Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans

• Based on the lab2, the lab3 implements an online symposium registration application “RegisterApp” which consists of a front session bean and two CMP supported by database tables.

• It can allow a new attendee to register one or more specific conferences online. The application calculates discount registration fee according to the attendee’s category and reports the conference registration confirmation with total fees for the attendee after a successful registration.

• This RegisterApp application has four components. The RegisterWar component is a J2EE Web application client that accesses the RegisterBean session bean through the bean’s remote interface.
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

- The *RegisterBean* talks to two entity beans — *AttendeeBean* and *ConferenceBean* — through their local interfaces since all beans are locally deployed at same server.
- The entity beans provide the logic for accessing the database by using container-managed persistence and relationship.
- The *AttendeeBean* and *ConferenceBean* entity beans have a many-to-many relationship.
- One conference may have many participants and one participant may join more than one conference.
- Each CMP bean has a relationship field whose value identifies the related bean instance.
- The diagram below shows the components and their relationships of the *RegisterApp* application.
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

• Step 1. Develop EJB Beans

1. Modify conference entity bean

   – The coding for conference entity bean component in this Lab is similar to the coding in the Lab2. But in the RegisterApp application a conference has many attendees, so a conference bean instance may be related to many attendee instances.

   – To specify this relationship the deployment descriptor of ConferenceBean defines a relationship field named attendees, a collection that represents the attendees of the conference.
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

– The ConferenceBean.java need to be modified to add two abstract access methods for relationship field.

– The code of the access methods for the attendees relationship field is as follows.

– Public abstract Collection getAttendees();

– Public abstract void setAttendees (Collection attendees);
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

2. Coding attendee entity bean

   Attendee entity bean component has following files:
   – Local interface: `LocalAttendee.java`
   – Local home interface: `LocalAttendeeHome.java`
   – Entity bean class: `AttendeeBean.java`

   In the attendee entity bean there are two business methods, `addConference()` and `removeConference()`, which invoke the `getConference()` access method to fetch the collection of related `LocalConference` object, and then invoke the add (or remove) method of the collection interface to add (or remove) the member from the collection.
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

//LocalAttendee.java is a Local interface for attendee
//CMP entity bean,
//It can be a remote interface if its client is remote.

package data;
import java.util.*;
import javax.ejb.*;
public interface LocalAttendee extends EJBLocalObject {
    public String getSsn();
    public String getName();
    public String getAttendeeType();
    public Collection getConferences();
    public void addConference(LocalConference conference);
    public void removeConference(LocalConference conference);
}
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

// LocalAttendeeHome.java is a Home interface of attendee CMP entity bean. It can be defined as a remote Home interface if it is accessed by a remote client.
package data;
import java.util.*;
import javax.ejb.*;
public interface LocalAttendeeHome extends EJBLocalHome {

    public LocalAttendee create (String ssn, String name, String attendeetype) throws CreateException;

    public LocalAttendee findByPrimaryKey (String ssn) throws FinderException;

}
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

//AttendeeBean.java CMP entity bean class implementation
package data;
import java.util.*;
import javax.ejb.*;
import javax.naming.*;
public abstract class AttendeeBean implements EntityBean
{
    private EntityContext context;
    // Access methods for persistent fields, implicit
    // definition of class members
    public abstract String getSsn();
    public abstract void setSsn(String ssn);
    public abstract String getName();
    public abstract void setName(String firstname);
public abstract String getAttendeeType();
    public abstract void setAttendeeType (String attendeeType);

    // Access methods for relationship fields, many to many
    // connection from Attendee to Conference entity bean
    public abstract Collection getConferences();
    public abstract void setConferences(Collection conferences);

