BEA WebLogic Enterprise

Guide to the EJB Sample Applications

WebLogic Enterprise 5.1
Document Edition 5.1
May 2000

Guide to the EJB Sample Applications

Document Edition	Date	Software Version
5.1	May 2000	BEA WebLogic Enterprise 5.1

Contents

About This Document

This document describes how programmers can implement key features in the BEA WebLogic Enterprise™ (WLE) product to design and implement scalable, high-performance, C++ server applications that run in a WLE domain.

This document covers the following topics:

- Chapter 1, “EJB Samples Overview,” introduces the WLE EJB applications, gives links to the EJB sample application Javadoc, and gives general information about building and running the sample applications.
- Chapter 2, “Stateless Session Bean Sample Application,” shows a stateless session bean in which the client application must maintain any state across invocations to that bean.
- Chapter 3, “Stateful Session Bean Sample Application,”shows a session bean that uses stateful persistence.
- Chapter 4, “JDBC Sequence Bean Sample Application,”shows an entity bean that automatically generates its primary key by calling directly to a database using a connection pool and JDBC.
- Chapter 5, “Oracle Sequence Bean Sample Application,”shows an entity bean that automatically generates its primary key by calling directly to a database using a connection pool and an Oracle database.
- Chapter 6, “Parent Bean Sample Application,”shows a stateless session bean called `ParentBean` that is the parent class for another bean, `ChildBean`.
- Chapter 7, “Child Bean Sample Application,”shows a stateless session bean called `ChildBean` that inherits methods from a `ParentBean`, and also shows one bean calling another bean.

- Appendix A, “JNDI Utility Application,” describes a utility application that you can use with your running EJB applications to search the JNDI tree and display information about each EJB.

What You Need to Know

This document is intended for programmers who are interested in creating secure, scalable, transaction-based server applications. It assumes you are knowledgeable with the BEA Tuxedo® system, Enterprise JavaBeans, and Java programming.

e-docs Web Site

The BEA WebLogic Enterprise product documentation is available on the BEA Systems, Inc. corporate Web site. From the BEA Home page, click the Product Documentation button or go directly to the “e-docs” 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 file at a time, by using the File—>Print option on your Web browser.

A PDF version of this document is available on the WebLogic Enterprise 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 Enterprise documentation Home page, click the PDF Files button, and select the document you want to print.

If you do not have Adobe Acrobat Reader installed, you can download it for free from the Adobe Web site at http://www.adobe.com/.

Related Information

For more information about CORBA, Java 2 Enterprise Edition (J2EE), BEA Tuxedo, distributed object computing, transaction processing, C++ programming, and Java programming, see the *Bibliography* in the WebLogic Enterprise online documentation.

Contact Us!

Your feedback on the BEA WebLogic Enterprise 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 WebLogic Enterprise documentation.

In your e-mail message, please indicate that you are using the documentation for the BEA WebLogic Enterprise 5.1 release.

If you have any questions about this version of BEA WebLogic Enterprise, or if you have problems installing and running BEA WebLogic Enterprise, contact BEA Customer Support through BEA WebSUPPORT at 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	Item
boldface text	Indicates terms defined in the glossary.
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
italics	Indicates emphasis or book titles.

`monospace text`	Indicates code samples, commands and their options, data structures and their members, data types, directories, and filenames and their extensions. Monospace text also indicates text that you must enter from the keyboard.
	Examples:
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

`monospace boldface text`	Identifies significant words in code. *Example*: `void` **`commit`** `( )`
`monospace italic text`	Identifies variables in code. *Example*: `String` *`expr`*

Convention	**Item**
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Example*s: LPT1 SIGNON OR
`{ }`	Indicates a set of choices in a syntax line. The braces themselves should never be typed.

`[ ]`	Indicates optional items in a syntax line. The brackets themselves should never be typed.
	Example:
	`buildobjclient [-v] [-o name ] [-f` *`file-list`*`]...`
	`[-l` *`file-list`*`]...`

`|`	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.

`...`	Indicates one of the following in a command line:
	■ That an argument can be repeated several times in a command line
	■ That the statement omits additional optional arguments
	■ That you can enter additional parameters, values, or other information
	The ellipsis itself should never be typed.
	Example:
	`buildobjclient [-v] [-o name ] [-f` *`file-list`*`]...`
	`[-l` *`file-list`*`]...`

. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

CHAPTER

1 EJB Samples Overview

This topic includes the following sections:

- EJB Sample Applications Provided with WebLogic Enterprise
- EJB Samples Javadoc
- Before you Build and Run the EJB Sample Applications

EJB Sample Applications Provided with WebLogic Enterprise

WebLogic Enterprise provides the following sample applications:

- `samples.j2ee.ejb.basic.statelessSession`

  Shows a stateless session bean in which the client application must maintain any state across invocations to that bean.
- `samples.j2ee.ejb.basic.statefulSession`

  Shows a session bean that uses stateful persistence.
- `samples.j2ee.ejb.sequence.jdbc`

  Shows an entity bean that automatically generates its primary key by calling directly to a database using a connection pool and JDBC.
- `samples.j2ee.ejb.sequence.oracle`

  Shows an entity bean that automatically generates its primary key by calling directly to a database using a connection pool and an Oracle database.

- `samples.j2ee.ejb.subclass.parent`

  Shows a stateless session bean called `ParentBean` that is the parent class for another bean, `ChildBean`.

- `samples.j2ee.ejb.subclass.child`

  Shows a stateless session bean called `ChildBean` that inherits methods from a `ParentBean`, and also shows one bean calling another bean.

- `samples.j2ee.ejb.utils`

  Contains a Java application that creates a list of all the EJBs deployed in a WebLogic Enterprise server process.

This document describes each of these sample applications, and also explains how to build and run them.

EJB Samples Javadoc

Javadoc for the EJB sample applications is installed in the locations shown in Table 1-1. If you are viewing this document in a browser, you can click the name in the left column to display the related Javadoc.

Table 1-1 EJB Sample Application Javadoc

Sample Application	Location

All packages	**Windows NT**
	`%TUXDIR%\samples\j2ee\ejb\docs\index.html`
	UNIX
	`$TUXDIR/samples/j2ee/ejb/docs/index.html`

Stateless session bean	**Windows NT**
	`%TUXDIR%\samples\j2ee\ejb\docs\samples\j2ee\ejb\basic\statelessSession\package-summary.html`
	UNIX
	`$TUXDIR/samples/j2ee/ejb/docs/samples/j2ee/ejb/basic\statelessSession/package-summary.html`

Sample Application	Location

Stateful session bean	**Windows NT**
	`%TUXDIR%\samples\j2ee\ejb\docs\samples\j2ee\ejb\basic\statefulSession\package-summary.html`
	UNIX
	`$TUXDIR/samples/j2ee/ejb/docs/samples/j2ee/ejb/basic\statefulSession/package-summary.html`

JDBC sequence bean	**Windows NT**
	`%TUXDIR%\samples\j2ee\ejb\docs\samples\j2ee\ejb\sequence\jdbc\package-summary.html`
	UNIX
	`$TUXDIR/samples/j2ee/ejb/docs/samples/j2ee/ejb/sequence\jdbc/package-summary.html`

Oracle sequence bean	**Windows NT**
	`%TUXDIR%\samples\j2ee\ejb\docs\samples\j2ee\ejb\sequence\oracle\package-summary.html`
	UNIX
	`$TUXDIR/samples/j2ee/ejb/docs/samples/j2ee/ejbs/equence\oracle/package-summary.html`

Parent bean	**Windows NT**
	`%TUXDIR%\samples\j2ee\ejb\docs\samples\j2ee\ejb\subclass\parent\package-summary.html`
	UNIX
	`$TUXDIR/samples/j2ee/ejb/docs/samples/j2ee/ejb/subclass\parent/package-summary.html`

Child bean	**Windows NT**
	`%TUXDIR%\samples\j2ee\ejb\docs\samples\j2ee\ejb\subclass\child\package-summary.html`
	UNIX
	`$TUXDIR/samples/j2ee/ejb/docs/samples/j2ee/ejb/subclass\child/package-summary.html`

Before you Build and Run the EJB Sample Applications

Each of the subsequent chapters in this guide explain all the steps for building and running each sample. In general, though, note the following about these sample applications:

1. The instructions described here are available in the WebLogic online documentation and also in the following location on the software CD:

   Windows NT

   `%TUXDIR%\samples\j2ee\ejb`

   UNIX

   `$TUXDIR/samples/j2ee/ejb`

2. Create a work directory on your machine, and copy the contents of the following directory into the work directory:

   Windows NT

   `%TUXDIR%\samples\j2ee\ejb`

   UNIX

   `$TUXDIR/samples/j2ee/ejb`

   You need to build each sample one at a time. All the samples use the following common build and execute a script, which are located in the preceding directory, for each platform:

   Windows NT

   Execute the batch file `runme.cmd`.

   UNIX

   Execute shell script file `runme.ksh`.

3. Before running the build script/batch file for each sample application, make sure that the following system environment variables are set:

Variable	Description

`TUXDIR`	The directory path where you installed the WebLogic Enterprise software. For example:
	Windows NT
	`TUXDIR=c:\wledir`
	UNIX
	`TUXDIR=/usr/local/wledir`

`JAVA_HOME`	The directory path where you installed the JDK software. For example:
	Windows NT
	`JAVA_HOME =c:\JDK1.2`
	UNIX
	`JAVA_HOME =/usr/local/JDK1.2`

`ORACLE_HOME`	The directory path where you installed the Oracle software. For example: `ORACLE_HOME=/usr/local/oracle` You need to set this environment variable on UNIX operating systems only.

4. You may optionally set the system environment variables listed in Table 1-2 prior to running the build command to change their default value. See the *Administration Guide* for more information about selecting appropriate values for these environment variables.

Table 1-2 System Environment Variables

Variable	Description
HOST	The host name portion of the TCP/IP network address used by the ISL process to accept connections from Java clients. The default value is the name of the local machine.
PORT	The TCP port number at which the ISL process listens for incoming requests; it must be a number between 0 and 65535. The default value is 2468.

