Exercises for Chapter 4

Exercise 4.1: A Simple Entity Bean

This exercise demonstrates the basic functionality of an entity bean with CMP (container-managed persistence) 2.0. The Cabin EJB is mapped to a single table in the database. The following section outlines the steps needed to build an application archive composed of an application client, a web client, and the EJB itself. We will use the same *.EAR* file to deploy the application from Chapters 4 through 6.

Building the Application for Exercise 4.1

1. Download the archive *ex04-1.jar* from the Titan Books download site. Downloads may be available for the complete bundle of exercises.
2. Use *jar* with `xvf` flags set, or another archive utility, to extract the file to an "examples home" directory we'll refer to as *$EXAMPLES_HOME*; for example, *c:\EJBBook*.
3. Open a command prompt and `cd` to *$EXAMPLES_HOME\src\ex04_1*, which we'll refer to as *$EX04_1_HOME*.
4. At the command prompt type `Ant` or `Ant -buildfile build.xml`.

This process should create the build directory, *$EXAMPLES_HOME\build\ex04_1*, and compile the appropriate files.

For this exercise and the next one we will go through a tedious process of using the deployment tool to create the EJBs, client application, and web application, and then examine the deployment descriptors. *$EX04_1_HOME* also contains a pre-built *.ear* file that can be opened in the deployment tool.

Database Schema for the Cabin EJB

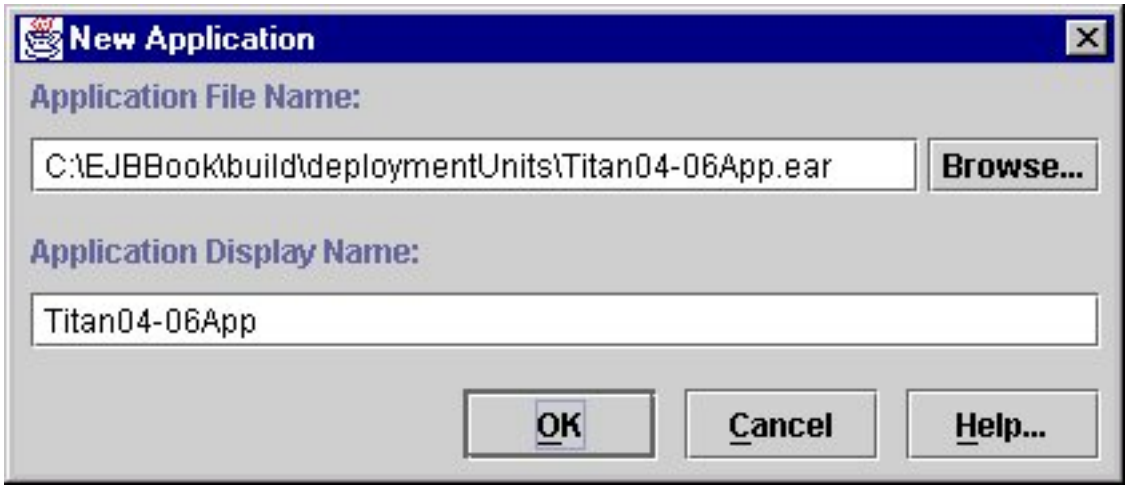
We will also create the required `Cabin` table on the fly, while deploying the EJB. Readers need to note the deviation from the EJB book with respect to the database schema. This is primarily because of the restrictions placed by the J2EE reference implementation CMP 2.0 container. The stubs and skeletons generated for the EJBs would not persist the data if not following these rules.

Note: J2EE SDK forces the CMP engine to use the database tables with the naming convention of <`EJBName`>Table. For example, `CabinBean` must map to `CabinBeanTable` in the database.

Building the Application Archive for Ex04_1

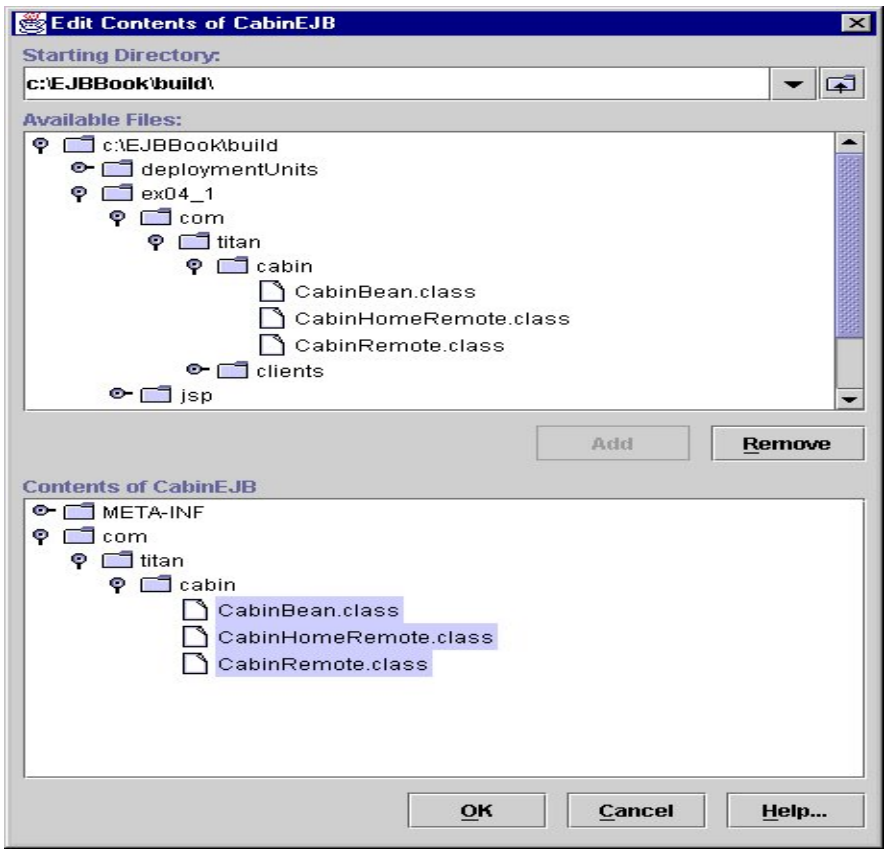
1. Run the deployment tool by typing at the command prompt `deploytool`.
2. Create a new application by selecting from the Start menu **File→New→Application.** and the following dialog will appear:

