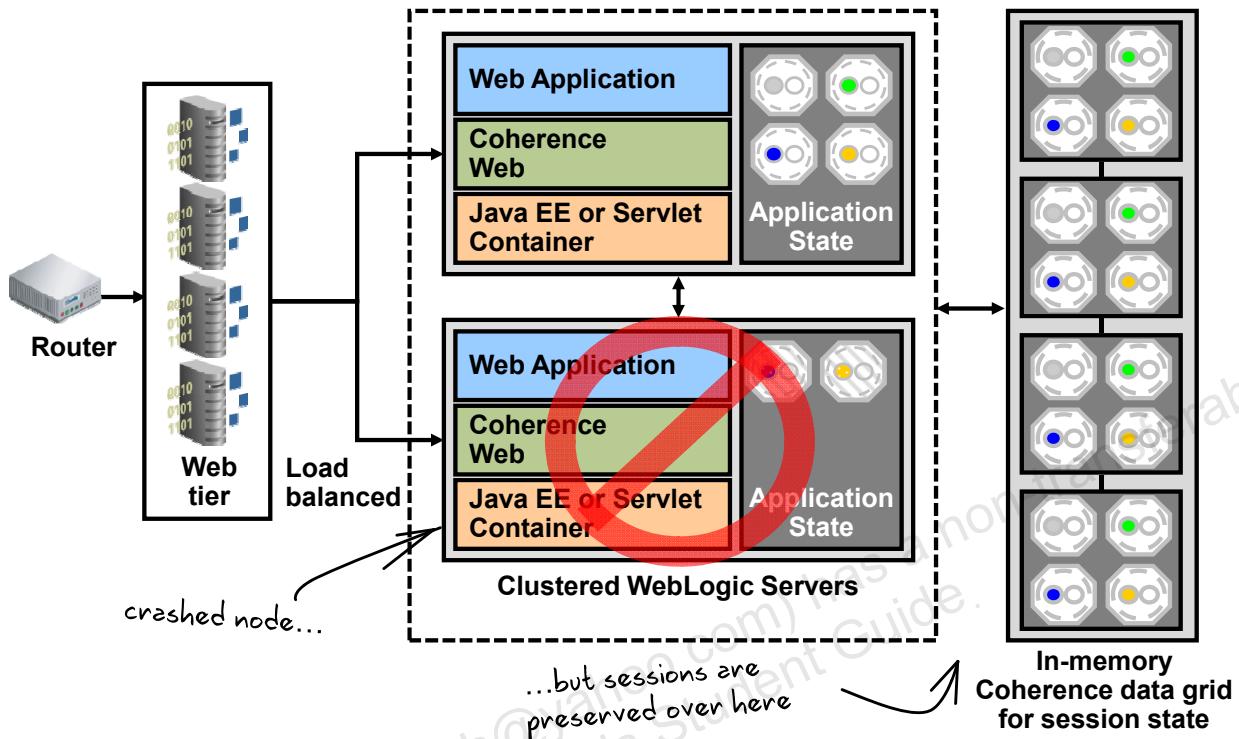
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Coherence*Web is not a replacement for WebLogic Server's in-memory HTTP state replication services. However, Coherence*Web should be considered when an application has large HTTP session state objects, when running into memory constraints due to storing HTTP session object data, or if you have an existing Coherence cluster and want to offload HTTP Session storage to a Coherence cluster.

Coherence*Web is configured with local storage disabled. This means that although the WebLogic server instance is a member of the Coherence cluster, it is not a data storage member of the cluster. Storage-disabled members rely on other Coherence cluster members to manage and replicate data.

By default, Coherence*Web creates a single HTTP session across all web applications for each client and scopes the session attributes to each web application. This means that if a session is invalidated in one web application, that same session is invalidated for all web applications in WebLogic Server that are using Coherence*Web. If you do not want this behavior, a potential work-around is to edit the `<cookie-path>` element of the `weblogic.xml` descriptor.

Coherence*Web Session Failover



ORACLE

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

The diagram in the slide illustrates that if a server were to fail, if you lose the entire application server or a JVM, Coherence still maintains the data. Therefore, you do not lose any session state information. Using your existing infrastructure, the user session fails over to one of the other application servers, or the one that is surviving. Therefore, no data is lost. The end user may not even notice that there is an outage. The user session continues without the user even noticing that anything went wrong because Coherence manages the data.

One common use case for Coherence clustering is to manage user sessions (conversational state) in the cluster. This capability is provided by the Coherence*Web module, which is a built-in feature of Oracle Coherence. Coherence*Web provides linear scalability for HTTP Session Management in clusters of hundreds of production servers. It can achieve this linear scalability because, at its core, it is built on Oracle Coherence dynamic partitioning.

Configuring Coherence*Web in WebLogic

1. Start one or more Coherence cache servers on your network.
2. Update each application's `weblogic.xml` descriptor file and set the session persistence type to `coherence-web`.

```
<weblogic-web-app>
  <session-descriptor>
    <persistence-store-type>
      coherence-web
    </persistence-store-type>
  </session-descriptor>
</weblogic-web-app>
```

weblogic.xml

ORACLE

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

The WebLogic Server Coherence*Web plug-in consists of the `coherence-web.jar` and `coherence.jar` files, located in the `coherence/lib` directory in the Coherence distribution. When WebLogic is installed, these files are included in the WebLogic system classpath. The `coherence-web.jar` file will load application classes from the appropriate WebLogic classloader so you do not have to include Coherence JAR files in the web application's classpath. All Coherence*Web-enabled applications running on WebLogic 12c (12.1.2) have application server scope. As a result, all applications become part of the same Coherence node.

The Coherence caches used by Coherence*Web are configured by the `default-session-cache-config.xml` file included in the `coherence-web.jar` file. Although the default configuration should be sufficient for most deployments, it can be overridden by placing a `session-cache-config.xml` file in your web application's `WEB-INF/classes` folder. If you are using the default session cache configuration file, you can package and deploy your applications just like any other Java EE application. However, if you are using a custom session cache configuration file, you must package and deploy your application in a GAR file.

Note: GAR files are covered later in this lesson.

Quiz

What is the primary purpose of Coherence*Web?

- a. To use Coherence caching for application data
- b. To integrate Coherence with application servers
- c. To use Coherence caching for HTTP sessions
- d. To use Coherence caching for client connections



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

Answer: c

Practice 16-1 Overview: Configuring Coherence*Web

This practice covers the following topics:

- Configuring the default Coherence cluster in WebLogic to use multicast networking
- Configuring Coherence stand-alone cache servers to use the same cluster settings as the WebLogic default Coherence cluster
- Starting Coherence cache servers
- Deploying the ShoppingCart application to WebLogic
- Testing Coherence*Web HTTP sessions



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

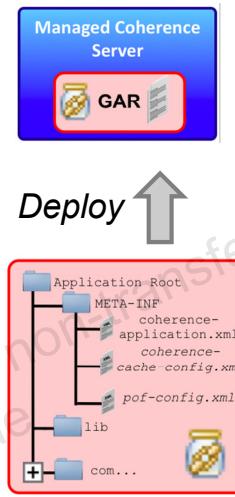
Agenda

- Coherence*Web Session Replication
- Managed Coherence Servers
 - What Are Managed Coherence Servers?
 - What Are Coherence Container Applications?
 - What Are the Benefits of the Coherence Container
 - Installing WebLogic with Coherence Support
 - Coherence Grid Archives
 - Coherence Cluster

Coherence and WebLogic Server

Oracle WebLogic 12c provides comprehensive support for Coherence including:

- **Managed Coherence servers:** The ability to configure, start, stop, and otherwise manage Coherence servers natively within a WebLogic environment
- **Coherence Grid Archives (GARs):** A Java EE-like artifact that encapsulates all the elements of a Coherence application into a single deployable unit



ORACLE

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

WebLogic 12c provides two new features for Coherence.

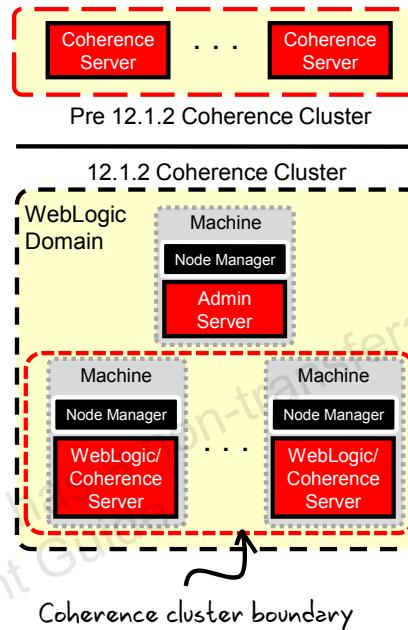
- **Managed Coherence servers:** With earlier versions of WebLogic Server, developers could manage Coherence by using WebLogic node manager. However, the process was often difficult and error-prone, and ultimately took advantage of the management capabilities of only a WLS cluster to manage Coherence instances. Coherence was still a stand-alone process. With WebLogic Server 12c, Coherence is fully integrated into WebLogic as a first-class member of the cluster, with the ability to take advantage of all aspects of the Java EE as well as all aspects of WebLogic Server management.
- **Coherence Grid Archives (GARs):** Additionally, Coherence applications are now first-class citizens of a WebLogic Server cluster. Coherence artifacts, including all configuration and deployment information, are packaged into a single file, known as a GAR. A GAR contains everything required to deploy and run Coherence applications. To make this happen, WebLogic now supports an additional container, often referred to as the Coherence Container. The Coherence Container provides all the same services as an EJB or similar container to Coherence applications. Lifecycle, injection, startup, shutdown, class management, isolation, and similar capabilities are provided for Coherence.

WebLogic Managed Coherence Servers

Operations

The WebLogic Server Container for Oracle Coherence:

- Is a simplified model for managing operational configuration
- Is conceptually similar to other Java EE containers
- Is fully integrated and installed with WebLogic and includes Coherence
- Is managed via the configuration wizard, WebLogic console, WLST, JMX, and Fusion Middleware Control
More on what a GAR is later
- Manages Grid Archives (GARs)



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

Traditionally, Coherence has been deployed as a JAR, incorporated into a Java application along with a tier of stand-alone cache server JVMs. For example, the use of embedded Coherence from within a WAR, was referred to as clients and the stand-alone cache servers were referred to as servers. The life cycles of the “clients” and cache servers were managed separately, often manually. Application development and deployment in this model was a complicated process involving many moving parts that required custom management processes.

The WebLogic Server Coherence Container for applications provide tight integration between WebLogic Server and Coherence and eliminate the difficulty of managing Coherence instances and applications in a WebLogic environment. This integration allows for simplified and streamlined development and management environments of distributed applications. The Coherence Container allows end users to build a new archive type (GAR) that can be deployed and managed via standard WebLogic Server methods. Using the Coherence Container:

- Developers can now streamline their build processes to Coherence applications
- Operations departments can now standardize deployment of Coherence instances and Coherence servers

What Is a Grid Archive (GAR)?

Coherence applications or GARs:

- Are structurally similar to enterprise/web applications, but are packaged as a Grid Archive with the extension .gar
- Define the elements required for deployment to the WebLogic Coherence Container
- Are composed of:
 - A deployment descriptor, coherence-application.xml
 - A cache configuration, named in the deployment descriptor
 - Java classes including entry processors, event handlers, filters, application classes, and others
 - Optional support libraries

Defines application name, cache configuration file name, POF configuration, lifecycle handler, and a variety of other information



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

Coherence applications are packaged into Grid Archive or GAR files. In general, a GAR contains all the artifacts required to service a particular cache or invocation request and mirrors, to a high degree, the structure of other Java EE artifacts, such as web archives. Specifically, a GAR contains:

- A deployment descriptor, coherence-application.xml, which defines the location of a cache configuration as well as a variety of other content such as POF configuration. The deployment descriptor may include references to several other optional elements, such as lifecycle handlers.
- A cache configuration, referenced from the deployment descriptor
- Java classes representing the application. These classes typically include the entities and other application classes, POF support, entry processors, event handlers, and similar classes.
- A set of optional library JAR files

The entire set of classes is encapsulated into a JAR known as a Coherence Grid Archive, which mirrors similar constructs for web applications, enterprise applications, and JAR files in general.

GAR files are also deployable directly to stand-alone Coherence (without WebLogic present at all), reinforcing their JAR-like structure.

Coherence Application Deployment on WebLogic

Coherence applications can be deployed in several ways:

- **Stand-alone:** GAR is deployed to a server by itself. It is analogous to the traditional default cache server.
- **Java EE:** The application is included as part of:
 - Web Application (WAR): GAR resources available to WAR only
 - Enterprise Application (EAR): GAR resources available to WAR and other Java EE resources, such as EJBs
- **Shared library:** GAR resources are available to all WARs and EARs located on the server.



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

Coherence applications can be deployed in several ways. Deployment models include:

- **Stand-alone:** A stand-alone deployed Coherence application exposes caches to applications, but does not expose resources directly to WAR or EAR files. Such a deployment is most common when configuring a data tier.
- **Java EE:** GAR files can be deployed as components of a WAR or an EAR file. Such deployments make Coherence resources available to the WAR and/or EAR directly. EJBs, web applications, JSPs, and other Java EE artifacts can use dependency injection to access a cache resource. Such a deployment scopes the caches to the EAR itself. An identical deployment, to a different EAR, would result in a similar set of caches, but scoped to the EAR/WAR.
- **Shared library:** With a shared library deployment, similar to a stand-alone deployment, all WAR and EAR artifacts can access the cache resources directly.

Coherence Container: Benefits

- Simplified installation: Coherence and WebLogic installed together
 - Simplified operations management:
 - Configure, manage, and deploy Coherence applications from the WebLogic administration console and WLST.
 - Manage Coherence resources centrally.
-

WebLogic to Coherence	Coherence benefits to WebLogic
<ul style="list-style-type: none"> • Instance management • Dependency injection • Application lifecycle management • Scripting support (deployment etc) 	<ul style="list-style-type: none"> • Simplified application packaging • Multiple deployment models <ul style="list-style-type: none"> – Stand-alone (GAR) – GAR within web application – GAR within Enterprise application – Shared library



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

With WebLogic Server 12c, Coherence is provided and installed with WebLogic Server. Both WebLogic Server and Coherence provide services to each other.

WebLogic Server provides several services to Coherence, including:

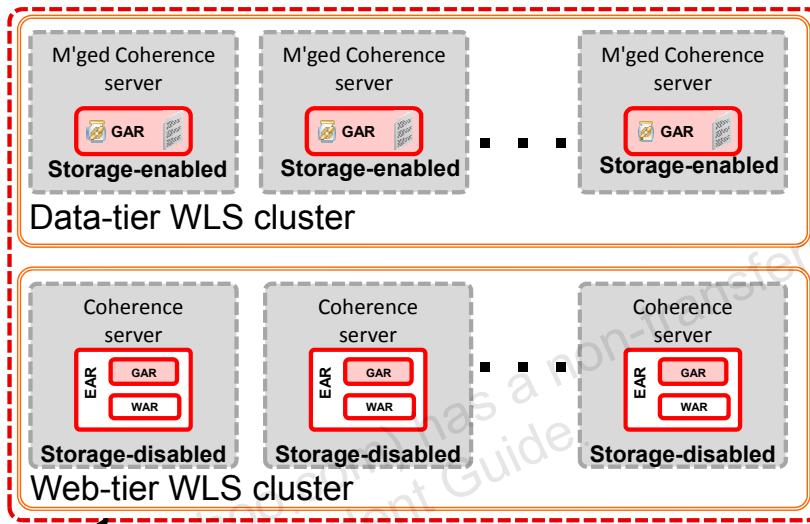
- Coherence, separate from WebLogic, cluster, member, and application management and scoping. Applications can be scoped to be stand-alone, WAR, EAR, or shared library scoped. Coherence also provides a cache-level scoping mechanism much like namespaces for caches and data.
- **Packaging:** Available on the system classpath, simplifying packaging
- **Management:** Coherence is now a first-class member of the cluster and fully supported by the console and WLST.
- **Instance support:** Full support for creating and managing Coherence instances
- **Application support:** Full support for application development and runtime life cycle
- **Cluster support:** Full support for defining and managing Coherence Clusters

Coherence Cluster

Coherence clusters consist of multiple, managed Coherence server instances.

- Managed Coherence server can:
- Be storage enabled or disabled
 - Share configuration and overrides
 - Have one or more deployed Coherence applications

Coherence Cluster spans multiple WLS clusters.



ORACLE

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

Coherence clusters are a mechanism for logically grouping sets of managed Coherence servers. In general, Coherence clusters consist of multiple managed Coherence servers, which together form a distributed data grid. Coherence clusters are different from WebLogic Clusters in several ways. First and foremost Coherence clusters often span multiple WebLogic clusters and are intended to represent Coherence servers, which logically represent a single data grid. Additionally, Coherence clusters use different clustering protocols, ports, and are configured separately from WebLogic Clusters. Managed Coherence servers can be associated with a Coherence cluster or with a WebLogic Cluster, which itself is associated with a Coherence cluster.

Typically, a Coherence cluster is associated with one or more WebLogic Clusters, which together represent a tiered architecture, typically having data, web, and proxy tiers. In the sample shown, two WebLogic Clusters, a data-tier cluster and a web-tier cluster, are defined. These two WebLogic Clusters are bound together into a single Coherence cluster. Typically, such a cluster shares a set of Coherence configurations and applications. In the example shown, the same Coherence GAR is deployed to each Coherence managed server, but packaged differently. On the data tier, the application is packaged as a simple GAR and deployed stand-alone. On the web tier, the same GAR is packaged as a component of an EAR and accessible from any other Java EE application within that EAR.

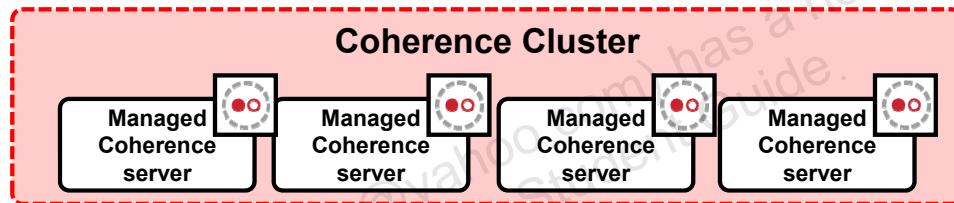
Managed Coherence Server

A domain can have zero or more managed Coherence servers.

A managed Coherence server:

- Is a WebLogic managed server that encapsulates a Coherence instance
- Can be storage enabled or disabled
- Can host Coherence applications
- Is normally a member of a Coherence Cluster

May be a member of
the default
Coherence cluster



ORACLE

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

Managed Coherence servers are server instances that are associated with a Coherence cluster. These servers work together and collectively form a Coherence cluster. Members of a Coherence cluster are historically referred to as Coherence instances, and are referred to as *managed Coherence servers* within a WebLogic installation. However, managed Coherence servers are *not* the same as Coherence servers. Managed Coherence servers are specific to the WebLogic Server Coherence Container integration. The use of Coherence servers, a resource defined in the previous ActiveCache integration, is deprecated. Managed Coherence servers conceptually are no different from earlier Coherence server instances running stand-alone, but there are differences. Some of the differences include:

- Managed Coherence servers that are managed within a WebLogic Server domain must not join an external Coherence cluster comprised of stand-alone JVM cluster members.
- Stand-alone JVM cluster members cannot be managed within a WebLogic Server domain.
- The administration server is typically not used as a managed Coherence server in a production environment.

Managed Coherence servers are distinguished by their role in the cluster, and can be:

- **Storage enabled:** A managed Coherence server that is responsible for storing data in the cluster. Coherence applications are packaged as Grid Archives (GARs) and are deployed on storage-enabled managed Coherence servers.

- **Storage disabled:** A managed Coherence server that is not responsible for storing data and is used to host Coherence applications (cache clients). A Coherence application GAR is packaged within an EAR and deployed on storage-disabled managed Coherence servers.
- **Proxy:** A managed Coherence server that is storage disabled and allows external clients (non-cluster members) to use a cache. A Coherence application GAR is deployed on managed Coherence proxy servers.