Variable	Description
IPCKEY	The address of shared memory; it must be a number greater than 32769 unique to this application on this system. The default value is 55432.
DB_INSTANCE	Name of the database instance or server. The default value for Oracle is `Beq-Local`. This is needed only for the samples that use a database.
DB_USER	Name of the database user. The default is `scott`. This is needed only for the samples that use a database.
DB_PASSWORD	Password for the database user. The default is `tiger`. This is needed only for the samples that use a database.
DB_DRIVER	The Java class name of the database driver. The default is the Oracle 8i driver, `weblogic.jdbc20.oci815.Driver`.This is needed only for the samples that use a database.
DB_URL	Database connection URL. The default for Oracle is `jdbc:weblogic:oracle:Beq-Local`. If the database instance is not named, set `DB_INSTANCE` to `null`. This is needed only for the samples that use a database.

When you enter the `tmboot` command to start one of the EJB sample applications, the following server processes are started:

- `TMSYSEVT`

  The BEA Tuxedo system Event Broker.

- `TMFFNAME`

  The following `TMFFNAME` processes are started:

  The `TMFFNAME` server process with the `-N` option and the `-M` option is the `MASTER NameManager` service. The `-N` option starts the `NameManager` service; the `-M` option starts this name manager as a `MASTER`. This service maintains a mapping of application-supplied names to object references.

  The `TMFFNAME` server process with the `-N` option only is a `SLAVE NameManager` service.

The `TMFFNAME` server with the `-F` option contains the FactoryFinder object.

- `JavaServer`

  The JavaServer takes one or more EJB JAR files that were created for the application.

- `ISL`

  The IIOP Listener/Handler.

Restoring the Sample Applications Directory to Its Original State

You can restore the sample application directory to its original state by completing the following steps:

1. Set the directory to the directory containing the sample application.
2. Enter the following command:

   Windows NT

   ```
   prompt>%TUXDIR%\samples\j2ee\ejb\clean.cmd
   ```

   UNIX

   ```
   prompt>./$TUXDIR/samples/j2ee/ejb/clean.ksh
   ```

CHAPTER

2 Stateless Session Bean Sample Application

This topic includes the following sections:

- How the Stateless Session Bean Sample Works
- Building and Running the Stateless Session Bean Sample
- Stopping the Sample Application
- Stateless Session Bean Javadoc

How the Stateless Session Bean Sample Works

This sample demonstrates the usage of stateless session EJBs using a simple stock trader application. This sample demonstrates how the client must maintain any persistent state -- such as the change in the cash account -- across repeated calls to the session EJB.

The EJB in this sample provides basic trading methods such as buying and selling stocks. Since there are no persistent stores involved in this sample, all the stock data are set in the deployment descriptor of the EJB as environment properties. The container supplies the data to the EJB through JNDI lookup operation.

The stateless session bean sample application implements the following classes:

Class	Description
`Client`	This bean: ■ Creates a trader and performs repeated buying and selling of shares. ■ Shows persistence of state between invocations to the TraderBean class. ■ Maintains state between invocations.
`TraderBean`	This bean does not manage any persistence of state between invocations on it.
`TradeResult`	This bean contains the results of a buy/sell transaction.

Building and Running the Stateless Session Bean Sample

To build and run the stateless session bean sample application, complete the following steps:

1. Verify the environment variables.
2. Create a work directory on your machine under the directory described in step 2 in the section “Before you Build and Run the EJB Sample Applications” on page 1-4.
3. Change the protection attribute on the files for the stateless session bean sample application.
4. Execute the `runme` command.

The following sections describe these steps, and also explain the following:

- How to run the stateless session bean sample application
- Processes and files created by the stateless session bean sample application

Verifying the Settings of the Environment Variables

Before building and running the sample application, you need to ensure that certain environment variables are set on your system. In most cases, these environment variables are set as part of the installation procedure. However, you need to check the environment variables to ensure they reflect correct information.

The following table lists the environment variables required to run the stateless session bean sample application.

Environment Variable	Description

`TUXDIR`	The directory path where you installed the WebLogic Enterprise software. For example:
	Windows NT
	`TUXDIR=c:\wledir`
	UNIX
	`TUXDIR=/usr/local/wledir`

`JAVA_HOME`	The directory path where you installed the JDK software. For example:
	Windows NT
	`JAVA_HOME =c:\JDK1.2`
	UNIX
	`JAVA_HOME =/usr/local/JDK1.2`

You may optionally set the following system environment variables to change their default value prior to running the stateless session bean sample application `runme` command. See the *Administration Guide* for more information about selecting appropriate values for these environment variables.

The following table lists the optional environment variables required to run the stateless session bean sample application.

Environment Variable	Description
`HOST`	The host name portion of the TCP/IP network address used by the ISL process to accept connections from Java clients. The default value is the name of the local machine.
`PORT`	The TCP port number at which the ISL process listens for incoming requests; it must be a number between 0 and 65535. The default value is 2468.
`IPCKEY`	The address of shared memory; it must be a number greater than 32769 unique to this application on this system. The default value is 55432.

Verifying the Environment Variables

To verify that the information for the environment variables defined during installation is correct, complete the following steps:

Windows NT

1. From the Start menu, select Settings.
2. From the Settings menu, select the Control Panel.

   The Control Panel appears.
3. Click the System icon.

   The System Properties window appears.
4. Click the Environment tab.

   The Environment page appears.
5. Check the settings for `TUXDIR` and `JAVA_HOME`.

UNIX

1. Enter the `ksh` command to use the Korn shell.
2. Enter the `printenv` command to display the values of `TUXDIR` and `JAVA_HOME`, as in the following example:

```
ksh prompt>printenv TUXDIR
ksh prompt>printenv JAVA_HOME
```

Changing the Environment Variables

To change the environment variable settings, complete the following steps:

Windows NT

1. From the Start menu, select Settings.
2. From the Settings menu, select the Control Panel.

   The Control Panel appears.
3. Click the System icon.

   The System Properties window appears.
4. Click the Environment tab.

   The Environment page appears.
5. On the Environment page in the System Properties window, click the environment variable you want to change or enter the name of the environment variable in the Variable field.
6. Enter the correct information for the environment variable in the Value field.
7. Click OK to save the changes.

UNIX

1. Enter the `ksh` command to use the Korn shell.
2. Enter the `export` command to set the correct values for the `TUXDIR` and `JAVA_HOME` environment variables, as in the following example:

```
ksh prompt>export TUXDIR=directorypath
ksh prompt>export JAVA_HOME=directorypath
```

Copying the Files for the Stateless Session Bean Sample Application into a Work Directory

You need to copy the files for the stateless session bean sample application into a work directory on your local machine. The following steps describe how to copy all the example files into a work directory.

1. If you are using a UNIX system, enter the `ksh` command to use the Korn shell:

   ```
   prompt>ksh
   ksh prompt>
   ```

2. Create a work directory on your machine under the directory described in step 2 in the section “Before you Build and Run the EJB Sample Applications” on page 1-4.

3. Copy the contents of the following directory into the work directory:

   Windows NT

   ```
   %TUXDIR%\samples\j2ee\ejb\basic\statelessSession
   ```

   UNIX

   ```
   $TUXDIR/samples/j2ee/ejb/basic/statelessSession
   ```

The files copied into the work directory are in two categories:

- Files specific to the stateless session bean example
- Utility files used for building and running the example application

Sample Application Files

Table 2-1 lists and describes all the files for this sample application.

Table 2-1 Sample Application Files

File	Description
`ejb-jar.xml`	The XML deployment descriptor file used to help add the bean to the EJB container.

File	Description
`weblogic-ejb-extensions.xml`	A file containing the WebLogic Enterprise extensions to the deployment descriptor DTD.
`Client.Java`	The Java source code for the client.
`TraderBean.java`	The Java source code for the stateless session bean. This class contains the business logic method implementations and methods required by the Sun Microsystems, Inc. EJB 1.1 specification.
`Trader.java`	The Java source code for the Remote interface of the `TraderBean` class.
`TraderHome.java`	The Java source code for the Home interface of the `TraderBean` class.
`TradeResult.java`	Application-specific utility class used to carry a trade execution result between the EJB and the client.
`ProcessingErrorException.java`	Application-specific exception thrown by the `TraderBean` class for business methods.
`index.html`	File containing these instructions.

Utility Files

Table 2-2 lists and describes the utility files for this sample application. These files are generated based on the WebLogic Enterprise installation environment. The following files are generated in the same directory as where the source files are found.

Table 2-2 Utility Files

File	Description
`runme.cmd`	The Windows NT batch file that contains commands to set the environment, boot the server, and execute the client for this sample.

File	Description
`runme.ksh`	The UNIX Korn shell script that contains commands to boot the server and execute the client for this sample.
`run_client.cmd`	The batch file to run the client on Windows NT systems.
`run_client.ksh`	The script file to run the client on UNIX systems.
`setenv.cmd`	The batch file to set the necessary environment variables on Windows NT systems.
`setenv.ksh`	The script file to set the necessary environment variables on UNIX systems.
`ubbconfig`	The WebLogic Enterprise server configuration file to be used on UNIX systems.
`ubbconfig.nt`	The WebLogic Enterprise server configuration file to be used on Windows NT systems.
`ejb_basic_statelessSession.jar`	The EJB JAR file that contains the source file classes, the container-specific class files generated by the `ejbc` command, and the deployment descriptor files. This is the EJB JAR file that is deployed on the WebLogic Enterprise server.

Changing the Protection Attribute on the Files for the Stateless Session Bean Sample Application

During the installation of the WebLogic Enterprise software, the sample application files are marked read-only. Before you can edit or build the files in the stateless session bean sample application, you need to change the protection attribute of the files you copied into your work directory, as follows:

Windows NT

```
prompt>attrib /S -r drive:\workdirectory\*.*
```

UNIX

```
prompt>ksh
```

```
ksh prompt>chmod +w /workdirectory/*.*
```

On the UNIX operating system platform, you also need to change the permission of `runme.ksh` and `clean.ksh` to give execute permission to those files, as follows:

```
ksh prompt>chmod +x *.ksh
```

Executing the runme Command

The `runme` command automates the following steps:

1. Loads the `UBBCONFIG` file.
2. Compiles the code for the stateless session bean sample application.
3. Starts the server application using the `tmboot` command.
4. Starts the client application.
5. Stops the server application using the `tmshutdown` command.

To build and run the stateless session bean sample application, make sure you have copied the scripts described in "Before you Build and Run the EJB Sample Applications" on page 1-4, and enter the `runme` command, as follows:

Windows NT

```
prompt>cd workdirectory
```

```
prompt>runme basic statelessSession
```

UNIX

```
ksh prompt>cd workdirectory
```

```
ksh prompt>./runme.ksh basic statelessSession
```

A number of messages are displayed, along with whether or not the build procedure was successful. Note that the sample is not only built, but also the servers are booted and the client is run once.