Figure 1: Creating an application archive



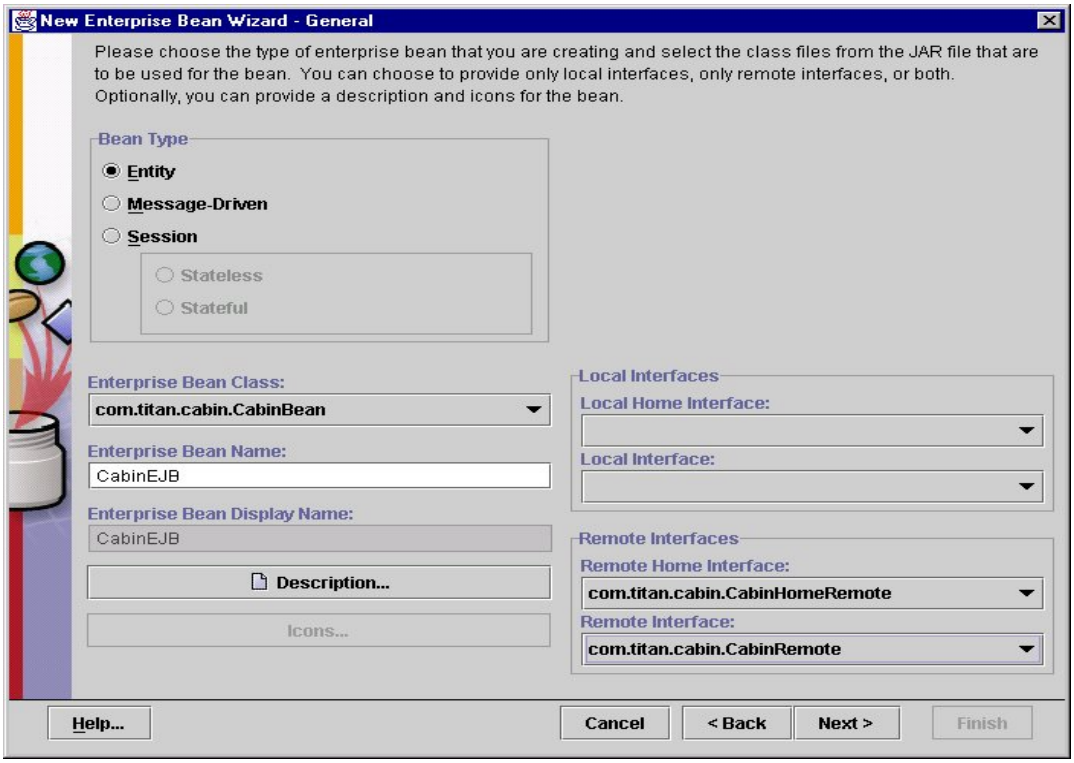
3. Save the application as *Titan04-06App.ear* in the directory *$EXAMPLES_HOME\build\deploymentUnits*. We will use this application archive for deploying examples from Chapters 4 through 6 of the EJB book.
4. Select the node that represents the *Titan04-06App*, then select from the menu **File→New→Enterprise Bean** to start a wizard that will walk through creation of the EJB.
5. Skip the introduction screen and go on to the next.
6. Enter **CabinEJB** in the **EJB Display Name** field, then in the contents pane click the **Edit** button to open another window, where you can select the EJB classes.

Figure 2: Cabin EJB packaging



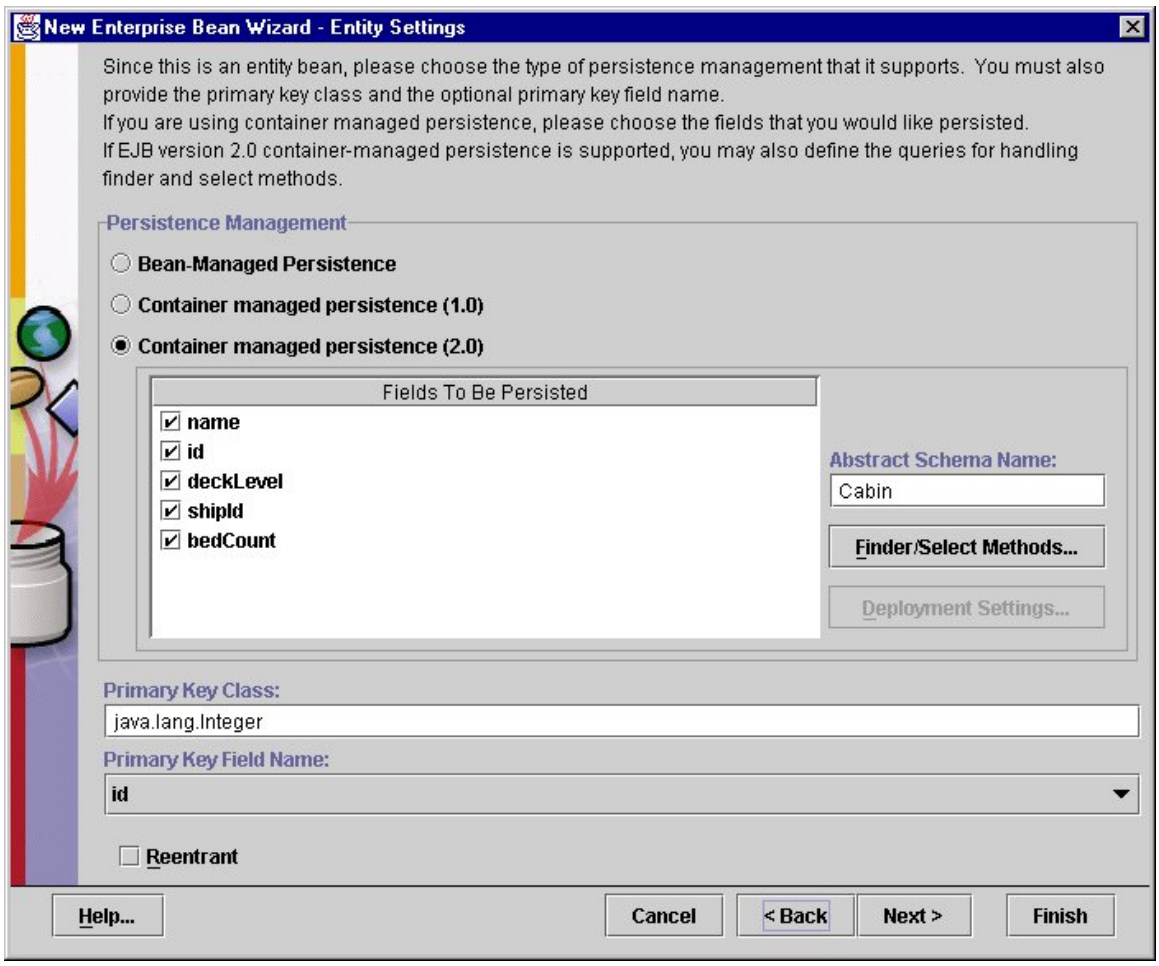
7. Select *CabinBean.class, CabinHomeRemote.class*, and *CabinRemote.class* from the *com.titan.cabin* package, then click **Add**, **OK**, and **Next**.

