BEA WebLogic Server™

BEA WebLogic Server Partners' Guide

Release 8.1 Beta
Document Revised: June 28 2002

Copyright

Restricted Rights Legend

Trademarks or Service Marks

BEA WebLogic Server Partners’ Guide

Part Number	Document Revised	Software Version
N/A	June 28, 2002	BEA WebLogic Server Version 8.1 beta

Contents

About This Document

1. Distributing WebLogic Server

2. Customizing WebLogic Server Configuration Files

3. Using JDBC Profiling MBeans

About This Document

This document describes how to acquire and install an Independent Software Vendors (ISV) license, which enables you to bundle BEA's core technologies with your application and distribute both items as a single product. The document also suggests development techniques for bundling BEA WebLogic Server™ with your applications.

The document is organized as follows:

- Chapter 1, "Distributing WebLogic Server," which describes how to acquire and install an ISV license and specifies which WebLogic Server files must be included in your distribution.
- Chapter 2, "Customizing WebLogic Server Configuration Files," which highlights typical modifications that partners and ISVs make to the WebLogic Server configuration files that they distribute with their applications.
- Chapter 3, "Using JDBC Profiling MBeans," which describes how to enable and use JDBC profiling.

Audience

This document is written for independent software vendors (ISVs) and other developers who are interested in creating custom applications that use BEA WebLogic Server core technologies. It is assumed that readers are already familiar with the BEA WebLogic Server platform, other guides in the WebLogic Server documentation set, and the Java programming language.

e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on Product Documentation or go directly to the WebLogic Server Product Documentation page at http://e-docs.bea.com.

How to Print the Document

You can print a copy of this document from a Web browser, one main topic at a time, by using the File—Print option on your Web browser.

A PDF version of this document is available on the WebLogic Server documentation Home page on the e-docs Web site (and also on the documentation CD). You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format. To access the PDFs, open the WebLogic Server documentation Home page, click Download Documentation, and select the document you want to print.

Adobe Acrobat Reader is available at no charge from the Adobe Web site at http://www.adobe.com.

Related Information

The BEA corporate Web site provides all documentation for WebLogic Server. The following BEA WebLogic Server documentation contains information that is relevant to understanding how to extend WebLogic Server.

- BEA WebLogic Server Documentation (available online):
  - *Administration Guide*
  - *Programming Guides*
  - WebLogic Server API

- The Sun Microsystems, Inc. Java site at http://java.sun.com/

For more information about BEA WebLogic Server and Java, refer to the Bibliography at http://edocs.bea.com/.

Contact Us!

Your feedback on BEA documentation is important to us. Send us e-mail at docsupport@bea.com if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the documentation.

In your e-mail message, please indicate the software name and version your are using, as well as the title and document date of your documentation. If you have any questions about this version of BEA WebLogic Server, or if you have problems installing and running BEA WebLogic Server, contact BEA Customer Support through BEA WebSupport at http://www.bea.com. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes
- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Usage
Ctrl+Tab	Keys you press simultaneously.
italics	Emphasis and book titles.

`monospace text`	Code samples, commands and their options, Java classes, data types, directories, and file names and their extensions. Monospace text also indicates text that you enter from the keyboard.
	Examples:
	`import java.util.Enumeration;`
	`chmod u+w *`
	`config/examples/applications`
	`.java`
	`config.xml`
	`float`

`monospace italic text`	Variables in code. *Example*: `String CustomerName;`
UPPERCASE TEXT	Device names, environment variables, and logical operators. *Examples*: LPT1 BEA_HOME OR
`{ }`	A set of choices in a syntax line.
`[ ]`	Optional items in a syntax line. *Example*: `java utils.MulticastTest -n name -a address [-p portnumber] [-t timeout] [-s send]`
`|`	Separates mutually exclusive choices in a syntax line. *Example*: `java weblogic.deploy [list|deploy|undeploy|update] password {application} {source}`

Convention	Usage
. . .	Indicates one of the following in a command line: ■ An argument can be repeated several times in the command line. ■ The statement omits additional optional arguments. ■ You can enter additional parameters, values, or other information
. . .	Indicates the omission of items from a code example or from a syntax line.

CHAPTER

1 Distributing WebLogic Server

Instead of requiring your customers to purchase, install, and maintain both your application and a J2EE application server, you can bundle BEA's core technologies with your application and distribute both items as a single product. The plug-and-play environment of WebLogic Server makes it an ideal choice for integration with your product.

