Developing a Real Java EE 5 Application

SAP®

Copyright

Icons in Body Text

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax

Additional icons are used in SAP Library documentation to help you identify different types of information at a glance. For more information, see *Help on Help* → *General Information Classes and Information Classes for Business Information Warehouse* on the first page of any version of *SAP Library*.

Typographic Conventions

Type Style	Description
Example text	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation.
Example text	Emphasized words or phrases in body text, graphic titles, and table titles.
EXAMPLE TEXT	Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text, for example, SELECT and INCLUDE.
`Example text`	Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
`Example text`	Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example text>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE TEXT`	Keys on the keyboard, for example, `F2` or `ENTER`.

Project Management and Employee Services Application

This documentation provides information about the architecture and the implementation of the Project Management and Employee Services application. See:

Scenario Overview

The Project Management and Employee Services application is explanatory software showing the possibilities for developing applications with the SAP NetWeaver Application Server, Java(TM) EE 5 Edition.

This section provides an introduction to the architecture and technologies used for the creation of the application.

The two main parts of the architecture are:

- Application Data Model [Page 7]
- Application Layers [Page 9]
- Scenario Walkthrough [Page 11]

Application Data Model

A data model is the database concept representing the application entities, entity attributes, and entity relations persisted in the database schema.

The entities (components) specified for the application are:

- Educational Data Model (EDM) entities:
  - Department

    This entity represents a department. The attributes are ID, name, manager, version, list of employees.
  - Employee

    This entity manages the records for the employee instances. The attributes for this entity are employeeId, firstName, lastName, email, skills, emoloyeeDepartment, deptMnager, and so on.
  - Project

    This entity is responsible for the records of the project instances. Its attributes are projectId, title, description, startDate, endDate, employess, leader, skills, and so on.
  - Skill

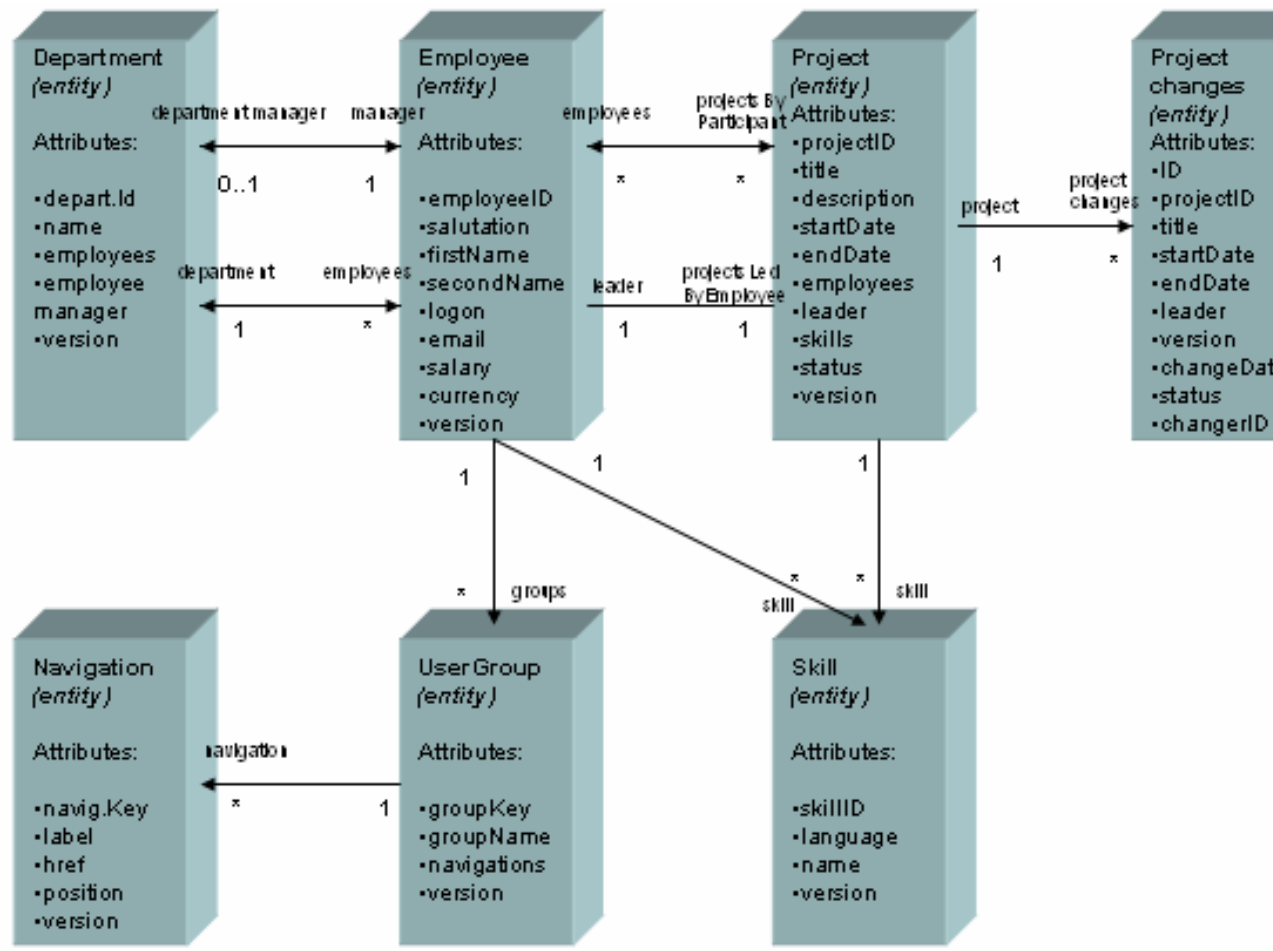
    This entity manages the database entries for the skill instances. The attributes are skill Id, description, language, version.
- Entities related to the application features:
  - Navigation

    This entity keeps the navigation sets defined for user groups. The available attributes are navigation key, label, href, position, and so on.
  - UserGroup

    This entity organizes the employees in user groups by connecting a single user to the set of navigation provided for its kind. The attributes are groupKey, groupName, version, and navigations.
  - ProjectChanges

    This entity keeps a record of the changes performed on the projects. For example, change of the corresponding dates, change of the list of employees, and so on. The entity attributes are id, project id, title, description, startDate, endDate, employee leader, version, change date, and so on.

Figure: RA data model describing entities, relations, and attributes



For the model presented in application, we have the following relations between the entities:

- Department – Employee

  Bidirectional one-to-many relationship for regular employees, and bidirectional one-to-one relationship for employees designated as department managers.

- Project – Employees

  Bidirectional many-to-many relationship for regular employees, and bidirectional one-to-one relationship for employees designated as project leads.

- Project – Skill

  Unidirectional one-to-many relationship.

- Employee – Skill

  Unidirectional one-to-many relationship.

- Employees – UserGroup

  Unidirectional one-to-many relationship.

- UserGroup – Navigation

  Unidirectional one-to-many relationship.

- Project-Projectchanges

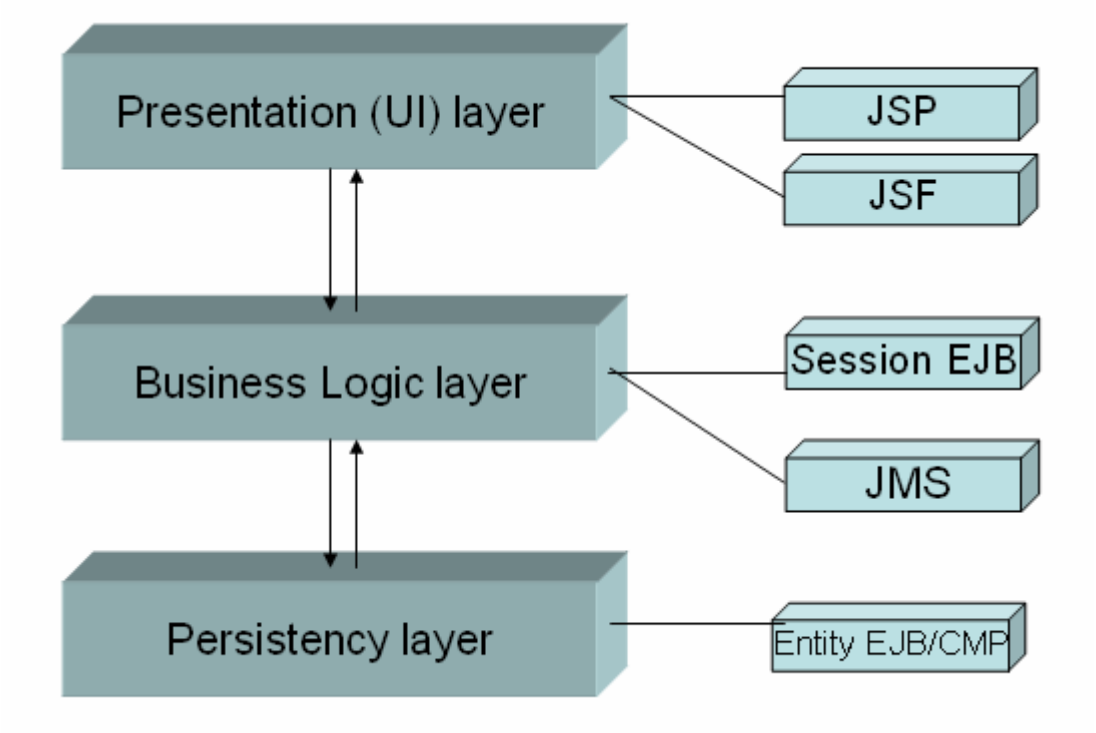
  Unidirectional one-to-many relationship.