Figure 3: Cabin EJB configuration



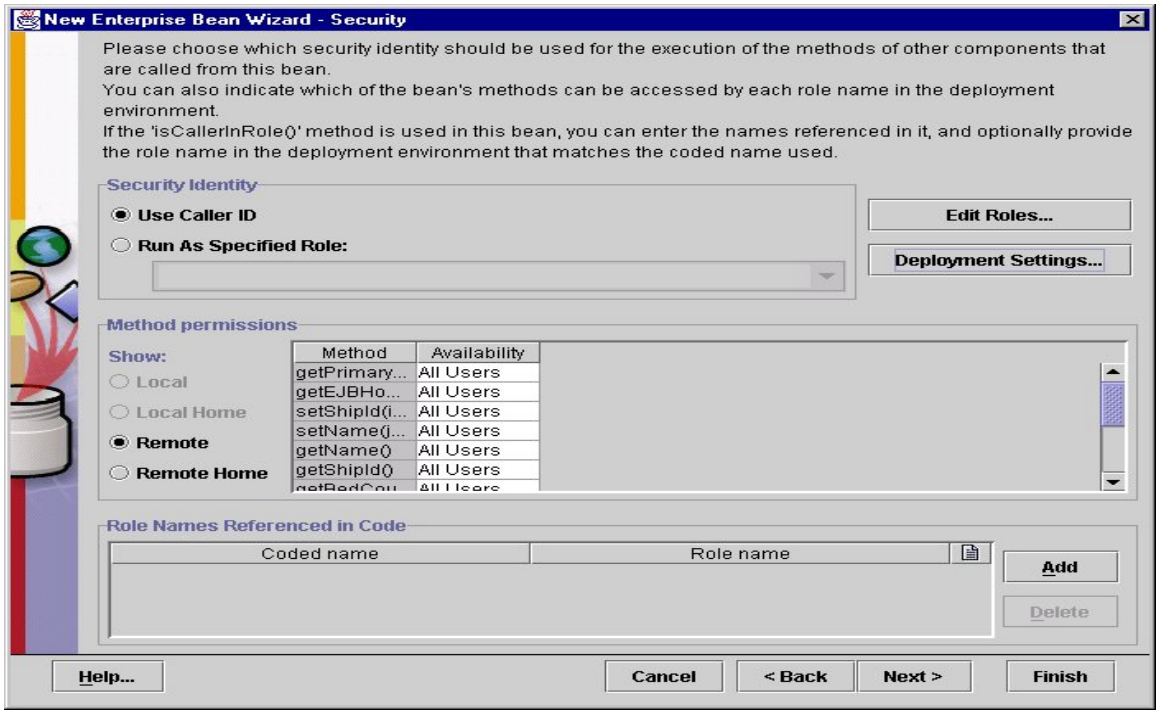
8. Set **Bean Type** to **Entity**. Select **Enterprise Bean Class** to be *com.titan.cabin.CabinBean*. For the **Enterprise Bean Name** enter **CabinEJB**.
9. Ignore the fields related to local interfaces; this example doesn't use them. From the **Remote Home Interface** drop-down list select **com.titan.cabin.CabinHomeRemote**. For **Remote Interface** select **com.titan.cabin.CabinRemote**. Click **Next**.

Figure 4: Cabin EJB O-R mapping



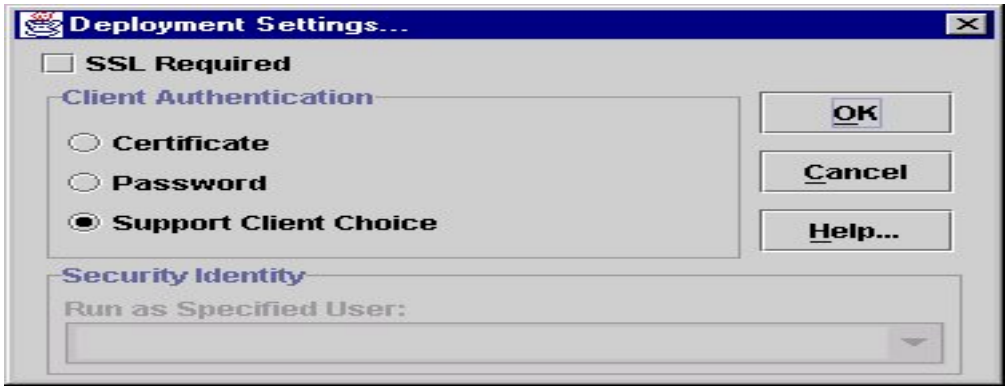
10. Select the **Container managed persistence (2.0)** radio button.
11. Select all the persistence fields: **name, id, deckLevel, shipId**, and **bedCount**.
12. In the **Abstract Schema Name** field enter **Cabin**.
13. For the **Primary Key** class enter **java.lang.Integer**, and for the **Primary key field name** select **id** from the drop down list. Click **Next**.
14. In the next screen, for **Transaction Management** select the **Container Managed** radio button. This example uses container-managed transactions.
15. Click **Next** till you reach the screen that shows the security requirements.

Figure 5: Caller Identity selection



16. For the **Security Identity** select **Use Caller ID**.
17. Click **Deployment Settings...** to bring up this small dialog:

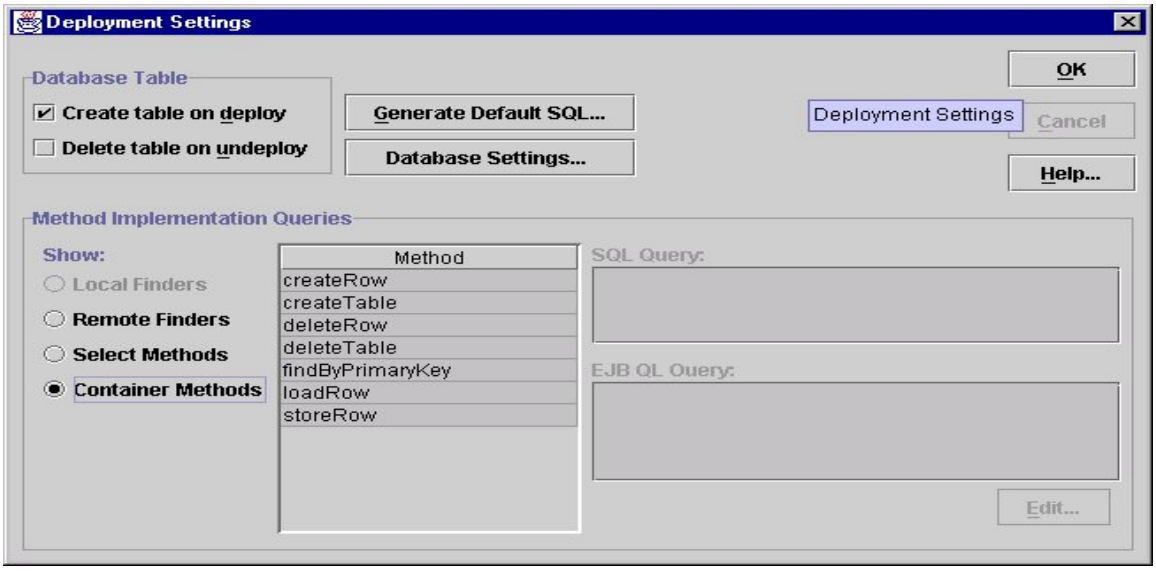
Figure 6: Caller Identity deployment settings



18. Make sure that **Support Client Choice** is selected and click OK.

19. Click **Next** and then **Finish** to create the Cabin enterprise bean in the application *Titan04-06App*.
20. Click the **Entity** tab in the deployment console for the Cabin EJB to configure the JDBC data source and the generation of SQL.
21. Click the **Deployment Settings** button to open the configuration window for the *CabinEJB*.

Figure 7: Cabin EJB entity settings



22. Select the **Create table on deploy** checkbox, then click the **Database Settings** button.

Figure 8: Database settings

23. Enter **jdbc/Titan** for the **Database JNDI Name**, **scott** for the **User Name** and **tiger** for the **Password** in the Database settings window. Click **OK**.