To distribute WebLogic Server, you must obtain and install a special license called an **ISV license**. (You do not need an ISV license to develop your application or to configure WebLogic Server.) Installing an ISV license for a WebLogic Server modifies the server and inextricably links files. Your distribution must include these modified WebLogic Server files.

To set up a WebLogic Server that you can distribute, complete the following tasks:

- Enroll in the BEA Star Partner Program
- Install the Partner Development Kit
- Install the ISV License
- Distribute Files

Enroll in the BEA Star Partner Program

BEA Systems, Inc. manages its relationships with Independent Software Vendors (ISVs) and Application Software Providers (ASPs) through the Star Partner Program. For more information about this program, including information on how to enroll, refer to the following topics:

- BEA WebLogic Server Platform Support, http://www.weblogic.com/platforms/index.html, to verify that your target platform is certified for use with WebLogic Server
- BEA Star Partner Program, http://www.bea.com/partners/index.shtml
- Become a BEA Star Partner, http://www.bea.com/partners/become.shtml

Install the Partner Development Kit

After you enroll in the program, BEA ships a CD collection of all major BEA products. When the Partner Development Kit arrives, install the software from the CDs. For information on installing WebLogic Server, refer to the Installation Guide on the BEA documentation Web site, http://edocs.bea.com.

Caution: If you already have BEA products installed on the computer that you want to host your distributable WebLogic Server, **back up your current *`BEA_HOME`*`\license.bea` file** before installing the Partner Development Kit. For more information about the BEA home directory and the `license.bea` file, refer to BEA Home Directory in the *Installing BEA WebLogic Server* guide.

Instead of waiting for the CDs, you can download BEA software from the BEA Systems Download Center, http://commerce.beasys.com/downloads/products.jsp. If you have an active WebSUPPORT account, you can use your WebSUPPORT login password for software downloads.

Install the ISV License

After verifying your eligibility for the Star Partner Program, BEA sends an email that includes your customized ISV license in an attached file named `isv.jar`. This section describes how to install the ISV license file **for WebLogic Server version 7.0 only**. If you are installing an ISV license for other versions of WebLogic Server, please consult the relevant installation instructions for your software version.

There are three main steps to installing an ISV license:

- Step 1: Preparing to Install an ISV License
- Step 2: Extracting the License Data and Linking WebLogic Server Files
- Step 3: Updating the WebLogic Server License

Step 1: Preparing to Install an ISV License

Before you install an ISV license file, do the following:

1. If you have not already done so, install WebLogic Server version 7.0 as described in the previous section, "Install the Partner Development Kit" on page 1-2.

   Note the location of the BEA home directory that the BEA installer uses. It contains a `license.bea` file, which will be updated in subsequent steps of this process. For more information about the BEA home directory and the `license.bea` file, refer to BEA Home Directory in the *Installing BEA WebLogic Server* guide.

2. Copy the `isv.jar` file from your email to the BEA home directory that the installer used.

3. Open a command shell and change directories to `BEA_HOME`, where `BEA_HOME` is the name of your BEA home directory.

4. Add `isv.jar` to the computer's `CLASSPATH` by entering one of the following commands:

   - `set CLASSPATH=.\isv.jar;%CLASSPATH%` (Windows systems)

- `export CLASSPATH=./isv.jar:$CLASSPATH` (UNIX systems)

5. Add the WebLogic Server JDK to the computer's `PATH` by entering one of the following commands:
   - `set PATH=.\jdk131_03\bin;%PATH%` (Windows systems)
   - `export PATH=./jdk131_03/bin:$PATH` (UNIX systems)

You are now ready to extract the ISV license data and link it to WebLogic Server files.

Step 2: Extracting the License Data and Linking WebLogic Server Files

To extract the ISV license data and link it to WebLogic Server files, enter one of the following commands from `BEA_HOME`:

- `java -Xmx128m -Dbea.home=`*`BEA_HOME`* `-Dbea.jar=`*`WL_HOME`*`\server\lib\weblogic.jar install` (Windows systems)
- `java -Xmx128m -Dbea.home=`*`BEA_HOME`* `-Dbea.jar=`*`WL_HOME`*`/server/lib/weblogic.jar install` (UNIX systems)

where *`BEA_HOME`* is an absolute pathname for your BEA home directory, and *`WL_HOME`* is an absolute pathname for the directory in which you installed WebLogic Server.