    // Select methods, Business methods, EntityBean methods
    public void addConference(LocalConference conference) {
        try {
            Collection conferences = getConferences();
            conferences.add(conference);
        } catch (Exception ex) {
            throw new EJBException(ex.getMessage());
        }
    }
}
public void removeConference(LocalConference conference) {
    try {
        Collection conferences = getConferences();
        conferences.remove(conference);
    } catch (Exception ex) {
        throw new EJBException(ex.getMessage());
    }
}

public String ejbCreate(String ssn, String name, String attendeetype) throws CreateException {
    setSsn(ssn);
    setName(name);
    setAttendeeType(attendeetype);
    return null;
}
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

```java
public void ejbPostCreate (String ssn, String name,
     String attendeeType ) throws CreateException {
}

class AttendeeBean {
    public void setEntityContext(EntityContext ctx) {
        context = ctx;
    }

    public void unsetEntityContext() {
        context = null;
    }

    public void ejbRemove() {}
    public void ejbLoad() {}
    public void ejbStore() {}
    public void ejbPassivate() {
    }
    public void ejbActivate() {
    }
} // AttendeeBean class
```
3. The *Register* entity bean component includes following files:
   - Remote interface: *Register.java*
   - Remote home interface: *RegisterHome.java*
   - Entity bean class: *Register.java*
   - Helper class: *ConferenceDetails.java* (same as in the Lab2)
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

- The Register session bean is a revised version of DiscountCalc session bean in the Lab2.
- In the RegisterApp application the register bean is a stateful session bean which can access two entity beans, and in turn to access database.
- Four new business methods are added to the session bean. The createAttendee() invokes attendeeHome.create() to create a LocalAttendee instance.
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

- The `addRegistration()` and `cancelRegistration()` method locates the `AttendeeBean` and `ConferenceBean` instances by calling `findByPrimaryKey()`, then invokes the `addConference()` (or `removeConference()` ) methods on the instance to update the relationship between the conference and attendee.

- Database manipulations are automatically implemented by the container.

- The `getConfOfAttendee()` method returns the `ArrayList` of `ConfDetails` object for a attendee.
//Register.java Remote interface of Register session bean
package register;
import java.util.*;
import javax.ejb.EJBObject;
import java.rmi.RemoteException;
public interface Register extends EJBObject {

    public void createAttendee(String ssn, String name, 
            String attendeeType) throws RemoteException;

    public ConfDetails getConfDetail( String confId)
        throws RemoteException;

    public void addRegistration(String ssn, String 
            conferenceId) throws RemoteException;
}
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

```java
public void cancelRegistration(String ssn, String conferenceId) throws RemoteException;

public ArrayList getConfsOfAttendee(String ssn) throws RemoteException;

public ArrayList getAllConferences() throws RemoteException;

public double getDiscountedFee (double registFee, String attendeeType) throws RemoteException;
```

// register
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

//RegisterHome.java is a Remote home interface of
//Register session bean
package register;
import java.rmi.RemoteException;
import javax.ejb.*;
public interface RegisterHome extends EJBHome {
    Register create() throws RemoteException, CreateException;
}

//RegisterBean.java
package register;
import java.util.*;
import javax.ejb.*;
import javax.naming.*;
import java.rmi.RemoteException;
import data.*;
import java.rmi.RMIException;
public class RegisterBean implements SessionBean {

    private LocalAttendeeHome attendeeHome = null;
    private LocalConferenceHome conferenceHome = null;

    // attendee business methods
    public void createAttendee(String ssn, String name,
                                 String attendeeType) {
        try {
            LocalAttendee attendee =
                attendeeHome.create(ssn, name, attendeeType);
        } catch (Exception ex) {
            throw new EJBException(ex.getMessage());
        }
    }
}
public ConfDetails getConfDetail( String confId) throws RemoteException {
    ConfDetails confDetail;
    LocalConference conference = null;
    try {
        conference = conferenceHome.findByPrimaryKey(confId);
        confDetail = new ConfDetails(confId,
                                      conference.getConferenceName(),
                                      conference.getRegistrationFee());
    } catch ( Exception ex) {
        throw new EJBException (ex.getMessage());
    }
    return confDetail;
}
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

```java
// register for a conference
public void addRegistration(String ssn, String conferenceId) {
    try {
        LocalConference conference =
            conferenceHome.findByPrimaryKey(conferenceId);
        LocalAttendee attendee =
            attendeeHome.findByPrimaryKey(ssn);
        attendee.addConference(conference);
    } catch (Exception ex) {
        throw new EJBException(ex.getMessage());
    }
}
```
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

```java
// cancel a conference registration
public void cancelRegistration(String ssn, String conferenceId) {
    try {
        LocalAttendee attendee = 
            attendeeHome.findByPrimaryKey(ssn);
        LocalConference conference =
            conferenceHome.findByPrimaryKey(conferenceId);
        attendee.removeConference(conference);
    } catch (Exception ex) {
        throw new EJBException(ex.getMessage());
    }
}
```
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

