Getting Started with Enterprise Beans

This chapter shows how to develop, deploy, and run a simple Java EE application named `converter`.

The purpose of `converter` is to calculate currency conversions between Japanese yen and Eurodollars.

The `converter` application consists of an enterprise bean, which performs the calculations, and a web client.

Here's an overview of the steps you'll follow in this chapter:

1. Create the enterprise bean: `ConverterBean`.
2. Create the web client.
3. Deploy `converter` onto the server.
4. Using a browser, run the web client.

Before proceeding, make sure that you've done the following:

- Read Chapter 1, Overview
- Become familiar with enterprise beans (see Chapter 22, Enterprise Beans)
- Started the server (see Starting and Stopping the GlassFish Server)

The following topics are addressed here:

- Creating the Enterprise Bean
- Modifying the Java EE Application

Creating the Enterprise Bean

The enterprise bean in our example is a stateless session bean called `ConverterBean`.

The source code for `ConverterBean` is in the *tut-install*`/examples/ejb/converter/src/java/` directory.

Creating `ConverterBean` requires these steps:

1. Coding the bean's implementation class (the source code is provided)

2. Compiling the source code

Coding the Enterprise Bean Class

The enterprise bean class for this example is called `ConverterBean`.

This class implements two business methods: `dollarToYen` and `yenToEuro`.

Because the enterprise bean class doesn't implement a business interface, the enterprise bean exposes a local, no-interface view.

The public methods in the enterprise bean class are available to clients that obtain a reference to `ConverterBean`.

The source code for the `ConverterBean` class is as follows:

```
package
com.sun.tutorial.javaee.ejb;
import java.math.BigDecimal;
import javax.ejb.*;
@Stateless
public class ConverterBean {
private BigDecimal yenRate =
new BigDecimal("83.0602");
private BigDecimal euroRate =
new BigDecimal("0.0093016");
```

```
public BigDecimal
dollarToYen(BigDecimal dollars) {
BigDecimal result =
dollars.multiply(yenRate);
return result.setScale
(2, BigDecimal.ROUND_UP);
}
public BigDecimal
yenToEuro(BigDecimal yen) {
BigDecimal result =
yen.multiply(euroRate);
```

```
return result.setScale
(2, BigDecimal.ROUND_UP);
}
}
```

Note the `@Stateless` annotation decorating the enterprise bean class.

This annotation lets the container know that `ConverterBean` is a stateless session bean.

Creating the converter Web Client

The web client is contained in the following servlet class:

tut-install/examples/ejb/converter/src/java/converter/web/ConverterServlet.java

A Java servlet is a web component that responds to HTTP requests.

The `ConverterServlet` class uses dependency injection to obtain a reference to `ConverterBean`.

The `javax.ejb.EJB` annotation is added to the declaration of the private member variable `converterBean`, which is of type `ConverterBean`.

`ConverterBean` exposes a local, no-interface view, so the enterprise bean implementation class is the variable type:

```
@WebServlet
public class ConverterServlet
extends HttpServlet {
@EJB
ConverterBean converterBean;
...
}
```

When the user enters an amount to be converted to yen and euro, the amount is retrieved from the request parameters; then the `ConverterBean.dollarToYen` and the `ConverterBean.yenToEuro` methods are called:

```
...
try {
String amount =
request.getParameter("amount");
```

```
if (amount != null &&
amount.length() > 0) {
// convert the amount
// to a BigDecimal from
// the request parameter
BigDecimal d =
new BigDecimal(amount);
// call the
// ConverterBean.dollarToYen()
// method to get the amount
// in Yen
```

```
BigDecimal yenAmount =
converter.dollarToYen(d);
// call the
// ConverterBean.yenToEuro()
// method to get the amount
// in Euros
BigDecimal euroAmount =
converter.yenToEuro(yenAmount);
```

```
. . .
}

. . .
}
```

The results are displayed to the user.

Building, Packaging, Deploying, and Running the `converter` Example

Now you are ready to compile the enterprise bean class (`ConverterBean.java`) and the servlet class (`ConverterServlet.java`) and to package the compiled classes into a WAR file.

To Build, Package, and Deploy the `converter` Example in NetBeans IDE

1. From the File menu, choose Open Project.

2. In the Open Project dialog, navigate to:

`tut-install/examples/ejb/`

3. Select the `converter` folder.

4. Select the Open as Main Project and Open Required Projects check boxes.

5. Click Open Project.

6. In the Projects tab, right-click the `converter` project and select Deploy.

A web browser window opens the URL `http://localhost:8080/converter`.

To Build, Package, and Deploy the `converter` Example Using Ant

1. In a terminal window, go to:

`tut-install/examples/ejb/converter/`

2. Type the following command:

```
ant all
```

This command calls the `default` task, which compiles the source files for the enterprise bean and the servlet, placing the class files in the `build` subdirectory (not the `src` directory) of the project.

The default task packages the project into a WAR module: `converter.war`.

For more information about the Ant tool, see Building the Examples.

Note - When compiling the code, the `ant` task includes the Java EE API JAR files in the classpath.

These JARs reside in the `modules` directory of your GlassFish Server installation.

If you plan to use other tools to compile the source code for Java EE components, make sure that the classpath includes the Java EE API JAR files.

To Run the `converter` Example

1. Open a web browser to the following URL:

```
http://localhost:8080/converter
```

The screen shown in Figure 23-1 appears.

Figure 23-1 The converter Web Client

2. Type `100` in the input field and click Submit.

A second page appears, showing the converted values.

Modifying the Java EE Application

The GlassFish Server supports iterative development.

Whenever you make a change to a Java EE application, you must redeploy the application.

To Modify a Class File

To modify a class file in an enterprise bean, you change the source code, recompile it, and redeploy the application.

For example, to update the exchange rate in the `dollarToYen` business method of the `ConverterBean` class, you would follow these steps.

To modify `ConverterServlet`, the procedure is the same.

1. Edit `ConverterBean.java` and save the file.

2. Recompile the source file.

- To recompile `ConverterBean.java` in NetBeans IDE, right-click the `converter` project and select Run.

  This recompiles the `ConverterBean.java` file, replaces the old class file in the build directory, and redeploys the application to GlassFish Server.

- Recompile `ConverterBean.java` using Ant:

a. In a terminal window, go to the *tut-install*`/examples/ejb/converter/` subdirectory.

b. Type the following command:

```
ant all
```

This command repackages, deploys, and runs the application.