Application Layers

As a business application based on Java EE 5 technology, the application consists of the following application layers:

- Presentation (UI) layer
- Business layer
- Persistence layer

See the figure below to become familiar with the model of structuring our application.



Persistency Layer Technologies

This layer implements the data model of the application. The components we introduced in the Data Model section are implemented as entity beans using Enterprise JavaBeans (EJB) 3.0 technology. EJB version 3.0 presents a new Java Persistency API. It simplifies and standardizes the object-relational mapping. The new Java Persistence API makes the application independent from the specific vendor's mapping requirements. The developer can either use annotations or XML descriptors to specify object-relational mapping information.

The entity beans are POJOs (Plain Old Java Objects) annotated with metadata. So they can be serialized and distributed across the network even in a persistence-unaware environment. As a consequence of this change, the data transfer objects (DTOs) become obsolete.

Business Layer Technologies

This layer performs the session operation between the components and implements the actual entity management. Generally, the layer implements the following business tasks:

- Employee management: finding objects (employees, skills, departments) by different attributes; creating employee and skill instances.
- Project management: finding projects by different attributes; creating and updating a project (for example, changing the employees assigned to the project, changing the start and end dates, and so on).

When creating or updating a project, the session bean that implements project management capabilities sends a JMS message to a configured JMS destination (a JMS queue). Message-driven beans enable the asynchronous consumption of JMS messages. A concurrent consumption of messages is possible with the EJB container architecture. The typical message driven bean comprises two parts:

- A “sender” part that sends the message to the queue (the session bean calls this part of the bean)
- A “receiver” part that processes the received messages in the queue (the EJB container calls this part automatically upon a message receipt in the queue). In the application, the MDB just converts the message into a text message and stores it in the Projectchanges entity as a history of the project change.

Presentation (UI) Layer Technologies

The presentation layer has the required elements to visualize the controller logic presented in the application business layer. For the implementation of this layer we use JavaServer Faces (JSF) 1.2 and JavaServer Pages (JSP) 2.1 technologies.

Unlike pure servlets and JSP technologies that mix the HTML presentation (VIEW) logic with the programming MODEL and CONTROLLER logic in the application, user interface (UI) designers work with XML-like tags and programmers writes the logic using Java beans. Then designers use these beans properties for the application UI.

Scenario Walkthrough

This section provides basic instructions about managing projects and employees in the application.

See also:

- Managing Projects [Page 12]
- Managing Employees [Page 13]

The option *Reset Application Data* resets the application to the state it is, before you deploy it on your server.

Managing Projects

Procedure

1. Creating New Projects

1. In the main application menu, choose *Create New Project*.
2. Type the required data:
   - Project Name
   - Project Description
   - Start date
   - End date
   - Project lead's name
3. Choose *Continue*.
4. Review the summary information and choose *Save*.

2. Assigning Employees to a Project

5. For the corresponding project, choose  (*Edit*).

   The project details appear in a separate window.
6. Choose *Edit Details*.
7. Choose *Assign Employees.*
8. Select the needed employees and by using the arrow button (->), move them to the right window.
9. Choose *Save.*

3.Assigning Skills to Project

10. For the corresponding project, choose  (*Edit*).

    The project details appear in a separate window.
11. Choose *Edit Details* .

    The Assign Project Skills field appears.
12. Select the needed skills and by using the arrow button (->), move them to the right window.
13. Choose *Save.*

Managing Employees

Procedure

1.Creating New Employees

1. In the main application menu, choose *Create New Employee*.
2. Enter the required data:
   - Salutation
   - Firstname
   - Lastname
   - Login
   - Email
   - Salary
   - Currency
   - Department
3. Choose *Continue*.

2. Creating New Skills

4. In the main application menu, choose *Create New Skill*.
5. Type the required data:
   - skill description
   - skill language
   - version
6. Choose *Continue*.

Rebuilding the Application Persistence and Business Logic Layers

Use

In this tutorial you learn how you implement the Project Management and Employee Services application components that build up the persistence and the business logic layers using the Java EE tools that the SAP NetWeaver Developer Studio provides. In the end you can deploy and run the application.

Procedure

The implementation comprises the following steps:

1. Developing the Persistence Entities [Page 15]

   As a first step, you implement the application data model using persistence entities. Using the Dali object-relational (O/R) mapping tools, you can easily generate the entities from the database tables that already exist in your DB.

2. Developing the Session Beans [Page 31]

   Next you implement the business logic for project and employee management using session beans.

3. Developing the Message Driven Bean [Page 37]

   Using a message-driven bean, you enable the application to use Java Messaging Service (JMS) and to record each change that is made to projects in a project history.

4. Deploying and Running the Application [Page 41]

   Finally, you publish the application to the SAP NetWeaver Application Server, Java™ EE 5 Edition where you can run and test it.

Developing the Persistence Entities

Use

This tutorial shows you how to implement the data model of the Project Management and Employee Services application using persistent entities. It uses the Employee entity as an example.

Using the Dali Object-relational (O/R) mapping tools integrated in the SAP NetWeaver Developer Studio, you can develop persistent entities that comply with the Java Pesistence API (JPA) standard.

Procedure

To create the Employee entity, you complete the following steps:

5. Creating the EDMProjectEJBDemo [Page 16]

   Persistent entities are not developed in a specific project. As they are usually packaged in an Enterprise JavaBeans (EJB) module, they are therefore developed in an EJB 3.0 project.

6. Defining a Connection to the Database Schema [Page 18]

   This makes it possible to list all tables that are created for the application. You can generate entities used by the application, based on the table definitions.

7. Generating the Project Entities [Page 20]

   In this step you select the tables that you want to generate persistent entities from. The tools do that automatically for you and generate the Plain Old Java Objects (POJOs) that are mapped to the respective database table.

8. Implementing the Employee Entity [Page 24]

   In this step you define relationships between your entities to reflect the relations defined by the data model of the application.