Caution: Do not interrupt this process once it has started.

The command generates a file named `BEA_HOME\license_isv.bea`, which contains the ISV license data. It also links files within the `WL_HOME` directory to the specific ISV license. Only the files in the `WL_HOME` directory that you specified will be able to use the ISV license data that you extracted to `license_isv.bea`.

Note: With some platforms and JDKs, you might encounter an "Out of Memory Error." To address this error, increase the value for the `-Xmx` argument (which sets the maximum heap size in megabytes) and run the command again. For example, `-Xmx150m` increases the default heap size to 150 megabytes.

To complete the process for installing an ISV license, you must update the Weblogic Server license with the data in `license_isv.bea`.

Step 3: Updating the WebLogic Server License

To update the `license.bea` file with the newly generated `license_isv.bea` file, enter one of the following commands from `BEA_HOME`:

- `UpdateLicense license_isv.bea` (Windows systems)
- `sh UpdateLicense.sh license_isv.bea` (UNIX systems)

The `UpdateLicense` command merges the `license_isv.bea` file with the `license.bea` file. After you run `UpdateLicense`, you do not need to keep the `license_isv.bea` file.

Next Steps: Configuring Your Application and WebLogic Server

After you install your ISV license, start the ISV-licensed WebLogic Server, deploy your application, and configure the server components. For more information, refer to the following topics (available from http://edocs.bea.com):

- Starting and Stopping WebLogic Servers in the *WebLogic Server Administration Guide*.
- The Administration Console Online Help
- Using WebLogic JMX Services, which provides detailed information and code samples for working with the WebLogic Server management system.
- The remaining sections of this document, which provide development tips that are specific to ISVs.

Distribute Files

When you are ready to distribute WebLogic Server with your application, you must make sure that your installer includes the BEA license file (`BEA_HOME\license.bea`) and the `WL_HOME\lib\weblogic.jar` file that you specified in "Step 3: Updating the WebLogic Server License" on page 1-5.

If you do not install both of the files that you specified, your embedded WebLogic Server will not start.

You can use this same `license.bea-weblogic.jar` pair for all of your licensed installations.For information on using the WebLogic Server silent install process, see Installing WebLogic Server Using Silent Installation.

Upgrading an Embedded WebLogic Server

BEA does not support upgrades to the `license.bea-weblogic.jar` pair. For example, if you installed your application with a bundled release of WebLogic Server 6.1, you must do the following to upgrade your bundled WebLogic Server to release 7.0:

1. Contact BEA to receive a new `isv.jar` file.
2. Install WebLogic Server 7.0.
3. Install the new ISV license as described in "Install the ISV License" on page 1-3.
4. Update your installer to include the new `license.bea` and WebLogic Server files.

CHAPTER

2 Customizing WebLogic Server Configuration Files

WebLogic Server stores configuration information, such as security credentials and the list of deployable resources and applications, in a set of configuration files.

The following sections highlight typical modifications that partners and ISVs make to the WebLogic Server configuration files that they distribute with their applications:

- Customizing the config.xml File
- Example Configuration
- Customizing Files for Compatibility Security

Customizing the config.xml File

The `config.xml` file defines the majority of configuration settings for all WebLogic Servers in a management domain. For example, `config.xml` controls all details of a given domain, including the name, number and configuration of servers and cluster; the list of deployable resources and applications; and the mapping of deployable resources and applications to servers and clusters.

Usually, we recommend that you use such WebLogic Server tools as the Administration Console and `weblogic.Admin` utility to modify the `config.xml` file. Partners, however, may need to edit this file directly in order to customize an installation.

The following sections highlight elements of the `config.xml` file that partners might modify for their installations:

- Pre-Configuring Application Resources
- Deployment of Application Components

If you are unfamiliar with the role of the `config.xml` file or management domains, refer to the following topics:

- Overview of WebLogic Server Management in the *WebLogic Server Administration Guide*
- Understanding Cluster Configuration and Application Deployment in the *Using WebLogic Server Clusters* guide.

If you are unfamiliar with editing `config.xml` directly, see the BEA WebLogic Server Configuration Reference, which provides conventions for editing `config.xml` and a description of the file's Document Type Definition (DTD).

Pre-Configuring Application Resources

Partner applications typically rely on several WebLogic Server resources, each of which is defined in the config.xml file:

Resources	config.xml Elements	Notes
Domain	Domain	To act as a cohesive unit, all WebLogic Servers that host a component of your application must reside within a single WebLogic Server administrative domain.