// get registered conference list for an attendee
public ArrayList getConfsofAttendee(String ssn) {
    Collection conferences = null;
    try {
        LocalAttendee attendee =
            attendeeHome.findByPrimaryKey(ssn);
        conferences = attendee.getConferences();
    } catch (Exception ex) {
        throw new EJBException(ex.getMessage());
    }
    return confDetailList(conferences);
} // getConfsofAttendee
lab3: a j2ee application with stateful session bean and cmp entity beans (cont.)

public ArrayList getAllConferences() {
    Collection conferences = null;
    try {
        conferences = conferenceHome.findAll();
    } catch (Exception ex) {
        throw new EJBException(ex.getMessage());
    }
    return confDetailList(conferences);
} // getAllConferences

private ArrayList confDetailList(Collection conferences) {
    ArrayList detailsList = new ArrayList();
    Iterator i = conferences.iterator();

while (i.hasNext()) {
    LocalConference conference = (LocalConference) i.next();
    ConfDetails details = new ConfDetails
    (conference.getConferenceId(),
     conference.getConferenceName(),
     conference.getRegistrationFee());
    detailsList.add(details); }

return detailsList;
} // confDetailList

public double getDiscountedFee (double registFee,
                                String attendeeType) {

    int discountRate = 0;
    if (attendeeType.equals("Member") )  discountRate = 20;
    else if (attendeeType.equals("Student") )
        discountRate = 50;
    else discountRate = 0;

return  (registFee * (1 - (double)discountRate/100 ));}
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

```java
public void ejbCreate() throws CreateException {
    try {
        attendeeHome = lookupAttendee();
        conferenceHome = lookupConference();
    } catch (NamingException ex) {
        throw new CreateException(ex.getMessage());
    }
}

public void ejbActivate() {
    try {
        attendeeHome = lookupAttendee();
        conferenceHome = lookupConference();
    } catch (NamingException ex) {
        throw new EJBException(ex.getMessage());
    }
}
```
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

```java
public void ejbPassivate() {
    attendeeHome = null;
    conferenceHome = null;
}
public RegisterBean() {}
public void ejbRemove() {}
public void setSessionContext(SessionContext sc) {} 

// Private methods
private LocalAttendeeHome lookupAttendee() 
    throws NamingException {

    Context initial = new InitialContext();
    Object objref =
    initial.lookup("java:comp/env/ejb/SimpleAttendee");
    return (LocalAttendeeHome) objref;
}
```
private LocalConferenceHome lookupConference() throws NamingException {
    Context initial = new InitialContext();
    Object objref =
        initial.lookup("java:comp/env/ejb/SimpleConference");
    return (LocalConferenceHome) objref;
}
} // RegisterBean
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

- Step 2. Compiling the Source Files and generate database schema file
  - To compile the source files go to c:\ejb\lab3 directory and run the asant build command, 10 source files will be compiled.
  - Start up database
  - Type asant capture-db-schema to create the database schema file.
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

• Step 3. Packaging Entity Beans

1. Start up application server, deploytool and database.
2. Create a new application RegisterApp.ear under c:\ejb\lab3 directory.
3. Following the steps in Lab2 to Package entity bean ConferenceBean in the JAR file named DataJar.
4. Adding AttendeeBean Entity Bean to the Existing JAR File
   – Select File | New | Enterprise Bean to start the New Enterprise Bean wizard
   – In the EJB JAR dialog box select Add to Existing JAR File
   – In the Existing JAR File, select DataJar, click Next.
   – Follow the steps of creating entity bean of ConferenceBean to complete the AttendeeBean creation with the following settings.
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

- EJB JAR General Settings:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents</td>
<td>LocalAttendee.class, LocalAttendeeHome,AttendeeBean.class.</td>
</tr>
<tr>
<td>Enterprise Bean Class</td>
<td>Data.AttendeeBean</td>
</tr>
<tr>
<td>Enterprise Bean Name</td>
<td>AttendeeBean</td>
</tr>
<tr>
<td>Enterprise Bean Type</td>
<td>Entity</td>
</tr>
<tr>
<td>Local Home Interface</td>
<td>Data.LocalAttendeeHome</td>
</tr>
<tr>
<td>Local Interface</td>
<td>Data.LocalAttendee</td>
</tr>
</tbody>
</table>
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