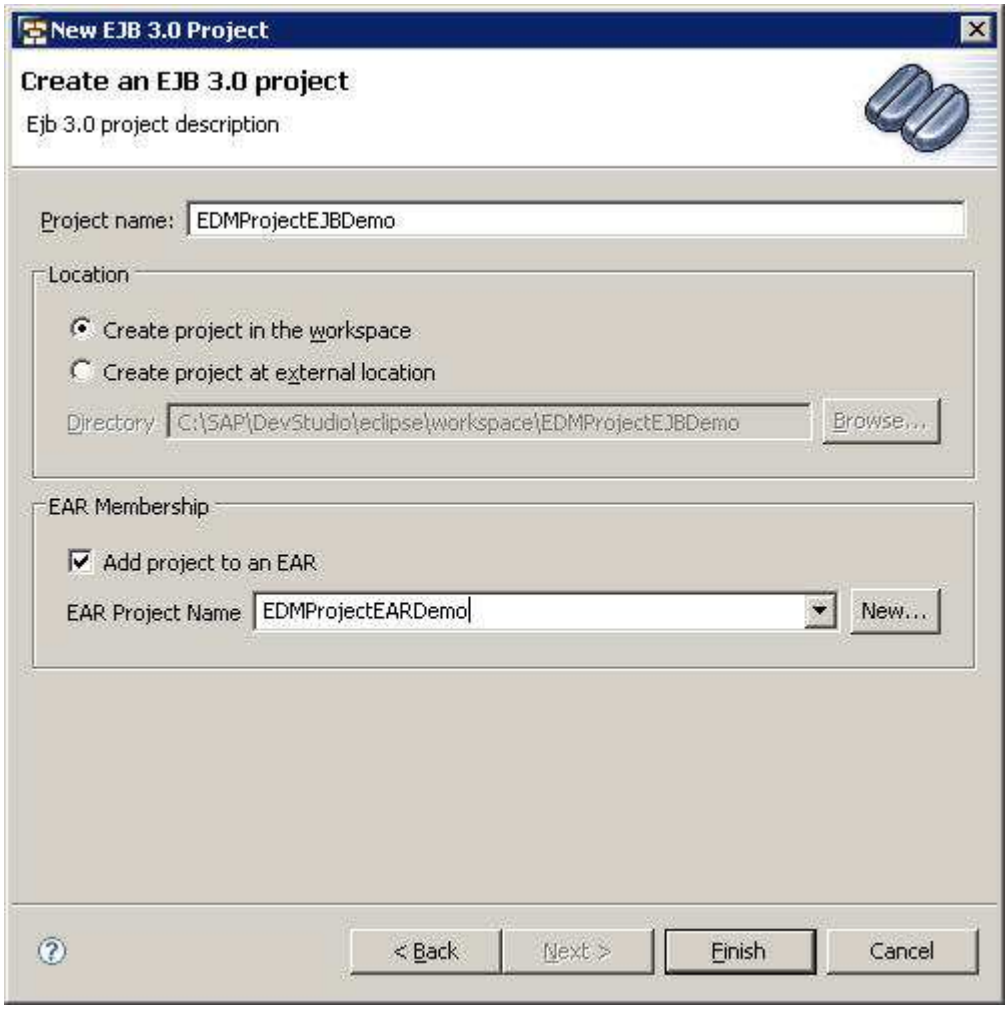
Creating the EDMProjectEJBDemo

Prerequisites

In the SAP NetWeaver Developer Studio, you have opened the J2EE perspective.

Procedure

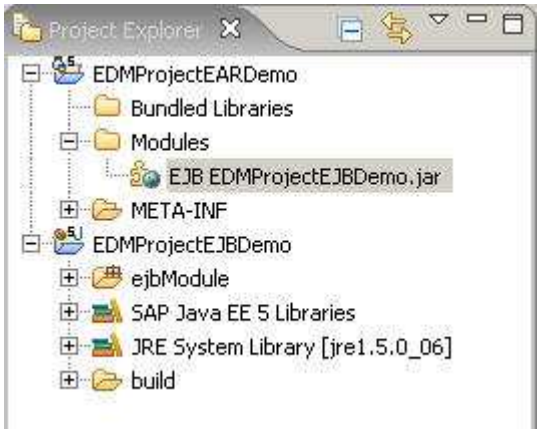
1. Choose *File* →*New* → *Project* from the main menu.
2. On the *New Project* screen, choose *EJB* → *EJB Project 3.0*.
3. On the next screen, do the following:
   - Enter **`EDMProjectEJBDemo`** in the *Project name* field
   - Select *Add project to an EAR*
   - Enter **`EDMProjectEARDemo`** in the *EAR Project Name* field



4. Choose *Finish* to complete the wizard.

Result

The Developer Studio generates the EDMProjectEJBDemo and EDMProjectEARDemo projects. The EAR project references the EJB one (the referred project is included in the list of modules that will be packaged in the EAR file).


Project Explorer
EDMProjectEARDemo
Bundled Libraries
Modules
EJB EDMProjectEJBDemo.jar
META-INF
EDMProjectEJBDemo
ejbModule
SAP Java EE 5 Libraries
JRE System Library [jre1.5.0_06]
build

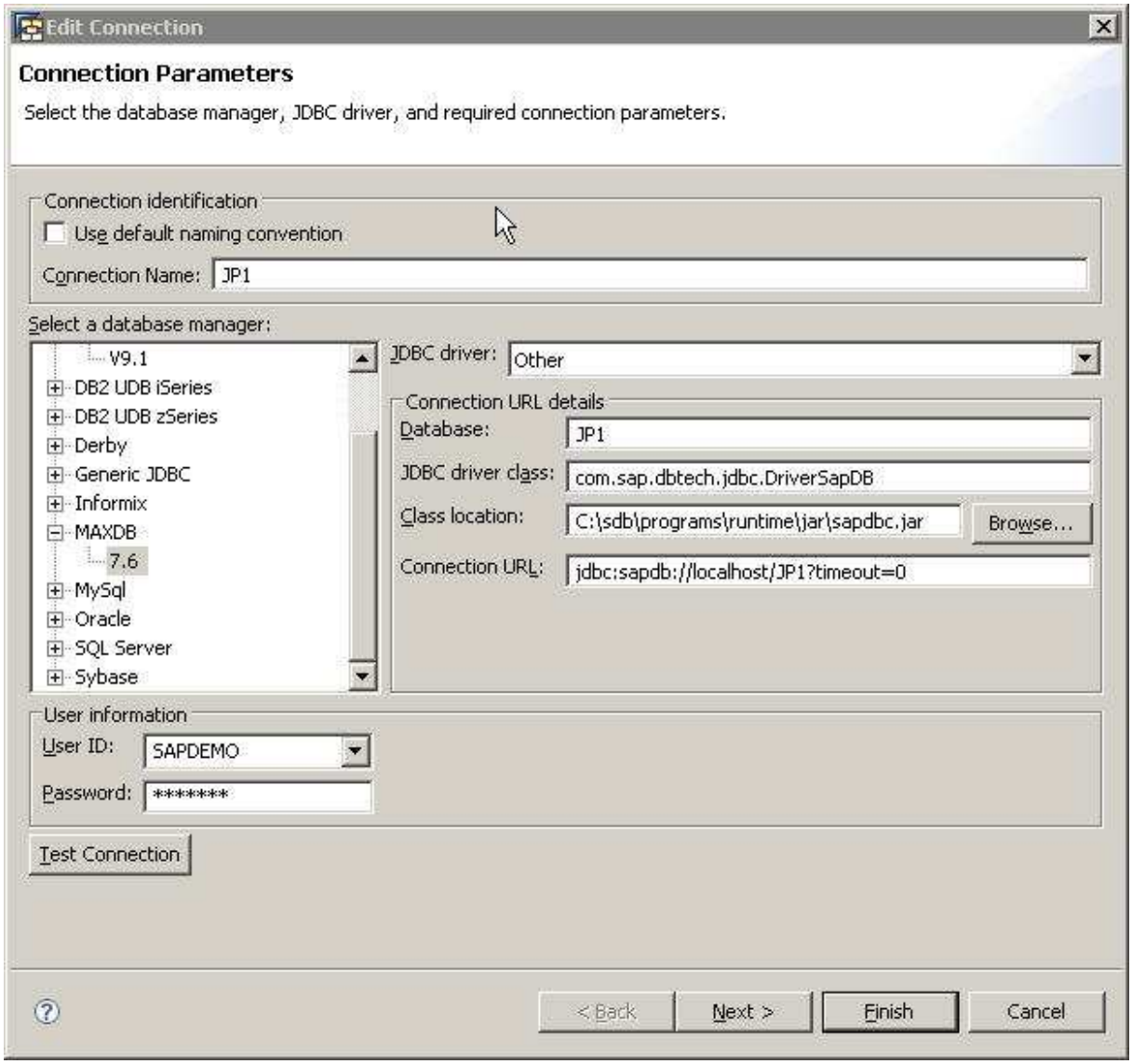
Defining a Connection to the Database Schema

Procedure

To define a connection to the database schema where the application tables are created, follow the procedure described in Creating Database Connections [External].

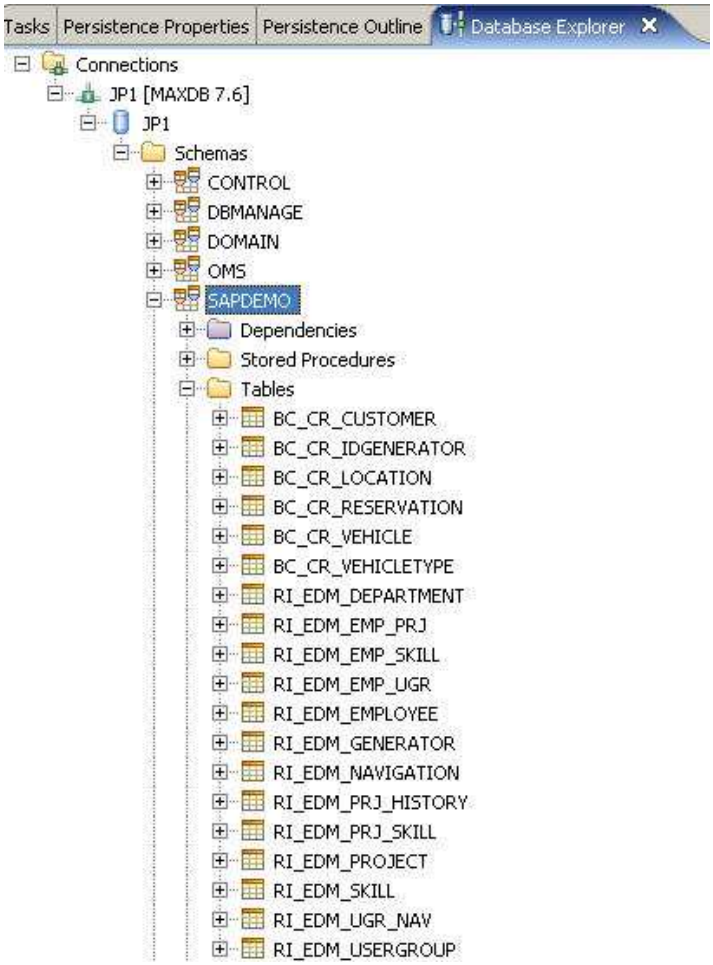
Connect to the **SAPDEMO** database schema with the following credentials:

- User ID: **SAPDEMO**
- Password: **sapdemo**



Result

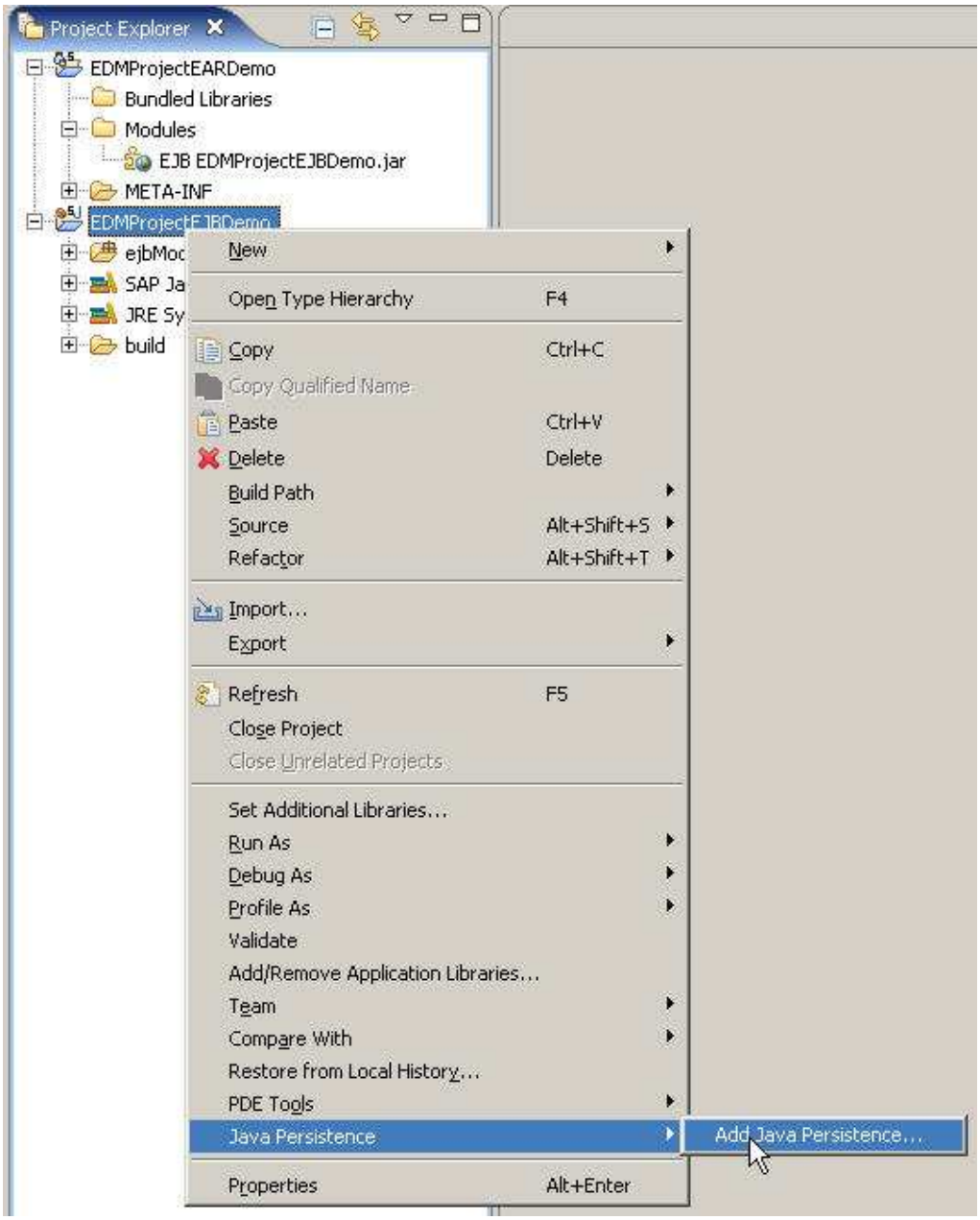
After you successfully define the connection, you should be able to list the content of the SAPDEMO database schema in the *Database Explorer* view, as shown below:


Tasks
Persistence Properties
Persistence Outline
Database Explorer
Connections
JP1 [MAXDB 7.6]
JP1
Schemas
CONTROL
DBMANAGE
DOMAIN
OMS
SAPDEMO
Dependencies
Stored Procedures
Tables
BC_CR_CUSTOMER
BC_CR_IDGENERATOR
BC_CR_LOCATION
BC_CR_RESERVATION
BC_CR_VEHICLE
BC_CR_VEHICLETYPE
RI_EDM_DEPARTMENT
RI_EDM_EMP_PRJ
RI_EDM_EMP_SKILL
RI_EDM_EMP_UGR
RI_EDM_EMPLOYEE
RI_EDM_GENERATOR
RI_EDM_NAVIGATION
RI_EDM_PRJ_HISTORY
RI_EDM_PRJ_SKILL
RI_EDM_PROJECT
RI_EDM_SKILL
RI_EDM_UGR_NAV
RI_EDM_USERGROUP

Generating the Project Entities

Procedure

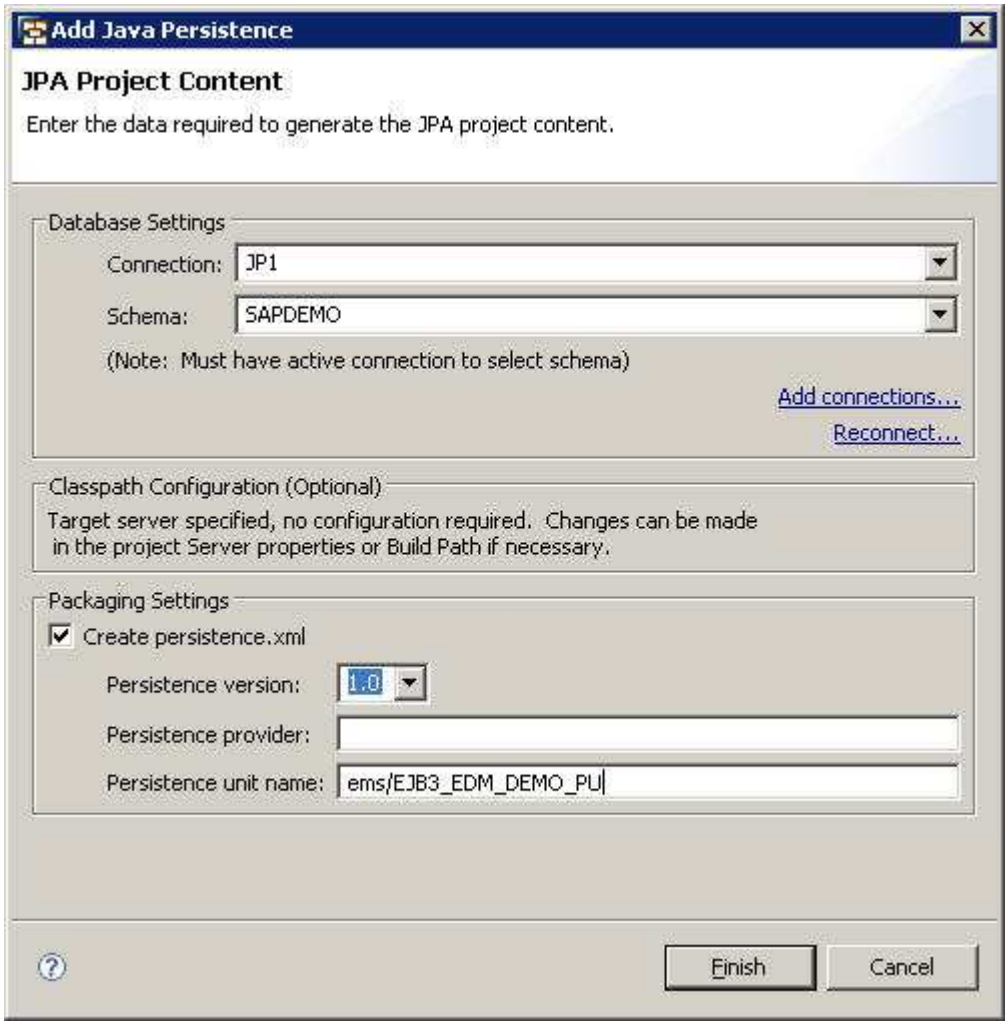
5. To enable your project to use persistence features, choose *Java Persistence* → *Add Java Persistence* from the context menu.