Resources	config.xml Elements	Notes
Server Names and Connection Information	Server	Partner applications can be configured to access one or more WebLogic Server names, IP addresses, and/or port numbers, or, if necessary for your application, you can hard-code a WebLogic Server domain to use specific server names and connection ports. IP Addresses can be configured dynamically by the application installer and embedded into a `config.xml` before installing the configuration.
JDBC Datasources	JDBCConnectionPool JDBCDataSource JDBCMultiPool JDBCTxDataSource	Partner applications that install WebLogic Server also frequently install an RDBMS or other datastore for maintaining the application data. If your product installer installs a datastore along with the application, you may want to pre-configure the installed WebLogic Server to set up a default datasource and connection pool for the datastore.

Deployment of Application Components

Partner applications can also be installed by adding the necessary elements to `config.xml`. Installing an application into a pre-configured WebLogic Server, however, requires coordination between the `config.xml` settings and the installed location of application component files (.war, .jar, .html and so forth).

The table below provides a summary of elements used to pre-deploy application components within WebLogic Server. See "Example Configuration" on page 2-4 for an example of how these elements correspond to the installed location of actual application component files.

Components	config.xml Elements	Notes
Startup Classes	StartupClass	WebLogic Server startup classes can be used to initialize resources required by other components of the partner application.
Webserver	WebServer	Web applications typically require standard web resources, such as static .html content, in addition to business logic. Use the `config.xml` file to configure the default location of these static files for the application.

Components	config.xml Elements	Notes
Web Applications	Application	EAR and WAR files can be stored anywhere in your application directory or the WebLogic Server directory. Reference the final installed location from within config.xml to deploy the application on startup.

Example Configuration

The WebLogic Server Pet Store is based on the Sun Microsystems Java Pet Store 1.3 demo. It includes four enterprise applications and one Web application that demonstrate various aspects of the J2EE platform.

WebLogic Server Pet Store provides a simple example of how to pre-configure an installation to support enterprise applications. For information about starting WebLogic Server Pet Store, refer to Samples and Examples on the WebLogic Server documentation Web site.

This section highlights key aspects of the `config.xml` file that configures WebLogic Server Pet Store.

Domain Configuration

The parent element in the `config.xml` file, `<Domain>`, provides the configuration for the `petstore` domain. All of the application's servers, resources, and components are defined within this element:

```
<Domain
  Name="petstore"
  >
```

Basic Server Setup

The WebLogic Server Pet Store uses a single server named `petstoreServer`. The connection properties for this server are preconfigured in the `Server` element attributes as follows:

```
<Server
    JavaCompiler="C:\bea\jdk131_02/bin/javac"
    ListenPort="7001"
    Name="petstoreServer"
    ServerVersion="7.0.0.0"
    IIOPEnabled="false"
  >
```

The remainder of the server setup configures the default Web Server and SSL configuration for the server.