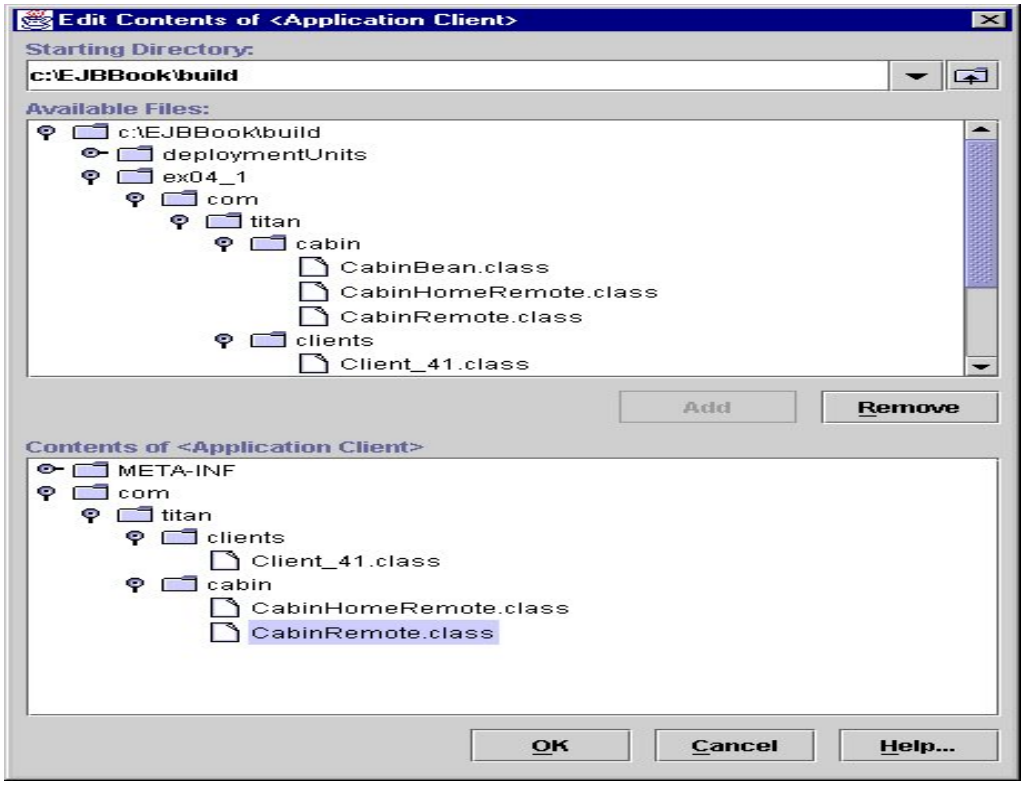
The EJB configuration should now be complete.

Creating an Application Client

An application client is a Java client that is bundled with remote stubs and EJB interfaces, so that it can connect to the EJB container and execute operations on the EJB. By definition, an application client is a single Java client program that can be executed.

1. Select **File→New→Application Client** from the main menu. Skip the next screen by clicking **Next**.
2. Click **Edit** to add the contents to the client *.jar*.

Figure 9: Client_41 configuration



3. Add three classes to the application client: *com.titan.clients Client_41.class, com.titan.cabin.CabinHomeRemote.class,* and *com.titan.cabin.CabinRemote.class*. Click the **Next** button.
4. Select **com.titan.clients.Client_41.class** as the application client's main class and enter **Client_41** in the **Display Name** field.

5. Make sure the call-back handler class is selected as **use container-managed transactions**.
6. Click the **Next** button and go to the EJB references screen. Click **Add**, and enter the following values in the fields:
   - **Coded Name:** **ejb/CabinHome**
   - **Type:** **Entity**
   - **Interfaces:** **Remote**
   - **Home Interface:** **com.titan.cabin.CabinHomeRemote**
   - **Local/Remote Interface:** **com.titan.cabin.CabinRemote**

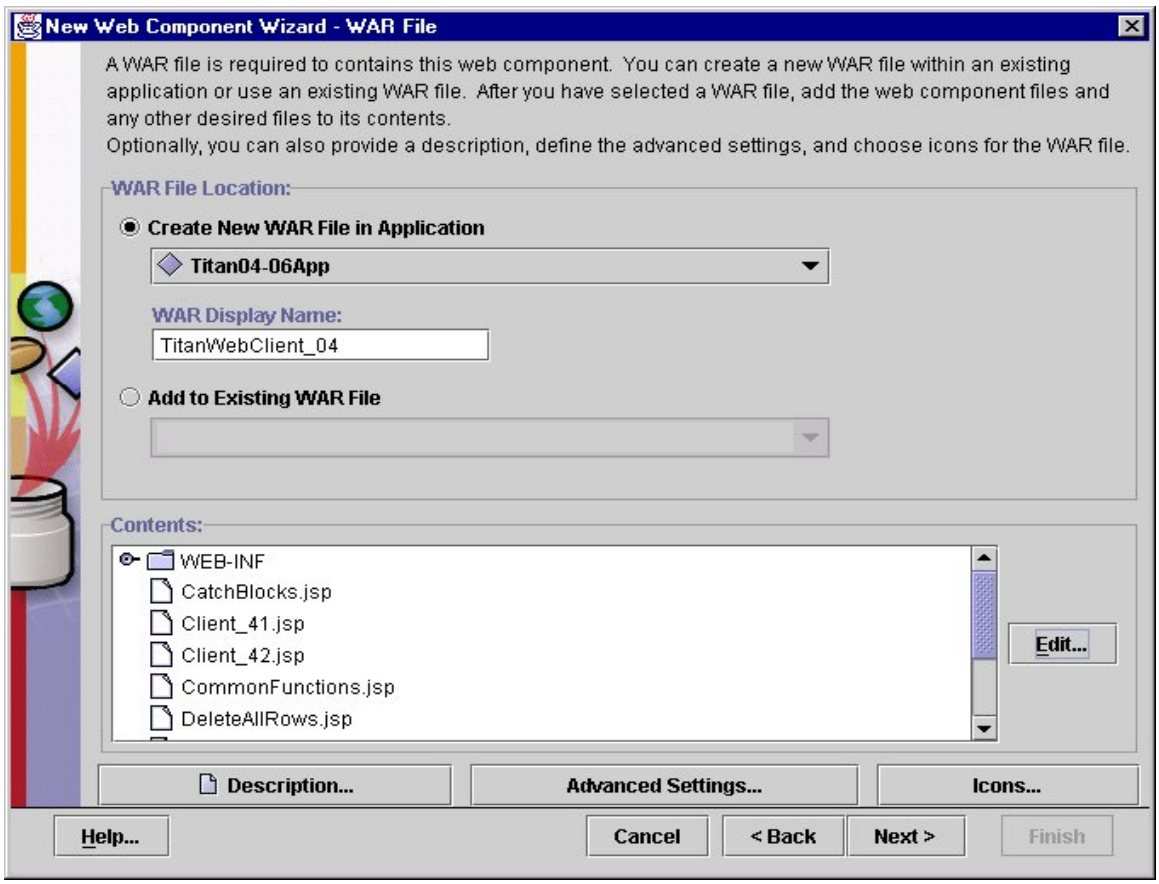
Coded Name is a reference in the JNDI from the root context. The root context for JNDI is defined as *java:comp/env*. To reference an EJB from a client, you can use either the JNDI name or a fully qualified JNDI reference.

7. Select the **JNDI Name** radio button and enter **CabinHome** as the JNDI name. For example, we could get a reference to the Cabin EJB by requesting a lookup on the Context object for *java:comp/env/ejb/CabinHome*, or by looking up the JNDI name *CabinHome*.
8. Click the **Next** button till the **Finish** screen appears. You should now have an application client for the *Titan04_06App* enterprise archive.
   - In Chapter 4 we are discussing the Java client and the JSP client. For other exercises we will provide different *.jar*s to be loaded that were pre-built using the above-mentioned steps.

