Demonstration

This demonstration shows how to represent the `DEPARTMENTS` table of `hr` schema using a container-managed persistent (CMP) entity bean.
Note: Start SQL*Plus, login with database account details and issue the command
`desc DEPARTMENTS`;
This is to show the students the various fields and the datatype of each field. Do not start SQL*Plus if you want to show the same with JDeveloper.
The complete workspace for this demo is under demos/lesson14.

Follow these steps to create a CMP entity bean using Oracle JDeveloper 10*g*.

1. Create a workspace with the name lesson14.
2. Create a project and name it departmentsdemo.
3. Right-click the project in the System-Navigator and select New..
4. Choose Enterprise JavaBeans from the Business tier of Categories
5. Select 'Entity Beans from Tables—Container-Managed Persistence' from Items. Click OK.
   Note: Please let the students know that they can also create an entity bean by selecting 'Entity Bean' from Items.
6. The wizard is displayed. Click the Next button. Choose Enterprise JavaBeans 2.0 from the drop down list of EJB Version. Click the Next button.
7. Database connection details will be displayed. Recollect that you have already created a connection by name hr, pointing to your database. The hr connection details will be displayed. If there is no connection existing, click on New to create a new connection.
8. If a connection already exists, click on Next. Wait till the query on database objects is complete.
9. Select DEPARTMENTS table. Move it to the Selected items and click the Next button.
10. We will be generating a remote interface for this CMP, therefore check the checkbox 'Generate Remote Interfaces' and uncheck the 'Generate Local Interfaces' checkbox. Click the Next button.
11. You can accept the name provided for the Entity bean, else you can change it in the field Entity Name. We will accept the default name Departments for this demo.
12. Click the Finish button.
13. Observe that the Departments bean, the ejb-jar.xml file, and the orion-ejb-jar.xml file are created. They are displayed in the System-Navigator.
14. Open the Departments.java file in the code editor and show the students that the set and get methods for each of the field in the DEPARTMENTS table is created.
15. Open the DepartmentsHome.java file in the code editor. Observe the two create methods and two finder methods that are created. JDeveloper picks the not null fields of the table to generate a create method with arguments. Another no argument create() method is always provided.
16. Open the DepartmentsBean.java file in the code editor and check the

implementation for create(), set, and get methods.

17. Observe that the set and get methods are abstract methods in the bean, without any implementation. The container implements these methods at run time.
18. Observe that the bean class does not contain any instance variables corresponding to the field of the table.
19. Double-click on the ejb-jar.xml file. Show the students that this file contains information about the various components (remote interface, home interface, primary-key class, and so on). Let the students know that there is a <cmp-field> tag for each of the field in the database table. For example,

```
<cmp-field>
       <field-name>departmentId</field-name>
</cmp-field>
```

20. Open the orion-ejb-jar.xml file and observe that the Datasource information, table information is provided here. Also show the mapping of each of the fields in the database table.
21. In the System-Navigator, right-click on the Departments bean and select New Sample Java Client. Check 'Connect to OC4J Embedded in JDeveloper' radio button and click the OK button.
22. Edit the DepartmentClient.java file. Comment some of the code in this file to make the demo simpler.
23. Comment the following lines in the client program to keep the demo simple.

```
/* Collection coll = departmentsHome.findAll();
Iterator iter = coll.iterator();
while (iter.hasNext())
{
departments = (Departments)iter.next();
System.out.println("department_id = " +
departments.getDepartmentId());
System.out.println("department_name = " +
departments.getDepartmentName());
System.out.println("manager_id = " +
departments.getManagerId());
System.out.println("location_id = " +
departments.getLocationId());
System.out.println();
} */
```

24. Add the following lines to create a department with department_id 404 and department_name security, and to invoke various methods.

System.out.println(" CREATING A NEW DEPARTMENT WITH ID 404...") ;
departments =departmentsHome.create(new Long(404),"Security");

```
System.out.println(" SUCCESSFUL ");
System.out.println("Getting the DEPARTMENT_NAME " +
departments.getDepartmentName());
System.out.println("Changing the DEPARTMENT_NAME to
Security Services " );
departments.setDepartmentName("Security Services");
System.out.println("PRINTING THE DEPARTMENT_ID AND
DEPARTMENT_NAME");
System.out.println(departments.getDepartmentId( ) + " " +
departments.getDepartmentName( ));
```

Compile and the run the Entity Bean by following these steps

1. Right-click on the Departments bean and select Make.
2. Right-click on the DepartmentClient.java and select Make.
3. Right-click on Departments bean and select Run.
4. Right-click on DepartmentsClient.java and select Run
   The following output will be displayed in the log window.
   ```
   CREATING A NEW DEPARTMENT WITH ID 404...
   SUCCESSFUL
   Getting the DEPARTMENT_NAME Security
   Changing the DEPARTMENT_NAME to Security Services
   PRINTING THE DEPARTMENT_ID AND DEPARTMENT_NAME
   404 Security Services
   Process exited with exit code 0.
   ```
5. You can issue commands in SQL*Plus to observe that 1 row is added to the departments table.
   ```
   SQL>SELECT count(*) FROM DEPARTMENTS;
   28
   SQL>SELECT department_id, department_name FROM DEAPARTMENTS
   WHERE department_id=404;
   ```

```
DEPARTMENT_ID DEPARTMENT_NAME
------------------------------------------------------------------
404 Security Services
```