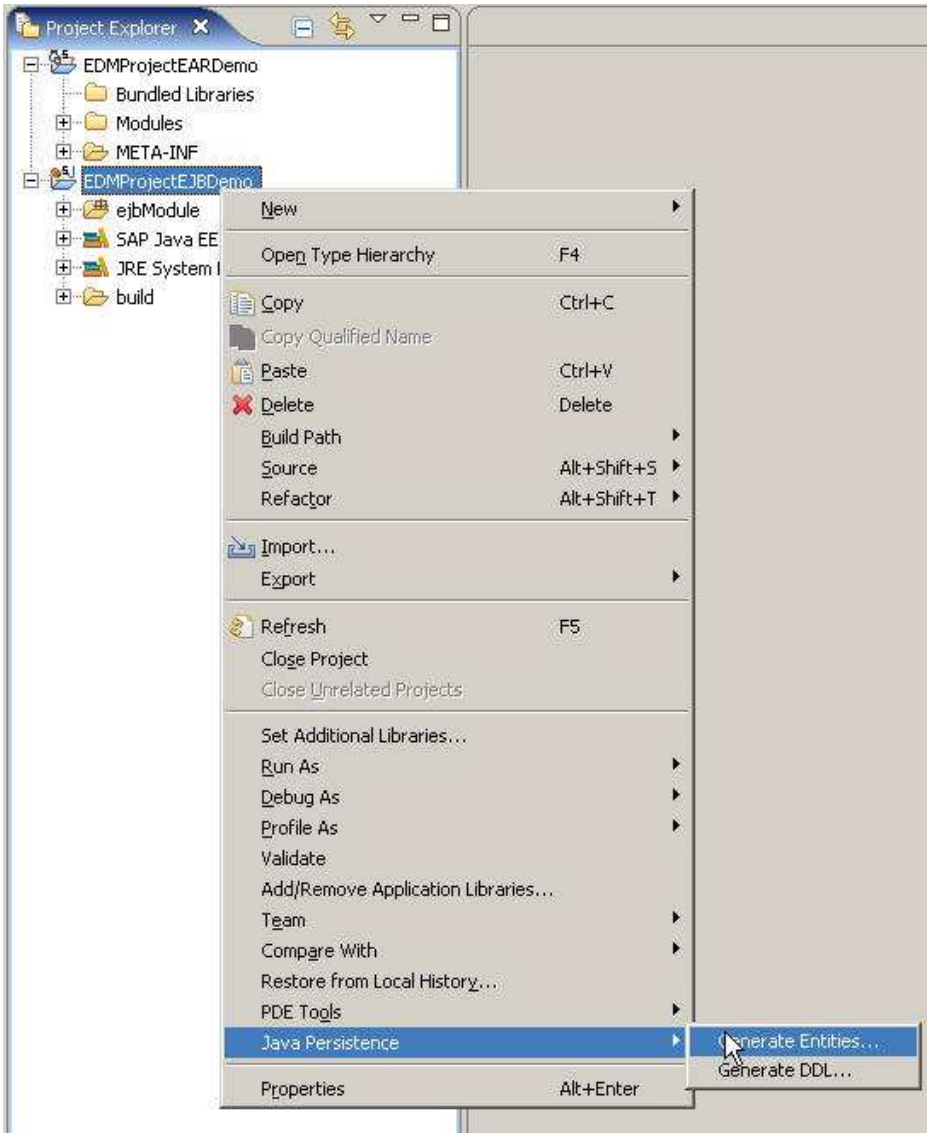
6. Specify the database to which your persistent entities should be mapped by providing the following details:

   a. In the *Database Explorer* view, select the name of the database from the *Connection* dropdown list.

   b. Select *SAPDEMO* schema from the *Schema* dropdown list.

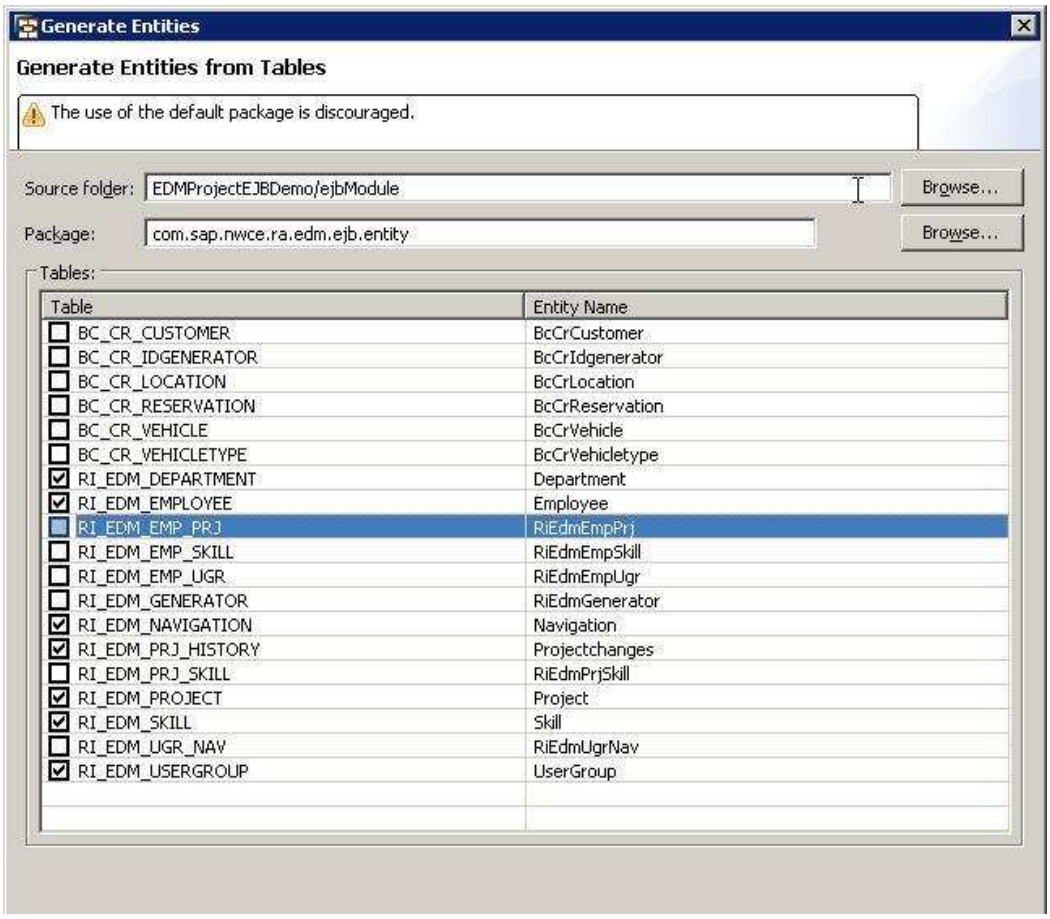


7. To create *persistence.xml*, packaged with your EJB 3.0 module, provide the following information in the *Packaging Settings*:
   a. Leave the Persistence provider field empty.
   b. Enter **`ems/EJB3_EDM_DEMO_PU`** in the *Persistence unit* name field. You use this name later on in your session EJBs code to inject the persistence unit.
   c. Choose *Finish*.
8. Choose *Java Persistence* → *Generate Entities* from the context menu of the EDMProjectEJBDemo.



9. On the *Create Entities* screen do the following:
   a. Enter **`com.sap.nwce.ra.edm.ejb.entity`** in the *Package* field.
   b. Select the DB tables that should be used to generate entities from. Change the default names of the entities in the *Entity Name* column as follows:

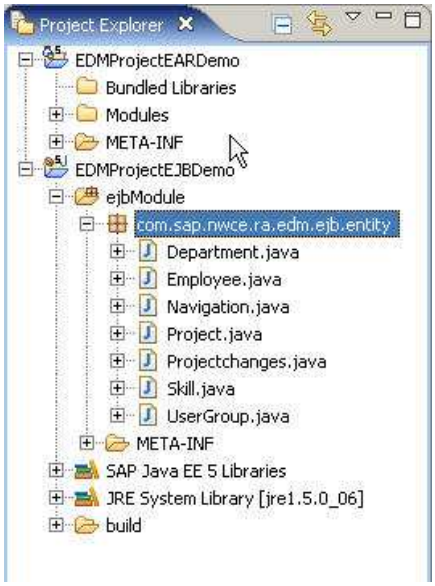
Table	**Entity Name**
RI_EDM_DEPARTMENT	Department
RI_EDM_EMPLOYEE	Employee
RI_EDM_NAVIGATION	Navigation
RI_EDM_PROJECT	Project
RI_EDM_SKILL	Skill
RI_EDM_USERGROUP	UserGroup
RI_EDM_PRJ_HISTORY	Projectchanges



10. To complete the procedure, choose *Finish*.

Result

The entities are generated and added to the EDMProjectEJBDemo:



Implementing the Employee Entity

Use

This procedure shows you how to implement the entities you generated in the previous step of the tutorial.

