

## EJB Hello World

In this lab, you will write the traditional first program that displays the string "Hello, world".

### Objectives:

- To write a simple stateless session Enterprise JavaBean
- To write a simple EJB client program
- To learn the EJB development process
- To learn how to deploy an EJB using the JBoss EJB container

### Steps:

1. Open a command-prompt (console) window and change to the `\ejbclass\helloworld\server` directory.
2. Create a file named **HelloWorld.java**, and in it, write the *HelloWorld* remote interface. Be sure to put the interface in the *hello* package.
3. Create a file named **HelloWorldHome.java**, and in it, write the *HelloWorldHome* interface. Be sure to put the interface in the *hello* package.
4. Create a file named **HelloWorldBean.java**, and in it, write the *HelloWorldBean* class. Be sure to put the interface in the *hello* package. This class should implement the required EJB session-bean lifecycle methods (with empty method bodies) and the "business logic" method from the *HelloWorld* interface.
5. Use Windows Explorer or the `mkdir` command to create a subdirectory named `\ejbclass\helloworld\server\META-INF`. In this subdirectory, create the `ejb-jar.xml` deployment descriptor. The home should be "hello>HelloWorldHome", remote should be "hello>HelloWorld" and so forth.
6. Edit the `servercp.bat` file and make sure the `JBOSS_HOME` variable matches your computer's drive and directory where JBoss is installed. Then run `servercp.bat` to set the `CLASSPATH` for server development. **Note:** This sets the `CLASSPATH` for the command window only -- you will need to re-run the batch file if you ever close the command window.
7. Compile your files:

```
javac -d . *.java
```

Be sure to put a space before the dot (after the "-d").

The "-d" switch indicates the directory in which the compiler should write the .class files (dot means the current directory). Since the interfaces and classes are in the *hello* package, the compiler will create a *hello* subdirectory and write the .class files into this new directory. You should examine this directory to ensure that the .class files have been successfully created.

Fix any compiler errors before continuing.

8. Create the JAR file that contains the Java classes and deployment descriptor:

```
jar cvf hello.jar hello\*.class META-INF
```

Look in the directory for the JAR file to ensure that it was created properly.

9. The JAR file should re-create the directory structure. To verify that the JAR is correct, type:

```
jar tvf hello.jar
```

You should see the .class files and the deployment descriptor in their proper directories. The *manifest* is added automatically by the JAR utility -- it lists the contents of the JAR.

10. Open another command window and change to the `\ejbclass\jboss\jboss\bin` directory and type "run" to start the EJB container. Watch its startup messages and wait for the "JBoss started" message. You should keep this console available so you can monitor the EJB container.
11. Now it's time to deploy the EJB. With JBoss, that's as simple as copying the JAR file to a special directory. Use Windows Explore or the appropriate command-line command to copy the **hello.jar** file to the `\ejbclass\jboss\jboss\deploy` directory. Watch the console where the EJB container is running and ensure that the EJB was loaded correctly. Fix any problems.

12. Open yet another command window and change to the `\ejbclass\helloworld\client` directory.

13. Create a file named **HelloWorldClient.java** and in it, write an EJB client program with a *main* method that:

- Creates an initial context
- Retrieves the home reference
- Casts the reference to the correct type
- Retrieves a reference to the remote EJB object
- Calls the remote method and displays the returned string to the console

Use the sample in the course notes as a guide. **Note:** the client program class does not need to be in a package.

14. Edit the **clientcp.bat** file and make sure the `JBOSS_HOME` variable matches your computer's drive and directory where JBoss is installed. Then run `clientcp.bat` to set the `CLASSPATH` for client development. **Note:** This sets the `CLASSPATH` for the command window only -- you will need to re-run the batch file if you ever close the command window.

15. Compile `HelloWorldClient.java`:

```
javac -d . *.java
```

Look in the `\ejbclass\helloworld\client` directory for the resulting .class file and fix any compile errors before continuing.

16. Run the client:

```
java HelloWorldClient
```

Do you see the message from the EJB? If you wrote any `System.out.println` in the Bean implementation, you should see them in the JBoss console.