Quiz

Which two of the following are supported by GAR deployments?

- a. GAR files do not support deployment
- b. Deployment to standalone Coherence without WebLogic
- c. Deployment to a WebLogic server within an EAR file
- d. None of the above



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

Answer: b, c

Summary

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

- Coherence*Web
- Managed Coherence Servers

Practice 16-2 Overview: Configuring Managed Coherence Servers

This practice covers the following topics:

- Configuring a new Coherence cluster
- Deploying a stand-alone GAR application
- Deploying an EAR application with an embedded GAR
- Configuring Coherence cluster settings
- Testing the Coherence 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.

Tuning HotSpot JVM



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Describe JVM performance characteristics
- Describe JVM garbage collection algorithms
- Monitor and tune JVM garbage collection
- Use HotSpot ergonomics
- Tune JVM heap settings
- Use command-line JVM monitoring tools
- Use GUI JVM monitoring tools



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Assumptions and Expectations

This lesson makes the following assumptions:

- Students already know what Java is.
- Students already know what a JVM is and have a basic understanding of its architecture.
- Students are already familiar with basic concepts of Java garbage collection.
- *This lesson provides an overview of JVM tuning and how it relates to WebLogic Server. Students should refer to the three-day JVM Tuning course provided by Oracle University for complete details.*



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Agenda

- JVM Performance: Overview
- The JVM and Java Garbage Collection
- Command-Line JVM Tools
- GUI JVM Tools

What Is Performance?

This table shows the different aspects of JVM performance:

Aspect	Description
Memory Footprint	The amount of memory used by an application and the JVM
Startup Time	The time taken for an application to start
Scalability	How an application performs as the load and resources increase
Responsiveness	How quickly an application responds with requested data
Throughput	The amount of work an application can perform in a specific period of time



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

There are several ways in which you can define performance. Some of these aspects of performance impact the JVM.

Each of these aspects should be a well-defined requirement for an application. In addition, the importance should be prioritized for the application. This clarification is an important input to the folks who are tackling the application's performance issues.

Memory Footprint: It is very important to know the environment and ecosystem in which your application runs. Shared resources can have a significant impact on performance.

Virtual memory swapping should be minimized for any Java application. Consider a scenario where a garbage collection occurs with a large portion of the Java heap in virtual memory. Scanning for referenced objects would take much longer than when the heap is in physical memory. It is very important to configure a system and the Java heap to avoid virtual memory swapping. Questions to ask yourself include:

- Does the application run on a dedicated system?
- Does the application share the machine with other applications and JVMs?

Startup Time: Startup time is very important for client applications that tend to get started over and over again. Server applications do not place the same importance on fast startup time because they are expected to run for longer periods of time after starting up initially. However, when servers are restarted frequently during the iterative development process, fast startup time is important.

Scalability: Scalability can apply to not only an application but also to many other aspects of a system of which the application is a part.

Tip: It is very important to have a development or qualification environment that replicates production environment situations. This reduces the chance to be caught off-guard with an application that does not scale well.

Responsiveness: Responsiveness can be measured in a number of ways. But the bottom line is that a responsive application returns requested data quickly. Client applications are typically more focused on responsiveness. Long pause time are not acceptable.

Throughput: Server applications typically focus more on throughput and less on responsiveness. High pause times are more acceptable, but still avoided if at all possible.

Performance Focus for This Lesson

This lesson focuses on:

- Java application performance
- Optimizing the JVM for throughput or responsiveness
- Discovering, troubleshooting, and tuning Java performance issues



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

HotSpot JVM Monitoring Tools

Management tools provided with HotSpot JVM include:

Tool	Description
VisualVM, jconsole, Mission Control, and Flight Recorder	JMX-compliant JVM monitoring and management consoles
jps	JVM process status tool
jcmd	JVM process status and management tool
jmap and jhat	Tools that dump and display heap and shared object memory map
jstack	Tool that prints Java stack traces of Java threads



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

You can use the tools listed in the slide to monitor JVM performance and resource consumption. You can also use some of the tools for troubleshooting. Java VisualVM, jconsole, Mission Control, and Flight Recorder are GUI-based tools that require X-Windows in Linux and UNIX environments. All of the other tools are command line-based tools.

Section Summary

In this section, you should have learned how to describe JVM performance characteristics.

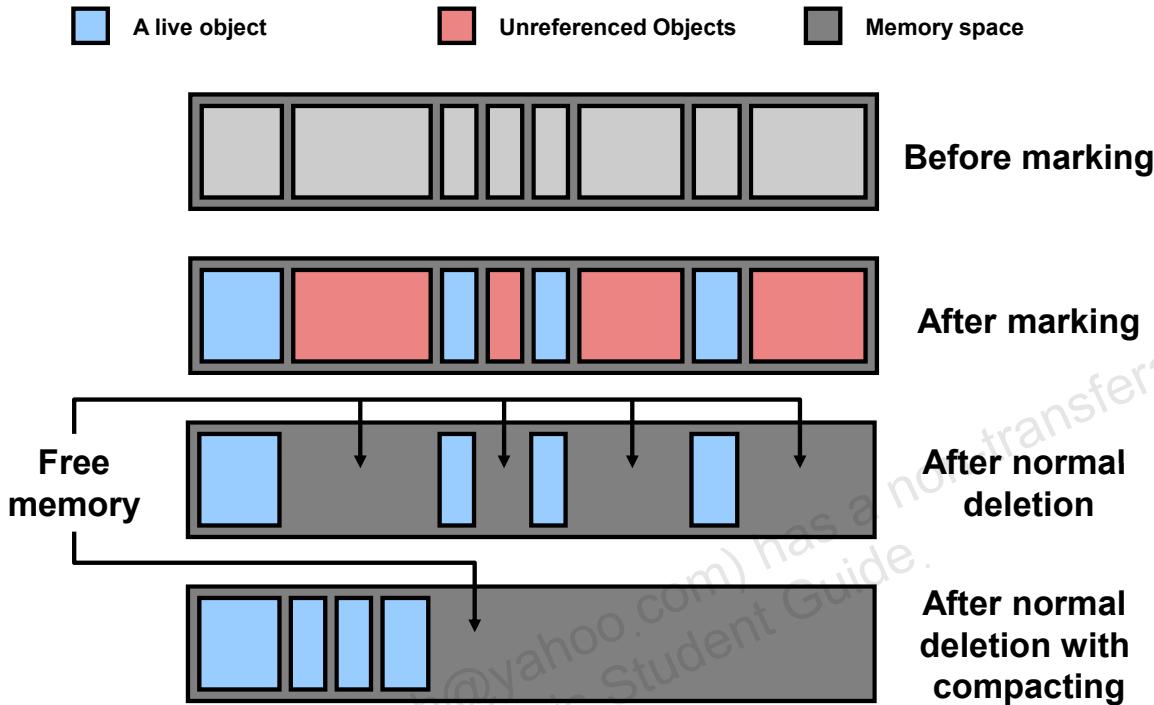


Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Agenda

- JVM Performance: Overview
- The JVM and Java Garbage Collection
- Command-Line JVM Tools
- GUI JVM Tools

Garbage Collection Basics



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Garbage collection is a way in which Java recollects the space occupied by loitering objects.

In general, a garbage collector is responsible for three tasks:

- Allocating memory for new objects
- Ensuring that any referenced objects (live objects) remain in memory
- Recovering memory used by objects that are no longer reachable (dead objects)

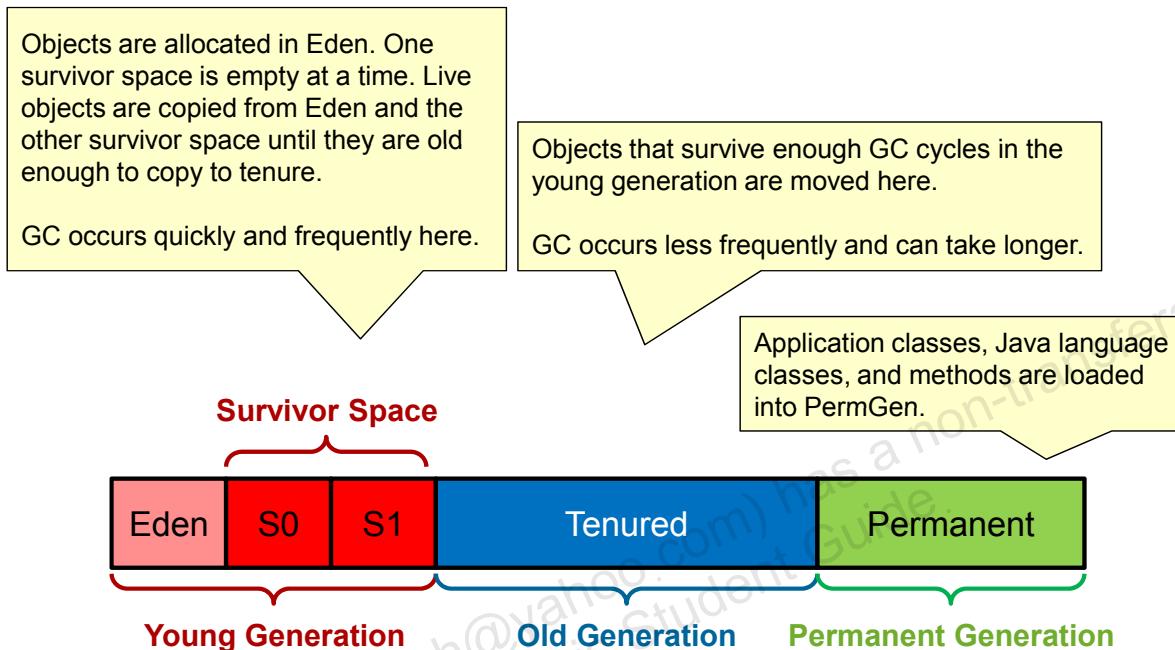
Marking: The first step that the garbage collector performs is called marking. The garbage collector iterates each object one by one through the application graph, checks whether the object is being referenced, and if so marks the object as being used. The marked objects will not be deleted in the sweeping stage.

Normal Deletion: The deletion of objects happens in the sweeping stage. The traditional and easiest way is to mark the space as free and let the allocator use complex data structures to search the memory for the required free space.

Compacting: It is obvious that the traditional way of freeing memory has many problems associated with it. An improved way is by providing a defragmenting system that compacts memory by moving objects closer to each other and removes fragments of free space, if any. In this way, the allocation of future objects is much faster.

Generational Garbage Collection

The Java heap is divided into three sections:



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Non-generational garbage collectors iterate over every object in the heap and check whether or not the object has some active references to it. As the number of objects increases in the heap, this process would take a longer time to complete, and therefore, would be inefficient.

A careful observation of a typical object would tell us the following two characteristics:

- Most allocated objects will die young.
- Few references from older to younger objects exist.

To take advantage of this, the Java HotSpot VM splits the heap into different physical areas, which are called generations. HotSpot uses a type of garbage collection that is termed generational. This means that the Java heap is partitioned into generational spaces. The default arrangement of generations (for all collectors with the exception of the throughput collector) looks something like the image in the slide.

There are three types of generational spaces:

- Young Generation
- Tenured Generation
- Permanent Generation

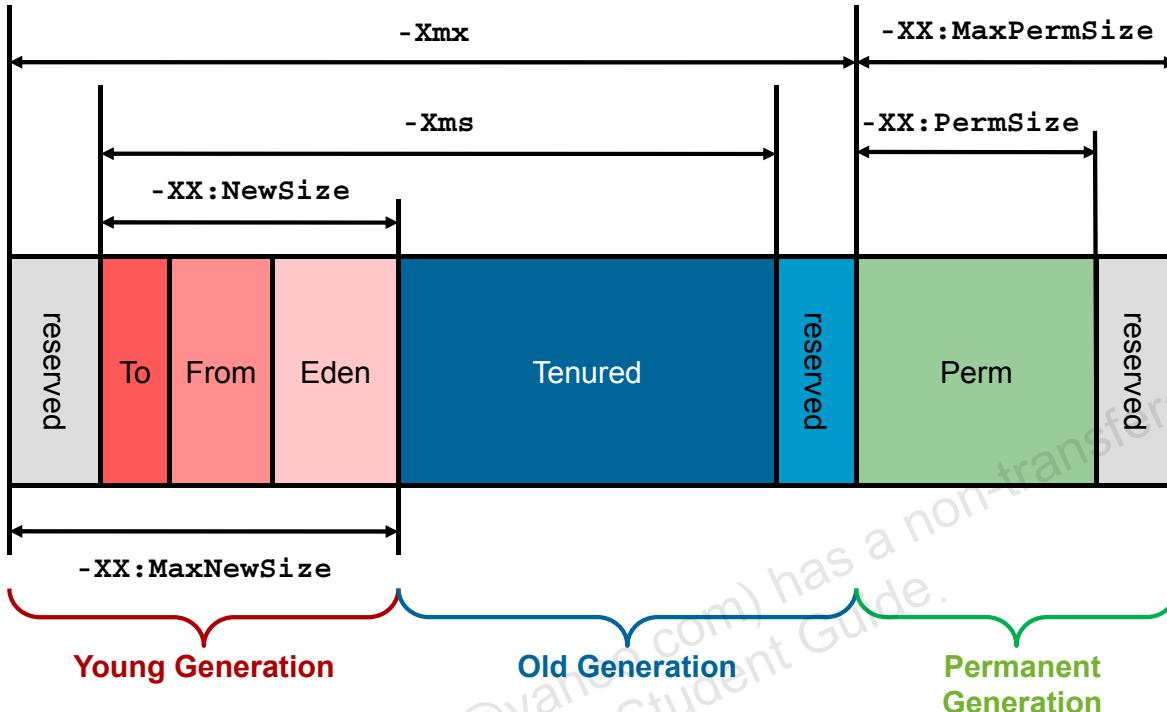
Each of these generations hold objects of different ages. Different algorithms are used to perform garbage collection in different generations, and each algorithm is optimized based on commonly observed characteristics for that particular generation.

Garbage collections for the young generation occur relatively frequently and, therefore, must be fast. So the young generation space is usually small and likely to contain objects that are no longer referenced. The young generation consists of Eden and two smaller survivor spaces. Most objects (except some large objects) are initially allocated in Eden. The survivor spaces hold objects that have survived at least one young generation collection. At any given time, one of the survivor spaces holds such objects, while the other is empty and remains unused until the next collection.

Objects that survive some number of young generation collections are tenured. This generation is typically larger than the young generation and its occupancy grows more slowly. The garbage collections in tenured generation are infrequent, but may take significantly longer to complete.

The permanent generation is where application Java classes, language classes, and method objects are stored. If an application loads a very large number of classes, the size of the permanent generation might need to be increased by using the `-XX:MaxPermSize` option.

Garbage Collectors: Java Heap Options



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The various options available to size the heap are as follows:

- **-Xmx<size>** is the maximum size of Java heap (young generation + tenured generation).
- **-Xms<size>** is the initial size of Java heap (young generation + tenured generation).
- Applications with emphasis on performance usually set **-Xms** and **-Xmx** to the same value.
- When **-Xmx != -Xms**, Java heap growth or shrinkage requires a full garbage collection.
- **-Xmn<size>** is the size of the young generation heap space.

Setting Heap Memory Size

- The `-Xms` value is the space in memory that is committed to the VM at startup.
- The JVM can grow up to the size of the `-Xmx` value.
- The difference between `-Xmx` and `-Xms` is virtual memory (virtually committed).

```
$ export USER_MEM_ARGS="-Xms1g -Xmx2g"  
$ startManagedWebLogic.sh myserver ...
```

Command Line

- For maximum performance, set minimum and maximum heap values the same.

```
$ export USER_MEM_ARGS="-Xms2g -Xmx2g"  
$ startManagedWebLogic.sh myserver ...
```

Command Line

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Setting Heap Size Options

Property	Description
<code>-Xms<size></code>	Initial heap size
<code>-Xmx<size></code>	Maximum heap size
<code>-Xmn<size></code>	Sets initial NewSize and MaxNewSize the same
<code>-XX:PermSize=<size></code>	Initial size of the permanent generation
<code>-XX:MaxPermSize=<size></code>	Maximum size of the permanent generation
<code>-XX:NewSize=<size></code>	Initial size of the young generation
<code>-XX:MaxNewSize=<size></code>	Maximum size of the young generation
<code>-XX:NewRatio=<ratio></code>	Ratio of young generation space to tenured generation space
<code>-XX:SurvivorRatio=<ratio></code>	Ratio of Eden and survivor spaces in young generation



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Configuring the Java HotSpot heap size options when starting WebLogic Server increases performance for most applications. These options may differ depending on your architecture and operating system.

- `-Xms`
 - This parameter configures the starting and minimum size of the Java heap.
 - As a general rule, set this value to be equivalent to the maximum heap size parameter (`Xmx`) to avoid expensive growing and shrinking of the heap.
- `-Xmx`
 - This parameter configures the maximum size of the Java heap.
 - Set this value to accommodate all the objects of your application. When the objects in memory require more space than specified by the `-Xmx` value, you get “Out of memory” errors.
 - If a value is too large, it can deprive memory for other services running on the computer and can cause paging and thrashing.

- -Xmn
 - Applications with emphasis on performance tend to use -Xmn to size the young generation because it combines the use of -XX:MaxNewSize and -XX:NewSize and almost always explicitly sets -XX:PermSize and -XX:MaxPermSize to the same value. A growing or shrinking permanent generation space requires a full garbage collection.
- -XX:PermSize
 - This parameter configures the initial size of the permanent generation.
 - Set this value to be equivalent to the maximum permanent generation size to avoid expensive growing and shrinking of the heap.
- -XX:MaxPermSize
 - This parameter configures the maximum size of the permanent generation.
 - If this value is not set high enough, your entire application server can crash with an out of memory error.
 - Unintentionally referenced objects left behind by component classloaders can cause memory leaks in the permanent generation space.
- -XX:NewSize
 - This parameter configures the initial size of the young generation.
 - Set this value to be one-fourth the size of the maximum heap size. Increase the value of this option for larger numbers of short-lived objects.
- -XX:MaxNewSize
 - This parameter configures the maximum size of the young generation.
 - Set this value to be one-fourth the size of the maximum heap size. Increase the value of this option for larger numbers of short-lived objects.
- -XX:NewRatio
 - Specifies the ratio between the young and tenured generation heap sizes.
 - Set the ratio to 3 to set the young generation to 1/4 of the heap—one part young and three parts old.
- -XX:SurvivorRatio
 - Specifies the ratio between survivor and Eden areas in the new generation.
 - The New generation area is divided into three subareas: Eden and two survivor spaces that are equal in size. Set the ratio to 6 to set each survivor space to 1/8 of the Eden size.
 - If survivor spaces are too small, copying collection overflows directly into the old generation. If survivor spaces are too large, they will be empty. At each GC, the JVM determines the number of times an object can be copied before it is tenured, called the tenure threshold. This threshold is chosen to keep the survivor space half full. Use the -XX:+PrintTenuringDistribution option to show the threshold and ages of the objects in the new generation.

G1 Garbage Collector

The Garbage-First (G1) collector is a server-style garbage collector, targeted for multi-processor machines with large memories.



Meets GC pause time goals with a high probability, while achieving high throughput

- Operates concurrently with application threads like the CMS collector
- Compacts free space without lengthy GC-induced pause times
- Provides more predictable GC pause durations



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The heap is one memory area split into many fixed sized regions. Region size is chosen by the JVM at startup. The JVM generally targets around 2000 regions varying in size from 1 to 32 MB. Regions are not required to be contiguous like the older garbage collectors.

A remember set is kept for each region, which is a list with references to objects. Threads inform the garbage collector when they change a reference, which can change the remember set. When a garbage collection occurs, the areas containing the most free space (or garbage) are collected first, hence the nickname, garbage-first. The idea behind this method is that the most common scenario for garbage collection will be that several regions will exist that contain no living objects. The garbage collector can then simply free the regions without requiring the mark-and-sweep or compacting processes, which speeds up GC performance.