The automatic generation detects all columns of the tables and generates persistent fields in the entity accordingly. In this case, the RI_EDM_EMPLOYEE table contains a column DEPARTMENT_ID, which is a foreign key in a one-to-many relationship between the Department and Employee entities. This means that the Employee entity should not contain a persistent entity mapped to this column.

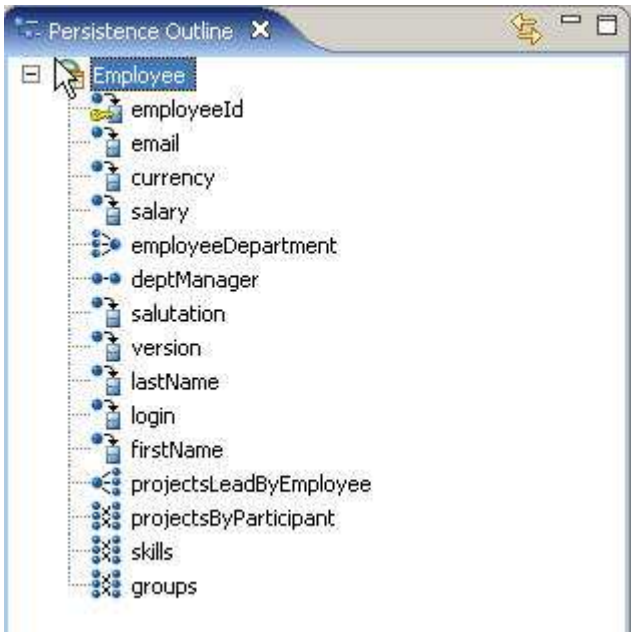
For each relationship the Employee entity participates in, you must define new variables that hold the foreign keys of the inverse side of the relationships.

Procedure

1. Adjusting the Persistent Fields

1. Open the entity for editing.

   The *Persistence Outline* view displays all persistent fields this entity has.



2. Delete the *departmentId* field and the corresponding getter and setter methods using the source code editor.
3. Add the following new variables:

```
private List<Project> projectsLeadByEmployee = new
ArrayList<Project>();

private List<Project> projectsByParticipant = new
ArrayList<Project>();

private List<Skill> skills = new ArrayList<Skill>();

private List<UserGroup> groups = new ArrayList<UserGroup>();

private Department employeeDepartment;

private Department deptManager;
```

4. Import the necessary classes by choosing *Source* → *Organize Imports* from the context menu.

   Make sure you select the `java.util.List` from the proposed options.

   At this point you can ignore the errors that you get for the *employeeDepartment* and the *deptManager* entries. They are handled later on when you set relationships between the entities.

5. To generate getter and setter methods for the new fields, choosing *Source* → Generate *Getters and Setter*.

   Select all variables in the list except *serialVersionUID*, and choose *OK*.

6. Add the following two methods that are used to manage an employee's set of skills:

```
public void addSkill(Skill s) {
  this.getSkills().add(s);
}
public void removeSkill(Skill s) {
  this.getSkills().remove(s);
}
```

Perform the same steps for the rest of the entities to define the fields that hold the corresponding foreign keys for the relationship they participate in.

2. Defining ID Generation

7. Select the *employeeId* field from the Persistence Outline view.

   This field is already marked as one that holds the primary key of the Employee entity based on the settings of the RI_EDM_EMPLOYEE table.

8. Open the Persistence Properties view by choosing *Window* → *Show View* → *Other* → *Java Persistence* → *Persistence Properties.*

9. To define the primary key generation strategy, open the *PK Generation* tab and proceed as follows:

   a. Choose *Primary Key Generation*.

   b. Select *Table* from the *Strategy* dropdown list.

   c. Enter *IdGenerator* in the *Generator Name* field.

   d. Expand the *Table Generator* list and select the *Table Generator* checkbox.

   e. Select *RI_EDM_GENERATOR* from the *Table* dropdown list.

f. Select *BEAN_NAME* from the *Primary Key Column.*

g. Select *MAX_ID* from the *Value* column.

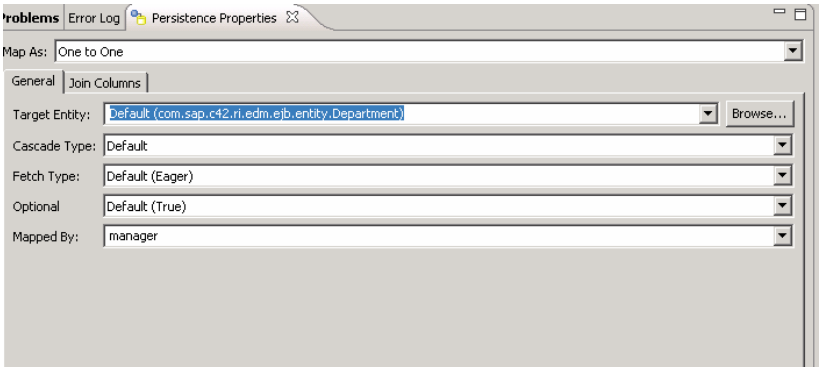
3. Defining Relationships

1. Define a **one-to-one** relationship between the *Employee* and *Department* entities, in which Employee is the owning side of the relationship:

   a. Select the *deptManager* field from the *Persistence Outline* view.

   b. Provide the following information in the *Persistence Properties* view

Property	**Value**
Map As	One to One
Fetch Type	Lazy
Mapped By	Manager



This generates the following annotation for the *deptManager* field:

```
@OneToOne(mappedBy="manager")
```

2. Define a **many-to-one** relationship between the Employee and the Department entities for the employeeDepartment variable. Complete the following steps:

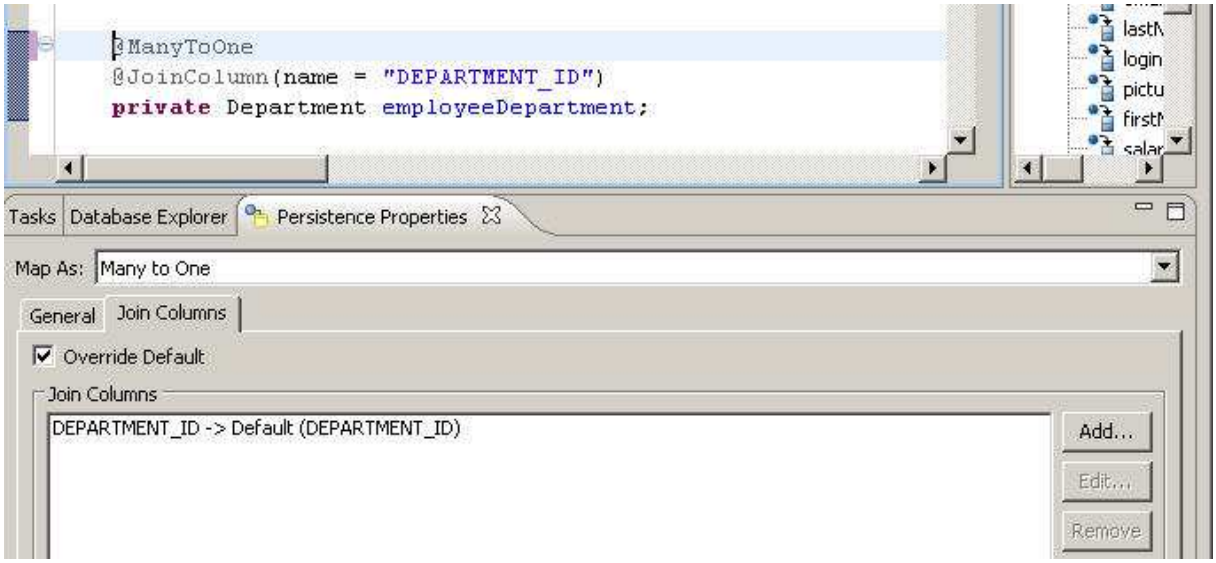
   a. Select the *deptManager* field from the *Persistence Outline* view.

   b. On the *General* tab page of the *Persistence Properties* view, enter the following data:

Property	**Value**
Map As	Many to one
Fetch Type	Lazy

   c. On the *Join Columns* tab page, enable the *Override Default* indicator, and choose *Add*. Select *DEPARTMENT_ID* from the *Name* dropdown list, leave *Referenced Column Name* empty, and choose *OK*.

   d. Remove the default entry.