Creating a Web Client

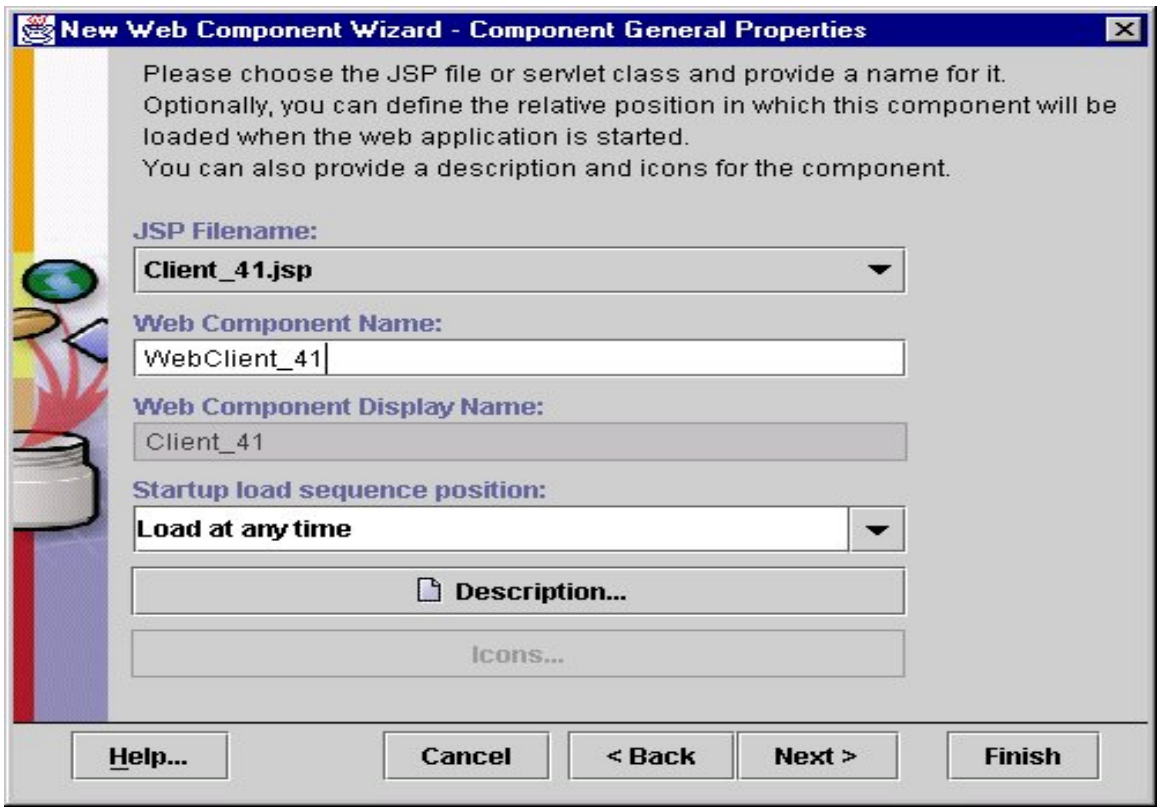
1. Select **File→New→WebComponent** to create a new web application (*.war*). Give it the name *TitanWebClient_04*, then click **Next**.
2. ?? to associate a client JSP file with a web component name. Add all the JSPs from the *$EXAMPLES_HOME\build\ex04_1\jsp* folder. This directory contains JSP clients for two client tests. Click **Next**.

Figure 10: Web client creation



3. Set the **Component Type** to **JSP**, then click **Next**.

Figure 11: Web component settings



4. From the **JSP Filename** drop-down list, select *Client_41.jsp*
5. For the **Web Component Name** enter *WebClient_41*, then click **Next** until you reach the Component Security screen.
6. Select **Use Caller ID** for security Identity, then click **Next** until you reach the screen for EJBs referenced in the code.
7. Click **Add**, then enter the same information for the JSP code as you did for the application client:
   - **Coded Name:** **ejb/CabinHome**
   - **Type:** **Entity**
   - **Interfaces:** **Remote**
   - **Home Interface:** **com.titan.cabin.CabinHomeRemote**
   - **Local/Remote Interface:** **com.titan.cabin.CabinRemote**

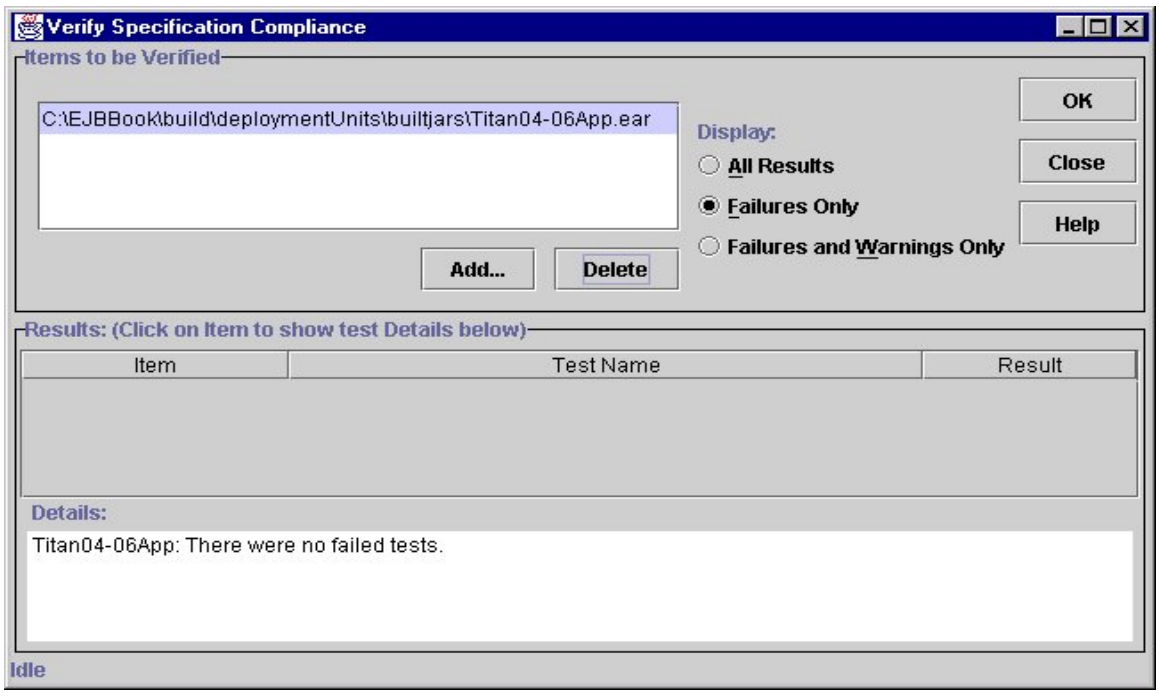
8. Specify the JNDI name to be **CabinHome**.
9. Click **Next** and finally click **Finish** to create the web archive.

Deploying the Enterprise Archive

Now it is time to deploy the *.ear* file. We begin by ensuring that everything is set up as it needs to be, and then proceed with deployment:

1. So that the directory we need is in place, begin by saving the enterprise archive as *Titan04_06App.ear*.
2. Make sure that SQL has been generated for the EJB:
   - Start the Cloudscape and J2EE servers.
   - Connect the deployment tool to the server.
   - Go to CabinBean's **Entity** tab.
   - Click **Deployment** settings.
   - Make sure that **Create Table on Deploy** is checked and **Delete Table on Undeploy** is unchecked
   - Click **Generate Default SQL**.
3. Run the verification tool to test for failures and other warnings. Open **Tools→Verifier,** select the application archive, and click **OK** to run the test.