The G1 collector provides settings to define overhead and pause time targets. The collector then sweeps only the number of areas it calculates it can complete within that target setting.

These regions are mapped into logical representations of Eden, Survivor, and old generation spaces. The colors in the picture shows which region is associated with which role. Live objects are evacuated (copied or moved) from one region to another. Regions are designed to be collected in parallel with or without stopping all other application threads.

Regions can be allocated into Eden, Survivor, and old generation regions. In addition, there is a fourth type of object known as Humongous regions. These regions are designed to hold objects that are 50% the size of a standard region or larger. They are stored as a set of contiguous regions. Finally, the last type of regions would be the unused areas of the heap.

GC Algorithms

These are the HotSpot VM command-line switches that affect GC algorithms:

Command-Line Switch	GC Method Chosen
-XX:+UseSerialGC	Serial
-XX:+UseParallelGC	Parallel
-XX:+UseParallelOldGC	Parallel compacting
-XX:+UseParNewGC	Parallel Young collector
-XX:+UseConcMarkSweepGC	Concurrent mark-sweep
-XX:+UseG1GC	Garbage First (G1)



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

With the advent of Ergonomics, most command-line switches are redundant. However, in some specific cases, you may need to use these switches. The table shows HotSpot VM options with the option description and the associated generation.

- **Serial:** The serial collector is a single-threaded, stop the world, garbage collector. It is best used on single CPU systems where multithreading collection would not provide increased performance.
- **Parallel:** Considering that hardware resources (CPU and memory) are getting to be relatively inexpensive, the parallel collector, also known as the throughput collector, enables you to take advantage of available CPUs rather than leaving most of them idle while only one does garbage-collection work.
- **Parallel compacting:** The difference between the parallel compaction collector and the parallel collector is that the parallel compaction collector uses a young algorithm for garbage collection of the old generation. With the parallel compaction collector, the old and permanent generations are collected in a mostly parallel fashion with sliding compaction. By default, only minor collections are executed in parallel and major collections are performed with a single thread. This option enables parallel compaction for major and minor collections.

- **Parallel young collector:** This parallel collector performs parallel garbage collection on the young generation. It is used in conjunction with the CMS collector for the old generation.
- **Concurrent mark-sweep (CMS):** For many applications, fast response time is the most important performance criteria. To facilitate this, you can set up a CMS collector, also known as the low-latency collector. Generally, GC for the young generation does not typically cause long pauses. However, old generation collections, though infrequent, can impose long pauses. So most of the collection of the old generation using the CMS collector is done concurrently with the execution of the application.
- **Garbage first:** The G1 collector is the newest collection algorithm and is meant for high-performance server-side applications. It optimizes collection time by avoiding expensive processing associated with other collection methods. It also optimizes collection time by only collecting what it can within specified performance targets. The permanent generation is still collected by using the CMS collector.

GC Performance Goals

The following are the different ways to measure GC performance:

Performance Goal	Description
Throughput	Percentage of time not spent in GC over a long period of time
Responsiveness	How quickly an application responds in a given scenario
Footprint	Overall memory a process uses to execute

WebLogic Server applications almost
NEVER focus on footprint performance!



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Throughput is the percentage of total time not spent in garbage collection, considered over long periods of time. Throughput includes time spent in allocation (but tuning for speed of allocation is generally not needed). Pauses are the times when an application appears unresponsive because garbage collection is occurring. A responsive application attempts to reduce the length of these pauses.

Users have different requirements of garbage collection. For example, some consider the right metric for a web server to be throughput, because pauses during garbage collection may be tolerable, or simply obscured by network latencies. However, in an interactive graphics program even short pauses may negatively affect the user experience.

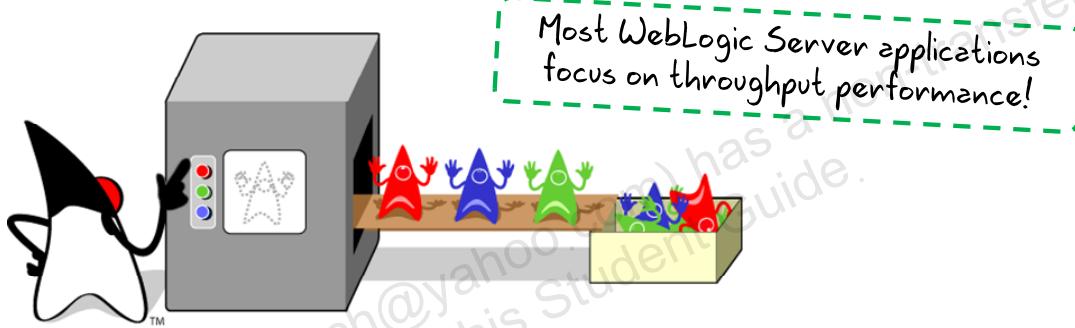
Some users are sensitive to other considerations. Footprint is the working set of a process, measured in pages and cache lines. On systems with limited physical memory or many processes, footprint may dictate scalability.

There are numerous ways to size generations. The best choice is determined by the way the application uses memory as well as user requirements.

Focusing on Throughput

Throughput:

- Has as highest priority the raw throughput of the information or data being processed
- Maximizes application throughput even at the expense of responsiveness
- Will tolerate high pause times in order to maximize throughput



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

A Java application that focuses on throughput emphasizes the raw throughput of the information or data being processed. This is the most important quality for the application. Pause times resulting from JVM garbage collection events are not an issue, or of very little interest. As long as the overall throughput of the application over a period of time is not sacrificed, long pause times are allowed. Examples of applications that focus on throughput include:

- A large phone company printing bills or statements
- A large credit card company printing statements
- A bank calculating interest for accounts

Focusing on Responsiveness

Responsiveness

- Have as highest priority the servicing of requests within a predefined maximum time
- Raw throughput of data or speed of processing requests are secondary to max response time goal
- Are sensitive to GC pause time



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

A Java application that focuses on responsiveness emphasizes how quickly an application responds in a given scenario rather than focusing on the raw throughput of the application. Most applications emphasizing responsiveness have a maximum pause time the application can tolerate. Examples of applications that focus on responsiveness include:

- Applications connected to a user interface such as a web browser or an IDE
- Financial trading applications
- Telecommunication applications

Java applications focusing on responsiveness are sensitive to the time it takes for garbage collection events to complete.

Evaluating GC Algorithm

Criteria	Description
Pause time	Does the collector introduce pauses by stopping the world?
Memory footprint	What is the total memory allocated versus the total memory available to the application?
Virtual memory usage and interaction	How efficiently does the garbage collector manage the reference for pages outside the resident memory during a full GC?
Parallel	Does the collector make use of multiple CPUs if available?
CPU usage	What is the percentage of CPU spent in garbage collection?
Concurrency	Is there a contention for a CPU between application threads and the garbage collector? Can this be overcome by use of systems with multiple CPUs?
Compiler	Are compile-time operations causing issues?

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

GC Tuning Tips

Heap size influences:

- GC frequency and the pause during collections
- The number of short- and long-term objects
- Fragmentation and locality problems

An undersized heap with the concurrent collector leads to full GCs with an increase in load and fragmentation problems.

Size the heap to handle peak and burst loads.



An oversized heap leads to increased collection times and locality problems.

Increase memory as you increase the number of processors, because allocation can be parallelized.

Sizing the permanent generation is important for applications that dynamically generate and load many classes.

Setting PermSize and MaxPermSize to the same value is recommended!

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Increasing the size for permanent generation can be helpful if there are numerous dynamically generated classes. However, you should bear in mind that you should increase -XX:MaxPermSize along with -XX:PermSize. To tune this, you may set -XX:PermSize as high as -XX:MaxPermSize and then reduce it to an appropriate level when you know your typical requirements.

GC Tuning Tips

Configure as much memory as possible to the virtual machine unless you have problems with pauses.



Explicit garbage collection calls (`System.gc()`) force major garbage collections.



Increased object lifespan increases GC frequency because live objects take up heap space. So keep live objects to a minimum.

Measure the effectiveness of explicit GC calls by disabling them using `-XX:+DisableExplicitGC`

Use the default serial collector for smaller applications.

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

If you want to disable calls to request GC manually, set `-XX:+DisableExplicitGC` on the command line while starting the WebLogic Server. This disables calls to `System.gc()` and the distributed Remote Method Invocation (RMI) GC, but JVM performs garbage collection when necessary.

GC Tuning Tips



Applications that rely on finalization cause lags in garbage collection.
Avoid using finalization.

The concurrent collector advantages increase with the number of CPUs.



Use Intimate Shared Memory (ISM) and variable page sizes to reduce the smear problem where available:
`-XX:+UseISM`

For larger applications hosted on WebLogic Server, use the throughput collector:
`-XX:+UseParallelGC`

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

For more information about tuning HotSpot JVM, refer to GCPortal at:

<http://www.oracle.com/technetwork/articles/javase/gcportal-136937.html>

GCPortal enables a service, Application Modeling, and Performance Tuning from a GC perspective. It is implemented in J2EE and available as a J2EE WAR module that can run on any J2EE Application Server. It allows developers to submit log files and analyze application behavior.

You can use GCPortal to performance tune and size the application to run optimally under lean, peak, and burst conditions.

To find more information about aggressive options, use `-XX:+PrintCommandLine`.

Intimate Shared Memory (ISM) increases the page size from 8 KB to 4 MB. It uses real addresses not virtual addresses. It locks pages in shared memory and eliminates paging to disk. A performance boost of 5-10% is possible by using ISM. However, this is not guaranteed and you should monitor the performance of your application to be sure.

ISM warnings:

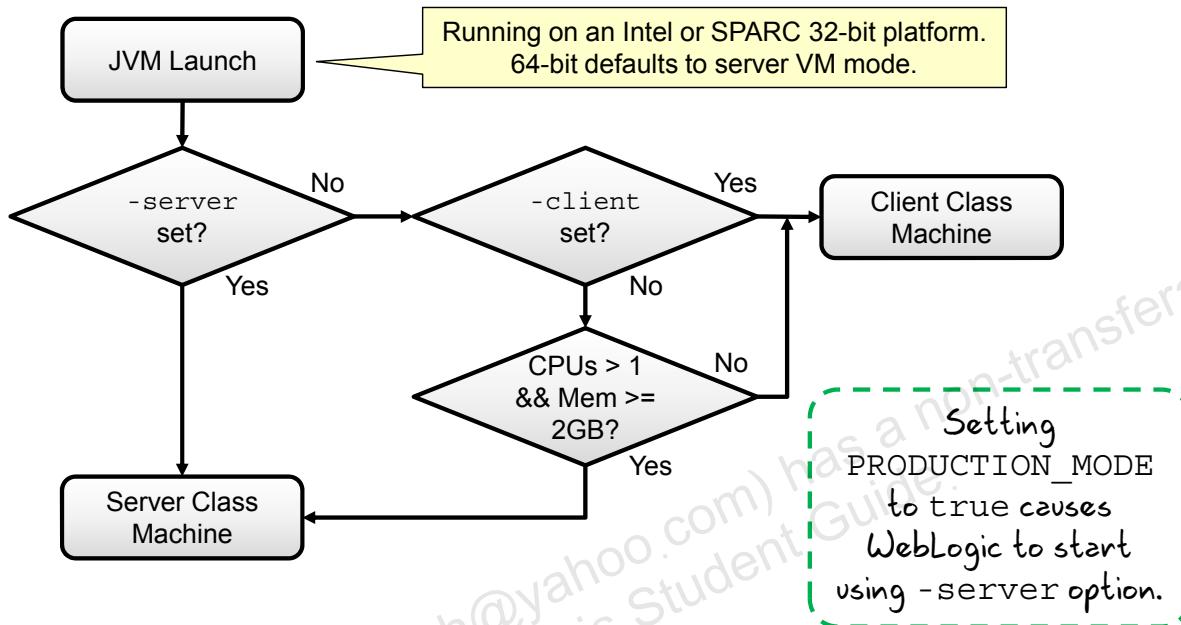
- Is a Solaris-only feature
- Locks memory and should be used only on dedicated systems
- Memory fragmentation may prevent allocation of contiguous 4-MB pages.
- Use `ipcs` or `ipcrm` to discover and delete any locked segments due to abnormal JVM terminations.

Refer to the following URL for further details about various parameters that you can use with Java 7 SE:

- <http://java.sun.com/javase/7/docs/technotes/tools/windows/java.html#nonstandard>

Server-Class Machine Detection

The JVM can detect if it is running on a server-class machine.



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

In J2SE 5.0 or later, when a machine with at least two CPUs and at least 2 GB of physical memory is used, it is said to be a server-class machine.

During JVM launch, the JVM detects whether the application or server is running in a server-class machine; if so, the Java HotSpot Server VM is used.

Server-class detection occurs if you do not specify `-server` or `-client` when launching the application on an Intel or SPARC 32-bit machine running Solaris or Linux. Windows machines always default to the client VM. All 64-bit platforms default to the server VM.

The `PRODUCTION_MODE` parameter in `setDomainEnv.sh` can be set to "true" to start server mode JVM.

The client VM focuses on faster start up time for iterative development and faster response time for users. This VM starts up quickly, but may not perform as fast as the server VM over time.

The server VM starts up more slowly than the client VM, but performs faster over time.

Ergonomics: What It Does

- Evaluates the system and auto-magically chooses defaults for the HotSpot JVM. No tuning required.
- Relies on definition of “server class machine”
 - 2 or more processor cores and 2 or more GB of physical RAM
- Server class machines will use -server JIT compiler.
- Nonserver class machines are considered "client class."



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The J2SE Platform version 1.5 or later offers a feature called ergonomics. The goal of ergonomics is good JVM performance with a minimum of command-line tuning. When ergonomics is enabled, the JVM attempts to match the best settings for an application.

On server-class machines, by default, the following are selected:

- Server JIT compiler
- Throughput garbage collector
- Heap sizes
- Initial heap size of 1/64 of physical memory up to 1 GB
- Maximum heap size of 1/4 of physical memory up to 1 GB

If not identified as a server class, then runs as a client class:

- Client JIT compiler
- Default serial collector, same as before

Section Summary

In this section, you should have learned how to:

- Tune JVM heap settings
- Describe JVM garbage collection algorithms
- Monitor and tune JVM garbage collection
- Use HotSpot ergonomics



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Practice 4-1 Overview: Tuning JVM Garbage Collection

This practice covers the following topics:

- Configuring different heap settings
- Configuring different garbage collectors
- Loading an application with each setting
- Capturing and comparing performance metrics
- Analyzing performance results and trying to answer some questions

Agenda

- JVM Performance: Overview
- The JVM and Java Garbage Collection
- Command-Line JVM Tools
- GUI JVM Tools

Using jps

- Command-line utility to find running Java processes
- Included in the HotSpot JDK
- Capable of local and remote monitoring

```
jps [-q] [-mlvV] [<hostid> where <hostid> =  
<hostname>[:<port>]]
```

Command Line



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

jps is very similar to ps in UNIX. Its purpose is to identify the process ID of Java processes. If jps is run without specifying a hostid, it looks for instrumented JVMs on the local host. If jps is started with a hostid, it looks for JVMs on the indicated host, using the specified protocol and port. This can be used to retrieve the virtual machine identifier (vmid) or Local Virtual Machine Identifier (lvmid), which can be used as an input for other commands.

Note: See the jps man page on oracle.com for details about -q, -mlvV options:

<http://download.oracle.com/javase/7/docs/technotes/tools/share/jps.html>

Using jps: Example

- Output from a `jps` command with no options lists each Java application's `lvmid` followed by the short form of the application's class name or JAR file name.
- The `-l` option outputs the full package name for the application's main class or the full path name to the application's JAR file.

```
$ jps
19518 NetworkServerControl
22287 Console
22334 Grinder
5155 jps
19719 Server
19539 Server
```

lvmids

These are WebLogic Server instances.
You use other command-line options with
`jps` to determine which server is which.

Command Line

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The list of JVMs produced by the `jps` command may be limited by the permissions granted to the principal running the command. The command lists only the JVMs for which the principal has access rights as determined by operating system-specific access control mechanisms.

This command is important to retrieve the `lvmids` of the JVMs running so that you can use them in other commands.

The screenshot shows the result of a `jps` call.

Using jcmd

- Utility to send diagnostic commands to a running JVM
 - Gets list of PIDs like `jps`
 - Provides a lot of query options for running JVMs
- Basic usage

```
$ jcmd  
$ jcmd <pid> command  
$ jcmd 0 command
```

Command Line

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The `jcmd` (pronounced "j command") utility is a new diagnostic tool that can look up process IDs of running JVMs and also send diagnostic commands to a running JVM. The following are some example `jcmd` commands.

- `jcmd`: Returns the PID of all running JVMs on the system
- `jcmd 1234 VM.flags`: Displays the command-line options used to start the JVM with a PID of 1234
- `jcmd java2 VM.flags`: Displays the command-line options used to start the JVM whose name includes the string "Java2"
- `jcmd 0 VM.flags`: Displays the command-line options used to start all the running JVM instances on this machine

Using jcmd: Examples

- Getting a list of options

```
$ jcmd 1234 help  
$ jcmd 1234 help -all
```

Command Line

- JVM information

```
$ jcmd 1234 VM.uptime  
$ jcmd 1234 VM.command_line  
$ jcmd 1234 VM.version  
$ jcmd 1234 VM.system_properties
```

Command Line

- Threads

Shows deadlocks if they exist.

```
$ jcmd 1234 Thread.print
```

Command Line

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

- `jcmd 1234 help`: Lists all the options available on this JVM
- `jcmd 1234 help -all`: Lists all the options available on this JVM and prints a brief description of each option
- `jcmd 1234 VM.uptime`: Displays the amount of time the JVM has been up
- `jcmd 1234 VM.command_line`: Shows the command used to start the JVM and the command-line options
- `jcmd 1234 VM.version`: Shows the version of the JVM
- `jcmd 1234 VM.system_properties`: Displays the system properties for the selected JVM

Using jcmd: Histogram

```
$ jcmd 1234 GC.class_histogram
num      #instances          #bytes  class name
-----
 1:        29775       3865168  <constMethodKlass>
 2:        29775       3825648  <methodKlass>
 3:        1110        3074312  [I
 4:        2309        2903976  <constantPoolKlass>
 5:        2309        2172472  <instanceKlassKlass>
 6:        2210        1985184  <constantPoolCacheKlass>
 7:        10778        859312  [C
 8:        3286        788120  [B
 9:        1177        525640  <methodDataKlass>
10:        2546        315248  java.lang.Class
11:        3836        261224  [S
12:        10376        249024  java.lang.String
```

Command Line

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

jcmd 1234 GC.class_histogram: Displays a list of classes taking up the most memory.
You may want to redirect the output to a file because it can be very long.

Using jinfo

jinfo prints Java configuration information for a given Java process or core file or a remote debug server.

```
$ jinfo 19719
Attaching to process ID 19719, please wait...
Debugger attached successfully.
Server compiler detected.
JVM version is 24.0-b56
Java System Properties:
java.vendor = Oracle Corporation
...
VM Flags:
-Xms256m -Xmx512m -XX:MaxPermSize=256m -Dweblogic.Name=server1 ...
-Dweblogic.ProductionModeEnabled=true ...
-Dwlshome=/u01/app/fmw/wlserver/server
-Dweblogic.home=/u01/app/fmw/wlserver/server
-Dweblogic.management.server=host01:7001
```

Command Line

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Refer to the following URL for more information:

<http://docs.oracle.com/javase/7/docs/technotes/tools/share/jinfo.html>

The example in the slide shows jinfo displaying the system properties and flags of a JVM running WebLogic.