Running the Sample Manually

After you have executed the `runme` command, you can run the sample manually if you like. To run the samples manually, complete the following steps:

1. Change to the sample's work directory, if necessary.
2. Make sure that your environment is set correctly by entering the following command:

   Windows NT

   ```
   prompt>setenv
   ```

   UNIX

   ```
   prompt>. ./setenv.ksh
   ```

3. Boot the server and run the client by entering the following commands:

   Windows NT

   ```
   prompt>tmboot -y
   prompt>run_client.cmd
   ```

   UNIX

   ```
   prompt>tmboot -y
   prompt>./run_client.ksh
   ```

Processes and Files Generated by the Sample Application

When the you enter the `tmboot` command to start one of the EJB sample applications, the following server processes are started:

- `TMSYSEVT`

  The BEA Tuxedo system Event Broker.

- `TMFFNAME`

  The following `TMFFNAME` processes are started:

  - The `TMFFNAME` server process with the `-N` option and the `-M` option is the `MASTER NameManager` service. The `-N` option starts the `NameManager` service; the `-M` option starts this name manager as a `MASTER`. This service maintains a mapping of application-supplied names to object references.

  - The `TMFFNAME` server process with the `-N` option is a `SLAVE NameManager` service.
  - The `TMFFNAME` server with the `-F` option contains the FactoryFinder object.

- `JavaServer`

  The JavaServer takes one or more EJB JAR files that were created for the application.

- `ISL`

  The IIOP Listener/Handler.

Stopping the Sample Application

You can stop the stateless session bean sample application by entering the following command:

```
prompt>tmshutdown -y
```

Stateless Session Bean Javadoc

The Javadoc for the stateless session bean example is in the following location:

Windows NT

```
%TUXDIR%/samples/j2ee/ejb/basic/statelessSession/index.html
```

UNIX

```
$TUXDIR\samples\j2ee\ejb\basic\statelessSession\index.html
```

If you are viewing this document in a browser, you can click the following link to display this Javadoc:

`Package samples.j2ee.ejb.basic.statelessSession`

CHAPTER

3 Stateful Session Bean Sample Application

This topic includes the following sections:

- How the Stateful Session Bean Sample Works
- Building and Running the Stateful Session Bean Sample
- Stopping the Sample Application
- Stateful Session Bean Javadoc

How the Stateful Session Bean Sample Works

This sample application shows how repeated calls to the same session bean have a persistent state -- the change in the cash account -- that is maintained across all the calls. Notice that neither the client nor the EJB do anything to maintain that state: the container handles it transparently. All the logic for the cash account is encapsulated in the bean, unlike the stateless session sample where all persistence is provided by the client.

The EJB in this sample provides basic trading methods such as buying and selling stocks. Since there are no persistent stores involved in this sample, all the stock data are set in the deployment descriptor of the EJB as environment properties. The container supplies the data to the EJB through the JNDI lookup operation.

This sample provides two types of clients: one is a simple single threaded client, and the other is a multithreaded client. The stateful session bean sample application implements the following classes:

Class	Description
`Client`	This class: ■ Creates a trader and performs repeated buying and selling of shares. ■ Shows persistence of state between calls to the `TraderBean`; the client does not do anything to maintain state between calls. ■ Searches the JNDI tree for an appropriate container.
`MultiClient`	This class demonstrates calling a stateful session bean using multiple colocated clients: each thread is a trader, and performs repeated buying and selling of shares Persistence of state between calls to the `TraderBean` bean. Like the single-threaded `Client` bean, the `MultiClient` bean does not do anything to maintain state between calls.
`TraderBean`	This bean does not manage any persistence of state between invocations on it.
`TradeResult`	This bean contains the results of a buy/sell transaction.

Building and Running the Stateful Session Bean Sample

To build and run the stateful session bean sample application, complete the following steps:

1. Verify the environment variables.
2. Copy the files for the stateful session bean sample application into a work directory.

3. Change the protection attribute on the files for the stateful session bean sample application.
4. Execute the `runme` command.

The following sections describe these steps, and also explain the following:

- How to run the stateful session bean sample application
- Processes and files created by the stateful session bean sample application

Verifying the Settings of the Environment Variables

Before building and running the sample application, you need to ensure that certain environment variables are set on your system. In most cases, these environment variables are set as part of the installation procedure. However, you need to check the environment variables to ensure they reflect correct information.

Table 3-1 lists the environment variables required to run the stateful session bean sample application.

Table 3-1 Environment Variables

Environment Variable	Description

`TUXDIR`	The directory path where you installed the WebLogic Enterprise software. For example:
	Windows NT
	`TUXDIR=c:\wledir`
	UNIX
	`TUXDIR=/usr/local/wledir`

`JAVA_HOME`	The directory path where you installed the JDK software. For example:
	Windows NT
	`JAVA_HOME =c:\JDK1.2`
	UNIX
	`JAVA_HOME =/usr/local/JDK1.2`

You may optionally set the following system environment variables to change their default value prior to running the stateful session bean sample application `runme` command. See the *Administration Guide* for more information about selecting appropriate values for these environment variables.

Table 3-2 lists the optional environment variables required to run the stateful session bean sample application.

Table 3-2 Optional Environment Variables

Environment Variable	Description
`HOST`	The host name portion of the TCP/IP network address used by the ISL process to accept connections from Java clients. The default value is the name of the local machine.
`PORT`	The TCP port number at which the ISL process listens for incoming requests; it must be a number between 0 and 65535. The default value is 2468.
`IPCKEY`	The address of shared memory; it must be a number greater than 32769 unique to this application on this system. The default value is 55432.

Verifying the Environment Variables

To verify that the information for the environment variables defined during installation is correct, complete the following steps:

Windows NT

1. From the Start menu, select Settings.
2. From the Settings menu, select the Control Panel.

   The Control Panel appears.
3. Click the System icon.

   The System Properties window appears.
4. Click the Environment tab.

   The Environment page appears.

5. Check the settings for `TUXDIR` and `JAVA_HOME`.

UNIX

1. Enter the `ksh` command to use the Korn shell.
2. Enter the `printenv` command to display the values of `TUXDIR` and `JAVA_HOME`, as in the following example:

   ```
   ksh prompt>printenv TUXDIR
   ksh prompt>printenv JAVA_HOME
   ```

Changing the Environment Variables

To change the environment variable settings, complete the following steps:

Windows NT

1. From the Start menu, select Settings.
2. From the Settings menu, select the Control Panel.

   The Control Panel appears.
3. Click the System icon.

   The System Properties window appears.
4. Click the Environment tab.

   The Environment page appears.
5. On the Environment page in the System Properties window, click the environment variable you want to change or enter the name of the environment variable in the Variable field.
6. Enter the correct information for the environment variable in the Value field.
7. Click OK to save the changes.

UNIX

1. Enter the `ksh` command to use the Korn shell.
2. Enter the `export` command to set the correct values for the `TUXDIR` and `JAVA_HOME` environment variables, as in the following example:

```
ksh prompt>export TUXDIR=directorypath
ksh prompt>export JAVA_HOME=directorypath
```

Copying the Files for the Stateful Session Bean Sample Application into a Work Directory

You need to copy the files for the stateful session bean sample application into a work directory on your local machine. The following steps describe how to copy all the example files into a work directory.

1. If you are using a UNIX system, enter the `ksh` command to use the Korn shell:

   ```
   prompt>ksh
   ksh prompt>
   ```

2. Create a work directory on your machine under the directory described in step 2 in the section “Before you Build and Run the EJB Sample Applications” on page 1-4.

3. Copy the contents of the following directory into the work directory:

   Windows NT

   `%TUXDIR%\samples\j2ee\ejb\basic\statefulSession`

   UNIX

   `$TUXDIR/samples/j2ee/ejb/basic/statefulSession`

The files copied into the work directory are in two categories:

- Files specific to the stateful session bean example
- Utility files used for building and running the example application

Sample Application Files

Table 3-3 lists and describes all the files for this sample application:

Table 3-3 Sample Application Files

File	Description
`ejb-jar.xml`	The XML deployment descriptor file used to help add the bean to the EJB container.
`weblogic-ejb-extensions.xml`	A file containing the WebLogic Enterprise extensions to the deployment descriptor DTD.
`Client.Java`	The Java source code for the client application.
`TraderBean.java`	The Java source code for the stateful session bean. This class contains the business logic method implementations and methods required by the EJB specification 1.1.
`Trader.java`	The Java source code for the Remote interface of the `TraderBean` class.
`TraderHome.java`	The Java source code for the Home interface of the `TraderBean` class.
`TradeResult.java`	Application-specific utility class used to carry a trade execution result between the EJB and the client.
`ProcessingErrorException.java`	Application-specific utility class used to carry a trade execution result between the EJB and the client.

Utility Files

Table 3-4 lists and describes the utility files for this sample application. These files are generated based on the WebLogic Enterprise installation environment. The following files are generated in the same directory as where the source files are found.

Table 3-4 Utility Files

File	Description
`runme.cmd`	The Windows NT batch file that contains commands to set the environment, boot the server, and execute the client for this sample.
`runme.ksh`	The UNIX Korn shell script that contains commands to boot the server and execute the client for this sample.
`run_client.cmd`	The batch file to run the client application on Windows NT systems.
`run_client.ksh`	The script file to run the client on UNIX systems.
`run_mclient.ksh`	The script file to run the multithreaded client application on UNIX systems.
`run_mclient.cmd`	The batch file to run the multithreaded client application on Windows NT systems.
`setenv.cmd`	The batch file to set the necessary environment variables on Windows NT systems.
`setenv.ksh`	The script file to set the necessary environment variables on UNIX systems.
`ubbconfig`	The WebLogic Enterprise server configuration file to be used on UNIX systems.
`ubbconfig.nt`	The WebLogic Enterprise server configuration file to be used on Windows NT systems.
`index.html`	File containing these instructions.