- Entity Bean Settings:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fields To Be Persisted</td>
<td>Name, ssn, attendeeType</td>
</tr>
<tr>
<td>Primary Key Class</td>
<td>Select an existing field - ssn</td>
</tr>
<tr>
<td>Abstract Schema Name</td>
<td>Attendee</td>
</tr>
<tr>
<td>Transactions</td>
<td>Container-managed</td>
</tr>
</tbody>
</table>
5. Defining Entity Relationships

After you create ConferenceBean and AttendeeBean you can define relationships that reside in the same EJB JAR file.

- Select DataJAR in the tree view and then select the relationships tab
- Click Add (or Edit to modify) button to open Add Relationship dialog box
- Select many to many in the combo box
- In the Enterprise Bean A, select ConferenceBean for Enterprise Bean Name, attendees for Field Referencing Bean B, and java.util.Collection for Field Type.
- In the Enterprise Bean B, select AttendeeBean for Enterprise Bean Name, conferences for Field Referencing Bean A, and java.util.Collection for Field Type.
6. Specify database mapping
   - Select DataJar from the tree in *deploytool*.
   - Select the General tab
   - Click Sun-specific Settings button.
   - In the JNDI Name field enter jdbc/PointBase.
   - Click Create DataBase Mappings button.
   - Select Map to Tables in Database Schema File
   - Choose register.dbschema for the Database Schema Files in Module
   - Click OK to go to Sun-specific Settings dialog box.
   - Under Persistent File Mapping, select AttendeeBean for Enterprise Bean, Attendee for Primary table. Verify field mappings for ssn, name and attendeeType, locate the row of conferences, select `<map relationship field>` in the Mapped Column, go to Map Relationship Field dialog box.
### CMP Database

- **JNDI Name:** jdbc:PointBase

### Persistent Field Mappings

#### Enterprise Bean:
- **AttendeeBean**

#### Mappings for Bean 'AttendeeBean'

<table>
<thead>
<tr>
<th>Field</th>
<th>Mapped Column</th>
<th>Type</th>
<th>Fetch</th>
</tr>
</thead>
<tbody>
<tr>
<td>attendeeType</td>
<td>ATTENDEE_ATTENDEETYPE</td>
<td>VARCHAR(30)</td>
<td>Default</td>
</tr>
<tr>
<td>conferences</td>
<td>&lt;current mapping&gt;</td>
<td>&lt;multiple&gt;</td>
<td>None</td>
</tr>
<tr>
<td>name</td>
<td></td>
<td>VARCHAR(30)</td>
<td>Default</td>
</tr>
<tr>
<td>ssn</td>
<td>&lt;current mapping&gt;</td>
<td>VARCHAR(9)</td>
<td></td>
</tr>
</tbody>
</table>

#### Advanced Settings...
### Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

- In the Initial Setup [1 of 3] page, verify the following settings:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>This Bean</td>
<td>AttendeeBean</td>
</tr>
<tr>
<td>This Field</td>
<td>conferences</td>
</tr>
<tr>
<td>Primary Table</td>
<td>ATTENDEE</td>
</tr>
<tr>
<td>Related Bean</td>
<td>ConferenceBean</td>
</tr>
<tr>
<td>Related Field</td>
<td>attendees</td>
</tr>
<tr>
<td>Primary Table</td>
<td>CONFERENCE</td>
</tr>
</tbody>
</table>
• Select Link Mapped Tables Using a join Table radio button, and then click Next.
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

• In the Map to key: Local to Join [2 of 3] page, verify that This Bean is Attendee and this field is conferences, and then select REGISTER as Join Table.

• Under Key Column Pairs, select ATTENDEE.SSN for Local Column, REGISTER.SSN for the Join Table Column.

• Click Next.
Map to Key: Local to Join [2 of 3]

Choose a join table. Then, map the relationship field to a key, described as a pair or pairs of columns. The most common way to map a relationship field is to map it to a foreign key or the inverse of a foreign key. If there is no foreign key defined, you might want to map to a reference key or define a custom key.