Syntax: jinfo [option] pid | executable core | [server-id@] remote-hostname-or-IP

where

- pid: Java process ID
- executable: Java executable from which the core dump was produced
- core: Core file for which the configuration information is to be printed
- server-id: Unique server ID
- remote-hostname-or-IP: Remote server's IP or hostname

Using jstat

- Command-line utility that runs on a local or remote JVM
- Included in the HotSpot JDK

```
$ jstat -<option> [<-t> [<-h<lines>>] <vmid> [<internal>  
[<count>]]
```

Command Line

- Garbage collection options:
`-gc, -gccapacity, -gccause, -gcnew, -gcnewcapacity,
-gcold, -gcoldcapacity, -gcpermcapacity, -gcutil`



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

jstat is a command-line tool that displays detailed performance statistics for a local or remote HotSpot VM. The `-gcutil` command-line option is used most frequently.

For information about garbage collection options, see the main page at:

<http://docs.oracle.com/javase/7/docs/technotes/tools/share/jstat.html>

Caution: When using the CMS collector (also known as concurrent collector), jstat reports two full GC events per CMS cycle, which is obviously misleading. However, young generation stats are accurate with the CMS collector.

Using jstack

- You can use `jstack` to print a stack trace of all the threads currently running in a VM.
- `jstack` is different from the normal thread dump process in the following ways:
 - Prints thread stack for a VM running remotely
 - Prints thread stack from the core file
 - Prints thread stack on `stdout` where the command is running, whereas thread dump (`kill -3 pid` or Ctrl-Break) prints thread stack trace on the VM's `stdout`
 - Is a vendor-dependent, platform-dependent implementation, whereas thread dump is not

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Syntax:

```
jstack [ option ] pid | executable core | [server-id@]remote-  
hostname-or-IP
```

- `pid`: Java process ID
- `executable`: Java executable from which the core dump was produced
- `core`: The core file for which the configuration information is to be printed
- `server-id`: Unique server ID
- `remote-hostname-or-IP`: Remote server's IP or hostname

For each Java frame, the full class name, method name, “bci” (byte code index), and line number, if available, are printed.

Refer to the following URL for more information:

<http://docs.oracle.com/javase/7/docs/technotes/tools/share/jstack.html>

jstack

`jstack` prints Java stack traces of Java threads for a given Java process, core file, or a remote debug server.

```
$ jstack -l 19719
2013-09-23 09:32:53
Full thread dump Java HotSpot(TM) 64-Bit Server VM (24.0-b56 mixed mode):
. .
"main" prio=10 tid=0x00007f90ac009800 nid=0x4d08 in Object.wait()
[0x00007f90b1cf000]
    java.lang.Thread.State: WAITING (on object monitor)
    at java.lang.Object.wait(Native Method)
    - waiting on <0x00000000e02c4c18> (a weblogic.t3.srvr.T3Srvr)
    at java.lang.Object.wait(Object.java:503)
    at weblogic.t3.srvr.T3Srvr.waitForDeath(T3Srvr.java:995)
    - locked <0x00000000e02c4c18> (a weblogic.t3.srvr.T3Srvr)
    at weblogic.t3.srvr.T3Srvr.run(T3Srvr.java:492)
    at weblogic.Server.main(Server.java:74)
    Locked ownable synchronizers: - None"
VM Thread" prio=10 tid=0x00007f90ac066800 nid=0x4d0b runnable
"GC task thread#0 (ParallelGC)" prio=10 tid=0x00007f90ac01f800 nid=0x4d09 runnable
"GC task thread#1 (ParallelGC)" prio=10 tid=0x00007f90ac021800 nid=0x4d0a runnable
"VM Periodic Task Thread" prio=10 tid=0x00007f90ac09e000 nid=0x4d12 waiting on
condition
JNI global references: 254
```

Command Line

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

A stack trace of all threads can be useful when you are trying to diagnose a number of issues such as deadlocks or hangs. Deadlocks and hangs may consume a server's resources more, which may let the user request deprive or starve. Diagnosing and eliminating these issues from applications deployed on WebLogic Server increases the performance of the server.

The following are the possible thread states:

- **NEW:** Thread has been created, but it has not started running yet.
- **IN_NATIVE:** Thread is running native code.
- **IN_VM:** Thread is running VM code.
- **IN_JAVA:** Thread is running (either interpreted or compiled) Java code.
- **BLOCKED:** Thread is blocked.
- **..._TRANS:** If you see any of the preceding states followed by “_TRANS,” it means that the thread is changing to a different state.
- **UNINITIALIZED:** Thread is not created. This will normally not happen (unless there is a serious bug such as memory corruption).

The screenshots show an example of running `jstack`.

Section Summary

In this section, you should have learned how to use command line JVM monitoring tools.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

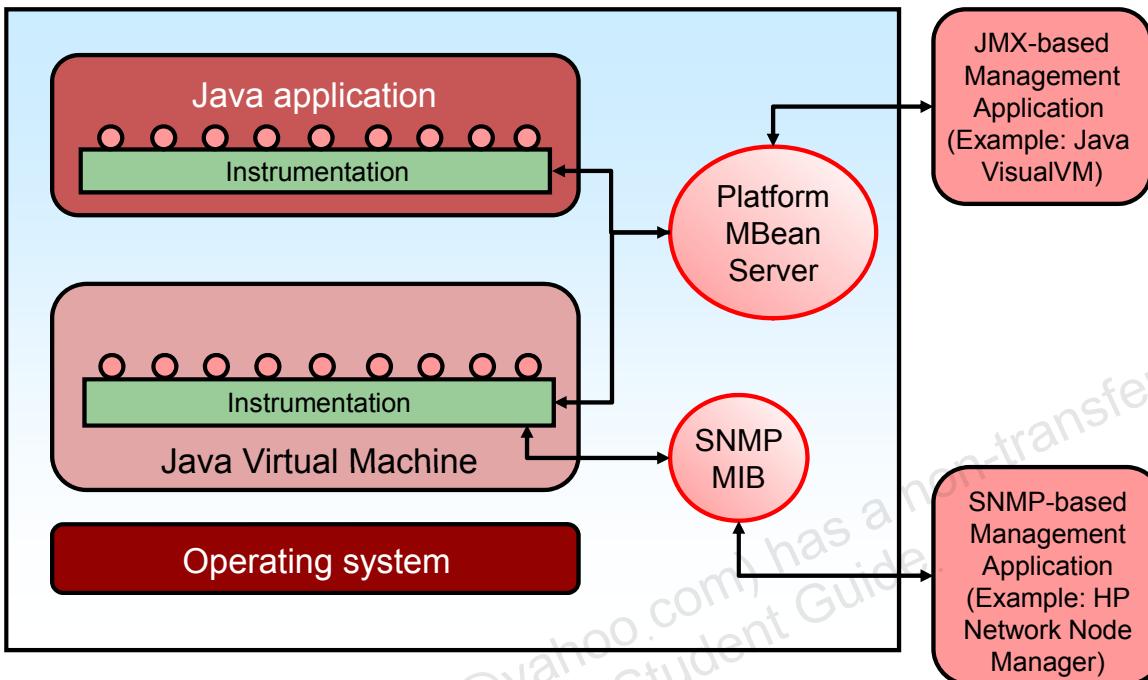
Practice 4-2 Overview: Using JVM Command Line Tools

This practice covers experimenting with different command-line tools to monitor WebLogic Server JVM processes.

Agenda

- JVM Performance: Overview
- The JVM and Java Garbage Collection
- Command-Line JVM Tools
- GUI JVM Tools

Java Monitoring and Management Architecture



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The graphic shows the Java Management Extensions (JMX)-based management applications and Simple Network Management Protocol (SNMP)-based management applications connecting to Java applications facilitating remote management.

Java has greatly expanded JVM monitoring and management support capabilities and included the Java VisualVM, jconsole, Mission Control, and Flight Recorder tools to take advantage of these capabilities.

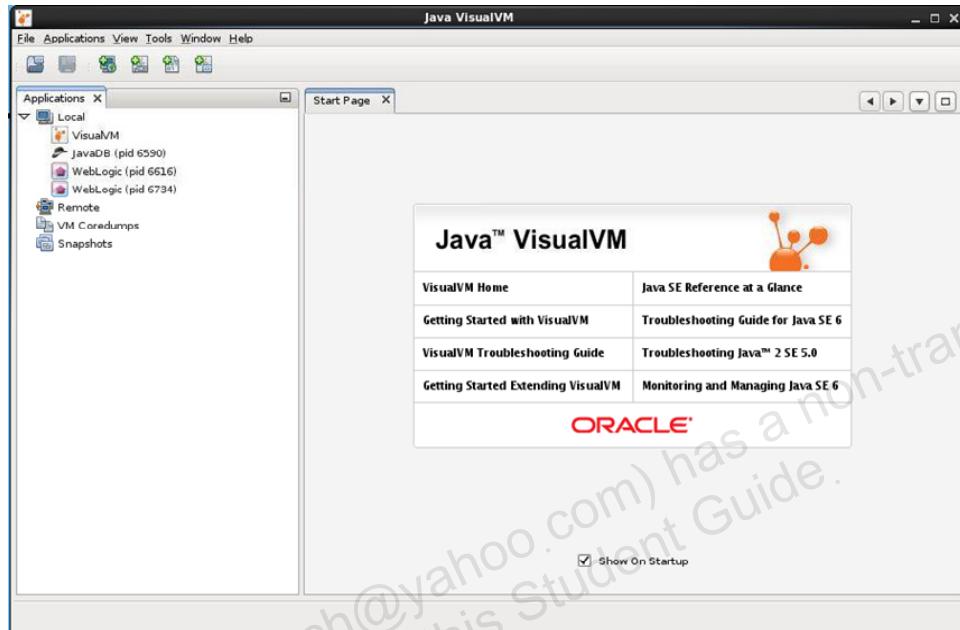
When an application is run with JMX enabled, a JMX Agent process is activated within the JVM to manage JMX requests. JMX clients can then locate and connect to the JMX agent and query it for the available management resources.

JMX was later complemented with the JMX Remote API (JSR 160), which allowed remote clients to access the JMX agent. The JMX Remote API includes access control and secure sockets layer (SSL) support, which makes it secure enough for use in production.

An MBean is a managed object that follows the design patterns conforming to the JMX specification. An MBean can represent a device, an application, or any resource that must be managed. The management interface of an MBean comprises a set of readable and writable attributes, and a set of invocable operations. MBeans can also emit notifications when predefined events occur.

Java VisualVM

Java VisualVM is a JMX-compliant GUI tool that enables you to monitor and manage Java applications and JVM.



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Java VisualVM, first made available with JDK version 6, update 7, is a graphical user interface that provides information about Java applications and the JVM on which the applications run. Java VisualVM federates several monitoring, troubleshooting, and profiling utilities such as `jmap`, `jinfo`, and `jstack` to obtain data from the JVM software, and then reorganizes and presents the information graphically.

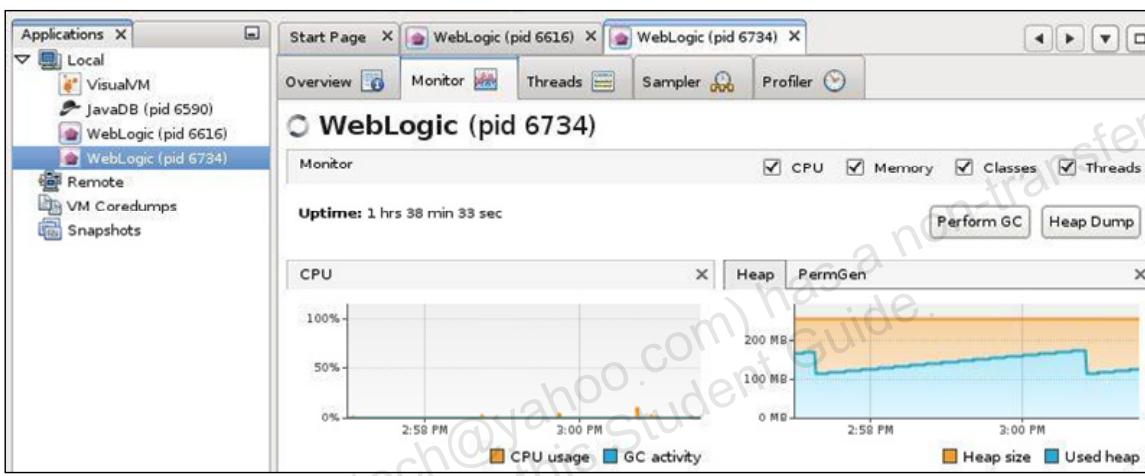
You can view different data about multiple Java applications uniformly, whether they are running locally or on remote machines.

Java VisualVM can be used by Java application developers to troubleshoot applications and to monitor and improve the applications' performance. Java VisualVM can allow developers to generate and analyze heap dumps, track down memory leaks, browse the platform's MBeans and perform operations on those MBeans, perform and monitor garbage collection, and perform lightweight memory and CPU profiling.

Java VisualVM Connections

Java VisualVM can connect to running JVMs:

- Local: Connects to JVMs running on the local system
- Remote: Connects to a JMX agent by using an RMI connector
- Advanced: Connects to a JMX agent by using a non-RMI connector



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

When started without any arguments, Java VisualVM automatically detects all JVMs running locally. In a production environment, it is recommended to run Java VisualVM on a separate machine so as to minimize contention for system resources. The screenshot shows Java VisualVM connected to two local JVM instances.

Note: A non-RMI connector could be SNMP based.

Remote Monitoring

- You can use Java VisualVM to monitor and manage remote applications and JVM.
- Ensure that the `jstatd` agent is running on the remote machine that you want to monitor.
- For security, when remote monitoring is enabled, password authentication over SSL is enabled by default.

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

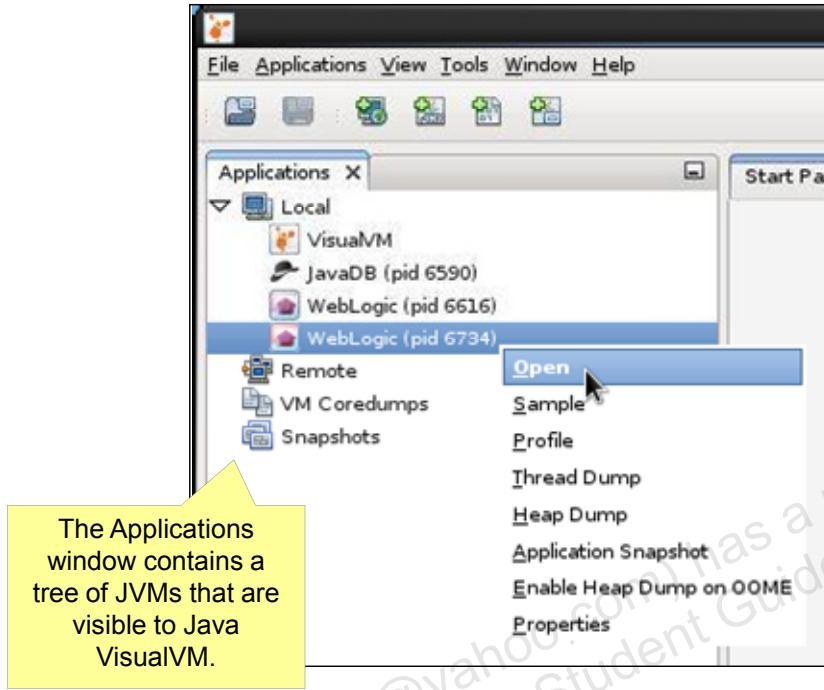
You can use Java VisualVM to monitor applications and JVMs running on remote hosts. Java VisualVM can display general data about the application's run-time environment and can monitor memory heap and thread activity; however, Java VisualVM cannot profile remote applications.

You can add a remote host as follows:

1. Right-click the Remote node in the Applications window.
2. Select Add Remote Host.
3. Enter the host name or IP address in the Add Remote Host dialog box. (You can also specify a display name that will be used to refer to the host when listed under the Remote node.)

Connected remote hosts are displayed as nodes and the applications running on the host as subnodes under the Remote branch of the Applications window.

Java VisualVM Interface



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

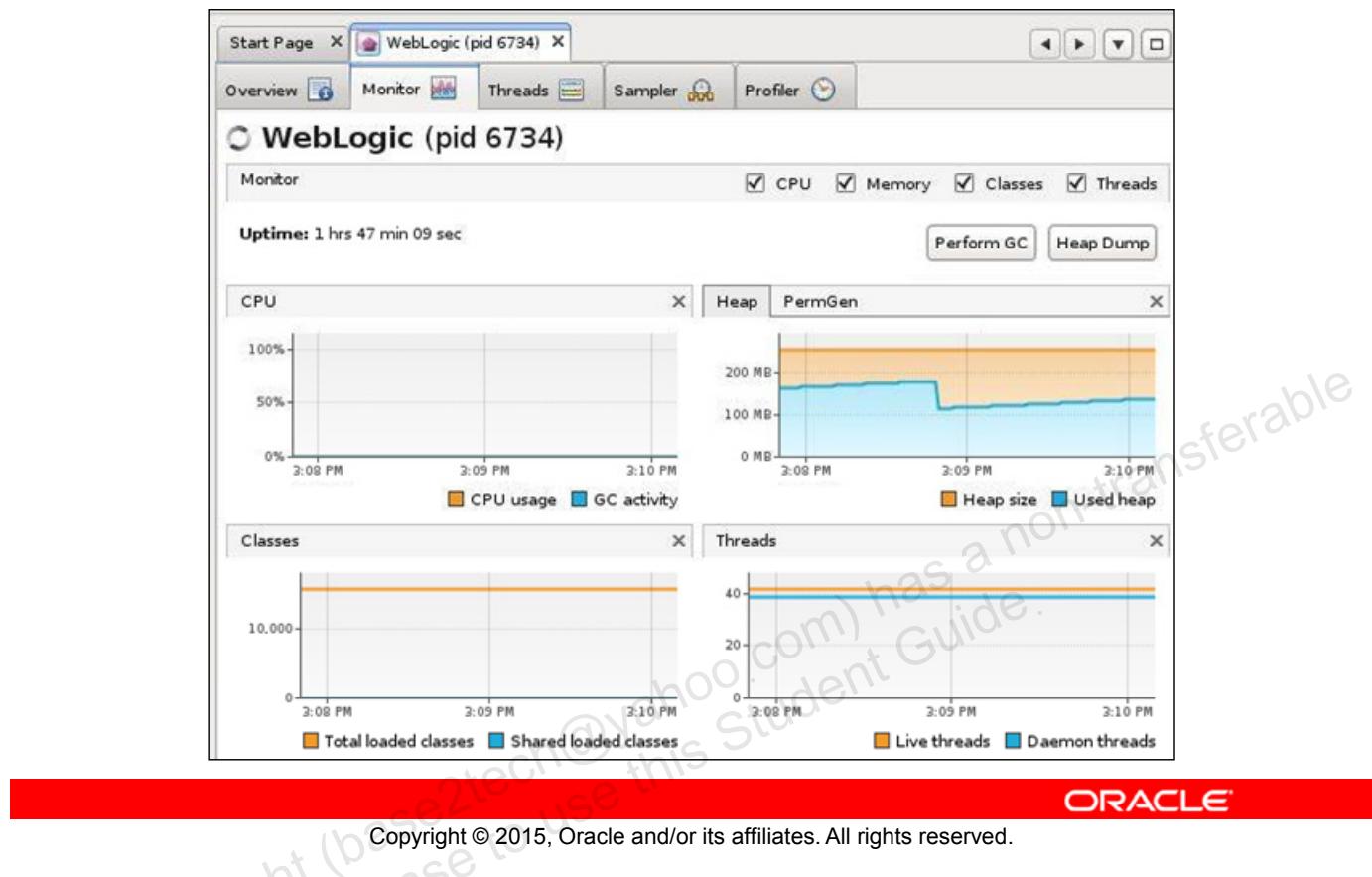
The screenshot shows the Java VisualVM interface after connecting to a HotSpot JVM. There are two main panels in this interface: the Applications window and the Monitoring Panel.