Figure 12: Running the Verifier tool



4. Select the **Tools→Deploy** menu item to start the deployment process. Walk through the following screens, identifying the client *.jar* to be returned, and specifying the EJB's JNDI name to be `CabinHome` and the web context root to be *Titan1*.

Figure 13: Deployment process – screen 1

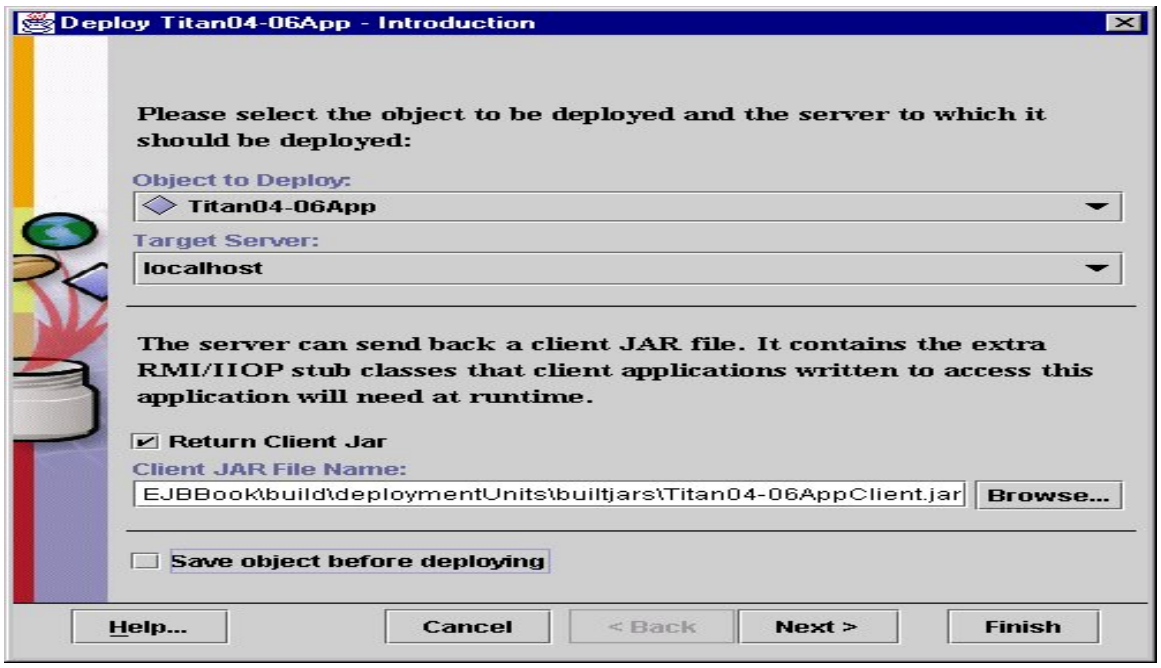


Figure 14: Deployment process – screen 2

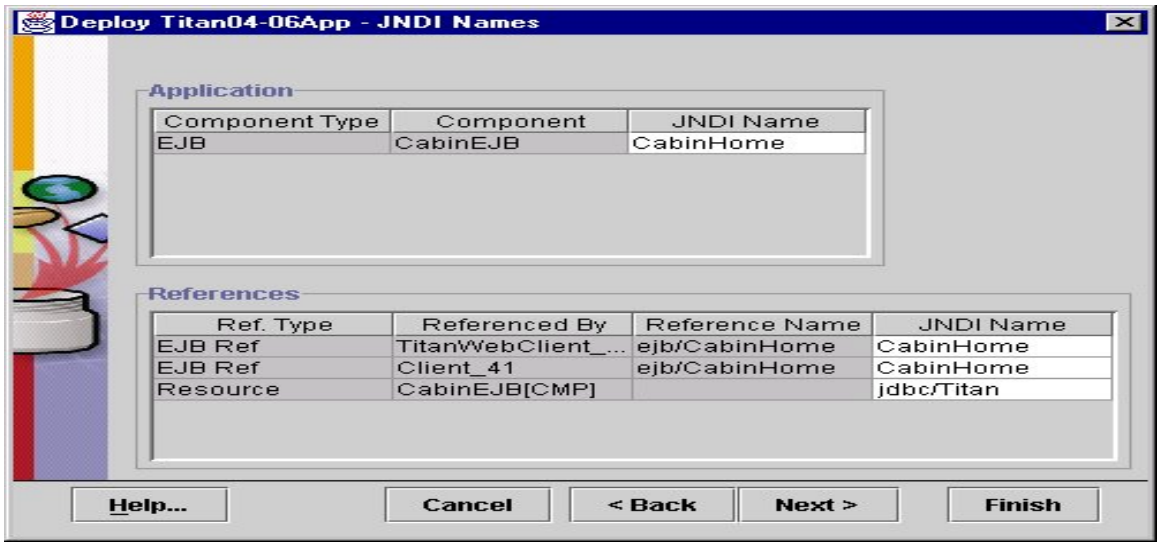


Figure 15: Deployment process – screen 3

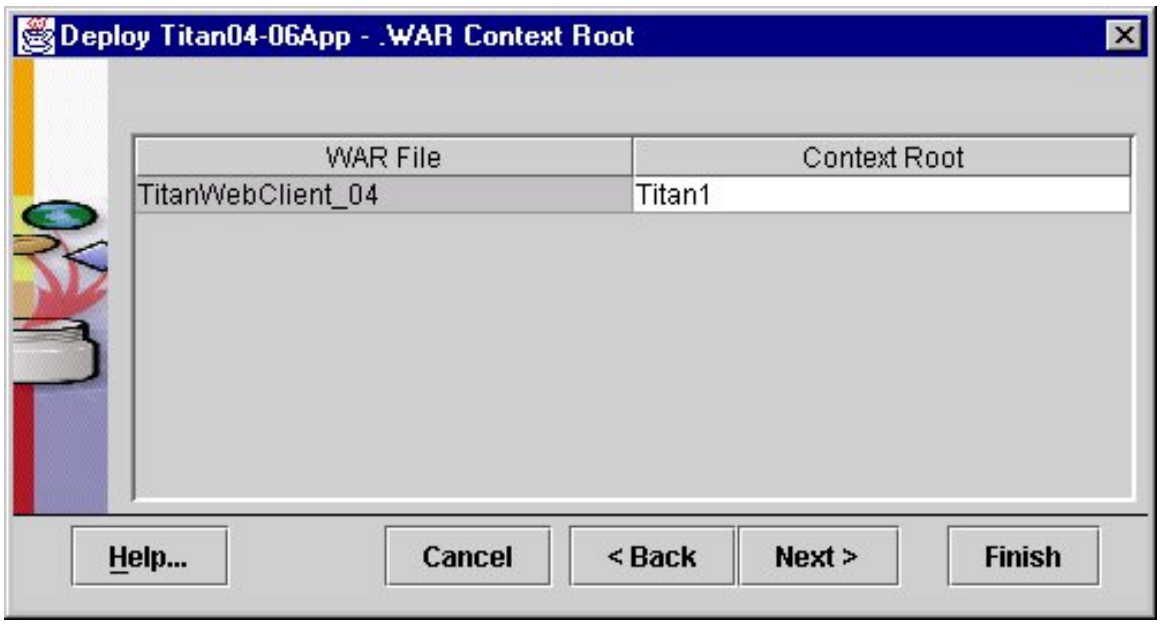
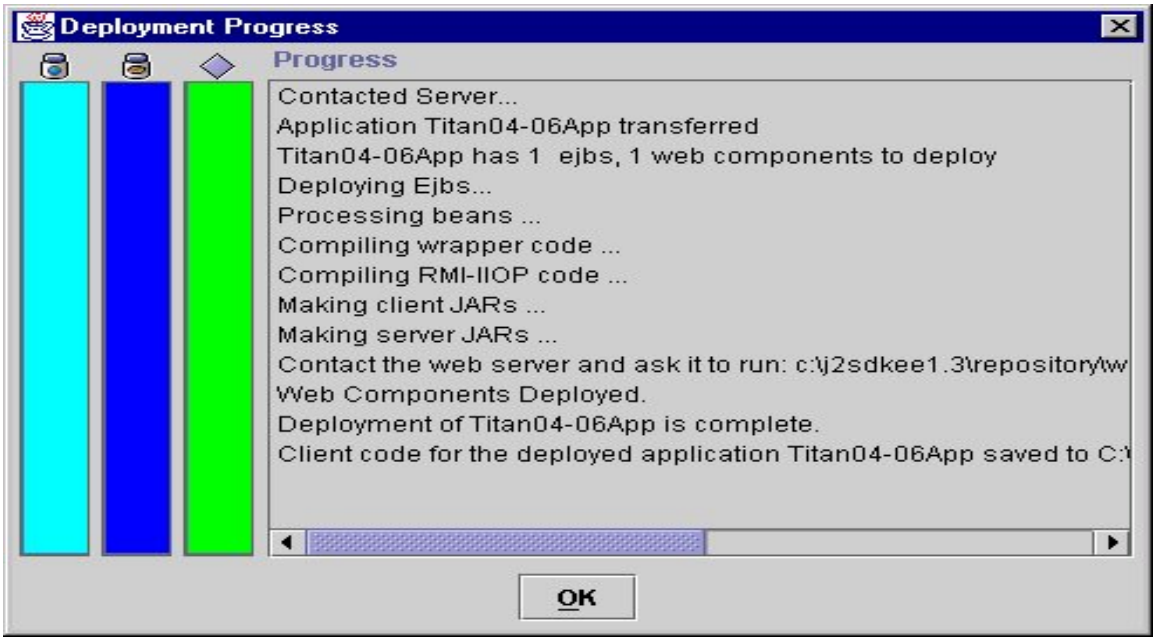


Figure 16: Deployment process – screen 4



You've now deployed a simple EJB with two client programs.

Updating the Application Archive with Client_42 Components

Before you run the client application, you need to add the *Client_42* components and redeploy the enterprise application. In the previous section we led you through an elaborate process of creating the Java client application as well as the web application. For your convenience, here we provide a pre-built *Client_42* application, and focus on the process of adding the *Client_42.jsp* to the web application.

Client_42 Application

1. Start the Cloudscape server: `cloudscape -start`
2. Start the J2EE server: `j2ee -verbose`
3. Start the deployment tool: `deploytool`
4. In the deployment tool, select **File → Open** and open the *ex04_06App.ear* that you saved before deployment in the preceding section.
5. Navigate **File→Open→Add to Application→Application Client jar** and select *Client_42.jar* from the pre-built clients directory you extracted from the downloaded application code. This step should add the *Client_42* application.
6. Click on the *Client_42* node and verify that the **EJB Ref** tab has entries for **Coded Name**, **Type**, **Interface**, **Home Interface**, and **Local/Remote Interface**.
7. Click on the *Titan04_06* WebClient node and go the **General** tab in the right pane.
8. Click **Edit**, go to the *build\ex04_1\jsp* folder, and add *Client_42.jsp* to the web application.
