Oracle® Fusion Middleware

Developing Java EE Management Applications for Oracle WebLogic Server

12*c* (12.2.1)

E55226-01

October 2015

This document describes the Java EE Management APIs which 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.

ORACLE

Oracle Fusion Middleware Developing Java EE Management Applications for Oracle WebLogic Server, 12*c* (12.2.1)

E55226-01

Contents

Preface

This preface describes the document accessibility features and conventions used in this guide—*Developing Java EE Management Applications for Oracle WebLogic Server*.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1 Introduction and Roadmap

This chapter describes the contents and audience for this guide—*Developing Java EE Management Applications for Oracle WebLogic Server*.

This chapter includes the following sections:

1.1 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 Section 1.3, "Related Documentation."

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 Section 1.3, "Related Documentation."

1.2 Guide to This Document

This document is organized as follows:

- This chapter, Chapter 1, "Introduction and Roadmap," describes the scope and organization of this guide.
- Chapter 2, "Using the Java EE Management APIs," introduces JMX and describes common ways to use it in conjunction with other WebLogic Server management features and describes WebLogic-specific extensions to JSR 77.

1.3 Related Documentation

Oracle has a Web site that provides links to books, white papers, and additional information on JMX: http://www.oracle.com/technetwork/java/javase/tech/javamanagement-140525.html.

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:

- *Adding WebLogic Logging Services to Applications Deployed on Oracle WebLogic Server* 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 Diagnostics Framework for Oracle WebLogic Server* 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 Using JMX for Oracle WebLogic Server* describes how to create and register custom MBeans.

1.4 New and Changed Features in this Release

For a comprehensive listing of the new WebLogic Server features introduced in this release, see *What's New in Oracle WebLogic Server 12.2.1*.

2

Using the Java EE Management APIs

This chapter describes the Java EE Management APIs, which 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 Java EE Management specification describes a standard data model for monitoring and managing the run-time 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).

This chapter includes the following sections:

2.1 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:

2.1.1 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`.

2.1.1.1 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.

2.1.1.2 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.

2.1.1.3 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.

2.1.1.4 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://docs.oracle.com/javaee/6/api/javax/management/j2ee/package-summary.html.

2.1.2 The Java EE Management Model on WebLogic Server

WebLogic Server implements only the required features of the Java EE Management Specification, version 1.1. 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.

2.1.3 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.

2.1.3.1 Example: Querying Names of JMOs

The example class in Example 2–1 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.

Example 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();
       }
    }
}
```

2.2 WebLogic Server Extensions

WebLogic Server implements an extension to JSR 77 that gives you access to WebLogic-specific deployment descriptors using the MEJB, just like the standard Java EE deployment descriptors. The `productSpecificDeploymentDescriptor` attribute returns the XML contents of the WebLogic-specific descriptor file. Example 2–2 illustrates calling the method.

Example 2–2 productSpecificDeploymentDescriptor

```
// Get the WLS specific deployment descriptor.
// This is similar to the call for the standard descriptor
// (i.e., the "deploymentDescriptor" attribute)
//
```

```
dd = (String) managementBean.getAttribute(objName,
"productSpecificDeploymentDescriptor");

// It returns a string containing the contents of the WLS specific deployment
// descriptor. This is the XML file contents as a string.
```