The Applications window:

- The Applications window is the main point of entry for exploring the details of running JVMs. This window uses a tree structure to enable you to quickly view the JVMs (running on the local and any connected remote systems) that are visible to this Java VisualVM. In Solaris and Linux systems, you can also access core dumps and saved snapshots from the Applications window.
- You can right-click a node in the Applications window and use the pop-up menu to perform actions related to that node, including opening application tabs, taking heap and thread dumps, and opening snapshots in the main window.

The Monitoring Panel contains statistics and graphs on the applications to which you have connected.

Monitoring JVM

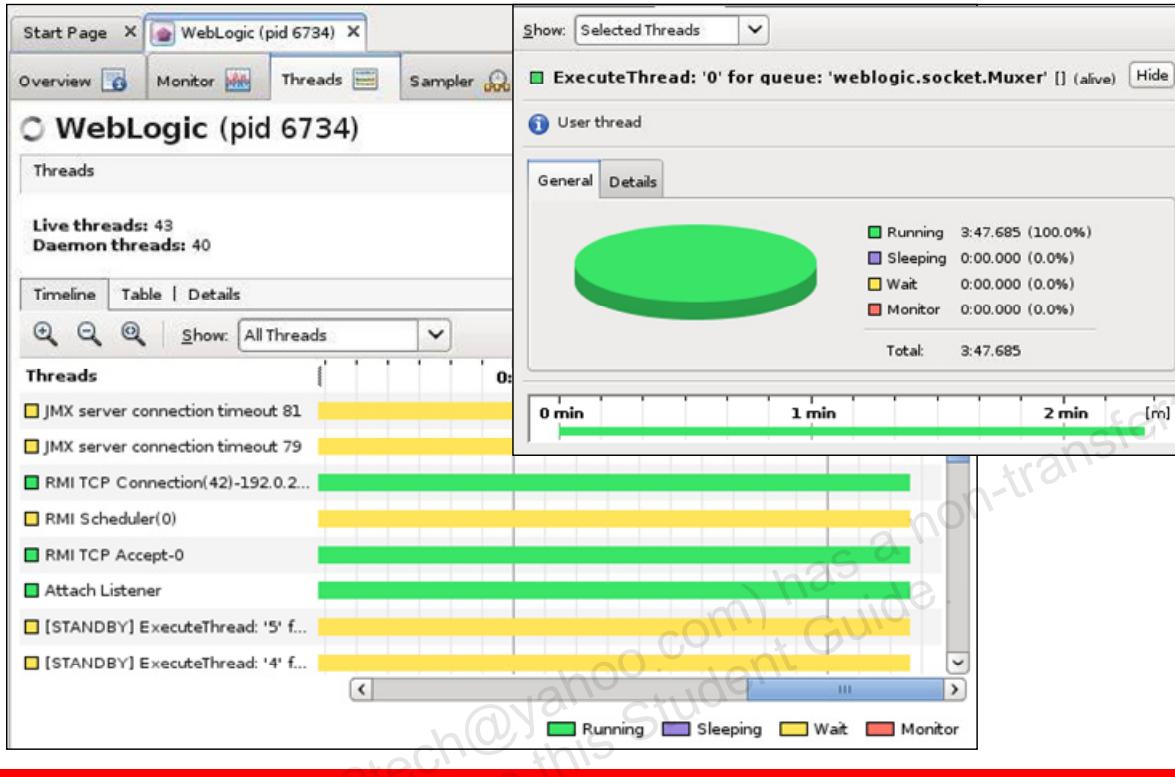


The Monitoring tab provides four detailed graph sections:

- **CPU:** The CPU graph shows CPU trends in historic intervals in real time, and GC activity in correlation to CPU statistics.
- **Heap/PermGen:** The Heap graph displays the total heap size and how much of the heap is currently used. The PermGen graph displays changes in the permanent generation area over time.
- **Classes:** The Classes graph displays an overview of the total number of loaded classes and shared classes.
- **Threads:** The Threads graph displays an overview of the number threads in the application's JVM. You can use Java VisualVM to take a thread dump if you want to capture and view more detailed data on application threads at a specific point in time.

In addition, you can manually force a garbage collection or dump heap. When you initiate a heap dump, a Heap Dump tab appears in the monitored JVM or application. Also, the heap dump node appears under the application or JVM node.

Monitoring Threads



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The Threads tab presents high-level data on thread activity. The Threads tab is visible if Java VisualVM can make a JMX connection and retrieve JMX instrumentation from the JVM.

If the target is local and based on Java 6 or later, then the JMX connection is made automatically. If the application is running on an older version, you may need to explicitly establish a JMX connection with the JVM software.

By default, the Threads tab displays a timeline of the current thread activity. You can click a thread in the timeline to view details about that thread on the Details tab.

Using the Thread Dump button, you can force a stack trace while a local application is running. A thread dump includes thread states for the active Java threads. It does not stop the application. You can use a stack trace to help diagnose a number of issues such as deadlocks or when an application hangs. Using Java VisualVM to take a thread dump can be very convenient where you do not have a command-line console for the application.

Profiler Snapshot

The screenshot shows the Java VisualVM interface for profiling a WebLogic application (pid 6734). The 'CPU' profile is active. A tooltip 'Take Snapshot of Collected Results' points to the 'Snapshot' button in the toolbar. The 'Profiling results' table lists the top four hot spots:

Hot Spots - Method	Self time	Invocations
weblogic.work.ExecuteThread.waitForRequest()	7,211,654 ms (65%)	1,502
weblogic.socket.NIOSocketMuxer.selectFrom(...)	1,528,346 ms (13.8%)	99
weblogic.timers.internal.TimerThread.doWait (long...)	571,923 ms (5.2%)	1,596
weblogic.socket.WeblogicServerSocket.accept()	557,443 ms (5%)	22

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

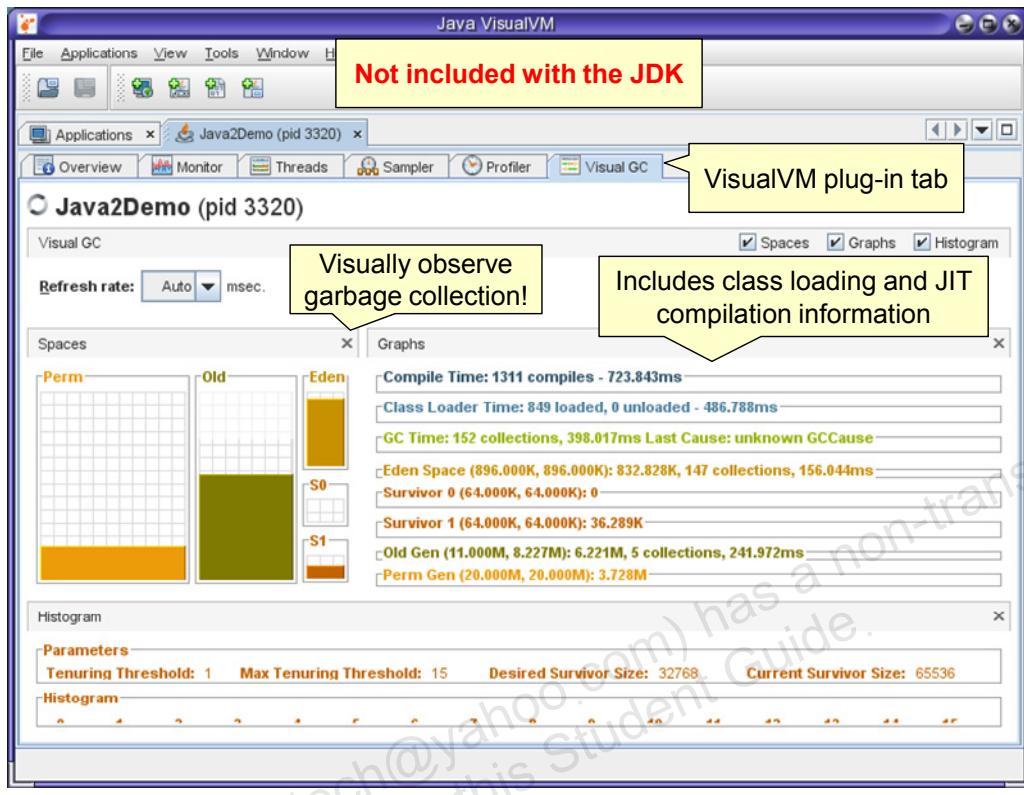
Profiler snapshots capture profiling data at the moment the snapshot is taken. You can take a profiler snapshot at any time during a profiling session. After you take the snapshot, stop the profiling session to reduce resource contentions.

When you take a profiler snapshot, a node representing the snapshot appears below the application node in the Applications window and a temporary snapshot file is written to your local system in the Java VisualVM user directory.

You can take the following types of profiler snapshots by using Java VisualVM:

- **Memory snapshot:** A memory snapshot captures profiling data on allocated objects. You can take a memory snapshot when you are using the profiler to analyze memory usage.
- **CPU snapshot:** A CPU snapshot captures data on the performance of the application. You can take a CPU snapshot when you are using the profiler to analyze application performance.

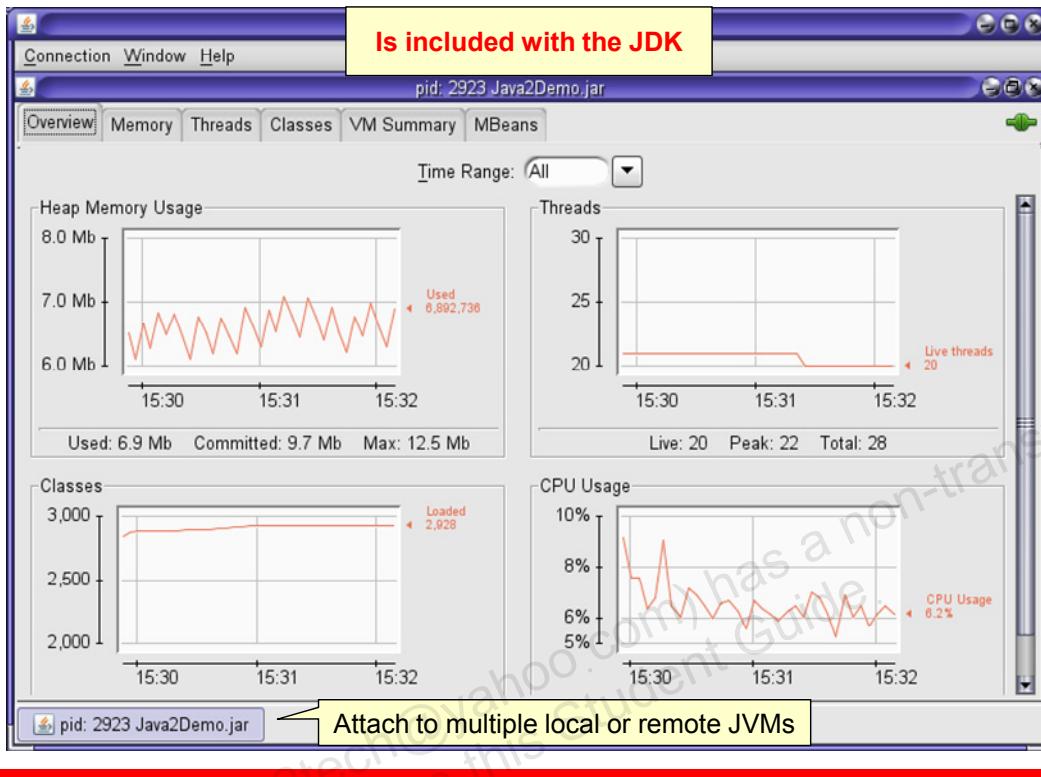
Using VisualGC



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

VisualGC is a stand-alone graphical JVM monitor or a VisualVM plug-in. In this course and typically, VisualGC is used as a VisualVM plug-in. You can install it directly by using the VisualVM plug-in center. With VisualGC, a picture is worth a thousand words, because you can see visually exactly what is going on with the garbage collector. In addition to garbage collection, VisualGC also provides information about class loading and JIT compilation.

Using jconsole



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

jconsole is a graphical monitoring and management console that comes with the HotSpot JDK. jconsole supports both Java Management Extensions (JMX) and MBean technology. This allows jconsole to monitor multiple JVMs at the same time. In addition, more than one jconsole session can monitor a single JVM session at the same time. jconsole can monitor the following JVM features:

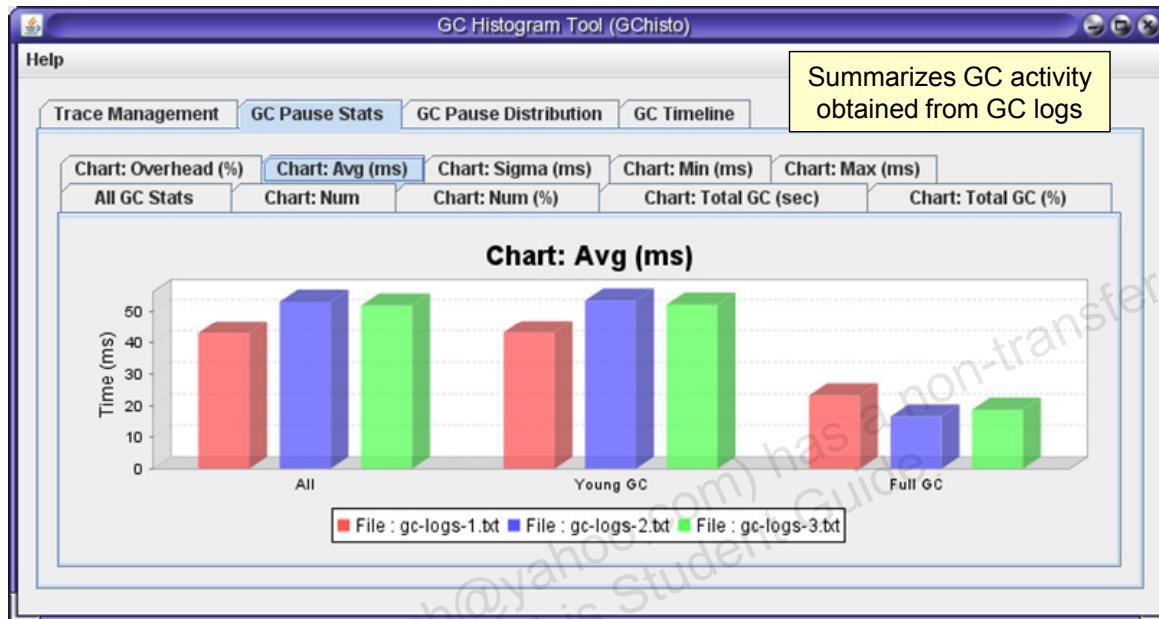
- Memory usage by memory pool/spaces
- Class loading
- JIT compilation
- Garbage collection
- Threading and logging
- Thread monitor contention

Note: MBeans are managed beans, which are Java objects that represent resources to be managed. They can be used with JMX applications. Multiple jconsole sessions can attach to a single JVM.

Using GCHisto

Open source project:
<http://gchisto.dev.java.net>

Not included with the JDK



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

GCHisto is a stand-alone GUI for analyzing GC log data. (There is also a VisualVM plug-in under development.) You can analyze multiple log files at the same time. GCHisto also allows the comparison of heap sizes or collector types for JVM tuning by comparing GC logs.

Open Source Project: <http://gchisto.dev.java.net>

Practice 4-3 Overview: Using Java VisualVM

This practice covers experimenting with Java VisualVM to monitor WebLogic Server JVM processes.

Practice 4-4 Overview: Using VisualGC, jconsole, and GCHisto

This practice covers experimenting with VisualGC, jconsole, and GCHisto to monitor WebLogic Server JVM processes.

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

ORACLE

Mission Control

Mission Control is a group of management and monitoring tools:

Tool	Description
Management Console	JMX-compliant monitoring tool
Flight Recorder	Records and analyzes the JVM and running applications

Now Hotspot Enabled:
Features from JRockit JVM are integrated



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Mission Control is a set of tools for managing, monitoring, profiling, and troubleshooting your Java applications. Mission Control uses data collected from the Hotspot JVM as a part of its normal operations, thus minimizing the performance overhead. The main tools available in Mission Control are:

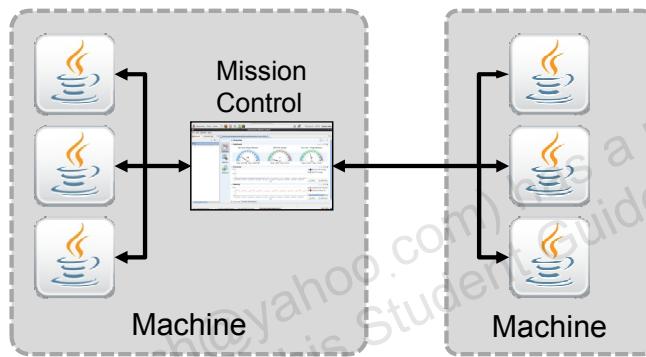
- **Management Console:** The Management Console views real-time behavior of your application and JVM. Features include the ability to create rules that trigger on certain events (for example, an email will be sent if the CPU reaches a 90% load).
- **Flight Recorder:** Records performance data over a period of time. Typical information recorded includes the Java heap distribution, garbage collections, method samples, latency data, and lock profiling information.
- **JMX Agent:** Provides access to all MBeans deployed in the platform MBean server. Using these MBeans, you can read attribute information, such as garbage collection pause times.

Java Discovery Protocol (JDP)

Java Discovery Protocol (JDP) is based on multicast and enables Mission Control to discover, connect, and manage JVMs remotely.

```
-Dcom.sun.management.jmxremote.port=7091  
-Dcom.sun.management.jmxremote.rmi.port=7091  
-Dcom.sun.management.jmxremote.authenticate=false  
-Dcom.sun.management.jmxremote.ssl=false  
-Dcom.sun.management.jmxremote.autodiscovery=true
```

Set these basic options on the remote JVM to set up remote connectivity.

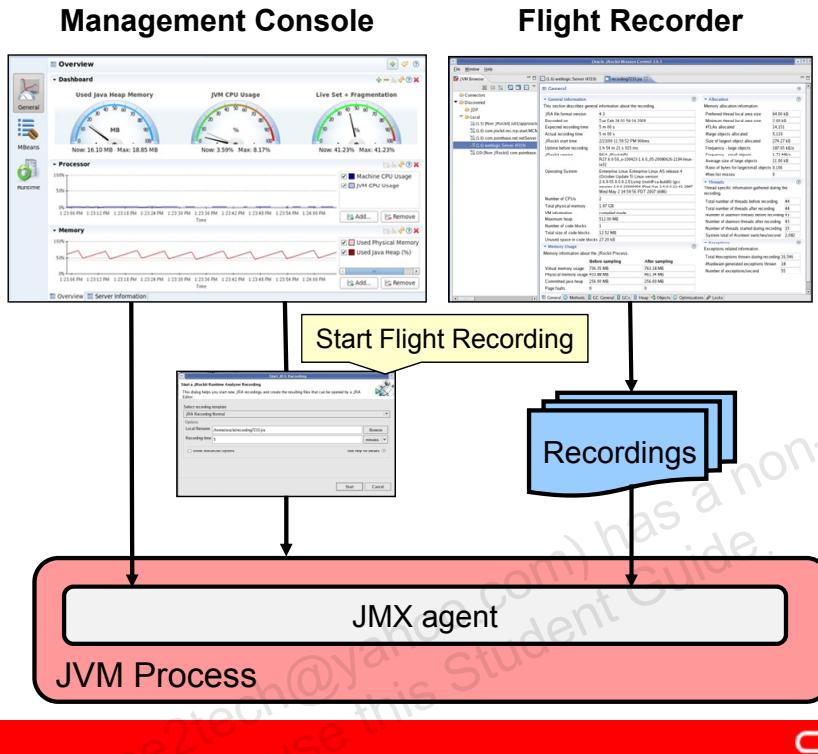


ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

You configure JMX remote connectivity with your JVMs using Java options that enable JDP. Mission Control is configured to automatically discover JDP-enabled JVMs. This connection uses SSL and a username and password login by default. The options in the slide bypass this security for development environments.

