Exercises for Chapter 5

Exercise 5.1: The Remote Component Interfaces

Exercise 4 explained in great detail the process of using *deploytool* to package an enterprise application with EJBs, application clients, and JSPs. We didn't explain the details of the deployment descriptors of the components. In this exercise and the next we will use the sample application archive. Exercise 5.1 walks through a number of features of the home interface, including EJB metadata.

Building the Application for Exercise 5.1

1. Download the *ex05-1.jar* file from our 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 the *$EXAMPLES_HOME*. directory you set up earlier, for example, *c:\> EJBBook*.
3. Change directory to *$EXAMPLES_HOME\src\ex05_1*, which we'll refer to as *$EX05_1_HOME*.
4. Open a command prompt and type `Ant` or `Ant –buildfile build.xml`

This process should create the build directory, *$EXAMPLES_HOME\build\ex05_1*, and compile the required files.

The *$EXAMPLES_HOME\pre-built\ex05_1* directory contains pre-packaged client application *.jar*s for all the examples in this chapter.

Database Schema for the Cabin EJB

This exercise will use the `CabinBeanTable` created in Exercise 4.1.

Building the Application Archive for Ex05_1

1. Run the deployment tool by typing the command `deploytool.`
2. Open the application *Titan04-06App.ear* you created in the preceding exercise.
3. Select the **TitanWebClient_04** node in the application and navigate to the **General** tab.
4. Click the **Edit** button to edit the contents of the web application.
5. Select all the JSPs from *$EXAMPLES_HOME/build/ex05_1/jsp* directory and click the **Add** button.
6. Select **File→Add to Application→ApplicationClient Jar** from the menu. Select the *$EXAMPLES_HOME/pre-built/ex05_1/client_51.jar* and add it to the application.

7. Run the verifier to make sure that the application package and components are free of any errors.
8. Select **Tools→Deploy** to deploy the application instead of the **Tools→Update and Redeploy** option. **Update and Redeploy** doesn't give you the chance to check the JNDI names while you walk through the wizard. Also, there is a bug that appears when you update the files of a deployed application; check the release notes.
9. Make sure that the JNDI names for the EJBs are specified properly.

You have finished deploying the application for running the *Ex05_1* examples.

Examine and Run the Ex05_1 Client Applications.

This exercise includes three client applications. The EJB book refers to two of them, and we provide an extra utility that enables you to re-run the JSPs or the *Client_51* program:

- *Client_51.java* and *Client_51.jsp* demonstrate the use of `remove` on the Cabin EJB Home interface.
- *Client_52.java* and *Client_52.jsp* demonstrate the use of EJB metadata methods.
- *Client_51_Undo.java* and *Client_51_undo.jsp* enables you to re-create the Cabin bean deleted in the *Client_51* program.

Examine and Run the Client_51 Program

In the exercises for Chapter 4 we did not discuss any details of the client code that accesses the EJBs. It will be worth your while to walk through the *Client_51.java* program before we run the client and examine the interesting features:

- This program obtains a reference to the EJB by performing a `lookup` on the `EJBHome` object.
- It uses that object to create a remote object `EJBObject`.
- It executes the remote method `listCabins` to instantiate or find the Cabin EJBs. This call represents a stateless service call, using the TravelAgent EJB.
- It then prints the listing to the output stream before deleting the requested *CabinEJB*.

```
// Obtain a list of all the cabins for ship 1 with bed count of 3.
Object ref = jndiContext.lookup("TravelAgentHome");
TravelAgentHomeRemote agentHome = (TravelAgentHomeRemote)
PortableRemoteObject.narrow(ref,TravelAgentHomeRemote.class);

TravelAgentRemote agent = agentHome.create();
String list [] = agent.listCabins(1,3);
System.out.println("1st List: Before deleting cabin number 30");
for(int i = 0; i < list.length; i++)
{
   System.out.println(list[i]);
}
...
```