Changing the Protection Attribute on the Files for the Stateful Session Bean Sample Application

During the installation of the WebLogic Enterprise software, the sample application files are marked read-only. Before you can edit or build the files in the stateful session bean sample application, you need to change the protection attribute of the files you copied into your work directory, as follows:

Windows NT

```
prompt>attrib /S -r drive:\workdirectory\*.*
```

UNIX

```
prompt>ksh
```

```
ksh prompt>chmod +w /workdirectory/*.*
```

On the UNIX operating system platform, you also need to change the permission of `runme.ksh` and `clean.ksh` to give execute permission to those files, as follows:

```
ksh prompt>chmod +x *.ksh
```

Executing the runme Command

The `runme` command automates the following steps:

1. Loads the `UBBCONFIG` file.
2. Compiles the code for the stateful session bean sample application.
3. Starts the server application using the `tmboot` command.
4. Starts the client application.
5. Stops the server application using the `tmshutdown` command.

To build and run the stateless session bean sample application, make sure you have copied the scripts described in "Before you Build and Run the EJB Sample Applications" on page 1-4, and enter the `runme` command, as follows:

Windows NT

```
prompt>cd workdirectory
```

```
prompt>runme basic statefulSession
```

UNIX

```
ksh prompt>cd workdirectory
```

```
ksh prompt>./runme.ksh basic statefulSession
```

A number of messages are displayed, along with whether or not the build procedure was successful. Note that the sample is not only built, but also the servers are booted and the client is run once.

Running the Sample Manually

After you have executed the `runme` command, you can run the sample manually if you like. To run the samples manually, complete the following steps:

1. Change to the sample's work directory, if necessary.
2. Make sure that your environment is set correctly by entering the following command:

   Windows NT

   ```
   prompt>setenv
   ```

   UNIX

   ```
   prompt>. ./setenv.ksh
   ```

3. Boot the server and run the client by entering the following commands:

   Windows NT

   ```
   prompt>tmboot -y
   prompt>run_client.cmd
   ```

   UNIX

   ```
   prompt>tmboot -y
   prompt>./run_client.ksh
   ```

Processes and Files Generated by the Sample Application

When the you enter the `tmboot` command to start one of the EJB sample applications, the following server processes are started:

- `TMSYSEVT`

  The BEA Tuxedo system Event Broker.

- `TMFFNAME`

  The following `TMFFNAME` processes are started:

  - The `TMFFNAME` server process with the `-N` option and the `-M` option is the `MASTER NameManager` service. The `-N` option starts the `NameManager` service; the `-M` option starts this name manager as a `MASTER`. This service maintains a mapping of application-supplied names to object references.
  - The `TMFFNAME` server process with the `-N` option is a `SLAVE NameManager` service.
  - The `TMFFNAME` server with the `-F` option contains the FactoryFinder object.

- `JavaServer`

  The JavaServer takes one or more EJB JAR files that were created for the application.

- `ISL`

  The IIOP Listener/Handler.

Stopping the Sample Application

You can stop the stateful session bean sample application by entering the following command:

```
prompt>tmshutdown -y
```

Stateful Session Bean Javadoc

The Javadoc for the stateleful session bean example is in the following location:

Windows NT

```
%TUXDIR%/samples/j2ee/ejb/basic/statefulSession/index.html
```

UNIX

```
$TUXDIR\samples\j2ee\ejb\basic\statefulSession\index.html
```

If you are viewing this document in a browser, you can click the following link to display this Javadoc:

`Package samples.j2ee.ejb.basic.statefulSession`

CHAPTER

4 JDBC Sequence Bean Sample Application

This topic includes the following sections:

- How the JDBC Sequence Bean Sample Works
- Building and Running the JDBC Sequence Bean Sample
- Stopping the Sample Application
- JDBC Sequence Bean Javadoc

How the JDBC Sequence Bean Sample Works

This sample demonstrates creating unique identification numbers using EJBs. You can use these techniques when you create an entity bean that requires a unique primary key. This sample uses a container-managed, persistence-based entity bean with a JDBC connection pool to show using a database table to create unique keys. The table and ranges for the keys are defined in the deployment descriptor.

The JDBC sequence bean sample application implements the following classes:

Class	Description
`Client`	This class demonstrates calling an entity bean, followed by creating two new accounts with an opening balance, and an automatically-generated primary key.
`AutoAccountBean`	This is the container-managed, persistence-based entity bean that uses a database table to create unique keys.
`AutoAccountPK`	This class serves as the interface of the primary key for the `AutoAccountBean` EJB.

Building and Running the JDBC Sequence Bean Sample

To build and run the JDBC sequence bean sample application, complete the following steps:

1. Verify the environment variables.
2. Copy the files for the JDBC sequence bean sample application into a work directory.
3. Change the protection attribute on the files for the JDBC sequence bean sample application.
4. Execute the `runme` command.

The following sections describe these steps, and also explain the following:

- How to run the JDBC sequence bean sample application
- Processes and files created by the JDBC sequence bean sample application

Verifying the Settings of the Environment Variables

Before building and running the sample application, you need to ensure that certain environment variables are set on your system. In most cases, these environment variables are set as part of the installation procedure. However, you need to check the environment variables to ensure they reflect correct information.

Table 4-1 lists the environment variables required to run the JDBC sequence bean sample application.

Table 4-1 Environment Variables

Environment Variable	Description

`TUXDIR`	The directory path where you installed the WebLogic Enterprise software. For example:
	Windows NT
	`TUXDIR=c:\wledir`
	UNIX
	`TUXDIR=/usr/local/wledir`

`JAVA_HOME`	The directory path where you installed the JDK software. For example:
	Windows NT
	`JAVA_HOME =c:\JDK1.2`
	UNIX
	`JAVA_HOME =/usr/local/JDK1.2`

`ORACLE_HOME`	The directory path where you installed the Oracle software. For example: `ORACLE_HOME=/usr/local/oracle` You need to set this environment variable on UNIX operating systems only.

You may optionally set the following system environment variables to change their default value prior to running the JDBC sequence bean sample application `runme` command. See the *Administration Guide* for more information about selecting appropriate values for these environment variables.

Table 4-2 lists the optional environment variables required to run the JDBC sequence bean sample application.

Table 4-2 Optional Environment Variables

Environment Variable	Description
`HOST`	The host name portion of the TCP/IP network address used by the ISL process to accept connections from Java clients. The default value is the name of the local machine.
`PORT`	The TCP port number at which the ISL process listens for incoming requests; it must be a number between 0 and 65535. The default value is 2468.
`IPCKEY`	The address of shared memory; it must be a number greater than 32769 unique to this application on this system. The default value is 55432.
`DB_INSTANCE`	Name of the database instance or server. The default value for Oracle is `Beq-Local`. This is needed only for the samples that use a database.
`DB_USER`	Name of the database user. The default is `scott`. This is needed only for the samples that use a database.
`DB_PASSWORD`	Password for the database user. The default is `tiger`. This is needed only for the samples that use a database.
`DB_DRIVER`	The Java class name of the database driver. The default is the Oracle 8i driver, `weblogic.jdbc20.oci815.Driver`.This is needed only for the samples that use a database.
`DB_URL`	Database connection URL. The default for Oracle is `jdbc:weblogic:oracle:Beq-Local`. If the database instance is not named, set `DB_INSTANCE` to `null`.This is needed only for the samples that use a database.

Verifying the Environment Variables

To verify that the information for the environment variables defined during installation is correct, complete the following steps:

Windows NT

1. From the Start menu, select Settings.
2. From the Settings menu, select the Control Panel.

   The Control Panel appears.
3. Click the System icon.

   The System Properties window appears.
4. Click the Environment tab.

   The Environment page appears.
5. Check the settings for `TUXDIR` and `JAVA_HOME`.

UNIX

1. Enter the `ksh` command to use the Korn shell.
2. Enter the `printenv` command to display the values of `TUXDIR` and `JAVA_HOME`, as in the following example:

   ```
   ksh prompt>printenv TUXDIR
   ksh prompt>printenv JAVA_HOME
   ```

Changing the Environment Variables

To change the environment variable settings, complete the following steps:

Windows NT

1. From the Start menu, select Settings.
2. From the Settings menu, select the Control Panel.

   The Control Panel appears.
3. Click the System icon.

   The System Properties window appears.
4. Click the Environment tab.

   The Environment page appears.

5. On the Environment page in the System Properties window, click the environment variable you want to change or enter the name of the environment variable in the Variable field.

6. Enter the correct information for the environment variable in the Value field.

7. Click OK to save the changes.

UNIX

1. Enter the `ksh` command to use the Korn shell.

2. Enter the `export` command to set the correct values for the `TUXDIR` and `JAVA_HOME` environment variables, as in the following example:

   ```
   ksh prompt>export TUXDIR=directorypath
   ksh prompt>export JAVA_HOME=directorypath
   ```

Copying the Files for the JDBC Sequence Bean Sample Application into a Work Directory

You need to copy the files for the JDBC sequence bean sample application into a work directory on your local machine. The following steps describe how to copy all the example files into a work directory.

1. If you are using a UNIX system, enter the `ksh` command to use the Korn shell:

   ```
   prompt>ksh
   ksh prompt>
   ```

2. Create a work directory on your machine under the directory described in step 2 in the section “Before you Build and Run the EJB Sample Applications” on page 1-4.

3. Copy the contents of the following directory into the work directory:

   Windows NT

   ```
   %TUXDIR%\samples\j2ee\ejb\sequence\jdbc
   ```

   UNIX

   ```
   $TUXDIR/samples/j2ee/ejb/sequence/jdbc
   ```

The files copied into the work directory are in two categories:

- Files specific to the JDBC sequence bean example
- Utility files used for building and running the example application

Sample Application Files

Table 4-3lists and describes all the files for this sample application.

Table 4-3 Sample Application Files

File	Description
`ejb-jar.xml`	The XML deployment descriptor file used to help add the bean to the EJB container.
`weblogic-ejb-extensions.xml`	A file containing the WebLogic Enterprise extensions to the deployment descriptor DTD.
`Client.Java`	The Java source code for the client application.
`AutoAccountBean.java`	The Java source code for the container-managed entity bean. This class contains the business logic method implementations and methods required by the EJB specification 1.1.
`AutoAccount.java`	The Java source code for the remote interface of the `AutoAccountBean` class.
`AutoAccountHome.java`	The Java source code for the home interface of the `AutoAccountBean` class.
`AutoAccountPK.java`	The primary key for the EJB.
`ProcessingErrorException.java`	Application-specific exception thrown by the `AutoAccountBean` class for business methods.
`index.html`	File containing these instructions.

Utility Files

Table 4-4 lists and describes the utility files for this sample application. These files are generated based on the WebLogic Enterprise installation environment. The following files are generated in the same directory as where the source files are found.