This Bean: AttendeeBean
Join Table: REGISTER
This Field: conferences

Key Column Pairs:

<table>
<thead>
<tr>
<th>Local Column</th>
<th>Join Table Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTENDEE.SSN</td>
<td>REGISTER.SSN</td>
</tr>
</tbody>
</table>

Add Pair  Remove

Review the default key or define your own custom key. A key contains pairs of columns from the tables mapped to this bean. The columns in each pair are expected to have the same value. To create a compound key, use the Add Pair button.
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

- In the Map to Key: Join to Foreign {3 of 3} page, verify that Join Table is REGISTER, Related Bean is Conference Bean, and Related Field is attendees.
- Under Key Column Pairs, select REGISTER.CONFERENCE_ID as Join Table Column, CONFERENCE.CONFERENCE_ID as Foreign Column.
- Click Finish.
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

- Packaging the Session Bean – RegisterBean
- In the RegisterApp application the session bean is a stateful session bean and handled by container-managed. Follow the steps in the Lab1 to package the session bean with the following settings.

**EJB JAR and session bean Settings:**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>JAR Name</td>
<td>RegisterJar</td>
</tr>
<tr>
<td>Contents</td>
<td>Register.class, RegisterHome.class,</td>
</tr>
<tr>
<td></td>
<td>RegisterBean.class, ConfDetails.class</td>
</tr>
<tr>
<td>Enterprise Bean Class</td>
<td>register.RegisterBean</td>
</tr>
<tr>
<td>Enterprise Bean Name</td>
<td>RegisterBean</td>
</tr>
<tr>
<td>Bean Type</td>
<td>Stateful</td>
</tr>
<tr>
<td>Remote Home interface</td>
<td>register.RegisterHome</td>
</tr>
<tr>
<td>Remote Interface</td>
<td>register.Register</td>
</tr>
</tbody>
</table>
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

- Specify the Session Bean Reference:
  - In the tree, select RegisterBean
  - Select the EJB Refs tab and Click Add

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coded Name</td>
<td>ejb/SimpleAttendee</td>
</tr>
<tr>
<td>Type</td>
<td>Entity</td>
</tr>
<tr>
<td>Interface</td>
<td>Local</td>
</tr>
<tr>
<td>Home Interface</td>
<td>data.LocalAttendeeHome</td>
</tr>
<tr>
<td>Local/Remote Interface</td>
<td>data.LocalAttendee</td>
</tr>
<tr>
<td>Target EJB/Enterprise Bean Name</td>
<td>ejb-jar-ic.jar#AttendeeBean</td>
</tr>
</tbody>
</table>
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

• Add another line for conferenceBean.

Specifying Transaction Settings
  – Select RegisterBean in the tree.
  – Click Transactions tab
  – Select Container-Managed
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

• Step 6. Implement Web Client

  – The Web component includes a html file index.html and a JSP file register.jsp.
  – The index.html is a simple html page, which enables users to enter their social security number, name, attendee type, and select the conference they are interested in.
  – When the user submits the form, the page index.jsp passes the request to register.jsp.
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

```html
<!-- index.jsp -->
<html><center>
<head>
<title>Register</title>
<h4>Computing Symposium On-line Registration</h4>
</head>
<body>
<form method="get" action="register.jsp">
<P>Please Enter your SSN, Name and Attendee Type: </P>
<table width="320">
<tr><td>SSN: </td><td><input type="TEXT" name="ssn"></input></td></tr>
<tr><td>Name: </td><td><input type="TEXT" name="name"></input></td></tr>
<tr><td>Attendee Type: </td><td><input type="radio" name="attendeeType" value="Non-Member" checked>
```
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

```html
<p>Please select the conference you are interested in, and then click Submit to register for the conference.</p>
<select name="confId" size="1">
</select>
```
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

<option value="001">Distributed Computing</option>
<option value="002">Component Based Computing</option>
<option value="003">Grid Computing</option>
</select>
<br>
<input type="submit" name="newAttendee" value="Submit"/>
<input type="reset" name="Reset" value="Reset"/>
</form>
</body>
</html>
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

- The `register.jsp` uses the method `jspInit()` to looks up the session bean home interface and create its instance. When the `register.jsp` receives a request from `index.html` the Register Web client invokes the `createAttendee()` business method of `RegisterBean` session bean to create a new attendee.