9. Make sure the Cabin EJB's deployment settings are correct. Go to the Cabin EJB **Entity** tab and click **Deployment Settings**. Make sure **Delete on Undeploy** is unchecked.
10. Click on the *Titan04_06App* node and select **Tools→Deploy** to deploy the application.
11. Following the deployment procedure you've already seen, deploy the *TitanAppClient_42.jar*, set the JNDI references if you haven't already, and set and the context root to *Titan1***.**
12. Deploy the complete enterprise application.

Examine and Run the Client Applications

Now we can test and run the Java client or the web client. This chapter has two client applications:

- *Client_41* creates a single Cabin bean and populates its attributes, then validates the existence of the Cabin bean by performing a lookup on its primary key.

- *Client_42* creates 100 Cabin beans and populates the table with the data. This data will be used in subsequent exercises through Chapter 6.

We will demonstrate the use of the Java version of the *Client_41* application and the web version of *Client_42*.

To execute the Java application client, change to *$EXAMPLES_HOME\build\deploymentunits* and execute the following command:

```
runclient -client Titan04_06App.ear -name Client_41
```

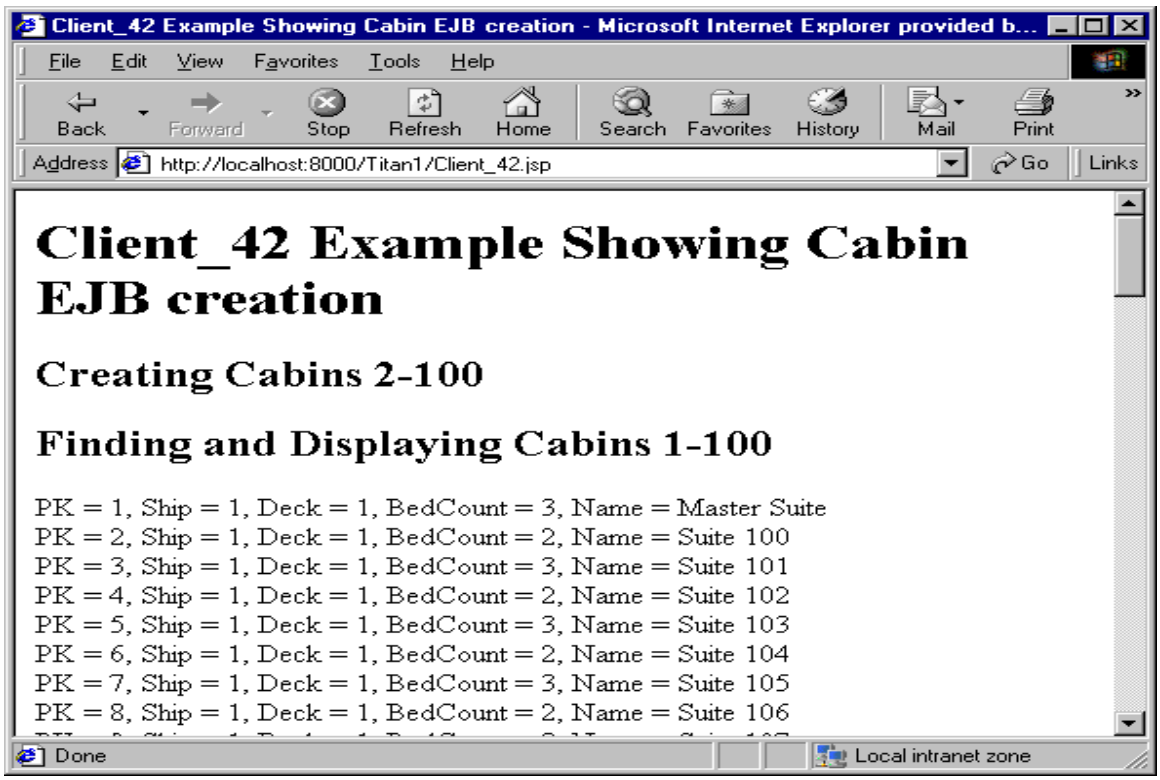
A login window will pop up. Enter the username as **scott** and the password as **tiger**. You should see the following output:

```
Initiating login ...
Username = null
Binding name:`java:comp/env/ejb/CabinHome`
 Found Cabin Home
Master Suite
1
1
3
Unbinding name:`java:comp/env/ejb/CabinHome`
```

Examine and Run the Web Client (JSP Pages)

1. Open a web browser and type in the URL *http://localhost:8000/titan1*. If the server is running on a different machine, specify its name rather than *localhost*: *http://<hostname>:8000/Titan1/*.
2. Click on *Client_42.jsp*. to run the web version of the *Client_42* example, which will display a window that shows the Cabin beans created.