Table 4-4 Utility Files

File	Description
`runme.cmd`	The Windows NT batch file that contains commands to set the environment, boot the server, and execute the client for this sample
`runme.ksh`	The UNIX Korn shell script that contains commands to boot the server and execute the client for this sample.
`run_client.cmd`	The batch file to run the client application on Windows NT systems.
`run_client.ksh`	The script file to run the client on UNIX systems.
`setenv.cmd`	The batch file to set the necessary environment variables on Windows NT systems.
`setenv.ksh`	The script file to set the necessary environment variables on UNIX systems.
`ubbconfig`	The WebLogic Enterprise server configuration file to be used on UNIX systems.
`ubbconfig.nt`	The WebLogic Enterprise server configuration file to be used on Windows NT systems.
`ejb_sequence_jdbc.jar`	The EJB JAR file that contains the source file classes, the container-specific class files generated by the `ejbc` command, and the deployment descriptor files. This is the EJB JAR file that is deployed on the WebLogic Enterprise server.

Changing the Protection Attribute on the Files for the JDBC Sequence Bean Sample Application

During the installation of the WebLogic Enterprise software, the sample application files are marked read-only. Before you can edit or build the files in the JDBC sequence bean sample application, you need to change the protection attribute of the files you copied into your work directory, as follows:

Windows NT

```
prompt>attrib /S -r drive:\workdirectory\*.*
```

UNIX

```
prompt>ksh
```

```
ksh prompt>chmod +w /workdirectory/*.*
```

On the UNIX operating system platform, you also need to change the permission of `runme.ksh` and `clean.ksh` to give execute permission to those files, as follows:

```
ksh prompt>chmod +x *.ksh
```

Creating Database Records to Use With the Sample

Before you build and run the JDBC sequence bean sample application, make sure you do the following:

1. Create the table in your database using SQL statements such as the following:

```
"create table ejbAccounts (id varchar(15), bal float, type varchar(15))"
"create table idGenerator (tablename varchar(32), maxKey int)"
```

2. Edit the `weblogic-ejb-extensions.xml` file to indicate the appropriate database information. The locations within the `weblogic-ejb-extensions.xml` file that you need to edit are identified by the following comments:

   ```
   *** DATABASE INFORMATION SPECIFIC TO INSTALLATION SITE ***
   ```

   The values you need to provide are for the database URL, user, and password.

Executing the runme Command

The `runme` command automates the following steps:

1. Loads the `UBBCONFIG` file.
2. Compiles the code for the JDBC sequence bean sample application.
3. Starts the server application using the `tmboot` command.
4. Starts the client application.
5. Stops the server application using the `tmshutdown` command.

To build and run the stateless session bean sample application, make sure you have copied the scripts described in “Before you Build and Run the EJB Sample Applications” on page 1-4, and enter the `runme` command, as follows:

Windows NT:

```
prompt>cd workdirectory

prompt>runme sequence jdbc
```

UNIX:

```
ksh prompt>cd workdirectory

ksh prompt>./runme.ksh sequence jdbc
```

A number of messages are displayed, along with whether or not the build procedure was successful. Note that the sample is not only built, but also the servers are booted and the client is run once.

Running the Sample Manually

After you have executed the `runme` command, you can run the sample manually if you like. To run the samples manually, complete the following steps:

1. Change to the sample's work directory, if necessary.
2. Make sure that your environment is set correctly by entering the following command:

Windows NT

```
prompt>setenv
```

UNIX

```
prompt>. ./setenv.ksh
```

3. Boot the server and run the client by entering the following commands:

   Windows NT

   ```
   prompt>tmboot -y
   prompt>run_client.cmd
   ```

   UNIX

   ```
   prompt>tmboot -y
   prompt>./run_client.ksh
   ```

Processes and Files Generated by the Sample Application

When the you enter the `tmboot` command to start the JDBC sequence bean sample application, the following server processes are started:

- `TMSYSEVT`

  The BEA Tuxedo system Event Broker.

- `TMFFNAME`

  The following `TMFFNAME` processes are started:

  - The `TMFFNAME` server process with the `-N` option and the `-M` option is the `MASTER NameManager` service. The `-N` option starts the `NameManager` service; the `-M` option starts this name manager as a `MASTER`. This service maintains a mapping of application-supplied names to object references.
  - The `TMFFNAME` server process with the `-N` option is a `SLAVE NameManager` service.
  - The `TMFFNAME` server with the `-F` option contains the FactoryFinder object.

- `JavaServer`

  The JavaServer takes one or more EJB JAR files that were created for the application.

- `ISL`

  The IIOP Listener/Handler.

Stopping the Sample Application

You can stop the JDBC sequence bean sample application by entering the following command:

```
prompt>tmshutdown -y
```

JDBC Sequence Bean Javadoc

The Javadoc for the JDBC sequence bean example is in the following location:

Windows NT

```
%TUXDIR%/samples/j2ee/ejb/sequence/jdbc/index.html
```

UNIX

```
$TUXDIR\samples\j2ee\ejb\sequence\jdbc\index.html
```

If you are viewing this document in a browser, you can click the following link to display this Javadoc:

`Package samples.j2ee.ejb.sequence.jdbc`

CHAPTER

5 Oracle Sequence Bean Sample Application

This topic includes the following sections:

- How the Oracle Sequence Bean Sample Works
- Building and Running the Oracle Sequence Bean Sample
- Stopping the Sample Application
- Oracle Sequence Bean Javadoc

How the Oracle Sequence Bean Sample Works

This sample demonstrates creating unique identication numbers using EJBs. You could use these techniques when you create entity beans, which require a unique primary key. This sample uses a container-managed persistence (CMP) based entity bean with a JDBC connection pool to demonstrate the use of the Oracle database's sequence procedure to create unique primary keys. This sample depends on system tables that are shipped with Oracle.

This sample shows:

- How to implement an entity bean that uses container-managed persistence in WebLogic Enterprise 5.1 according to the Sun Microsystems, Inc. EJB Specification 1.1.
- The required deployment descriptor file.
- The environment and `UBBCONFIG` file for booting such an application as a WebLogic Enterprise 5.1 server application.

The Oracle sequence bean sample application implements the following classes:

Class	Description
`Client`	This class demonstrates calling an entity bean, followed by creating two new accounts with an opening balance and an automatically-generated primary key.
`OracleBean`	This is the container-managed persistence based entity bean that uses a database table to create unique keys.
`OraclePK`	This class serves as the interface of the primary key for the `OracleBean` EJB.

Building and Running the Oracle Sequence Bean Sample

To build and run the Oracle sequence bean sample application, complete the following steps:

1. Verify the environment variables.
2. Copy the files for the Oracle sequence bean sample application into a work directory.
3. Change the protection attribute on the files for the Oracle sequence bean sample application.

4. Execute the `runme` command.

The following sections describe these steps, and also explain the following:

- How to run the Oracle sequence bean sample application
- Processes and files created by the Oracle sequence bean sample application

Verifying the Settings of the Environment Variables

Before building and running the sample application, you need to ensure that certain environment variables are set on your system. In most cases, these environment variables are set as part of the installation procedure. However, you need to check the environment variables to ensure they reflect correct information.

Table 5-1 lists the environment variables required to run the Oracle sequence bean sample application.

Table 5-1 Environment Variables

Environment Variable	Description

`TUXDIR`	The directory path where you installed the WebLogic Enterprise software. For example:
	Windows NT
	`TUXDIR=c:\wledir`
	UNIX
	`TUXDIR=/usr/local/wledir`

`JAVA_HOME`	The directory path where you installed the JDK software. For example:
	Windows NT
	`JAVA_HOME =c:\JDK1.2`
	UNIX
	`JAVA_HOME =/usr/local/JDK1.2`

Environment Variable	Description
`ORACLE_HOME`	The directory path where you installed the Oracle software. For example: `ORACLE_HOME=/usr/local/oracle` You need to set this environment variable on UNIX operating systems only.

You may optionally set the following system environment variables to change their default value prior to running the Oracle sequence bean sample application `runme` command. See the *Administration Guide* for more information about selecting appropriate values for these environment variables.

Table 5-2 lists the optional environment variables required to run the Oracle sequence bean sample application.

Table 5-2 Optional Environment Variables

Environment Variable	Description
`HOST`	The host name portion of the TCP/IP network address used by the ISL process to accept connections from Java clients. The default value is the name of the local machine.
`PORT`	The TCP port number at which the ISL process listens for incoming requests; it must be a number between 0 and 65535. The default value is 2468.
`IPCKEY`	The address of shared memory; it must be a number greater than 32769 unique to this application on this system. The default value is 55432.
`DB_INSTANCE`	Name of the database instance or server. The default value for Oracle is `Beq-Local`. This is needed only for the samples that use a database.
`DB_USER`	Name of the database user. The default is `scott`. This is needed only for the samples that use a database.
`DB_PASSWORD`	Password for the database user. The default is `tiger`. This is needed only for the samples that use a database.

Environment Variable	Description
DB_DRIVER	The Java class name of the database driver. The default is the Oracle 8i driver, `weblogic.jdbc20.oci815.Driver`.This is needed only for the samples that use a database.
DB_URL	Database connection URL. The default for Oracle is `jdbc:weblogic:oracle:Beq-Local`. If the database instance is not named, set `DB_INSTANCE` to `null`. This is needed only for the samples that use a database.

Verifying the Environment Variables

To verify that the information for the environment variables defined during installation is correct, complete the following steps:

Windows NT

1. From the Start menu, select Settings.
2. From the Settings menu, select the Control Panel.

   The Control Panel appears.
3. Click the System icon.

   The System Properties window appears.
4. Click the Environment tab.

   The Environment page appears.
5. Check the settings for `TUXDIR` and `JAVA_HOME`.

UNIX

1. Enter the `ksh` command to use the Korn shell.
2. Enter the `printenv` command to display the values of `TUXDIR` and `JAVA_HOME`, as in the following example:

```
ksh prompt>printenv TUXDIR
ksh prompt>printenv JAVA_HOME
```

Changing the Environment Variables

To change the environment variable settings, complete the following steps:

Windows NT

1. From the Start menu, select Settings.
2. From the Settings menu, select the Control Panel.

   The Control Panel appears.
3. Click the System icon.

   The System Properties window appears.
4. Click the Environment tab.