Mission Control: Architecture



ORACLE

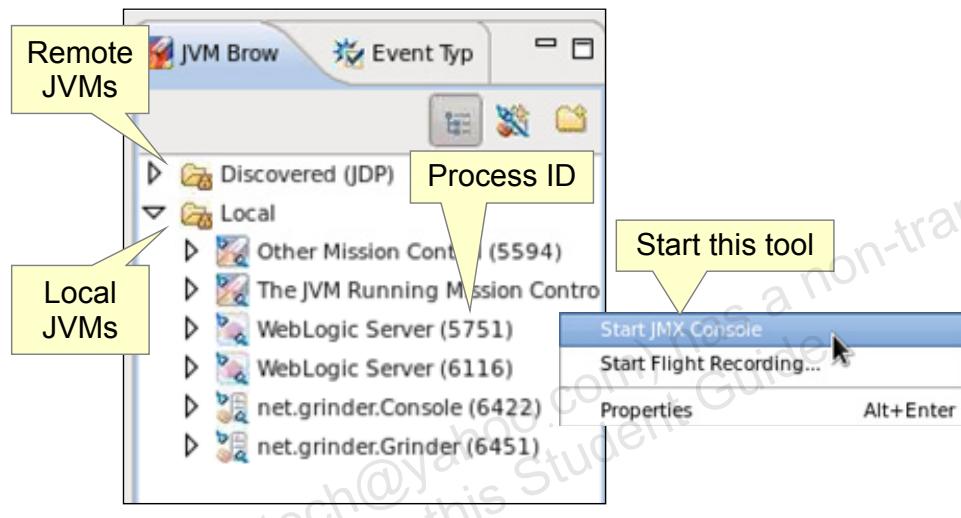
Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The main focus of Mission Control is to do the necessary instrumentation with the lowest possible impact on the running system. The technology used also enables the application to run at full speed after the tool is disconnected from the JVM, which makes Mission Control uniquely suitable for use in production environments.

The graphic shows that the management console interfaces with the JMX agent. The Flight Recorder is able to analyze and display results of generated recordings.

JVM Browser

- Invoking `jmc` brings up the JVM Browser window.
- Use the JVM Browser view to launch tools for a specific JVM process.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

ORACLE

When you launch Mission Control, the JVM Browser is opened. The JVM Browser is a tree view that you can use to manage your JVM connections. It automatically discovers locally running JVMs as well as JVMs on the network that have been started using JDP. Other plug-ins, such as the Flight Recorder tool, use the JVM Browser to access the connections to JVMs to record.

Although Mission Control should discover your JVMs automatically, you can also explicitly create a connection to a JVM. Within the JVM Browser pane, click the **Create a new custom JVM connection** button. Enter a **Host**, **Port**, and **Connection Name**. Mission Control can only connect to JVMs that have enabled remote JMX connectivity.

To start other Mission Control tools such as the Management Console, select a running JVM, right-click the JVM, and select an application.

To update user preferences for the JVM Browser or other plug-ins, select **Window > Preferences** from the main menu.

Mission Control: Management Console Overview Tab



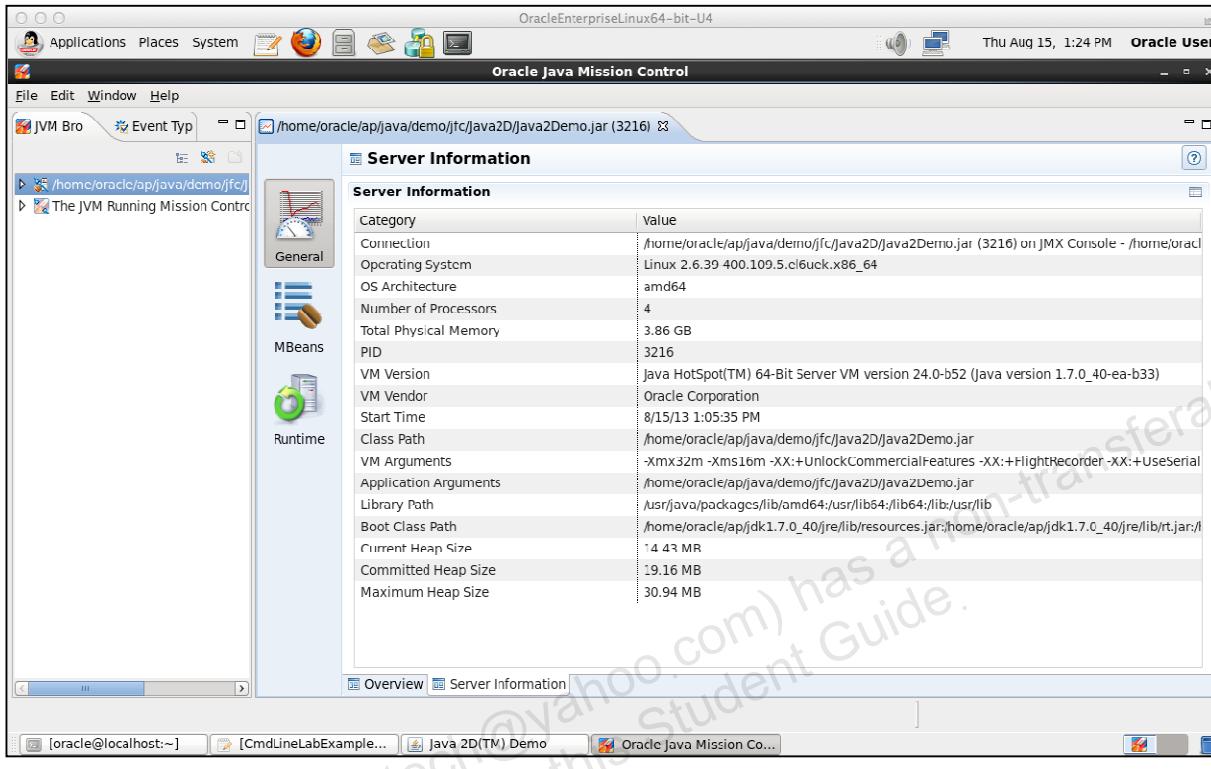
ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The Mission Control Management Console is displayed above. To open it, select a running JVM from the JVM Browser on the left side of the interface. Once selected, right-click and select **Start JMX Console**.

The General page is shown with the Overview tab by default. The page shows information about CPU Usage along with Garbage Collection memory. Any part of the Overview tab can be customized.

General: Server Info

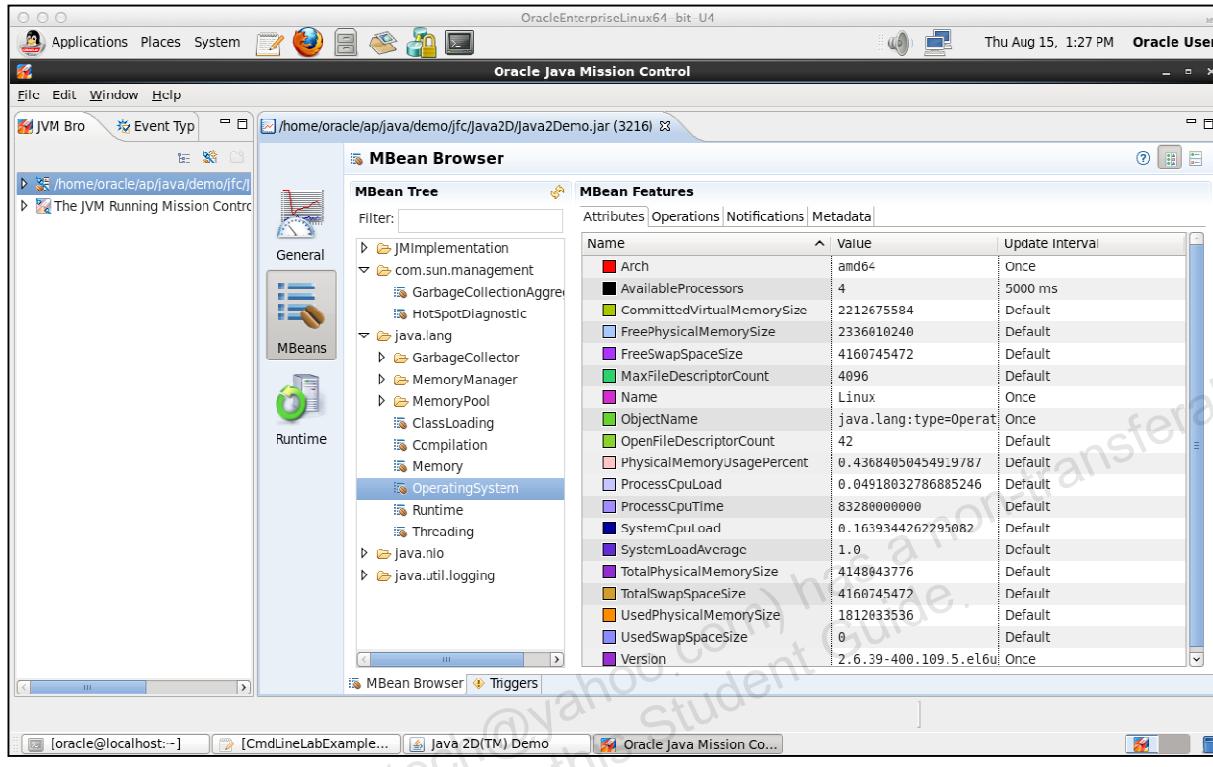


ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The Server Information tab shows key information about this JVM and the current operating system.

MBeans: Attributes

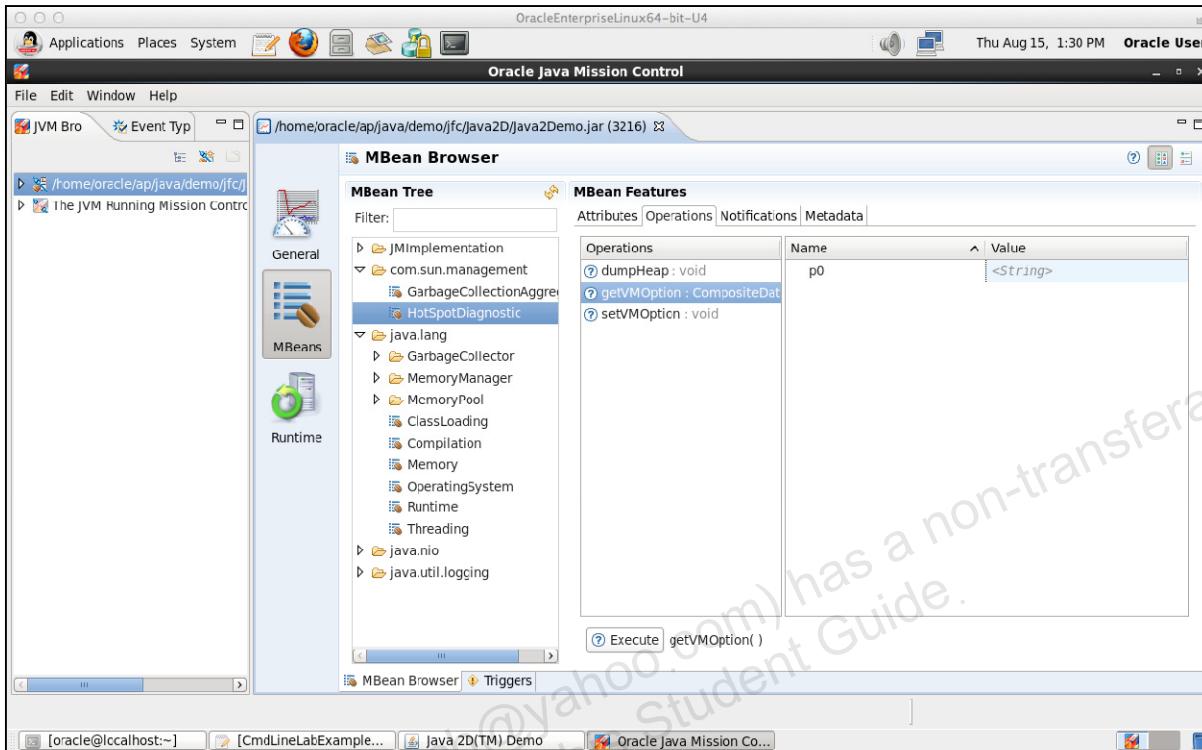


ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The MBean Browser tab provides access to detailed information about the selected JVM. The default attributes tab is shown in this screenshot. The selected **Operating System** attribute provides information about the current JVM and the operating system on which it is running.

MBeans: Operations

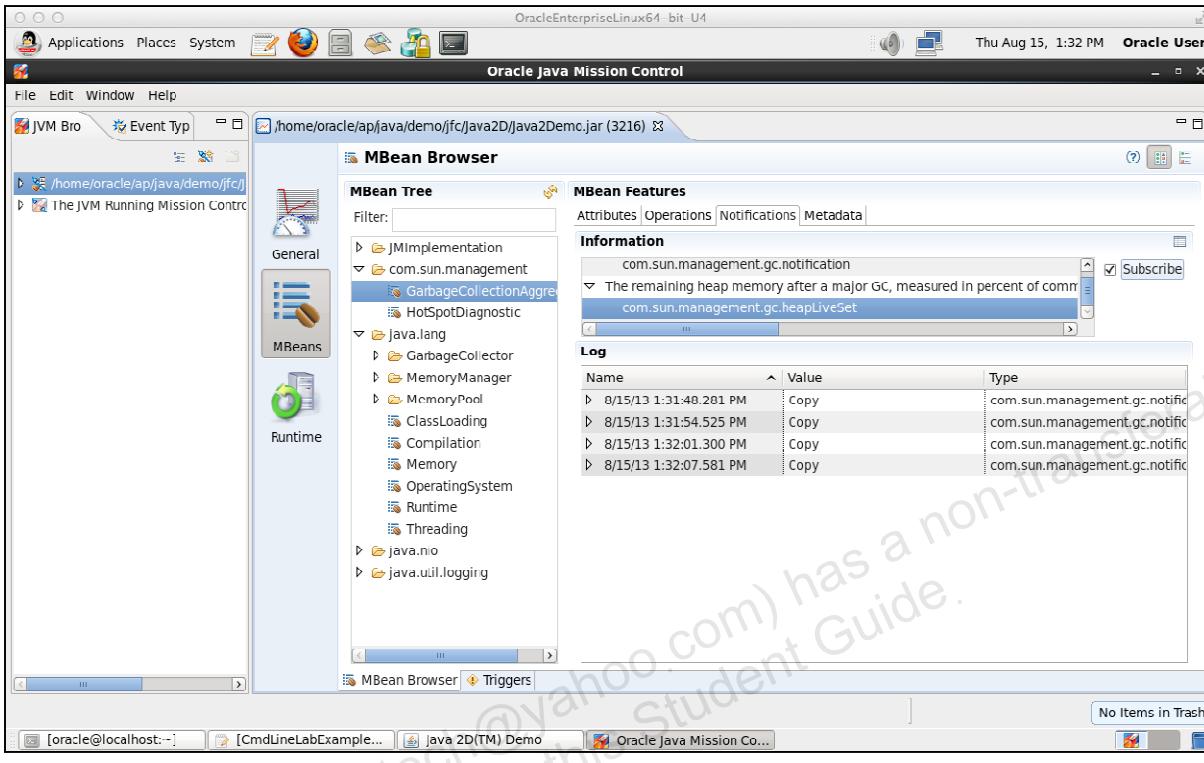


Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

ORACLE

The Operations tab provides access to special attributes that may be queried for information or changed via the interface. So it provides a way to make changes to a running JVM.

MBeans: Notifications

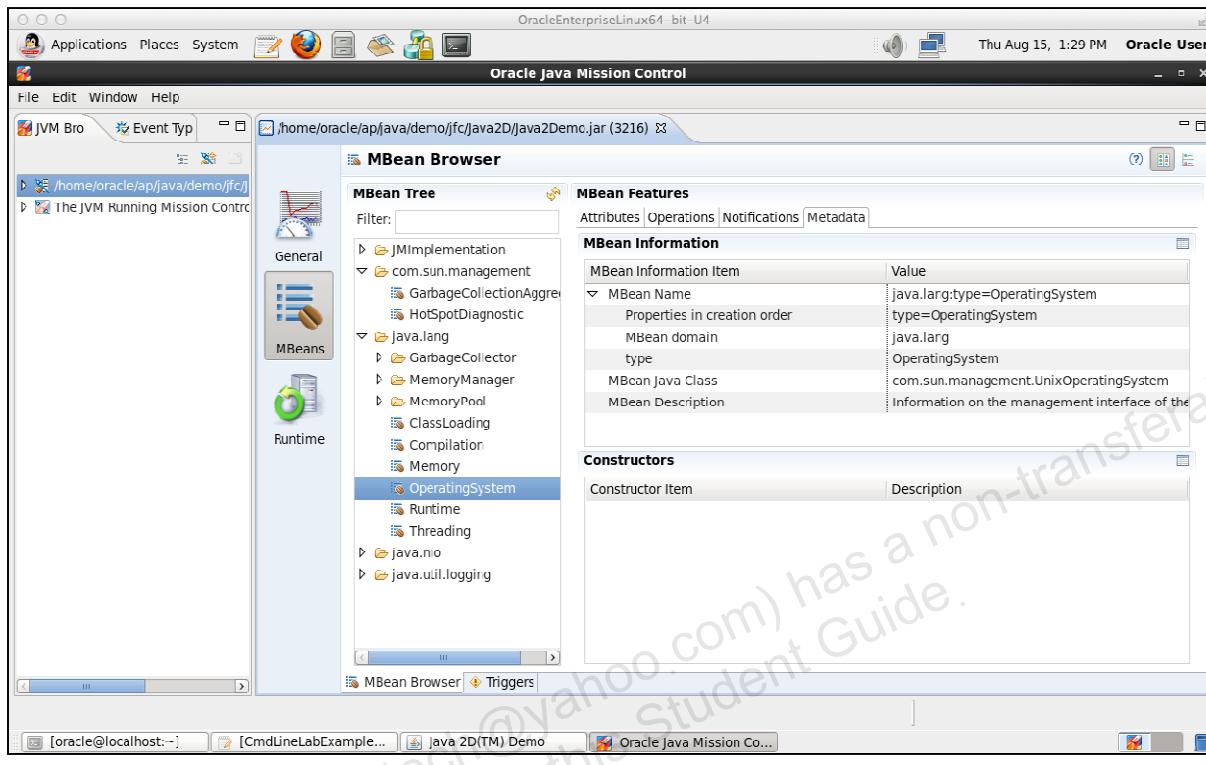


ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The Notifications tab allows you to subscribe to attributes that periodically publish data. In the example shown, data about minor garbage collections is recorded to the interface.