- The `register.jsp` page also accepts an input for conference. When user select a conference and click Add button, the web client invokes the `addRegistration()` method to add a relationship, and the table `register` will be updated. After the database is manipulated the method `getConfsOfAttendee()` returns the registered conference list for the user.
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

<%@ page import="javax.naming.*" %>
<%@ page import="javax.rmi.PortableRemoteObject"%>
<%@ page import="java.rmi.RemoteException"%>
<%@ page import="javax.ejb.*"%>
<%@ page import="java.util.*"%>
<%@ page import="register.*"%>
<%!
private Register theRegister = null;
String ssn = "";
String name = "";
String attendeeType = "";
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

String confId = null;
String confName = null;
ConfDetails confDetail;
double registFee = 0.00;
double totalFee = 0.00;
DecimalFormat twoDigits = new DecimalFormat("0.00");
ArrayList confsOfAttendeeList = null;
public void jspInit() {
    try {
        InitialContext ic = new InitialContext();
        Object objRef =
            ic.lookup("java:comp/env/ejb/SimpleRegister");
        RegisterHome home =
            (RegisterHome)PortableRemoteObject.
narrow(objRef, RegisterHome.class);
        theRegister = home.create();
    } catch (Exception ex) {
        System.out.println(ex.getMessage());
    }
}
public void jspDestroy() {
    theRegister = null;
}

<ssn = request.getParameter("ssn");
name = request.getParameter("name");
attendeeType = request.getParameter("attendeeType");
confId = request.getParameter("confId");
if (ssn.equals ("") || name.equals ("")) {
    <jsp:forward page = "index.html" />

Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

```jsp
<head>
    <title>Register</title>
    <h4>Computing Symposium On-line Registration </h4>
</head>
<body>
<% if ( request.getParameter("newAttendee") != null) {
    try {
        theRegister.createAttendee(ssn, name, attendeeType);
    } catch (Exception ex) {
        System.out.println(ex.getMessage());
    }
}%>

<P> <%= name %> , Welcome to Computing Symposium</P>
<% }
```
if (confId != null && ssn != null) {
    try {
        if (request.getParameter("Add") != null ||
            request.getParameter("newAttendee") != null)
            theRegister.addRegistration(ssn, confId);
        if (request.getParameter("Remove") != null)
            theRegister.cancelRegistration(ssn, confId);
    } catch (Exception ex) {
        System.out.println(ex.getMessage());
    }
}
try {ArrayList confsOfAttendeeList =
    theRegister.getConfsOfAttendee(ssn);
%
<p> You have registered the following conferences </p>
<table>
    <tr><th width="160">Conference</th>
        <th width="130">Registration Fee</th>
    </tr>
    <th width="130">Registration Fee</th>
    <%
    Iterator i = confsOfAttendeeList.iterator();
    totalFee = 0.00;
    while (i.hasNext()) {
        confDetail = (ConfDetails)i.next();
        registFee = theRegister.getDiscountedFee
            (confDetail.getRegistFee(), attendeeType);
    </%>
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

```java
int totalFee = 0;
for (Attendee attendee: attendees) {
    totalFee += theRegister.getDiscountedFee
            (confDetail.getRegistFee(), attendeeType);
    <tr><td> <%= confDetail.getConfName() %> </td><td align="right"> $ <%= twoDigits.format(registFee) %></td></tr>
</table>
<% } %>
```

```
<table>
<tr><td align="right">________________</td></tr>
<tr><td>Total registration fee: </td>
<td align="right"> $ <%= twoDigits.format(totalFee)%></td></tr>
</table>
<%}%>
```
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

<form method="get" action="register.jsp">
  <P> Please select the conference you are interested in, and then click Add to register for the conference, or click Remove to cancel the registration. </P>
  <select name="confId" size="1">
    <option value="001">Distributed Computing</option>
    <option value="002">Component Based Computing</option>
    <option value="003">Grid Computing</option>
  </select>
<br><br>
</form>
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

```html
<input type="hidden" name="ssn" value="<%= ssn %>">
<input type="hidden" name="name" value="<%= name %>">
<input type="hidden" name="attendeeType" value=""
    <%= attendeeType %>">
<input type="submit" name="Add" value="Add">
<input type="submit" name="Remove" value="Remove">
</form>
</body>
</center>
</html>
```
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