   The Environment page appears.
5. On the Environment page in the System Properties window, click the environment variable you want to change or enter the name of the environment variable in the Variable field.
6. Enter the correct information for the environment variable in the Value field.
7. Click OK to save the changes.

UNIX

1. Enter the `ksh` command to use the Korn shell.
2. Enter the `export` command to set the correct values for the `TUXDIR` and `JAVA_HOME` environment variables, as in the following example:

```
ksh prompt>export TUXDIR=directorypath
ksh prompt>export JAVA_HOME=directorypath
```

Copying the Files for the Oracle Sequence Bean Sample Application into a Work Directory

You need to copy the files for the Oracle sequence bean sample application into a work directory on your local machine. The following steps describe how to copy all the example files into a work directory.

1. If you are using a UNIX system, enter the `ksh` command to use the Korn shell:

   ```
   prompt>ksh
   ksh prompt>
   ```

2. Create a work directory on your machine under the directory described in step 2 in the section "Before you Build and Run the EJB Sample Applications" on page 1-4.

3. Copy the contents of the following directory into the work directory:

   Windows NT

   `%TUXDIR%\samples\j2ee\ejb\sequence\oracle`

   UNIX

   `$TUXDIR/samples/j2ee/ejb/sequence/oracle`

The files copied into the work directory are in two categories:

- Files specific to the Oracle sequence bean example
- Utility files used for building and running the example application

Sample Application Files

Table 5-3 lists and describes all the files for this sample application.

Table 5-3 Sample Application Files

File	Description
`ejb-jar.xml`	The XML deployment descriptor file used to help add the bean to the EJB container.
`weblogic-ejb-extensions.xml`	A file containing the WebLogic Enterprise extensions to the deployment descriptor DTD.
`Client.Java`	The Java source code for the client application.
`OracleBean.java`	The Java source code for the entity bean that uses container-managed persistence. This class contains the business logic method implementations and methods required by the EJB specification 1.1.

File	Description
`Oracle.java`	The Java source code for the remote interface of the `OracleBean` class.
`OracleHome.java`	The Java source code for the home interface of the `OracleBean` class.
`OraclePK.java`	Primary key for the bean.
`ProcessingErrorException.java`	Application-specific exception thrown by the `OracleBean` class for business methods.
`index.html`	File containing these instructions.

Utility Files

Table 5-4 lists and describes the utility files for this sample application. These files are generated based on the WebLogic Enterprise installation environment. The following files are generated in the same directory as where the source files are found.

Table 5-4 Utility Files

File	Description
`runme.cmd`	The Windows NT batch file that contains commands to set the environment, boot the server, and execute the client for this sample.
`runme.ksh`	The UNIX Korn shell script that contains commands to boot the server and execute the client for this sample.
`run_client.cmd`	The batch file to run the client application on Windows NT systems.
`run_client.ksh`	The script file to run the client on UNIX systems.
`setenv.cmd`	The batch file to set the necessary environment variables on Windows NT systems.
`setenv.ksh`	The script file to set the necessary environment variables on UNIX systems.

File	Description
`ubbconfig`	The WebLogic Enterprise server configuration file to be used on UNIX systems.
`ubbconfig.nt`	The WebLogic Enterprise server configuration file to be used on Windows NT systems.
`ejb_sequence_oracle.jar`	The EJB JAR file that contains the source file classes, the container-specific class files generated by the `ejbc` command, and the deployment descriptor files. This is the EJB JAR file that is deployed on the WebLogic Enterprise server.

Changing the Protection Attribute on the Files for the Oracle sequence Bean Sample Application

During the installation of the WebLogic Enterprise software, the sample application files are marked read-only. Before you can edit or build the files in the Oracle sequence bean sample application, you need to change the protection attribute of the files you copied into your work directory, as follows:

Windows NT

```
prompt>attrib /S -r drive:\workdirectory\*.*
```

UNIX

```
prompt>ksh
```

```
ksh prompt>chmod +w /workdirectory/*.*
```

On the UNIX operating system platform, you also need to change the permission of `runme.ksh` and `clean.ksh` to give execute permission to those files, as follows:

```
ksh prompt>chmod +x *.ksh
```

Creating Database Records to Use With the Sample

Before you run the Oracle sequence bean sample application, make sure you do the following:

1. Create a sequence that includes one or more appropriate SQL statements, such as the following:

   ```
   "create sequence ejbSequence start with 8001"
   ```

2. Edit the `weblogic-ejb-extensions.xml` file to indicate the appropriate database information. The locations within the `weblogic-ejb-extensions.xml` file that you need to edit are identified by the following comments:

   ```
   *** DATABASE INFORMATION SPECIFIC TO INSTALLATION SITE ***
   ```

   The values you need to provide are for the database URL, user, and password.

Executing the runme Command

The `runme` command automates the following steps:

1. Loads the `UBBCONFIG` file.
2. Compiles the code for the Oracle sequence bean sample application.
3. Starts the server application using the `tmboot` command.
4. Starts the client application.
5. Stops the server application using the `tmshutdown` command.

To build and run the stateless session bean sample application, make sure you have copied the scripts described in “Before you Build and Run the EJB Sample Applications” on page 1-4, and enter the `runme` command, as follows:

Windows NT

```
prompt>cd workdirectory
```

```
prompt>runme sequence oracle
```

UNIX

```
ksh prompt>cd workdirectory
```

```
ksh prompt>./runme.ksh sequence oracle
```

A number of messages are displayed, along with whether or not the build procedure was successful. Note that the sample is not only built, but also the servers are booted and the client is run once.

Running the Sample Manually

After you have executed the `runme` command, you can run the sample manually if you like. To run the samples manually, complete the following steps:

1. Change the directory to the sample's work directory, if necessary.
2. Make sure that your environment is set correctly by entering the following command:

   Windows NT

   ```
   prompt>setenv
   ```

   UNIX

   ```
   prompt>. ./setenv.ksh
   ```

3. Boot the server and run the client by entering the following commands:

   Windows NT

   ```
   prompt>tmboot -y
   prompt>run_client.cmd
   ```

   UNIX

   ```
   prompt>tmboot -y
   prompt>./run_client.ksh
   ```

Processes and Files Generated by the Sample Application

When the you enter the `tmboot` command to start the Oracle sequence bean sample application, the following server processes are started:

- `TMSYSEVT`

  The BEA Tuxedo system Event Broker.

- `TMFFNAME`

  The following `TMFFNAME` processes are started:

  - The `TMFFNAME` server process with the `-N` option and the `-M` option is the `MASTER NameManager` service. The `-N` option starts the `NameManager` service; the `-M` option starts this name manager as a `MASTER`. This service maintains a mapping of application-supplied names to object references.
  - The `TMFFNAME` server process with the `-N` option is a `SLAVE NameManager` service.
  - The `TMFFNAME` server with the `-F` option contains the FactoryFinder object.

- `JavaServer`

  The JavaServer takes one or more EJB JAR files that were created for the application.

- `ISL`

  The IIOP Listener/Handler.

Stopping the Sample Application

You can stop the Oracle sequence bean sample application by entering the following command:

```
prompt>tmshutdown -y
```

Oracle Sequence Bean Javadoc

The Javadoc for the Oracle sequence bean example is in the following location:

Windows NT

```
%TUXDIR%/samples/j2ee/ejb/sequence/oracle/index.html
```

UNIX

```
$TUXDIR\samples\j2ee\ejb\sequence\oracle\index.html
```

If you are viewing this document in a browser, you can click the following link to display this Javadoc:

`Package samples.j2ee.ejb.sequence.oracle`

CHAPTER

6 Parent Bean Sample Application

This topic includes the following sections:

- How the Parent Bean Sample Works
- Building and Running the Parent Bean Sample
- Stopping the Sample Application
- Parent Bean Javadoc

How the Parent Bean Sample Works

The parent bean sample and the child bean sample together show subclassing an EJB. Each sample shows a stateless session bean. The parent bean contains methods that are overwritten by the subclassed child bean, and the methods that are specific to the parent bean.

The parent bean sample application implements the following classes:

Class	Description
`Client`	This class is used with the `ChildBean` EJB to illustrate using an EJB that subclasses from a another bean.
`ParentBean`	`ParentBean` is a stateless session bean.

Building and Running the Parent Bean Sample

To build and run the parent bean sample application, complete the following steps:

1. Verify the environment variables.
2. Copy the files for the parent bean sample application into a work directory.
3. Change the protection attribute on the files for the parent bean sample application.
4. Execute the `runme` command.

The following sections describe these steps, and also explain the following:

- How to run the parent bean sample application
- Processes and files created by the parent bean sample application

Verifying the Settings of the Environment Variables

Before building and running the parent bean sample application, you need to ensure that certain environment variables are set on your system. In most cases, these environment variables are set as part of the installation procedure. However, you need to check the environment variables to ensure they reflect correct information.

The following table lists the environment variables required to run the parent bean sample application.

Environment Variable	Description

`TUXDIR`	The directory path where you installed the WebLogic Enterprise software. For example:
	Windows NT
	`TUXDIR=c:\wledir`
	UNIX
	`TUXDIR=/usr/local/wledir`

`JAVA_HOME`	The directory path where you installed the JDK software. For example:
	Windows NT
	`JAVA_HOME =c:\JDK1.2`
	UNIX
	`JAVA_HOME =/usr/local/JDK1.2`

You may optionally set the following system environment variables to change their default value prior to running the parent bean sample application `runme` command. See the *Administration Guide* for more information about selecting appropriate values for these environment variables.

Table 6-1 lists the optional environment variables required to run the parent bean sample application.

Table 6-1 Optional Environment Variables

Environment Variable	Description
`HOST`	The host name portion of the TCP/IP network address used by the ISL process to accept connections from Java clients. The default value is the name of the local machine.
`PORT`	The TCP port number at which the ISL process listens for incoming requests; it must be a number between 0 and 65535. The default value is 2468.
`IPCKEY`	The address of shared memory; it must be a number greater than 32769 unique to this application on this system. The default value is 55432.

Verifying the Environment Variables

To verify that the information for the environment variables defined during installation is correct, complete the following steps:

Windows NT

1. From the Start menu, select Settings.
2. From the Settings menu, select the Control Panel.

   The Control Panel appears.
3. Click the System icon.

   The System Properties window appears.
4. Click the Environment tab.

   The Environment page appears.
5. Check the settings for `TUXDIR` and `JAVA_HOME`.