JDBC Requirements

To demonstrate the use of multiple JDBC connection pools, WebLogic Server Pet Store defines and three JDBC connection pools. It also defines four JDBC datasources, one for each application within the Pet Store. All of the JDBC pools and datasources are deployed (targeted) on the default Pet Store server, `petstoreServer`.

```
<JDBCConnectionPool
    CapacityIncrement="1"
    DriverName="com.pointbase.jdbc.jdbcUniversalDriver"
    InitialCapacity="1"
    MaxCapacity="10"
    Name="petstorePool"
    Password="petstore"
    Properties="user=petstore"
    RefreshMinutes="0"
    ShrinkPeriodMinutes="15"
    ShrinkingEnabled="true"
    Targets="petstoreServer"
    TestConnectionsOnRelease="false"
```

```
    TestConnectionsOnReserve="false"
    URL="jdbc:pointbase:server://localhost/demo"
/>

<JDBCConnectionPool
    CapacityIncrement="1"
    DriverName="com.pointbase.jdbc.jdbcUniversalDriver"
    InitialCapacity="1"
    MaxCapacity="10"
    Name="petstoreopcPool"
    Password="petstoreopc"
    Properties="user=petstoreopc"
    RefreshMinutes="0"
    ShrinkPeriodMinutes="15"
    ShrinkingEnabled="true"
    Targets="petstoreServer"
    TestConnectionsOnRelease="false"
    TestConnectionsOnReserve="false"
    URL="jdbc:pointbase:server://localhost/demo"
/>

<JDBCConnectionPool
    CapacityIncrement="1"
    DriverName="com.pointbase.jdbc.jdbcUniversalDriver"
    InitialCapacity="1"
    MaxCapacity="10"
    Name="petstoresupplierPool"
    Password="petstoresupplier"
    Properties="user=petstoresupplier"
    RefreshMinutes="0"
    ShrinkPeriodMinutes="15"
    ShrinkingEnabled="true"
    Targets="petstoreServer"
    TestConnectionsOnRelease="false"
    TestConnectionsOnReserve="false"
    URL="jdbc:pointbase:server://localhost/demo"
/>


<JDBCTxDataSource
    EnableTwoPhaseCommit="true"
    JNDIName="datasource-petstorePool"
    Name="PetstoreDataSource"
    PoolName="petstorePool"
    Targets="petstoreServer"
/>
```

```
<JDBCTxDataSource
    EnableTwoPhaseCommit="true"
    JNDIName="datasource-petstoreopcPool"
    Name="PetstoreOPCDataSource"
    PoolName="petstoreopcPool"
    Targets="petstoreServer"
/>

<JDBCTxDataSource
    EnableTwoPhaseCommit="true"
    JNDIName="datasource-petstoresupplierPool"
    Name="PetstoreSupplierDataSource"
    PoolName="petstoresupplierPool"
    Targets="petstoreServer"
/>

<JDBCTxDataSource
    EnableTwoPhaseCommit="true"
    JNDIName="jdbc/CatalogDataSource"
    Name="CatalogDataSource"
    PoolName="petstorePool"
    Targets="petstoreServer"
/>
```

Application Components

The Pet Store application installs five application component files: `tour.war`, `petstore.ear`, `opc.ear`, `supplier.ear` and `petstoreadmin.ear`. These components are each installed within the WebLogic Server subdirectory and deployed to `petstoreServer`. For example, on Windows, `tour.war` is deployed using the following element:

```
<Application
   Deployed="true"
   Name="tour"
   Path="C:/bea/weblogic810/samples/server/stage/petstore">

  <WebAppComponent
   Name="tour"
   Targets="petstoreServer"
   URI="tour.war"
/>

</Application>
```

Note that the `c:\bea\wlserver810` portion of the application component path is determined during the WebLogic Server installation, while the remaining portion of the path is hard-coded. Your application installer can use a similar technique to install application components in a subdirectory unrelated to WebLogic Server, if necessary.

Startup Classes

On Windows systems, the Pet Store application uses a startup class to initiate the system web browser after the `petstoreServer` is booted. The definition for this startup class is mostly hard-coded in the installed `config.xml` file; only the port number is set dynamically by the WebLogic Server installation program:

```
<StartupClass </Application>
   Arguments="port=7001"
   ClassName="com.bea.estore.startup.StartBrowser"
   FailureIsFatal="false"
   Name="StartBrowser"
   Targets="petstoreServer"
   Notes="On Windows, this class automatically starts a browser after the server
has finished booting."
   />
```

Customizing Files for Compatibility Security

Compatibility security refers to the capability of running security configurations from WebLogic Server 6.x in WebLogic Server 8.1. If you run WebLogic Server with Compatibility security, your distribution must include the following:

- A `fileRealm.properties` file, which defines the ACLs, groups, and security principles for the default WebLogic Server security realm
- The following minimal set of elements in `config.xml`:

```
<Domain Name="mydomain">
<Security Name="mydomain" Realm="mysecurity"/>
<Realm Name="mysecurity" FileRealm="myrealm"/>
<FileRealm Name="myrealm"/>
<Server ListenPort="7001" Name="myserver">
</Server>
```

```
</Domain>
```

If your application requires integration with a third-party security realm (for example, single sign-on using the Windows NT security realm), you must also configure a caching realm.

For more information on WebLogic Server security, refer to the following topics:

- Using Compatibility Security in the *Managing WebLogic Security* guide.
- The Security page on the WebLogic Server documentation Web site.
- The BEA WebLogic Server Configuration Reference, which provides conventions for editing `config.xml` and a description of the file's DTD

CHAPTER

3 Using JDBC Profiling MBeans

The WebLogic Server management system uses Java Management Extensions (JMX) and Managed Beans (MBeans) to configure servers. The Using WebLogic JMX Services guide provides detailed information and code samples for working with WebLogic Server MBeans.