- Step 7. Packing the Web Component

  Following the steps in Lab1 to create War file RegisterWar

  The settings for the web component as follows:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAR Name</td>
<td>RegisterWar</td>
</tr>
<tr>
<td>Contents</td>
<td>Under c:\ejb\lab3\web Index.html, register.jsp</td>
</tr>
<tr>
<td>Component Type</td>
<td>JSP</td>
</tr>
<tr>
<td>JSP File Name</td>
<td>Register.jsp</td>
</tr>
<tr>
<td>Web Component display Name</td>
<td>register</td>
</tr>
<tr>
<td>Startup load sequence position</td>
<td>Load at any time</td>
</tr>
</tbody>
</table>
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

- Specifying the Web component Reference

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coded Name</td>
<td>ejb/SimpleRegister</td>
</tr>
<tr>
<td>Type</td>
<td>Session</td>
</tr>
<tr>
<td>Interface</td>
<td>Remote</td>
</tr>
<tr>
<td>Home Interface</td>
<td>register.RegisterHome</td>
</tr>
<tr>
<td>Local/Remote Interface</td>
<td>register.Register</td>
</tr>
<tr>
<td>TargetEJB/JNDI name</td>
<td>MyRegister</td>
</tr>
</tbody>
</table>
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

- Sun-Specific Settings (See Lab1)
  - Use MyRegister for the JNDI name.
  - Use register for the Context Root.

7. deployment
   - save the application
   - Click Tools | Deploy
Lab3: A J2EE Application with Stateful Session Bean and CMP Entity Beans (cont.)

• Step 5. Running the RegisterApp Application

To access the online registration application, point your browser to [http://localhost:8080/register](http://localhost:8080/register)

– Fill in a Social Security Number
– Fill in a Name
– Select Attendee Type
– Select the conference
– Click the Submit button

• The *index.html* will forward the request to the *register.jsp* after the Submit button is pushed.

• The following two screens show a sample execution of this Web application.
Computing Symposium On-line Registration

Please Enter your SSN, Name and Attendee Type:

SSN: 999999999
Name: David Walton
Attendee Type: 
   - Non-Member
   - Member
   - Student

Please select the conference you are interested in, and then click Submit to register for the conference.

Distributed Computing

Submit  Reset
• After you selected the conference of distributed computing and component based computing, the total registration fee is listed as $200 at the membership discount rate. You can add more conferences afterward.
Summary

• Sun Microsystems’ EJB2.x provides a complete component infrastructure similar to CORBA 3.0.
• The EJB technology is widely used for building Java Web enterprise application in distributed computing environment.
• Any EJB component has two interfaces (except MDB) and one bean class implementing the interfaces.
• The EJB home (can be local home or remote home) interface manages the lifecycle of an EJB object such as creating, removing an EJB instance and finding an entity bean instance by primary-key.
Summary (cont.)

• There are three EJB types: session bean, entity bean, and message driven bean.
• A session bean may be stateless or stateful and both of them act on the behaviors of their clients.
• The stateless EJB component does not hold any state information during its session such as an online calculator.
• On the other hand, stateful session beans need to hold some state information during the session.
• For instance, a shopping cart session bean needs to keep track of client’s items in a collection.
Summary (cont.)

• There are two types of entity beans, too, BMP and CMP ones, and both of them are used in the back-end tier to connect database and to support the persistence.

• Each entity bean has a relation table behind it. Each instance of the entity EJB component is stored in the table as a row.

• CMP entity beans are free of SQL database programming and the mapping to database implementation is completed by an enterprise application deployer at deployment time.
Summary (cont.)

• Any EJB component is hosted by an EJB container, which provides all necessary services and makes EJB more portable and scalable.
• An EJB component is packed, assembled, and finally deployed on a server.
• EJB components are server side reusable components. This compares to JavaBeans components discussed in Chapter 3, which are client side reusable components.
• EJB components are widely used in distributed computing applications in a LAN, WAN, Internet, or wireless network environment.
• The EJB component can work with other components such as CORBA components, Servlets, or JSP web components.