UNIX

1. Enter the `ksh` command to use the Korn shell.
2. Enter the `printenv` command to display the values of `TUXDIR` and `JAVA_HOME`, as in the following example:

   ```
   ksh prompt>printenv TUXDIR
   ksh prompt>printenv JAVA_HOME
   ```

Changing the Environment Variables

To change the environment variable settings, complete the following steps:

Windows NT

1. From the Start menu, select Settings.
2. From the Settings menu, select the Control Panel.

   The Control Panel appears.
3. Click the System icon.

   The System Properties window appears.

4. Click the Environment tab.

   The Environment page appears.

5. On the Environment page in the System Properties window, click the environment variable you want to change or enter the name of the environment variable in the Variable field.

6. Enter the correct information for the environment variable in the Value field.

7. Click OK to save the changes.

UNIX

1. Enter the `ksh` command to use the Korn shell.

2. Enter the `export` command to set the correct values for the `TUXDIR` and `JAVA_HOME` environment variables, as in the following example:

   ```
   ksh prompt>export TUXDIR=directorypath
   ksh prompt>export JAVA_HOME=directorypath
   ```

Copying the Files for the Parent Bean Sample Application into a Work Directory

You need to copy the files for the parent bean sample application into a work directory on your local machine. The following steps describe how to copy all the example files into a work directory.

1. If you are using a Unix system, enter the `ksh` command to use the Korn shell:

   ```
   prompt>ksh
   ksh prompt>
   ```

2. Create a work directory on your machine under the directory described in step 2 in the section “Before you Build and Run the EJB Sample Applications” on page 1-4.

3. Copy the contents of the following directory into the work directory:

   Windows NT

   ```
   %TUXDIR%\samples\j2ee\ejb\subclass\parent
   ```

UNIX

```
$TUXDIR/samples/j2ee/ejb/subclass/parent
```

The files copied into the work directory are in two categories:

- Files specific to the parent bean example
- Utility files used for building and running the example application

Sample Application Files

Table 6-2 lists and describes all the files for this sample application.

Table 6-2 Sample Application Files

File	Description
`ejb-jar.xml`	The XML deployment descriptor file used to help add the bean to the EJB container.
`weblogic-ejb-extensions.xml`	A file containing the WebLogic Enterprise extensions to the deployment descriptor DTD.
`Client.Java`	The Java source code for the client application.
`ParentBean.java`	The Java source code for the stateless session bean. This class contains the business logic method implementations and methods required by the EJB specification 1.1.
`Parent.java`	The Java source code for the remote interface of the `ParentBean` class.
`ParentHome.java`	The Java source code for the home interface of the the `ParentBean` class.
`index.html`	File containing these instructions.

Utility Files

Table 6-3 lists and describes the utility files for this sample application. These files are generated based on the WebLogic Enterprise installation environment. The following files generated in the same directory as where the source files are found.

Table 6-3 Utility Files

File	Description
`runme.cmd`	The Windows NT batch file that contains commands to set the environment, boot the server, and execute the client for this sample.
`runme.ksh`	The UNIX Korn shell script that contains commands to boot the server and execute the client for this sample.
`run_client.cmd`	The batch file to run the client application on Windows NT systems.
`run_client.ksh`	The script file to run the client on UNIX systems.
`setenv.cmd`	The batch file to set the necessary environment variables on Windows NT systems.
`setenv.ksh`	The script file to set the necessary environment variables on UNIX systems.
`ubbconfig`	The WebLogic Enterprise server configuration file to be used on UNIX systems.
`ubbconfig.nt`	The WebLogic Enterprise server configuration file to be used on Windows NT systems.
`ejb_subclass_parent.jar`	The EJB JAR file that contains the source file classes, the container-specific class files generated by the `ejbc` command, and the deployment descriptor files. This is the EJB JAR file that is deployed on the WebLogic Enterprise server.

Changing the Protection Attribute on the Files for the Parent Bean Sample Application

During the installation of the WebLogic Enterprise software, the sample application files are marked read-only. Before you can edit or build the files in the parent bean sample application, you need to change the protection attribute of the files you copied into your work directory, as follows:

Windows NT

```
prompt>attrib /S -r drive:\workdirectory\*.*
```

UNIX

```
prompt>ksh
```

```
ksh prompt>chmod +w /workdirectory/*.*
```

On the UNIX operating system platform, you also need to change the permission of `runme.ksh` and `clean.ksh` to give execute permission to those files, as follows:

```
ksh prompt>chmod +x *.ksh
```

Executing the runme Command

The `runme` command automates the following steps:

1. Loads the `UBBCONFIG` file.
2. Compiles the code for the parent bean sample application.
3. Starts the server application using the `tmboot` command.
4. Starts the client application.
5. Stops the server application using the `tmshutdown` command.

To build and run the stateless session bean sample application, make sure you have copied the scripts described in “Before you Build and Run the EJB Sample Applications” on page 1-4, and enter the `runme` command, as follows:

Windows NT

```
prompt>cd workdirectory
```

```
prompt>runme subclass parent
```

UNIX

```
ksh prompt>cd workdirectory
```

```
ksh prompt>./runme.ksh subclass parent
```

A number of messages are displayed, along with whether or not the build procedure was successful. Note that the sample is not only built, but also the servers are booted and the client is run once.

Running the Sample Manually

After you have executed the `runme` command, you can run the sample manually if you like. To run the samples manually, complete the following steps:

1. Change to the sample's work directory, if necessary.
2. Make sure that your environment is set correctly by entering the following command:

   Windows NT

   ```
   prompt>setenv
   ```

   UNIX

   ```
   prompt>. ./setenv.ksh
   ```

3. Boot the server and run the client by entering the following commands:

   Windows NT

   ```
   prompt>tmboot -y
   prompt>run_client.cmd
   ```

   UNIX

   ```
   prompt>tmboot -y
   prompt>./run_client.ksh
   ```

Processes and Files Generated by the Sample Application

When you enter the `tmboot` command to start one of the EJB sample applications, the following server processes are started:

- `TMSYSEVT`

  The BEA Tuxedo system Event Broker.

- `TMFFNAME`

  The following `TMFFNAME` processes are started:

  - The `TMFFNAME` server process with the `-N` option and the `-M` option is the `MASTER NameManager` service. The `-N` option starts the `NameManager` service; the `-M` option starts this name manager as a `MASTER`. This service maintains a mapping of application-supplied names to object references.
  - The `TMFFNAME` server process with the `-N` option is a `SLAVE NameManager` service.
  - The `TMFFNAME` server with the `-F` option contains the FactoryFinder object.

- `JavaServer`

  The JavaServer takes one or more EJB JAR files that were created for the application.

- `ISL`

  The IIOP Listener/Handler.

Stopping the Sample Application

You can stop the parent bean sample application by entering the following command:

```
prompt>tmshutdown -y
```

Parent Bean Javadoc

The Javadoc for the parent bean example is in the following location:

Windows NT

```
%TUXDIR%/samples/j2ee/ejb/subclass/parent/index.html
```

UNIX

```
$TUXDIR\samples\j2ee\ejb\subclass\parent\index.html
```

If you are viewing this document in a browser, you can click the following link to display this Javadoc:

`Package samples.j2ee.ejb.subclass.parent`

CHAPTER

7 Child Bean Sample Application

This topic includes the following sections:

- How the Child Bean Sample Works
- Building and Running the Child Bean Sample
- Stopping the Sample Application
- Child Bean Javadoc

How the Child Bean Sample Works

The parent bean sample and the child bean sample together show subclassing an EJB. Each sample shows a stateless session bean. The child bean overrides some methods from the parent bean, and also invokes methods directly on the parent bean. Thus, the child bean sample shows subclassing and one bean invoking another bean.

The parent bean sample application implements the following classes:

Class	Description
`Client`	This class is used with the `ParentBean` EJB to illustrate using an EJB that subclasses from a another bean.
`ChildBean`	`ChildBean` is a stateless session bean.

Building and Running the Child Bean Sample

To build and run the child bean sample application, complete the following steps:

1. Verify the environment variables.
2. Copy the files for the child bean sample application into a work directory.
3. Change the protection attribute on the files for the child bean sample application.
4. Execute the `runme` command.

The following sections describe these steps, and also explain the following:

- How to run the child bean sample application
- Processes and files created by the child bean sample application

Verifying the Settings of the Environment Variables

Before building and running the child bean sample application, you need to ensure that certain environment variables are set on your system. In most cases, these environment variables are set as part of the installation procedure. However, you need to check the environment variables to ensure they reflect correct information.

Table 7-1 lists the environment variables required to run the child bean sample application.

Table 7-1 Environment Variables

Environment Variable	Description

`TUXDIR`	The directory path where you installed the WebLogic Enterprise software. For example:
	Windows NT
	`TUXDIR=c:\wledir`
	UNIX
	`TUXDIR=/usr/local/wledir`

`JAVA_HOME`	The directory path where you installed the JDK software. For example:
	Windows NT
	`JAVA_HOME =c:\JDK1.2`
	UNIX
	`JAVA_HOME =/usr/local/JDK1.2`

You may optionally set the following system environment variables to change their default value prior to running the child bean sample application `runme` command. See the *Administration Guide* for more information about selecting appropriate values for these environment variables.

Table 7-2 lists the optional environment variables required to run the child bean sample application.

Table 7-2 Optional Environment Variables

Environment Variable	Description
`HOST`	The host name portion of the TCP/IP network address used by the ISL process to accept connections from Java clients. The default value is the name of the local machine.
`PORT`	The TCP port number at which the ISL process listens for incoming requests; it must be a number between 0 and 65535. The default value is 2468.

Environment Variable	Description
`IPCKEY`	The address of shared memory; it must be a number greater than 32769 unique to this application on this system. The default value is 55432.

Verifying the Environment Variables

To verify that the information for the environment variables defined during installation is correct, complete the following steps:

Windows NT

1. From the Start menu, select Settings.
2. From the Settings menu, select the Control Panel.

   The Control Panel appears.
3. Click the System icon.

   The System Properties window appears.
4. Click the Environment tab.

   The Environment page appears.
5. Check the settings for `TUXDIR` and `JAVA_HOME`.

UNIX

1. Enter the `ksh` command to use the Korn shell.
2. Enter the `printenv` command to display the values of `TUXDIR` and `JAVA_HOME`, as in the following example:

```
ksh prompt>printenv TUXDIR
ksh prompt>printenv JAVA_HOME
```

Changing the Environment Variables

To change the environment variable settings, complete the following steps:

Windows NT

1. From the Start menu, select Settings.
2. From the Settings menu, select the Control Panel.

   The Control Panel appears.
3. Click the System icon.