Figure 17: Client_42 results



Congrats! You now know how to build, deploy, and test a simple client interface to an enterprise bean.

Exercise 4.2: A Simple Session Bean

This exercise uses a stateless session bean, `TravelAgentEJB`, to encapsulate the process logic and act as a façade for the `CabinEJB` entity bean we developed in the first exercise. Exercise 4.2 extends the usage described in Exercise 4.1 to demonstrate interaction between distributed components.

Building the Example Programs

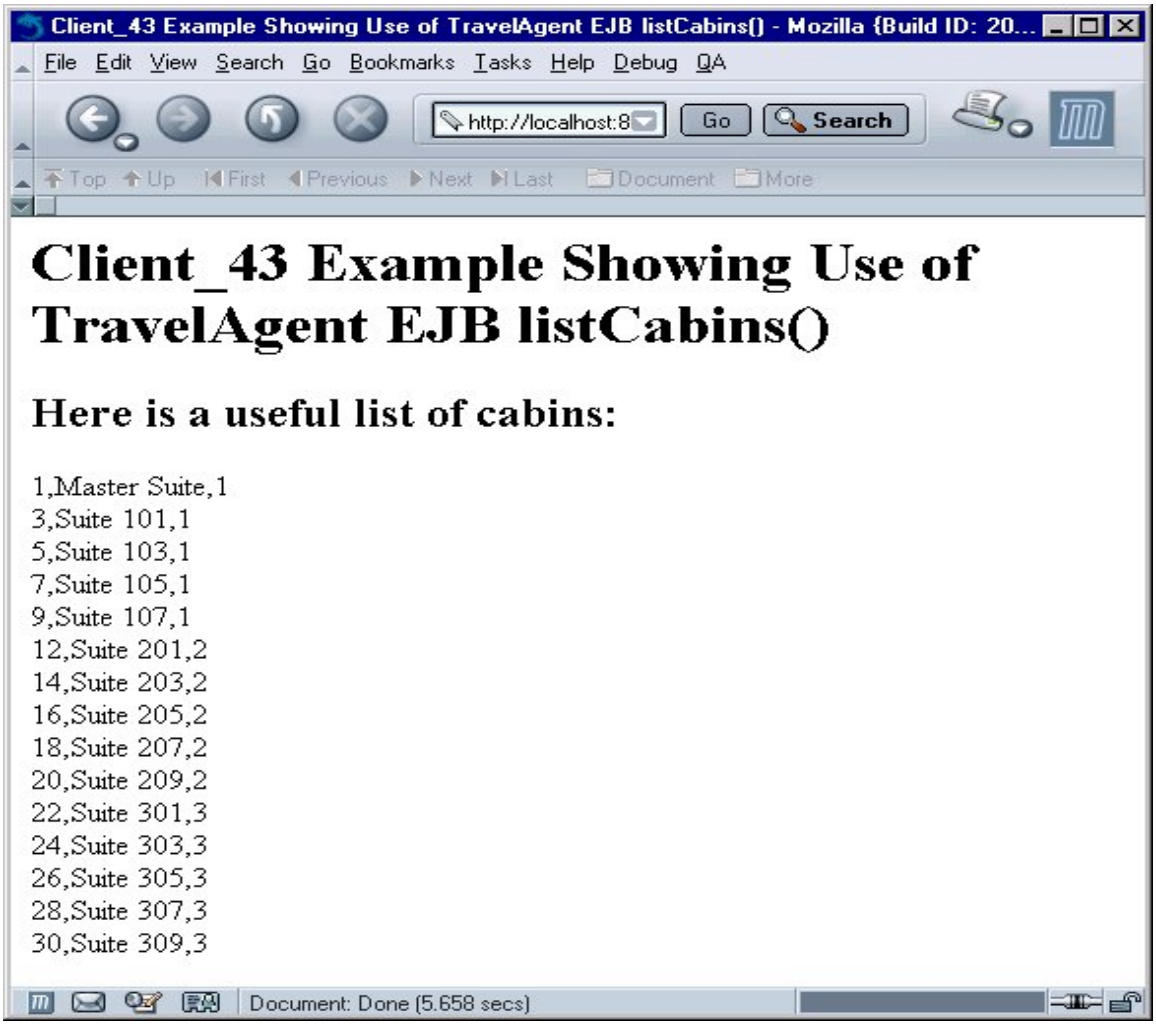
1. Open a command prompt and `cd` to *$EXAMPLES_HOME\src\ex04_2*.
2. Run *ant* to compile all the required classes, and place them in the directory *$EXAMPLES_HOME\build\ex04_2*.

If you want the practice, you can build the whole application yourself, following the detailed steps you've seen. For your convenience we have bundled *TravelAgentEJB.jar* in the directory *$EXAMPLES_HOME\pre-built*.

3. Start *Cloudscape*, the J2EE Server, and the deployment tool.
4. Open the *Titan04-06App.ear* we used in Ex04_1 exercise.
5. In the deployment tool select **File→Add to Application→EJB Jar** and select *TravelAgentEJB.jar*.
6. On the **General** tab, check the Tree view for the three TravelAgent EJB class files and the home and remote interfaces of the Cabin EJB. This example uses the TravelAgent bean as a session façade, hiding the details of the Cabin bean. *Client_43.jsp* accesses the TravelAgent EJB to retrieve requested Cabin EJBs.
7. `TravelAgentEJB` works as a client of `CabinEJB`, so in the **EJB references** tab of the `TravelAgentEJB` check for the EJB references of `CabinEJB`.
8. Make sure the entries are as you've seen them before:
   - **Coded Name:** **ejb/CabinHome**
   - **Type:** **Entity**
   - **Interfaces:** **Remote**
   - **Home Interface:** **com.titan.cabin.CabinHomeRemote**
   - **Local/Remote Interface:** **com.titan.cabin.CabinRemote**
9. Specify the JNDI name to be **CabinHome**.
10. Select **Tools→Deploy.** Using **Tools→Update and ReDeploy** might cause an error the verifier is unable to catch. This is basically "the JNDI name for TravelAgent EJB is not set" message which is requested by the deployment screens.

11. Open a browser and type in a URL in the pattern *http://<hostname>:8000/<webapp>*; for example, *http://localhost:8000/Titan1/*.
12. You will see a list of all the JSPs available in the webapp. Click *Client_43.jsp* and view the output, which should look like this:

Figure 18: Client_43 results



This trial run finishes up Chapter 4. Client programs are provided for Client_43 and is left as an exercise for the reader.