- The client program performs similar steps to obtain a reference to the Cabin entity bean by performing a JNDI lookup.
- It is important to know that the `create` and `remove` methods are executed on the EJBHome object. Typically, finders are also available through the `EJBHome` interface.
- Because you know the primary key value, the client program executes the `remove` method on the EJBHome object, passing the primary key object – in this case, the one whose value is `30`. This method deletes the cabin's information from `CabinBeanTable`.
- The client then performs another service call to the TravelAgent EJB, to get the updated listing of cabins so it can print their information.

```
...
// Obtain the home and remove cabin 30. Rerun the same cabin list.
ref = jndiContext.lookup("CabinHome");
CabinHomeRemote c_home = (CabinHomeRemote)
         PortableRemoteObject.narrow(ref, CabinHomeRemote.class);

Integer pk = new Integer(30);
c_home.remove(pk);
list = agent.listCabins(1,3);
System.out.println("2nd List: After deleting cabin number 30");
   for (int i = 0; i < list.length; i++)
   {
         System.out.println(list[i]);
   }
...
```

You will use the application client program to run this example. To run the *Client_51* Java program execute the *runclient* utility with appropriate arguments. A login window will pop up. Enter the same string for both the username and the password: **j2ee**.

```
runclient -client Titan04_06App.ear -name Client_51
```

The output should look like this:

```
Binding name:`java:comp/env/ejb/TravelAgentHome`
Binding name:`java:comp/env/ejb/CabinHome`
1st List: Before deleting cabin number 30
1,Master Suite,1
3,Suite 101,1
5,Suite 103,1
7,Suite 105,1
9,Suite 107,1
12,Suite 201,2
```

```
14,Suite 203,2
16,Suite 205,2
18,Suite 207,2
20,Suite 209,2
22,Suite 301,3
24,Suite 303,3
26,Suite 305,3
28,Suite 307,3
30,Suite 309,3
2nd List: After deleting cabin number 30
1,Master Suite,1
3,Suite 101,1
5,Suite 103,1
7,Suite 105,1
9,Suite 107,1
12,Suite 201,2
14,Suite 203,2
16,Suite 205,2
18,Suite 207,2
20,Suite 209,2
22,Suite 301,3
24,Suite 303,3
26,Suite 305,3
28,Suite 307,3
Unbinding name:`java:comp/env/ejb/TravelAgentHome`
Unbinding name:`java:comp/env/ejb/CabinHome`
```

Examine and Run the Client_52 Web Client

The client code for this application is very straightforward and we encourage you to glance through it.

Open the web browser and type in the URL *http://localhost:8000/Titan1*. If running on a different machine, specify its name rather than *localhost*: *http://<hostname>:8000/Titan1/*.

Click on the link to *Client_52.jsp* to run the *Client_52* example and display a window that shows its output:

```
Client_52 Example showing metadata functions
com.titan.cabin.CabinHomeRemote
com.titan.cabin.CabinRemote
java.lang.Integer
false
Master Suite
```

Examine and Run the Client_51_undo Web Client

If you try to run the application before running the *Client_51_undo* program, it may throw an object-not-found exception and generate report like this:

```
java.rmi.NoSuchObjectException: CORBA OBJECT_NOT_EXIST 9998 Maybe;
nested exception is:
        org.omg.CORBA.OBJECT_NOT_EXIST:   minor code: 9998
completed: Maybe
org.omg.CORBA.OBJECT_NOT_EXIST:   minor code: 9998 completed: Maybe


Unbinding name:`java:comp/env/ejb/TravelAgentHome`
Unbinding name:`java:comp/env/ejb/CabinHome`
```