MBeans: Metadata

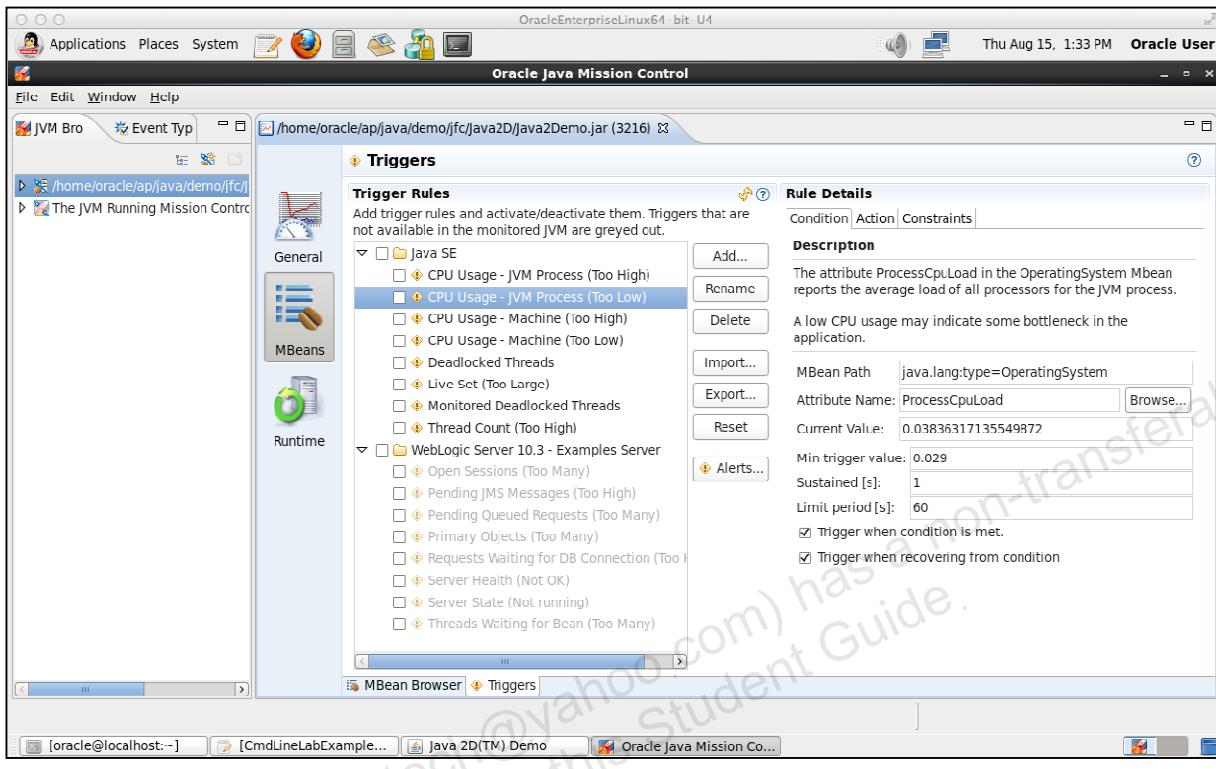


ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The Metadata tab shows detailed information about any attribute.

MBeans: Triggers

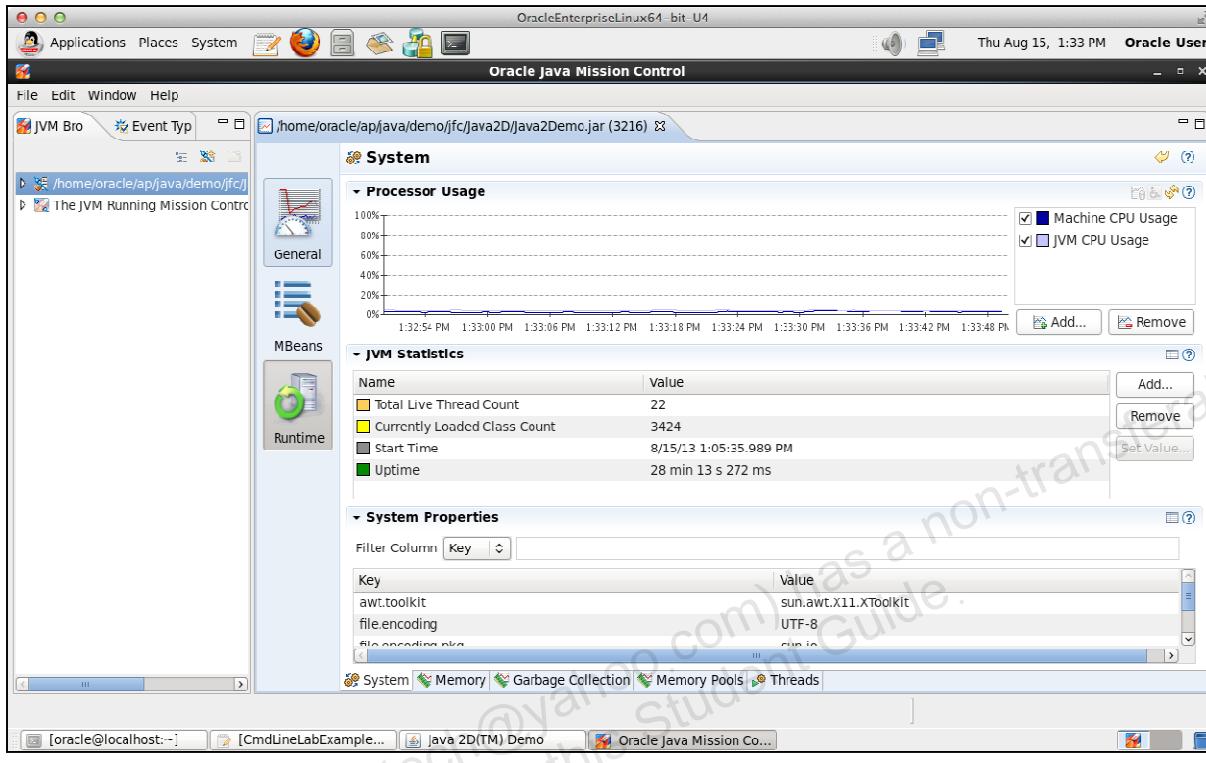


ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The Triggers tab allows you to set triggers based on JVM events. Set the conditions for the trigger as shown in this screenshot. Then, the Action tab allows you to specify how you get notified, anything from a dialog popup to an email.

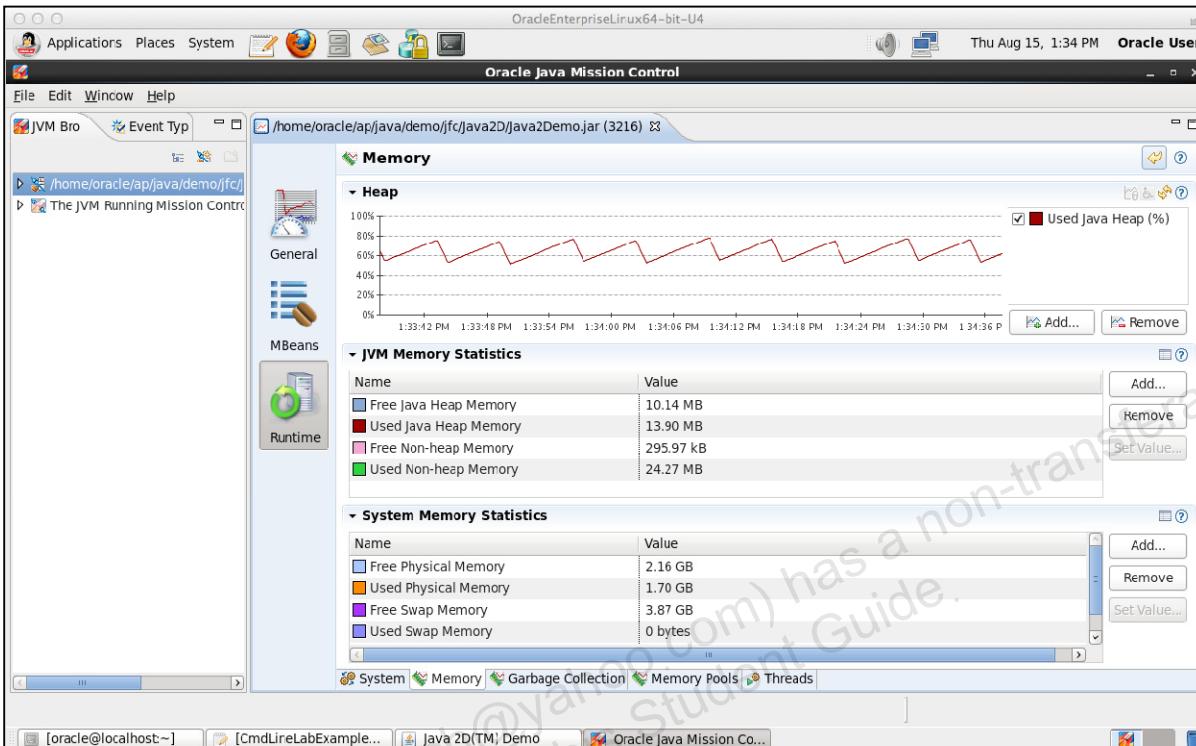
Runtime: Server



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

This is the System tab from the Runtime page. It contains general information about the JVM.

Runtime: Memory

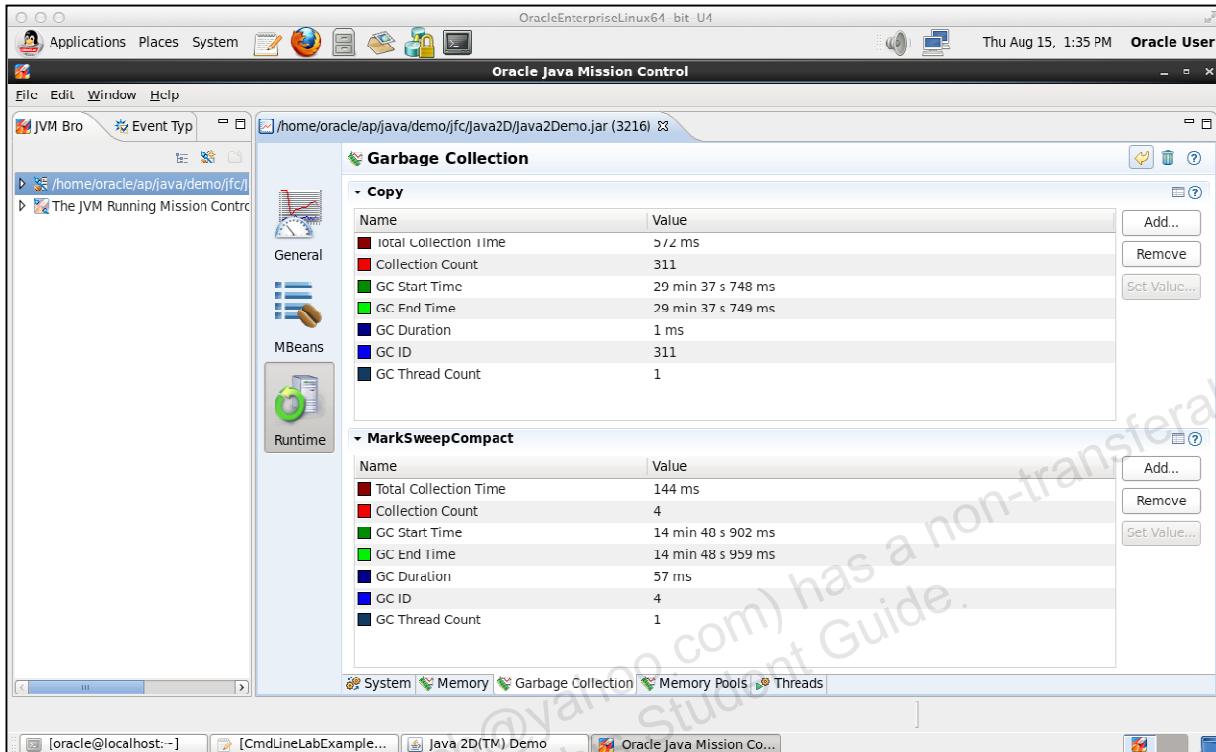


ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The Memory tab shows detailed information about the heap and memory.

Runtime: Garbage Collection

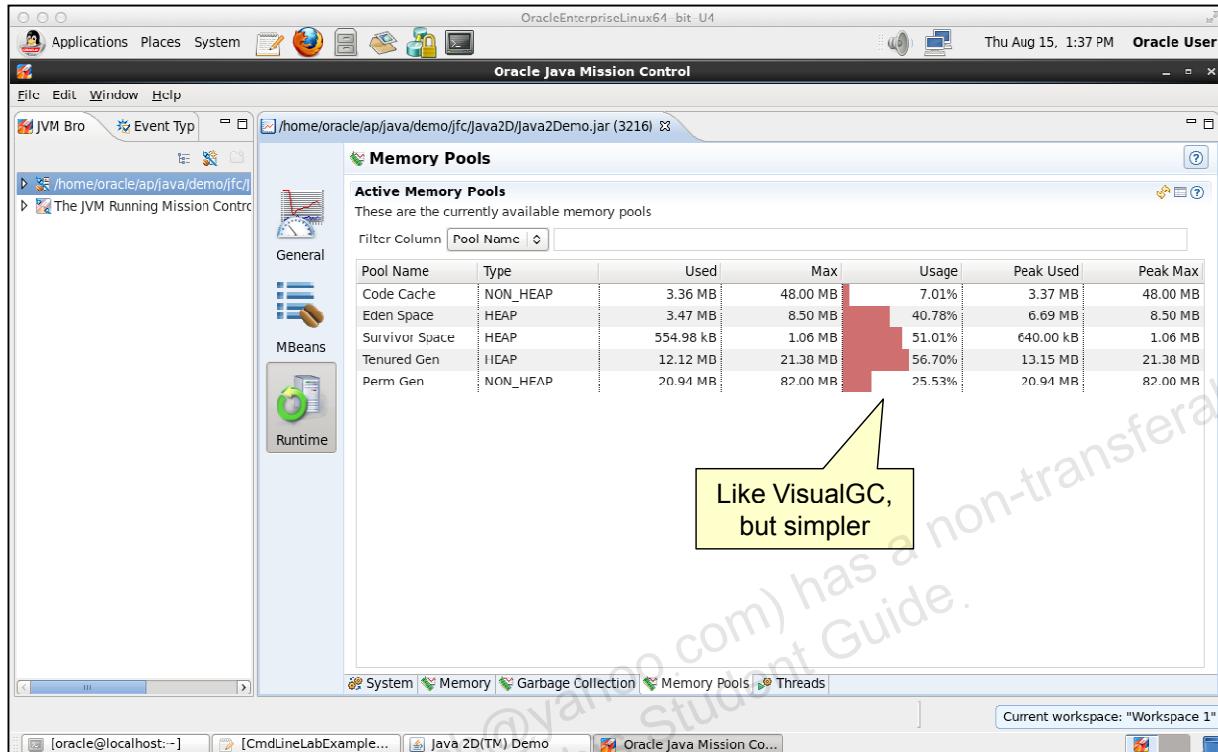


ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The Garbage Collection tab shows information about the number of collections and time-related information.

Runtime: Memory Pools

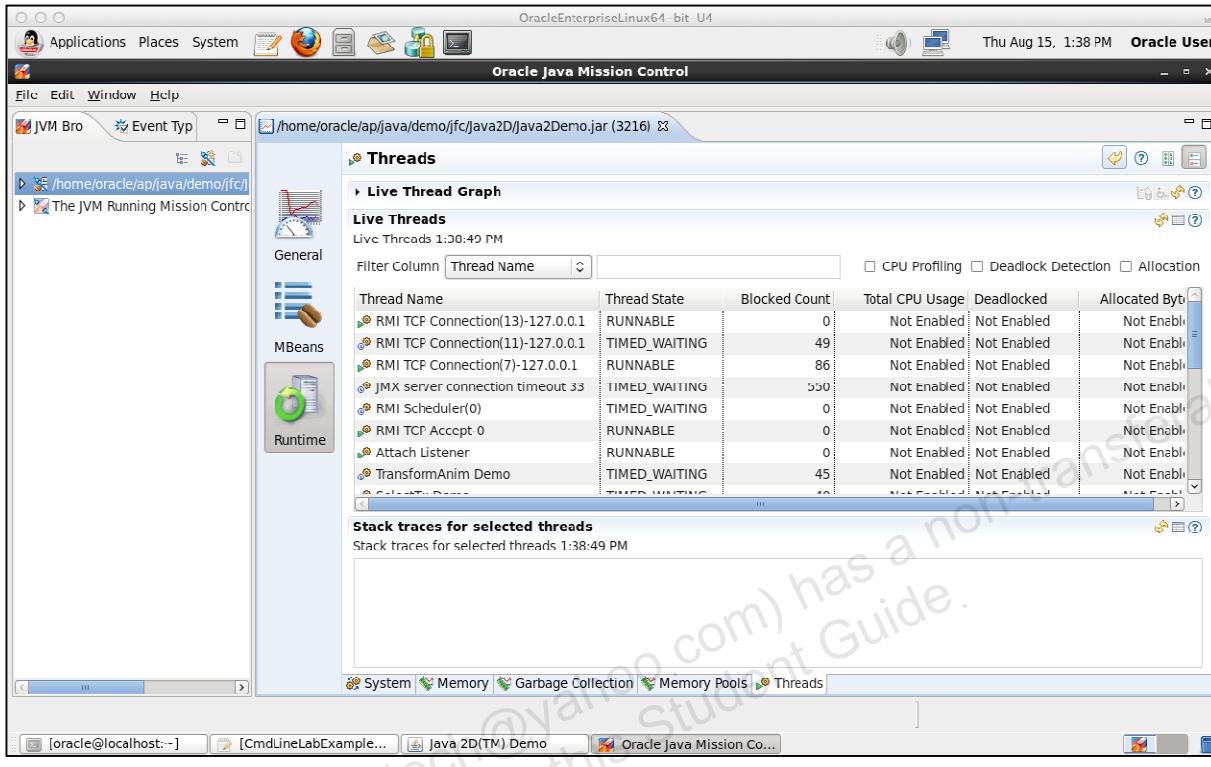


Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

ORACLE

The Memory Pools tab shows real-time information about the various memory components of the JVM.

Runtime: Threads



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The Thread tab shows information about threads in the JVM. You can enable a number of attractive features using the check boxes at the top of the table. Detailed information per thread about:

- CPU Usage
- Deadlocks
- Memory Allocation

Best Practices

- To prevent the GUI from contending for resources with WLS and its applications, run the Management Console remotely.
- To prevent unauthorized access to the JVM management server, implement restrictive mechanisms in the firewall.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

To minimize resource contention in production environments, you should run the management console from a machine different from those used to run applications. As a practice, you may want to dedicate a separate machine for running the management console and prevent other machines from accessing the management console.

Practice 4-5 Overview: Using Mission Control

This practice covers experimenting with Java Mission Control to monitor WebLogic Server JVM processes.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Flight Recorder

- The JRockit Flight Recorder is now integrated into the Hotspot JVM. It is now called the Java Flight Recorder (JFR).
- JFR consists of:
 - Runtime: The JFR runtime is the JVM engine that performs the actual recordings.
 - GUI: The user interface to view JVM recordings



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The Java Flight Recorder is a performance monitoring and profiling tool that makes diagnostics information always available, even in the wake of catastrophic failure, such as a system crash.

JFR is a rotating buffer of diagnostics and profiling data that is always available on demand. You might consider it a sort of time machine that enables you to go back in time to gather diagnostics data leading up to an event. The data stored in the rotating buffer includes JVM and application events. Because JFR is always on, using the default configuration will not result in any performance overhead.

- JFR runtime is the actual recording agent and comprises these components:
 - The Flight Recorder agent, which controls buffers, disk I/O, MBeans, and so on. This component provides a dynamic library that mixes C and Java code and also provides a JVM-independent pure Java implementation.
 - Producers, which insert data in the Flight Recorder buffer. Producers include the JVM itself, other Oracle Java applications and, through a Java API, events from third-party applications.
- The JFR graphical user interface is part of the Mission Control suite of diagnostic and profiling tools. It allows users to view JVM recordings, current recording settings, and run-time parameters.