This generates the following code for the *employeeDepartment* field:

```
@ManyToOne(fetch=FetchType.LAZY)
@JoinColumn(name="DEPARTMENT_ID")
private Department employeeDepartment;
```

3. Using the same procedure you can define the other relationships that Employee participates in. To input in the respective fields in the *Persistence Properties* view, use the tables below for the corresponding fields:

   - Persistent field *groups*

Property	**Value**
Tab Page: General	
Map As	Many to many
Fetch Type	Lazy
Tab Page: Join Table	
Name	RI_EDM_EMP_UGR
Join Columns	Name: EMPLOYEE_ID Referenced Column Name: EMPLOYEE_ID
Inverse Join Columns	Name: GROUP_ID Referenced Column Name: GROUP_ID
Generated Code	

`@ManyToMany(fetch=FetchType.LAZY)`
`@JoinTable(name="RI_EDM_EMP_UGR", joinColumns = @JoinColumn(name = "EMPLOYEE_ID", referencedColumnName = "EMPLOYEE_ID"), inverseJoinColumns = @JoinColumn(name = "GROUP_ID", referencedColumnName = "GROUP_ID"))`
`private List<UserGroup> groups = new ArrayList<UserGroup>();`	

   - Persistent field *skills*

Property	**Value**
Tab Page: General	
Map As	Many to many

Fetch Type	Eager
Tab Page: Join Table	
Name	RI_EDM_EMP_SKILL
Join Columns	Name: EMPLOYEE_ID Referenced Column Name: EMPLOYEE_ID
Inverse Join Columns	Name: SKILL_ID Referenced Column Name: SKILL_ID
Generated Code	

```
@ManyToMany(fetch=FetchType.EAGER)
@JoinTable(name="RI_EDM_EMP_SKILL", joinColumns = @JoinColumn(name =
"EMPLOYEE_ID", referencedColumnName = "EMPLOYEE_ID"),
inverseJoinColumns = @JoinColumn(name = "SKILL_ID",
referencedColumnName = "SKILL_ID"))
private List<Skill> skills = new ArrayList<Skill>();
```

- Persistent field *projectsLeadByEmployee*

Property	**Value**
Tab Page: General	
Map As	One to many
Fetch Type	Lazy
Mapped By	leader
Generated Code	

```
@OneToMany(mappedBy="leader", fetch = FetchType.LAZY)
private List<Project> projectsLeadByEmployee = new
ArrayList<Project>();
```

- Persistent field *projectsByParticipant*

Property	**Value**
Tab Page: General	
Map As	Many to many
Fetch Type	Lazy
Mapped By	employees
Generated Code	

```
@ManyToMany(mappedBy="employees", fetch = FetchType.LAZY)
private List<Project> projectsByParticipant = new
ArrayList<Project>();
```

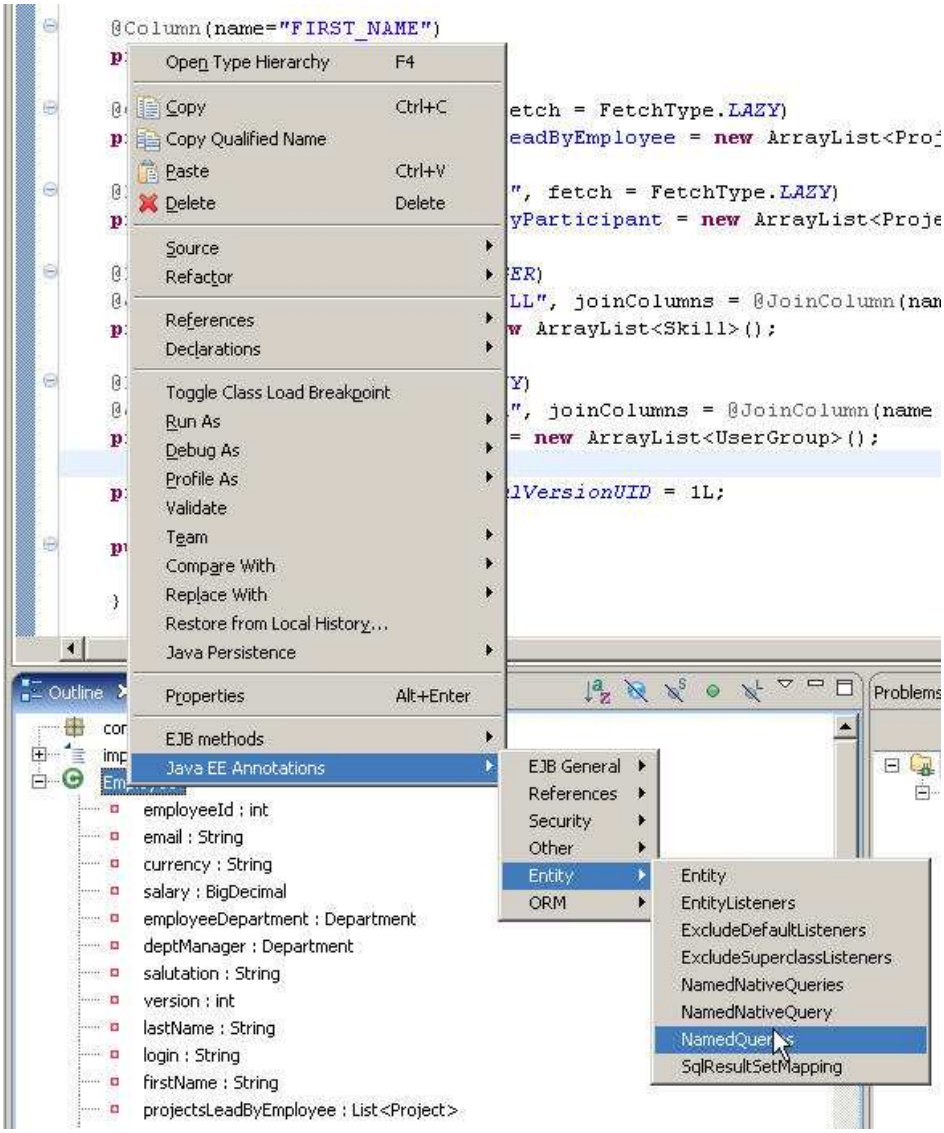
While creating the relationships in which Employee is the owning side, you can get error messages issued by the Developer Studio. The error messages disappear when you create the entity that is the inverse side of the relationship. To implement the other entities, copy the source codes from Reference [Page 44].

4. Defining Named Queries

4. Open the *Outline* view (*Window* → *Show view* → *Outline* from the main menu) and select *Employee*.

5. Choose *Java EE Annotations* → *Entity* → *NamedQueries* from the context menu for Employee.

   This generates the @NamedQueries annotation for the Employee class.



6. Enter the following as elements of the annotation:

Property	**Value**
Name	Employee.getAll
Query	SELECT e FROM Employee e

7. Repeat the same procedure to create the following named queries:

Name	**Query**
Employee.findByNamePart	SELECT e FROM Employee e WHERE lower(e.lastName) LIKE :namepart OR lower(e.firstName) LIKE :namepart1

Employee.findByLogin	SELECT e FROM Employee e WHERE e.login = :login

Developing the Session Beans

Use

This tutorial shows you how to implement the business logic layer of the reference application. It uses the ProjectManagement Session bean as an example. Using the the same procedure, you can implement the rest of the session beans copying the source codes from Reference [Page 44].

Procedure

To develop the ProjectManagementService bean, you need to go through the following steps:

1. Creating the ProjectManagementService Session Bean [Page 32]

   Using the session EJB wizard, you create the bean with its main requisites, such as session type and access type of the business interface.

2. Implementing the ProjectManagementService Session Bean [Page 33]

   a. Inject resources into the environment.

      You inject the persistence context to provide access to the EntityManager (EM) and use the EM's API to work with queries in the bean's business methods. You also inject a reference to another session bean, as well as make a session context available to the ProjectManagementService bean.

   b. Write queries to the database.

      You use the EM APIs to implement the bean's business methods that send queries to the database.

   c. Expose the business methods to the business interface.

      You use the Developer Studio tools to add the bean's methods to its local business interface.