Open the browser and go to *http://localhost:8000/Titan1/Client_51_undo.jsp*. The following output is typical:

```
Client_51_undo Re-create Cabin 30 to undo Client_51 change
1st List: Before re-creating cabin number 30
1,Master Suite,1
3,Suite 101,1
5,Suite 103,1
7,Suite 105,1
9,Suite 107,1
12,Suite 201,2
14,Suite 203,2
16,Suite 205,2
18,Suite 207,2
20,Suite 209,2
22,Suite 301,3
24,Suite 303,3
26,Suite 305,3
28,Suite 307,3
2nd List: After re-creating cabin number 30
1,Master Suite,1
3,Suite 101,1
5,Suite 103,1
7,Suite 105,1
9,Suite 107,1
12,Suite 201,2
14,Suite 203,2
16,Suite 205,2
18,Suite 207,2
20,Suite 209,2
22,Suite 301,3
24,Suite 303,3
26,Suite 305,3
28,Suite 307,3
30,Suite 309,3
```

Exercise 5.2: The EJBObject, Handle, and Primary Key

This exercise demonstrates additional capabilities of EJB components including serialization of handles.

Building the Example Programs

1. Open a command prompt and change directory to *$EJBBook\src\ex05_2*.
2. Run *Ant* to compile all the required classes.

The build process places the *.class* files in *$EJBBook\build\ex05_2*.

For your convenience we have bundled client *.jar* files in the *$EXAMPLES_HOME\pre-built\ex05_2* directory.

3. Start the Cloudscape server, the J2EE server, and *deploytool*.
4. Open the *Titan04-06App.ear* that you used in Exercise 5.1.
5. In the deployment tool click on **File→Add to Application→Application Client Jar** and select *Client_53.jar*.
6. Also add *Client_54.jar* and *Client_55.jar* files to create other application client programs.
7. Select the **TitanWebClient_04** node in the application and navigate to the **General** tab.
8. Click the **Edit** button to edit the contents of the web application.
9. Select all the JSPs from *$EXAMPLES_HOME/build/ex05_2/jsp* directory and click the **Add** button.
10. Select **Tools→Deploy** to deploy the application. Select the **return client Jar** check-box in the wizard to obtain the client-jar files.

    The serialization example will need the class-path settings for the home stubs and remote stubs for the EJBs. Make sure the class-path settings for this program are set up for these operations. There are many ways these can be set up. Running the client programs discusses an approach to making corresponding settings.

Examine the Client Applications for Exercise 5.2

Let's examine the client application code for Exercise 5.2

Client_53.java

This program demonstrates some interesting metadata information about EJB and also demonstrates mechanisms for obtaining EJB reference and EJBHome object via EJBObject reference.

```
...
Context jndiContext = getInitialContext();
Object ref = jndiContext.lookup("TravelAgentHome");
TravelAgentHomeRemote home = (TravelAgentHomeRemote)
    PortableRemoteObject.narrow(ref,TravelAgentHomeRemote.class);

// Get a remote reference to the bean (EJB object).
TravelAgentRemote agent = home.create();
// Pass the remote reference to some method.
getTheEJBHome(agent);
...
```

The `getTheEJBHome` method is a convenience provided so you can re-acquire a reference to the `EJBHome` object and display some metadata.

```
public static void getTheEJBHome(TravelAgentRemote agent)
         throws RemoteException
{
   // The home interface is out of scope in this method,
   // so it must be obtained from the EJB object.
   Object ref = agent.getEJBHome();
   TravelAgentHomeRemote home = (TravelAgentHomeRemote)

   PortableRemoteObject.narrow(ref,TravelAgentHomeRemote.class);

   // Print metadata information about the EJB
   EJBMetaData meta = home.getEJBMetaData();
   System.out.println(meta.getHomeInterfaceClass().getName());

   System.out.println(meta.getRemoteInterfaceClass().getName());
         System.out.println(meta.isSession());
}
```