   The System Properties window appears.
4. Click the Environment tab.

   The Environment page appears.
5. On the Environment page in the System Properties window, click the environment variable you want to change or enter the name of the environment variable in the Variable field.
6. Enter the correct information for the environment variable in the Value field.
7. Click OK to save the changes.

UNIX

1. Enter the `ksh` command to use the Korn shell.
2. Enter the `export` command to set the correct values for the `TUXDIR` and `JAVA_HOME` environment variables, as in the following example:

   ```
   ksh prompt>export TUXDIR=directorypath
   ksh prompt>export JAVA_HOME=directorypath
   ```

Copying the Files for the Child Bean Sample Application into a Work Directory

You need to copy the files for the child bean sample application into a work directory on your local machine. The following steps describe how to copy all the example files into a work directory.

1. If you are using a UNIX system, enter the `ksh` command to use the Korn shell:

   ```
   prompt>ksh
   ksh prompt>
   ```

2. Create a work directory on your machine under the directory described in step 2 in the section “Before you Build and Run the EJB Sample Applications” on page 1-4.

3. Copy the contents of the following directory into the work directory:

   Windows NT

   ```
   %TUXDIR%\samples\j2ee\ejb\subclass\child
   ```

   UNIX

   ```
   $TUXDIR/samples/j2ee/ejb/subclass/child
   ```

The files copied into the work directory are in two categories:

- Files specific to the child bean example
- Utility files used for building and running the example application

Sample Application Files

Table 7-3 lists and describes all the files for this sample application:

Table 7-3 Sample Application Files

File	Description
`ejb-jar.xml`	The XML deployment descriptor file used to help add the bean to the EJB container.
`weblogic-ejb-extensions.xml`	A file containing the WebLogic Enterprise extensions to the deployment descriptor DTD.
`Client.Java`	The Java source code for the client application.
`ChildBean.java`	The Java source code for the stateless session bean. This class contains the business logic method implementations and methods required by the EJB specification 1.1.

File	Description
`Child.java`	The Java source code for the remote interface of the `ChildBean` class.
`ChildHome.java`	The Java source code for the home interface of the the `ChildBean` class.
`index.html`	File containing these instructions.

Utility Files

Table 7-4 lists and describes the utility files for this sample application. These files are generated based on the WebLogic Enterprise installation environment. The following files are generated in the same directory as where the source files are found.

Table 7-4 Utility Files

File	Description
`runme.cmd`	The Windows NT batch file that contains commands to set the environment, boot the server, and execute the client for this sample
`runme.ksh`	The UNIX Korn shell script that contains commands to boot the server and execute the client for this sample.
`run_client.cmd`	The batch file to run the client application on Windows NT systems.
`run_client.ksh`	The script file to run the client on UNIX systems.
`setenv.cmd`	The batch file to set the necessary environment variables on Windows NT systems.
`setenv.ksh`	The script file to set the necessary environment variables on UNIX systems.
`ubbconfig`	The WebLogic Enterprise server configuration file to be used on UNIX systems.

File	Description
`ubbconfig.nt`	The WebLogic Enterprise server configuration file to be used on Windows NT systems.
`ejb_subclass_child.jar`	The EJB JAR file that contains the source file classes, the container-specific class files generated by the `ejbc` command, and the deployment descriptor files. This is the EJB JAR file that is deployed on the WebLogic Enterprise server.

Changing the Protection Attribute on the Files for the Child Bean Sample Application

During the installation of the WebLogic Enterprise software, the sample application files are marked read-only. Before you can edit or build the files in the child bean sample application, you need to change the protection attribute of the files you copied into your work directory, as follows:

Windows NT

```
prompt>attrib /S -r drive:\workdirectory\*.*
```

UNIX

```
prompt>ksh
```

```
ksh prompt>chmod +w /workdirectory/*.*
```

On the UNIX operating system platform, you also need to change the permission of `runme.ksh` and `clean.ksh` to give execute permission to those files, as follows:

```
ksh prompt>chmod +x *.ksh
```

Executing the runme Command

The `runme` command automates the following steps:

1. Loads the `UBBCONFIG` file.

2. Compiles the code for the child bean sample application.
3. Starts the server application using the `tmboot` command.
4. Starts the client application.
5. Stops the server application using the `tmshutdown` command.

To build and run the stateless session bean sample application, make sure you have copied the scripts described in "Before you Build and Run the EJB Sample Applications" on page 1-4, and enter the `runme` command, as follows:

Windows NT

```
prompt>cd workdirectory

prompt>runme subclass child
```

UNIX

```
ksh prompt>cd workdirectory

ksh prompt>./runme.ksh subclass child
```

A number of messages are displayed, along with whether or not the build procedure was successful. Note that the sample is not only built, but also the servers are booted and the client is run once.

Running the Sample Manually

After you have executed the `runme` command, you can run the sample manually if you like. To run the samples manually, complete the following steps:

1. Change to the sample's work directory, if necessary.
2. Make sure that your environment is set correctly by entering the following command:

   Windows NT

   ```
   prompt>setenv
   ```

   UNIX

   ```
   prompt>. ./setenv.ksh
   ```

3. Boot the server and run the client by entering the following commands:

   Windows NT

   ```
   prompt>tmboot -y
   prompt>run_client.cmd
   ```

   UNIX

   ```
   prompt>tmboot -y
   prompt>./run_client.ksh
   ```

Processes and Files Generated by the Sample Application

When the you enter the `tmboot` command to start one of the EJB sample applications, the following server processes are started:

- `TMSYSEVT`

  The BEA Tuxedo system Event Broker.

- `TMFFNAME`

  The following `TMFFNAME` processes are started:

  - The `TMFFNAME` server process with the `-N` option and the `-M` option is the `MASTER NameManager` service. The `-N` option starts the `NameManager` service; the `-M` option starts this name manager as a `MASTER`. This service maintains a mapping of application-supplied names to object references.
  - The `TMFFNAME` server process with the `-N` option is a `SLAVE NameManager` service.
  - The `TMFFNAME` server with the `-F` option contains the FactoryFinder object.

- `JavaServer`

  The JavaServer takes one or more EJB JAR files that were created for the application.

- `ISL`

  The IIOP Listener/Handler.

Stopping the Sample Application

You can stop the child bean sample application by entering the following command:

```
prompt>tmshutdown -y
```

Child Bean Javadoc

The Javadoc for the child bean example is in the following location:

Windows NT

```
%TUXDIR%/samples/j2ee/ejb/subclass/child/index.html
```

UNIX

```
$TUXDIR\samples\j2ee\ejb\subclass\child\index.html
```

If you are viewing this document in a browser, you can click the following link to display this Javadoc:

`Package samples.j2ee.ejb.subclass.child`

A JNDI Utility Application

This topic includes the following sections:

- How the JNDI Utility Application Works
- Building and Running the JNDI Utility Application
- Running the JNDI Utility Application and Examining the Output

How the JNDI Utility Application Works

This application is a utility program that creates a list of all the EJBs deployed on a WebLogic Enterprise server process. This application searches the JNDI tree, and checks each entry to see if it is an EJB; if so, this application displays the following information:

- The JNDI name of the EJB's home interface
- The class name of the service stub that the client application is accessing

From the name of the service stub, you can determine the package name of the EJB.

Building and Running the JNDI Utility Application

To build and run the JNDI utility application, complete the following steps:

1. Make sure that you have successfully built at least one EJB application; for example, one of the EJB samples documented in this guide.
2. Set your `CLASSPATH` to include the EJB JAR files for the EJB applications you have built.
3. Copy the files for the JNDI utility application into a work directory.
4. Change the protection attribute on the files for the JNDI utility application
5. Compile the source file for the JNDI utility application.
6. Run the JNDI utility application and examine the output.

The sections that follow provide details on steps 3 through 6.

Copying the Files for the JNDI Utility Application into a Work Directory

You need to copy the file for the JNDI utility application into a work directory on your local machine. The following steps describe how to copy all the file into a work directory.

1. If you are using a UNIX system, enter the `ksh` command to use the Korn shell:

   ```
   prompt>ksh
   ksh prompt>
   ```

2. Create a work directory on your machine under the directory described in step 2 in the section “Before you Build and Run the EJB Sample Applications” on page 1-4.
3. Copy the contents of the following directory into the work directory:

Windows NT

```
%TUXDIR%\samples\j2ee\ejb\utils
```

UNIX

```
$TUXDIR/samples/j2ee/ejb/utils
```

The files copied into the work directory are:

- The `ListAll.java` source file
- The file `index.html`, which includes the instructions appearing in this topic

Changing the Protection Attribute on the Files for the JNDI Utility Application

During the installation of the WebLogic Enterprise software, the sample application files are marked read-only. Before you can edit or build the file in the JNDI utility application, you need to change the protection attribute of the file you copied into your work directory, as follows:

Windows NT

```
prompt>attrib /S -r drive:\workdirectory\*.*
```

UNIX

```
prompt>ksh
ksh prompt>chmod +w /workdirectory/*.*
```

Compiling the Source File for the JNDI Utility Application

Compile the source file for the JNDI utility application. For example, on Windows NT, enter the following command:

```
prompt> javac ListAll.java
```

Running the JNDI Utility Application and Examining the Output

To run the JNDI utility application, complete the following steps:

1. Change to the EJB application's work directory, if necessary.
2. Boot the EJB application and run the client by entering the following commands:

   Windows NT

   ```
   prompt>tmboot -y
   prompt>run_client.cmd
   ```

   UNIX

   ```
   prompt>tmboot -y
   prompt>./run_client.ksh
   ```

3. Change to the JNDI application’s work directory.
4. Run the JNDI utility program, as in the following command:

   ```
   prompt> java ListAll corbaloc://host:port
   ```

   In the preceding command, `host:port` represent the TCP/IP address of the IIOP listener/handler for the WebLogic Enterprise domain.

5. Examine the output displayed by the JNDI utility application, which may appear similar to the following:

```
Starting to search tree for all EJBeans...

EJBean Binding Name: statelessSession.TraderHome
Classname: samples.j2ee.ejb.basic.statelessSession.TraderBeanHomeImpl_WLStub

Finished searching tree for all EJBeans...
```

Index