Oracle® WebLogic Server

Monitoring and Managing with the Java EE Management APIs

10*g* Release 3 (10.3)

July 2008

ORACLE

Oracle WebLogic Server Monitoring and Managing with the Java EE Management APIs, 10*g* Release 3 (10.3)

Copyright © 2007, 2008, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

1. Introduction and Roadmap

Document Scope and Audience 1-1

Guide to This Document 1-2

Related Documentation 1-2

2. Using the Java EE Management APIs on WebLogic Server

Understanding the Java EE Management Model and APIs 2-1

JMO Hierarchy 2-2

JMO Object Names. 2-2

Optional Features of JMOs 2-2

Accessing JMOs 2-3

The Java EE Management Model on WebLogic Server 2-3

Accessing the MEJB on WebLogic Server 2-3

Example: Querying Names of JMOs 2-4

CHAPTER 1

Introduction and Roadmap

The Java EE Management specification describes a standard data model for monitoring and managing the runtime state of any Java EE Web application server and its resources. It includes standard mappings of the model through a Java EE Management EJB Component (MEJB).

The following sections describe the contents and organization of this guide—*Monitoring and Managing with the J2EE Management APIs*:

- "Document Scope and Audience" on page 1-1
- "Guide to This Document" on page 1-2
- "Related Documentation" on page 1-2

Document Scope and Audience

This document is a resource for software developers who develop management services for Java EE applications and for software vendors who develop JMX-compatible management systems. It also contains information that is useful for business analysts and system architects who are evaluating WebLogic Server or considering the use of JMX for a particular application.

The information in this document is relevant during the design and development phases of a software project. The document does not address production phase administration, monitoring, or performance tuning topics. For links to WebLogic Server documentation and resources for these topics, see "Related Documentation" on page 1-2.

It is assumed that the reader is familiar with Java EE and general application management concepts. This document emphasizes a hands-on approach to developing a limited but useful set

of JMX management services. For information on applying JMX to a broader set of management problems, refer to the JMX specification or other documents listed in "Related Documentation" on page 1-2.

Guide to This Document

This document is organized as follows:

- This chapter, Introduction and Roadmap, describes the scope and organization of this guide.
- Chapter 2, "Using the Java EE Management APIs on WebLogic Server," introduces JMX and describes common ways to use it in conjunction with other WebLogic Server management features.

Related Documentation

The Sun Developer Network includes a Web site that provides links to books, white papers, and additional information on JMX:
http://java.sun.com/javase/technologies/core/mntr-mgmt/javamanagement/.

To view the JMX 1.2 specification and API documentation, download it from
http://jcp.org/aboutJava/communityprocess/final/jsr003/index3.html.

To view the JMX Remote API 1.0 specification and API documentation, download it from
http://jcp.org/aboutJava/communityprocess/final/jsr160/index.html.

For guidelines on developing other types of management services for WebLogic Server applications, see the following documents:

- *Using WebLogic Logging Services for Application Logging* describes WebLogic support for internationalization and localization of log messages, and shows you how to use the templates and tools provided with WebLogic Server to create or edit message catalogs that are locale-specific.
- *Configuring and Using the WebLogic Diagnostic Framework* describes how system administrators can collect application monitoring data that has not been exposed through JMX, logging, or other management facilities.

For guidelines on developing and tuning WebLogic Server applications, see the following documents:

- *Developing Applications with WebLogic Server* is a guide to developing WebLogic Server applications.
- *Developing Manageable Applications with JMX* describes how to create and register custom MBeans.

CHAPTER 2

Using the Java EE Management APIs on WebLogic Server

The Java EE Management APIs enable a software developer to create a single Java program that can discover and browse resources, such as JDBC connection pools and deployed applications, on any Java EE Web application server. The APIs are part of the Java EE Management Specification, which requires all Java EE Web application servers to describe their resources in a standard data model.

The following sections describe how to use the Java EE Management APIs on WebLogic Server:

- "Understanding the Java EE Management Model and APIs" on page 2-1
- "The Java EE Management Model on WebLogic Server" on page 2-3
- "Accessing the MEJB on WebLogic Server" on page 2-3

Understanding the Java EE Management Model and APIs

In the Java EE Management data model, each instance of a Web application server resource type is represented by a Java EE Managed Object (JMO). The Java EE Management Specification describes exactly which types of resources must be represented by a JMO. JMOs themselves contain only a limited set of attributes, which are used to describe the location of the object in the data model.

Download the Java EE Management Specification from http://jcp.org/aboutJava/communityprocess/final/jsr077/index.html.

JMO Hierarchy

The data model organizes JMOs hierarchically in a tree structure. The root JMO is `J2EEDomain`, which represents a collection of Web application server instances that are logically related. `J2EEDomain` contains the object names for all instances of the `J2EEServer` JMO, each of which represents a server instance in the collection.

Java applications can browse the hierarchy of JMOs, recursively querying for object names and looking up the JMOs that are named by the query results.