BEA provides several JDBC MBeans that you can use to store and analyze metrics for SQL statements, prepared statements, and JDBC connection leaks. The following sections describe how to enable and use JDBC profiling. For additional information, refer to the Javadoc for the following WebLogic Server MBeans and related classes:

- JDBCConnectionPoolMBean
- JDBCConnectionPoolRuntimeMBean
- JDBCStatementProfile
- JDBCConnectionLeakProfile

Enabling JDBC Profiling

Before you can analyze SQL statements or connection leak profiles, you must enable profiling for the connection pool you want to observe. When profiling is enabled, the connection pool stores metrics in an external repository for later analysis.

Applications enable and disable JDBC profiling options using the `JDBCConnectionPoolMBean`. In addition to providing get/set methods for standard connection pool properties, `JDBCConnectionPoolMBean` provides the following methods for enabling and disabling profiling:

- `setConnLeakProfilingEnabled()` enables or disables profiling for JDBC connection leaks. Connection leaks represent connections that were checked out of the connection pool but never returned with a `close()` method. It is important to analyze the connection leak profiles, as leaked connections cannot be used to fulfill later connection requests.
- `setSqlStmtProfilingEnabled()` enables or disables profiling for SQL statements. When this type of profiling is enabled, the connection pool stores both SQL statement text as well as the statement execution time and other metrics. You can analyze the SQL statement profile to determine which queries consume the most time in your applications.
- `setSqlStmtParamLoggingEnabled()` enables or disables profiling for the bind parameters of prepared and callable statements. Because statement parameters can be very large, you can optionally use `setSqlStmtMaxParamLength()` to limit the size of parameters that are stored in the profile.

For information on obtaining MBeans in WebLogic Server, see Accessing WebLogic Server MBeans. The following excerpt shows an application that obtains the `JDBCConnectionPoolMBean` and activates all profiling options. This example stores a maximum of 20 characters for each statement parameter:

```
// Obtain MBeanHome for the administration server.

...

JDBCConnectionPoolMBean mbean =
     (JDBCConnectionPoolMBean)home.getConfigurationMBean(poolName,
      "JDBCConnectionPoolConfig");

mbean.setConnLeakProfilingEnabled(true);

mbean.setSqlStmtParamLoggingEnabled(true);

mbean.setSqlStmtMaxParamLength(maxLen);

...
```

Accessing JDBC Profiles

Once you have enabled the desired profiling option(s), you can analyze the stored metrics using the `JDBCStatementProfile` and `JDBCConnectionLeakProfile` classes. Both of these profile classes can be easily obtained using the `JDBCConnectionPoolRuntimeMBean`.

`JDBCStatementProfile` stores the SQL statements and associated metrics (and optionally, bind parameters) for the connection pool. `JDBCConnectionLeakProfile` stores stack traces for leaked connections.

Obtaining all profiles at once may consume considerable resources. For this reason, applications should generally retrieve only a subset of profiles at a given time. You can accomplish this by first determining the total number of profiles in storage, then retrieving profiles in smaller subsets.

The following example shows a simple way to divide the number of profiles into smaller fractions.

```
// Obtain MBeanHome for the server that hosts the connection pool.

. . .

// Get the JDBCRuntimeMbean for the "testPool" connection pool.

String poolName = "testPool";

JDBCConnectionPoolRuntimeMBean mbean =
     (JDBCConnectionPoolRuntimeMBean)home.getRuntimeMBean
          (poolName,"JDBCConnectionPoolRuntime");

JDBCConnectionLeakProfile[] profiles = null;

// Get the total number of available prepared statement cache profiles
int profileCount = mbean.getConnectionLeakProfileCount();


// Request profilesPerStep number of profiles
int profilesPerStep = 10;

// Begin with profile number profileIndex
int profileIndex = 0;
boolean done = (profileCount > 0);
while (!done) {
```

```
     // Get profiles
     profiles = mbean.getConnectionLeakProfiles(profileIndex,
          profilesPerStep);

     // Go through retrieved profiles
          for (int index = 0; index < profiles.length; index++) {

          // Get pool name
          String poolName = profiles[index].getPoolName();

          // Get stack trace
          String stackTrace = profiles[index].getStackTrace();
          }

     profileIndex = profileIndex + profilesPerStep - 1;

     // Finish if number of retrieved profiles is
     // less then requested
     done = (profiles.lengh < profilesPerStep);
}
```