Client_54.java

This application demonstrates the usefulness of the primary key for an entity bean. It also demonstrates the equivalence of two entity beans with respect to references and value. Based on the identity of the entity bean with uniqueness established through the primary key you prove this in the example.

```
System.out.println("Creating Cabin 101 and retrieving additional
reference by pk");
CabinRemote cabin_1 = home.create(new Integer(101));
Integer pk = (Integer)cabin_1.getPrimaryKey();
CabinRemote cabin_2 = home.findByPrimaryKey(pk);

System.out.println("Testing reference equivalence");
// We now have two remote references to the same bean -- Prove it!
```

```
cabin_1.setName("Keel Korner");
if (cabin_2.getName().equals("Keel Korner"))
{
   System.out.println("Names match!");
}

// Test the isIdentical() function
if (cabin_1.isIdentical(cabin_2))
{
   System.out.println("cabin_1.isIdentical(cabin_2) returns
   true - This is correct");
}
else
{
   System.out.println("cabin_1.isIdentical(cabin_2) returns false -
   This is wrong!");
}
...
```

The `testSerialization` method shows how primary key objects are the serialized and deserialized. This process includes serializing the primary key class to a file system, retrieving the sample entity bean later using the deserialized version of the key, then removing the entity bean.

Client_55.java

This program demonstrates the serialization and deserialization of the `HomeHandle` and `Handle` objects. Refer to the code for a detailed description.

Run the Client Programs for Exercise 5.2

Before you can run the example you need to make some setup changes to facilitate the serialization of the handles for the EJB. For example, to run the *Client_54* and *Client_55* programs you need to add path settings to the *setenv.bat* file in *$J2EE_HOME/bin* directory, and copy some existing classes to a more convenient directory.

1. Close *deploytool*.
2. Shut down the J2EE server.
3. Add the following line to the *setenv.bat* file:

```
set CLIENT_APP_PATH=
    %J2EE_HOME%\repository\%COMPUTERNAME%\web\Titan1\WEB-INF\classes
```

4. Copy four class files – *CabinRemote.class, CabinHomeRemote.class*, *_CabinRemote_Stub.class*, and *_CabinHomeRemoteStub.class* – from the directory *$J2EE_HOME\repository\ <machinename> \gnrtrTMP\Titan04-06App\com\titan* to

$J2EE_HOME\repository\ <machinename> %COMPUTERNAME%\web\Titan1\WEB-INF\classes, with the package structure preserved for *com/titan/cabinbean.*

5. Similarly copy the stub classes and the interface classes for the TravelAgent EJB to the corresponding location.
6. Open the *catalina.policy* file in the *$J2EE_HOME/conf* directory and add the permission to access these files.

```
grant codeBase
    "file:${catalina.home}/webapps/Titan1/WEB-INF/classes/-" { };
```

You've completed the setup and are ready to run the example:

1. Open a browser and type in a URL in the pattern *http://<hostname>:8000/<webapp>*; for example, *http://localhost:8000/Titan1/.*
2. From the list of JSPs available in the web application, click *Client_53.jsp* and view the results:

```
Client_53 Example of using EJBObject to retrieve home interface
com.titan.travelagent.TravelAgentHomeRemote
com.titan.travelagent.TravelAgentRemote
true
```

3. Change to the *$EXAMPLES_HOME\build* directory and run the client program *Client_53.java* with the command:

```
runclient -client Titan04_06App.ear -name Client_53
```

The output should look like this:

```
Initiating login ...
Username = null
Binding name:`java:comp/env/ejb/CabinHome`
Binding name:`java:comp/env/ejb/TravelAgentHome`
com.titan.travelagent.TravelAgentHomeRemote
com.titan.travelagent.TravelAgentRemote
true
```

4. Click on the page *http://localhost:8000/Titan1/Client_54.jsp* to see the following:

```
Client_54 Example showing use of primary keys
Creating Cabin 101 and retrieving additional reference by pk
Testing reference equivalence
Names match!
cabin_1.isIdentical(cabin_2) returns true - This is correct
Testing serialization of primary key
Setting cabin name to Presidential Suite
Writing primary key object to file...
Reading primary key object from file...
Acquiring reference using deserialized primary key...
Retrieving name of Cabin using new remote reference...
```

```
Presidential Suite
Removing Cabin from database to clean up..
```

5. Recall that *Client_55* demonstrates the serialization and deserialization of the *HomeHandle* objects. Run both versions, the web application by clicking on the page *http://localhost:8000/Titan1/Client_55.jsp*, and the Java client by executing the following command:

   ```
   runclient -client Titan04_06App.ear -name Client_55
   ```

   The output of the two versions will be similar; the Java client's should look like this:

   ```
   Initiating login ...
   Username = null
   Binding name:`java:comp/env/ejb/CabinHome`
   Binding name:`java:comp/env/ejb/TravelAgentHome`
   Testing serialization of EJBObject handle
   Writing handle to file...
   Reading handle from file...
   Acquiring reference using deserialized handle...
   cabin_1.isIdentical(cabin_2) returns true - This is correct
   Testing serialization of Home handle
   Writing Home handle to file...
   Reading Home handle from file...
   Acquiring reference using deserialized Home handle...
   Acquiring reference to bean using new Home interface...
   cabin_1.isIdentical(cabin_3) returns true - This is correct
   Unbinding name:`java:comp/env/ejb/CabinHome`
   Unbinding name:`java:comp/env/ejb/TravelAgentHome`
   ```

Exercise 5.3: The Local Component Interfaces

This exercise demonstrates the use of local interfaces. The TravelAgent session bean will use them to access the Cabin entity bean. You will build a new application archive for this exercise. Before you proceed with this exercise, undeploy the Titan04-06App from the environment.

Building the Example Programs

1. Obtain the examples *.jar* for *ex05_3* from the download site.
2. Open a command prompt and `cd` to *$EXAMPLES_HOME\src\ex05_3*.
3. Run *Ant* to compile all the required classes.

The build process places the *.class* files in the *$EJBBook\build\ex05_2* directory. You can build the beans yourself, as in Chapter 4, or use the pre-built EJBs. For convenience we have bundled client, EJB, and web archives in *$EXAMPLES_HOME\pre-built\ex05_3*.

4. Start the Cloudscape server, the J2EE server, and the deployment tool.
5. Create a new application *Titan05_3App.ear* and add *Titan05_3EJBs.jar* to it.
6. If you are using the pre-compiled versions of the Cabin and TravelAgent beans, make sure that the local interfaces are selected for `CabinEJB` and remote interfaces for `TravelAgentEJB`.
7. Make sure that the TravelAgent bean's references to the Cabin EJB use its local interfaces.

According to the EJB 2.0 specification, the `<ejb-link>` tag is optional, but the deployment tool generates the tag for the EJBs. This discrepancy can cause verifier problems and problems with deployment.

To get a clean result from the verification process, the enterprise bean name for the referenced Cabin EJB should be set to its `<ejb-name>`. This example sets it to `CabinBean`.

8. In the tree view select the **CabinEJB** node, and in the **General** tab check the given name of **Enterprise Bean Name**.
9. In the tree view, select the **TravelAgentEJB** node and click on the **EJB Refs** tab.
10. Select the EJB reference row for **CabinEJB**, and in the **Enterprise Bean Name** field enter the name for the Cabin EJB. **JNDI Name** will be disabled for local access to the EJB.
11. Add to **Titan05_3App** the application client *Client_58.jar*.
12. Then add to it the web application *WebClient_58.war*.
13. Make sure that the deployment settings for the Cabin EJB are intact with the database settings information and that SQL generation is completed.
14. Uncheck the **Delete table on undeploy** option.

15. Run the application through the verifier and deploy the application with the **Return client Jar** option checked.
16. Select the context root for the web application as *Titan2*.
17. Open a browser and locate *http://localhost:8000/Titan2/* to get a listing of all the contents of the folder.
18. Click on the link *Client_58.jsp*

```
Client_58 Example Showing Use of TravelAgent EJB listCabins()
Here is a useful list of cabins:
1,Master Suite,1
3,Suite 101,1
5,Suite 103,1
7,Suite 105,1
9,Suite 107,1
12,Suite 201,2
14,Suite 203,2
16,Suite 205,2
18,Suite 207,2
20,Suite 209,2
22,Suite 301,3
24,Suite 303,3
26,Suite 305,3
28,Suite 307,3
```