JMO Object Names

Each JMO instance is identified by a unique object name of type `javax.management.ObjectName`. The names follow this pattern:

`domain``:`*`name`*`=j2eeType=`*`value`*`,name=`*`value`*`,`*`parent-j2eeType[,property=value]*`

For example, `mydomain:J2EEtype=J2EEDomain,name=mydomain`

The Java EE Management Specification describes exactly which name/value pairs must be in the object names for each JMO type.

The object name for each child JMO contains name/value pairs from its parent JMO's object name. For example, if the JMO for a server instance is named `mydomain:j2eeType=J2EEServer,name=myserver`

then the JMO for a servlet that is part of an application deployed on that server instance would be named:

```
mydomain:J2EEApplication=myapplication,J2EEServer=myserver,WebModule=myapp
_mywebmodule,j2eeType=Servlet,name=myservlet_name
```

The name/value pairs can appear in any order.

Optional Features of JMOs

The Java EE Management Specification, version 1.0, requires only that Web application servers implement JMOs and provide API access to the JMOs.

Optionally, you can implement the JMOs to provide performance statistics, management operations, and to emit notifications when specified events occur.

Accessing JMOs

A Java application accesses the JMOs through `javax.management.j2ee.Management`, which is the remote interface for the Management Enterprise Java Bean (MEJB).

The Java EE Management Specification requires that the MEJB's home interface be registered in a server's JNIDI tree as `ejb.mgmt.MEJB`.

See the API Reference for the `javax.management.j2ee` package: http://java.sun.com/j2ee/1.4/docs/api/javax/management/j2ee/package-summary.html.

The Java EE Management Model on WebLogic Server

WebLogic Server 9.0 implements only the required features of the Java EE Management Specification, version 1.0. Therefore, the following limitations are in place:

- None of the JMOs provide performance statistics, management operations, or emit notifications.
- There are no mappings to the Common Information Model (CIM).
- There are no mappings to an SNMP Management Information Base (MIB).

The MEJB and JMOs are available only on the Administration Server. This is consistent with the Java EE Management Model, which assumes that most Java EE Web servers exist within some logically connected collection and that there is a central point within the collection for accessing or managing the server instances. From the Administration Server, a Java application can browse to the JMO that represents any resource on any server instance in the WebLogic Server domain.

Because WebLogic Server implements its JMOs as a wrapper for its MBeans, any changes in a WebLogic Server MBean that corresponds to a JMO is immediately available through the Java EE Management APIs.

For all JMO object names on WebLogic Server, the *`domain`*`:` portion of the object name corresponds to the name of the WebLogic Server domain.

Accessing the MEJB on WebLogic Server

To retrieve monitoring data through the MEJB:

1. Look up the `javax.management.j2ee.ManagementHome` interface through the Administration Servers JNDI tree under the name `ejb.mgmt.MEJB`.

2. Use `ManagementHome` to construct an instance of `javax.management.j2ee.Management`, which is the MEJB's remote interface.

Example: Querying Names of JMOs

The example class in accesses the MEJB for a WebLogic Server domain and invokes `javax.management.j2ee.Management.queryNames` method. This method returns the object name for all JMOs in the domain.

Listing 2-1 Querying Names of JMOs

```
import java.io.IOException;
import java.net.MalformedURLException;
import java.util.Iterator;
import java.util.Set;
import java.util.Properties;

import javax.management.j2ee.Management;
import javax.management.j2ee.ManagementHome;
import javax.management.AttributeNotFoundException;
import javax.management.InstanceNotFoundException;
import javax.management.ObjectName;
import javax.management.QueryExp;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;

import javax.ejb.CreateException;

public class GetJMONames {

   static String url = "t3://localhost:7001";
   static String user = "weblogic";
   static String password = "weblogic";

   public static void main(String[] args) {
      try {
         getAllJMONames();
      }catch(Exception e){
      System.out.println(e);
```

```
   }
}

public static Management getMEJBRemote()
    throws IOException, MalformedURLException,
    NamingException,CreateException
{
   Context context = getInitialContext();
   ManagementHome home = (ManagementHome)
       context.lookup("ejb.mgmt.MEJB");
   Management bean = home.create();
   return bean;
}

public static Context getInitialContext()
       throws NamingException
{
   Properties p = new Properties();
   p.put(Context.INITIAL_CONTEXT_FACTORY,
       "weblogic.jndi.WLInitialContextFactory");
   p.put(Context.PROVIDER_URL, url);
   if (user != null) {
      p.put(Context.SECURITY_PRINCIPAL, user);
      if (password == null)
         password = "";
         p.put(Context.SECURITY_CREDENTIALS, password);
      }
   return new InitialContext(p);
}

public static void getAllJMONames()
{
   try {
      Management rhome = getMEJBRemote();
      String string = "";
      ObjectName name = new ObjectName(string);
      QueryExp query = null;

      Set allNames = rhome.queryNames(name, query);
      Iterator nameIterator = allNames.iterator();
```

```
            while(nameIterator.hasNext()) {
               ObjectName on = (ObjectName)nameIterator.next();
               System.out.println(on.getCanonicalName() + "\n");
            }
         } catch (Exception ex) {
               ex.printStackTrace();
         }
      }
   }
```