Flight Recorder: Benefits

The Flight Recorder enables you to go back historically to gather diagnostics data leading up to an event:

Flight Recorder Use:	Description
As a black box	Provides an account of events leading up to a failure of a system or application
For real-time diagnostics	When an application detects an error or warning, it can request Flight Recorder data from the time window leading up to the error
For diagnostic profiling	Profile an application's performance during testing or in production

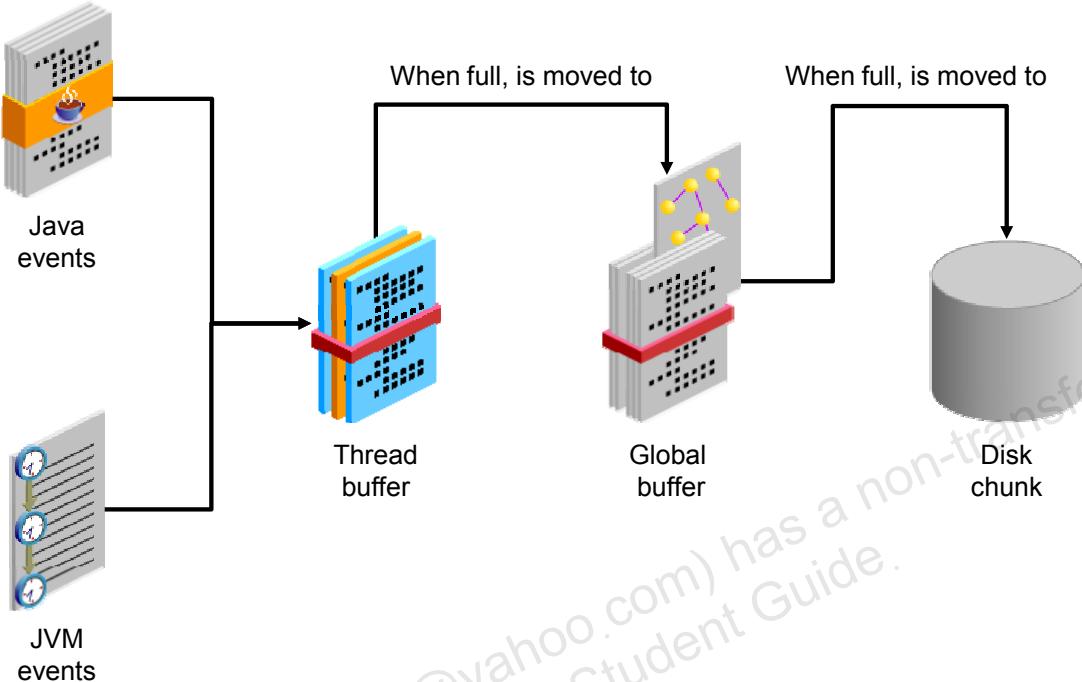


Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Flight Recorder offers a low-to-zero overhead process that has the following benefits:

- **Always on:** You no longer need to restart a broken application and create a recording. Because the Flight Recorder runs at the time the problem occurs, it provides you with information from before the time you were alerted of the problem. Runtime analysis data is always available in the buffer; providing a record of the problem you can diagnose it based upon the actual event. This feature helps diagnose intermittent problems because you do not have to start a recording and hope the problem happens again.
- **Allows for third-party event providers:** A set of APIs allow the Flight Recorder to monitor third-party applications, including WebLogic and other Oracle products. This means that you will receive JVM-level detail combined with logic from the Java application running on the JVM.
- **Reduces total cost of ownership:** Because you spend less time diagnosing and troubleshooting problems, the Flight Recorder helps reduce operating costs, reduce business interrupts, provide faster resolution time when problems occur, and improve system efficiency.

Data Flow in Flight Recorder



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Data is sent to the recorder by the JVM (through internal APIs), and by the Java application (through the Flight Recorder APIs). The Flight Recorder runtime stores this data in small thread-local buffers that are flushed to a global in-memory buffer. The data in the in-memory buffer will be moved from the in-memory buffer to a disk buffer when running in persistent storage mode.

Buffers are critical to the efficacy of the Flight Recorder. The size of the buffer determines when and how data is removed from the disk buffers. Data can be discarded after evaluating either its age or the size of the buffer.

You can configure the buffer sizing mode at the command line when you start the Flight Recorder. You can select from one of two modes:

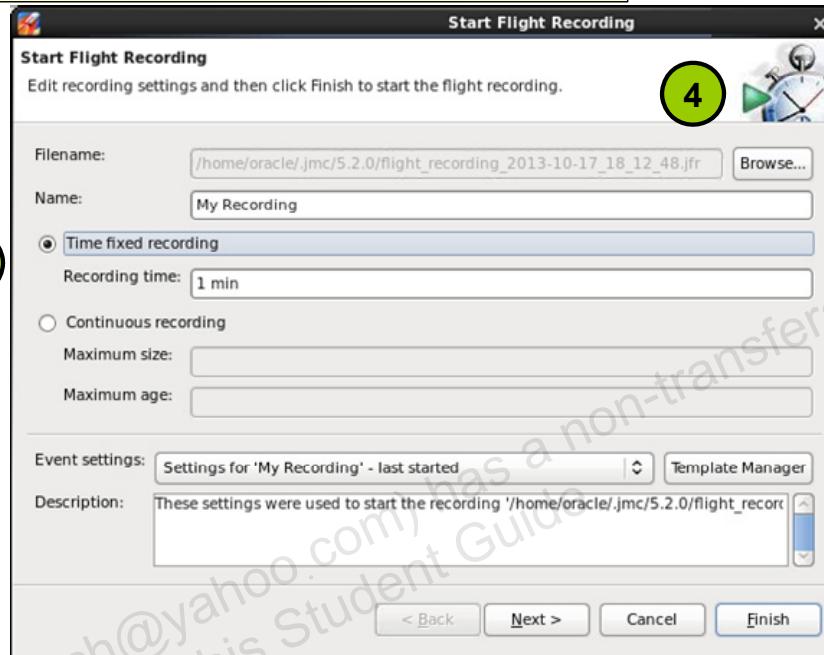
- **Time-bound mode:** In the Time-bound sizing mode, the Flight Recorder keeps data for a certain period of time. Data that has expired will be removed from the disk.
- **Space-bound sizing mode:** The Flight Recorder will always keep the size of the buffers bounded, so the maximum age of the events in the buffer may vary from time to time.

Initiating Recording Using the Management Console

Start your JVM by using the -XX:+UnlockCommercialFeatures and -XX:+FlightRecorder options.

Run your application under load.

Start JMX Console
Start Flight Recording...
Control the Remote JMX Agent



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The screenshot shows an example of initiating performance data recording by using the Management Console. After a recording is completed, Mission Control opens the recording in a new tab and shows the view of performance data. You can save this data for further analysis and comparison with other recordings.

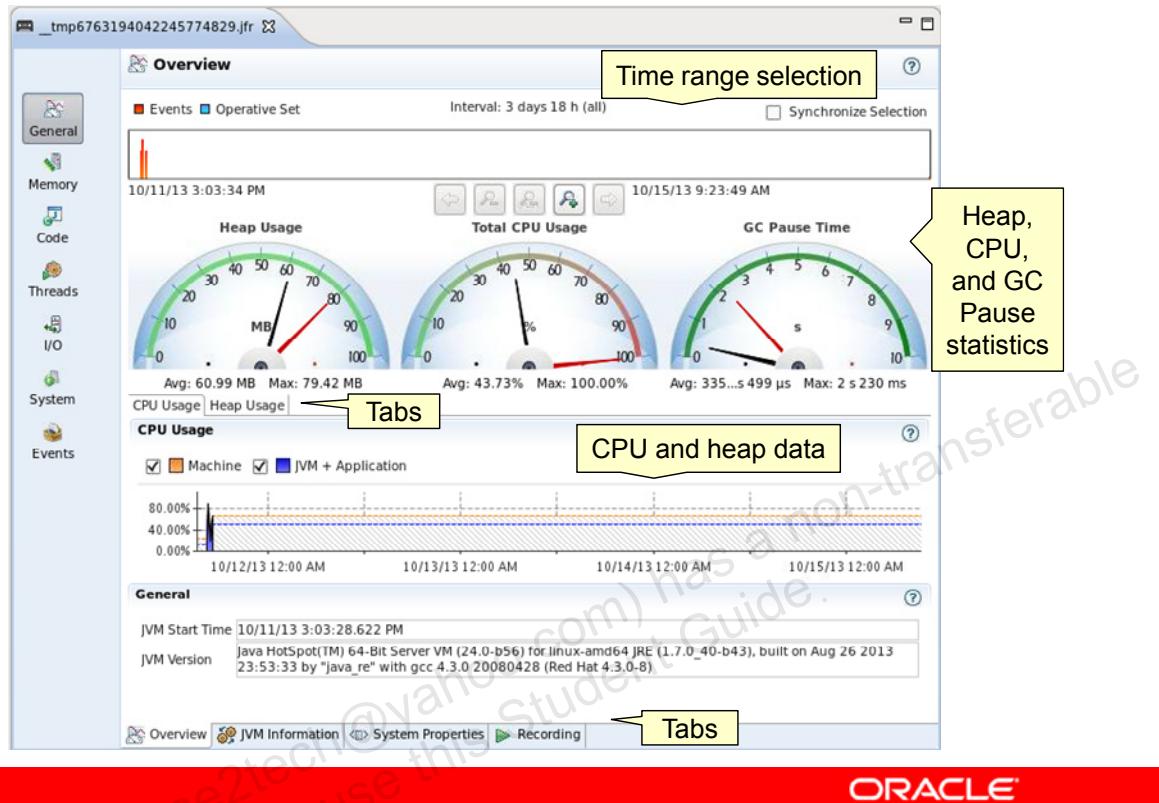
Tab Groups of Flight Recorder

Description	
 General	General information about the recording and the recorded application and JVM
 Memory	Information about memory management and garbage collection
 Code	Information about methods, optimizations, and exceptions
 Threads	Information about the threads and locks in the recorded application
 I/O	Information about file and network reads and writes
 System	Information about underlying hardware and system properties
 Events	Generic information about selected events based upon event types selected in the Events Type View

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Flight Recorder General Tab

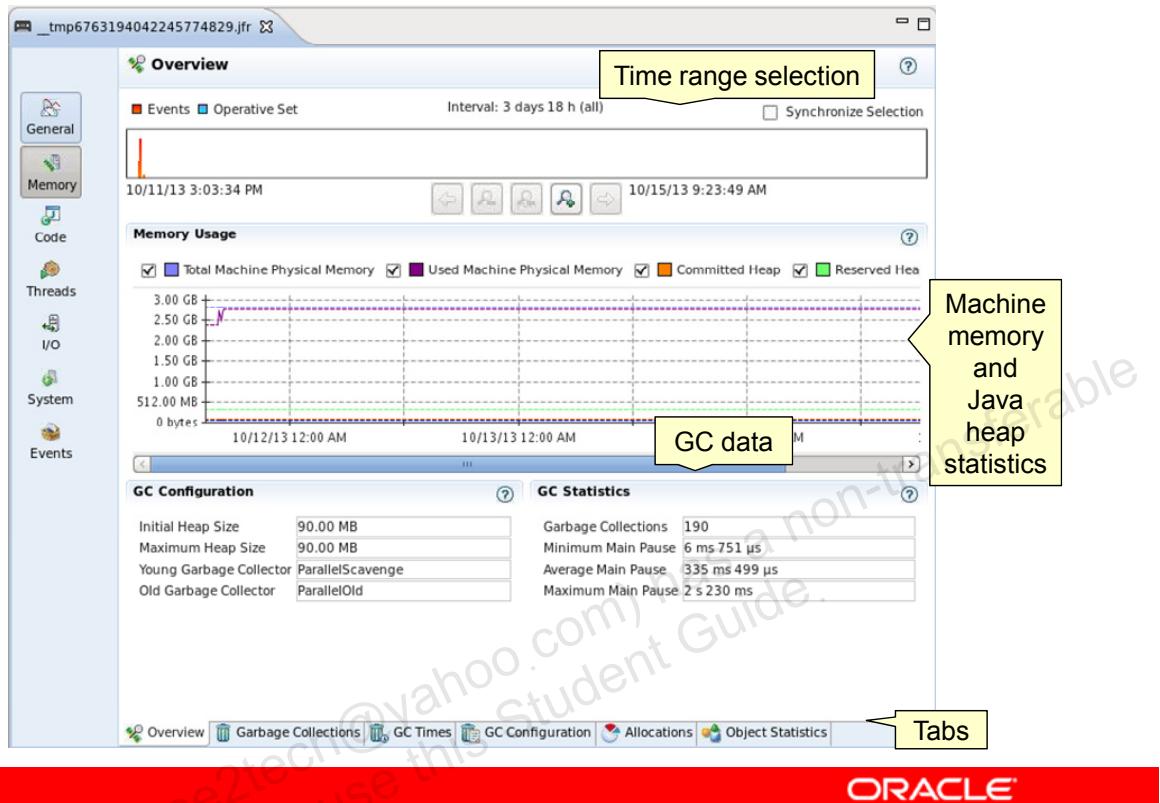


Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

ORACLE

The General tab of the Flight Recorder displays high-level information about the recording. It displays memory usage, CPU usage, and system properties of the JVM. The VM arguments that you specified on the command line when starting the JVM are also available on the General tab. You can achieve a very high level of understanding by investigating this data. For more information about the specific areas, further investigation is required on the subtabs on the bottom of the page, or the other tab groups.

Flight Recorder Memory: Overview Tab



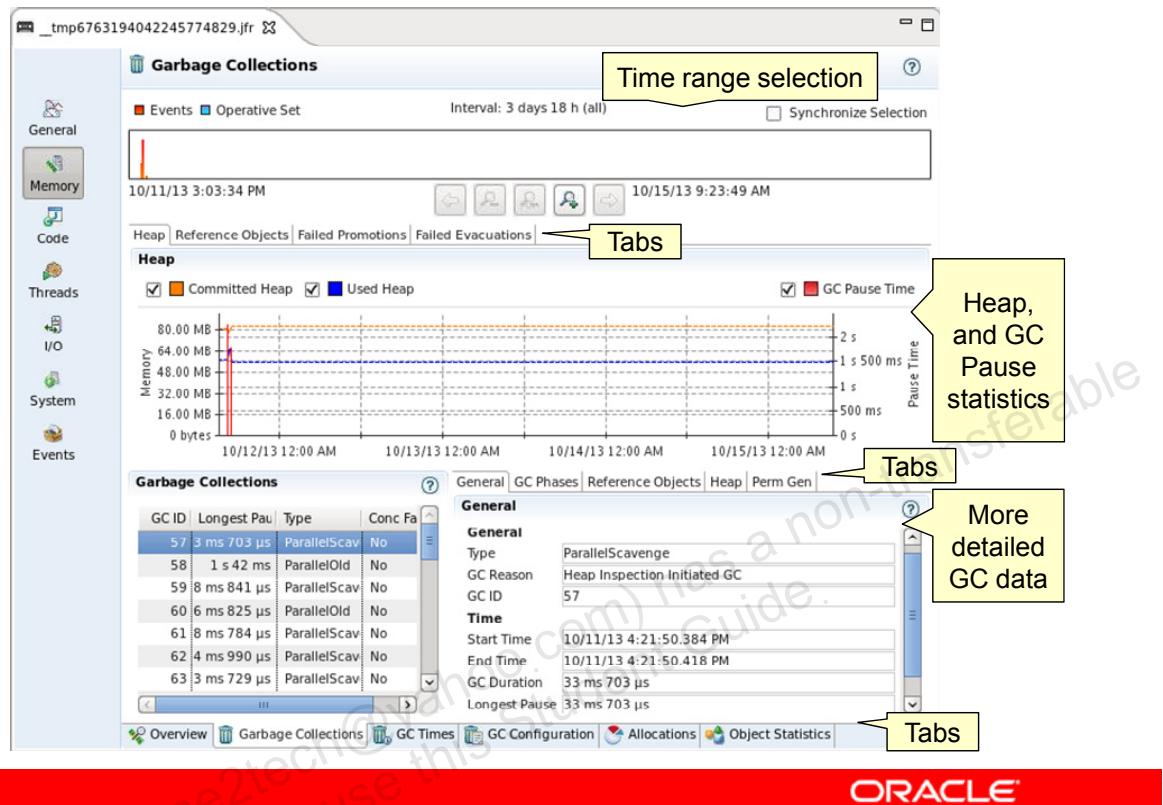
Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The Overview tab provides the broad picture of the memory usage during the Flight Recorder session. The tab has two graphs:

- **The Events and Operatives Set:** Shows the graph of when a particular event occurred. By default a set of events, such as garbage collection, are predefined. In addition, you can also define events based on your application logic.
- **Memory Usage:** Shows a graphic representation of the memory used during a Flight Recorder session.

The two tables on the bottom, namely GC Configuration and GC Statistics, provide average and maximum statistics relating to the memory (heap) usage and garbage collection during the session.

Flight Recorder Memory: GC Collections Tab

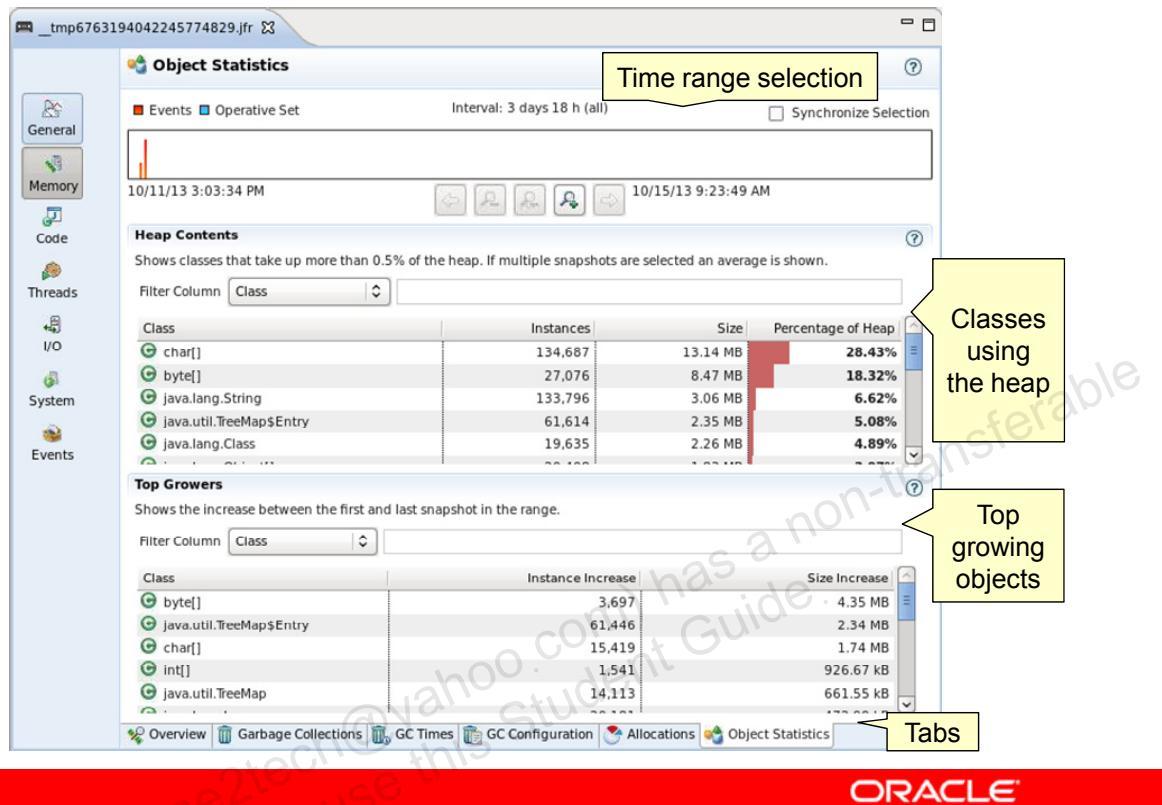


Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The GC Collections tab gives information about the garbage collected during the recording. The charts show heap usage.

In the Garbage Collections table, you see a list of garbage collection with the time taken and the generation of heap affected by garbage collection. When you select a GC in the list, you can find more details under the appropriate subtab in the Details table.

Flight Recorder Memory: Object Statistics



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The **Object Statistics** tab shows the number of instances and cumulative memory used by different classes during the Flight Recorder session. The Top Growers table provides details of the classes that are growing. Unbridled growth of objects can indicate a memory leak.

Section Summary

In this section, you should have learned how to use GUI JVM monitoring tools.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Practice 4-6 Overview: Using Flight Recorder

This practice covers experimenting with Java Flight Recorder to monitor WebLogic Server JVM processes.