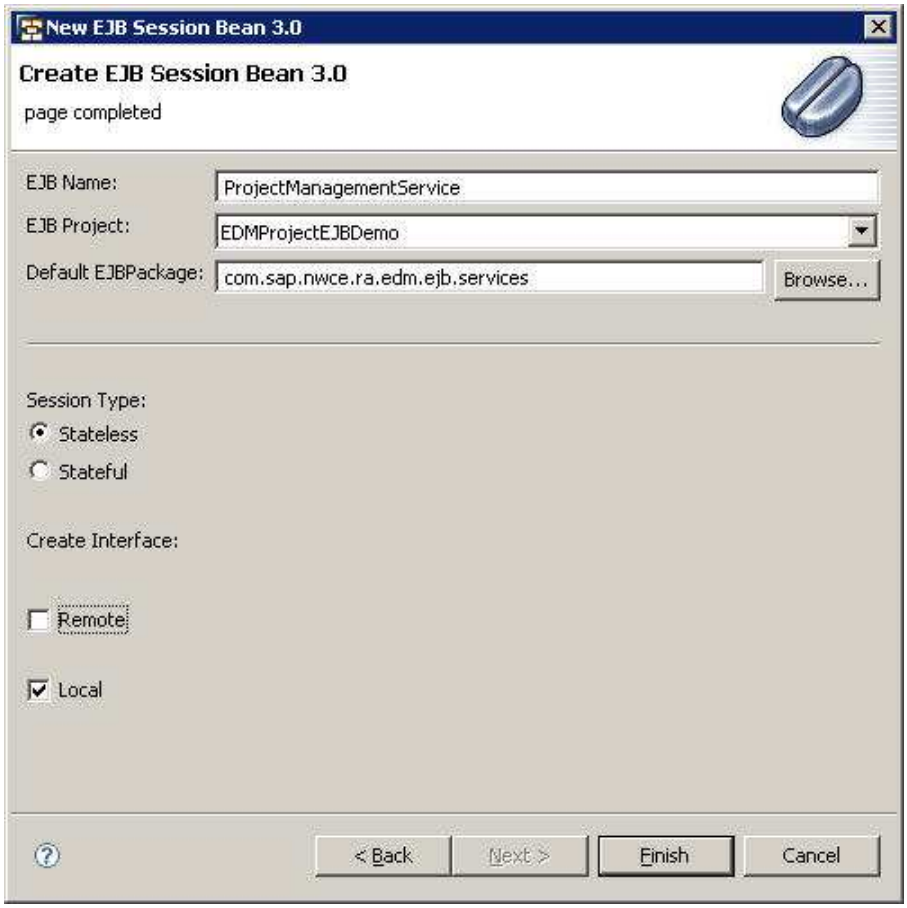
Creating the ProjectManagementService Session Bean

Prerequisites

You have opened the J2EE perspective in the SAP NetWeaver Developer Studio.

Procedure

1. Choose *File* → *New* → *Other* from the main menu.
2. Choose *EJB* → *EJB Session Bean 3.0* on the *Select a wizard* screen
3. On the *New EJB Session Bean 3.0* screen, proceed as follows:
   a. Enter **`ProjectManagementService`** in the *EJB Name* field.
   b. Enter **`com.sap.nwce.ra.edm.ejb.services`** in *Default EJBPackage*.
   c. Select *Stateless* for *Session Type*.
   d. Deselect *Remote* from *Create Interface*.



4. Choose *Finish* to complete the wizard.

Result

This creates the ProjectManagementService and the associated local business interface ProjectManagementServiceLocal. In addition, the bean is with container-managed transaction demarcation, which is the default demarcation for session beans. Therefore, you do not need to set the @Local and @TransactionManagement annotations explicitly.

Implementing the ProjectManagementService Session Bean

Procedure

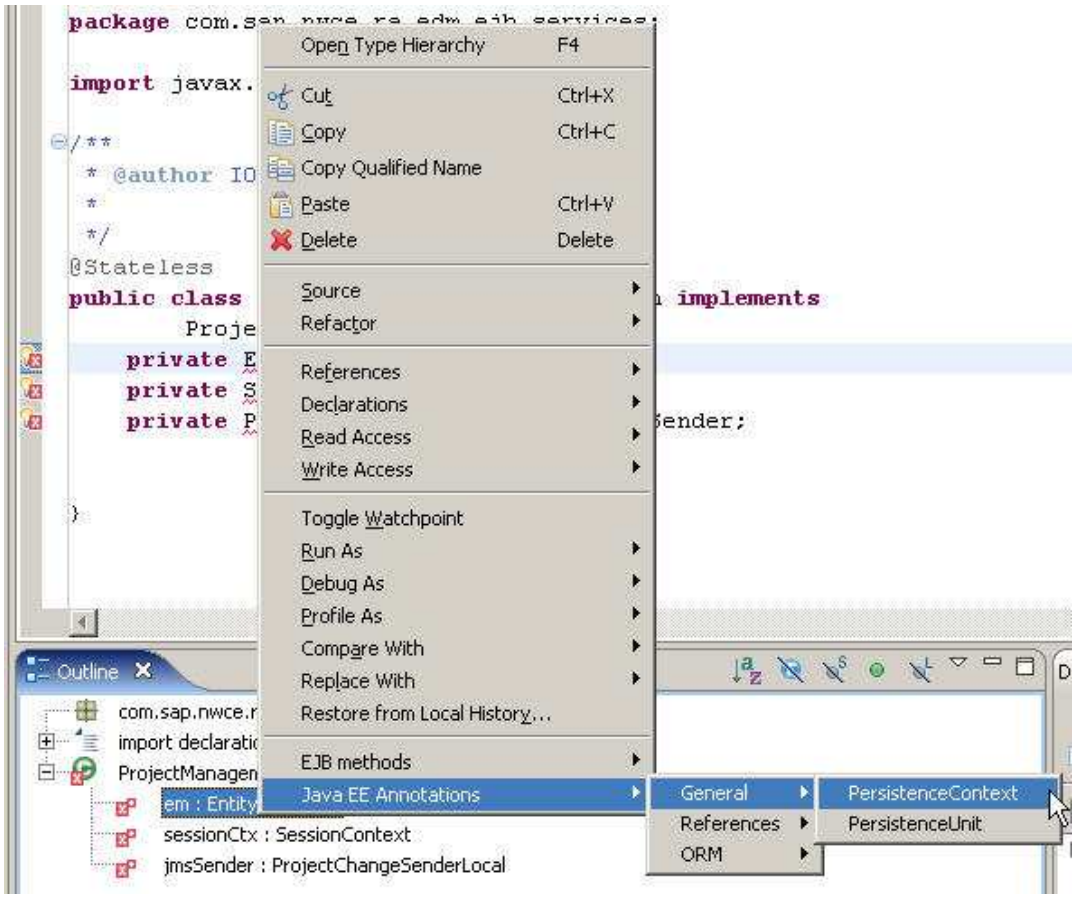
1. Injecting Resources to Bean's Environment

1. Open the ProjectManagementService bean in the source code editor and define the following variables that hold the injected resources:

```
private EntityManager em;
private SessionContext sessionCtx;
private ProjectChangeSenderLocal jmsSender;
```

2. To inject the persistence context, select the `em` variable in the *Outline* view and choose *Java EE Annotations* → *General* → *PersistenceContext* from the context menu.

   This generates the @PersistenceContext annotation.



3. Enter **`ems/EJB3_EDM_DEMO_PU`** as a value of the *unitName* attribute of the annotation.

4. Inject the remaining two resources using the same procedure.

Variable	Context Menu Path	Attribute Values
sessionCtx	*Java EE Annotations* → *References* → *Resource*	
jmsSender	*Java EE Annotations* → *References* → *EJB*	name="JMSSenderSessionLocal"

2. Querying the Database

Using the simplified EntityManager API, as well as the new named queries available in EJB3.0 query language, it is easy to implement business methods that query the database. For example, to find a project by its ID, you do not have to write your own query. EntityManager's `find()` method does that for you. You can also use EntityManager's `createNameQuery()` method to execute any of the named queries you defined in the Employee entity.

Use the following code to define and implement all methods of the ProjectManagementService bean:

```
/**
    * Get a project by its id.
    * @return project entity
    */

   public Project getProjectById(String pId){
      return em.find(Project.class, Integer.valueOf(pId));
   }

   /**
    * Get all projects in a single list.
    * @return list of all projects
    */
   @SuppressWarnings("unchecked")
   public List<Project> getAllProjects(){
      Query query = em.createNamedQuery("Project.getAll");
      List<Project> result = query.getResultList();
      return result;
   }

   /**
    * Get all assigned projects.
    * @return list of all assigned projects
    */
   @SuppressWarnings("unchecked")
   public List<Project> getAssignedProjects(int pEmployeeId){
      Query query = em.createNamedQuery("Project.getAllForEmployee");
      query.setParameter("employeeId", pEmployeeId);
      List<Project> result = query.getResultList();
      return result;
   }

   /**
    * Removes a project.
    * @param pProject
    */
   public void removeProject(Project pProject){
      em.remove(pProject);
   }

   /**
    * Persists a project (create or update) and sends a message to
    * the projectchangequeue to notify the system that the project
    * was changed
    */
   @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW)
   public Project createProject(Project pProject){
      Project result = em.merge(pProject);
      String caller = sessionCtx.getCallerPrincipal().getName();
      jmsSender.sendMessage(result, caller);
```

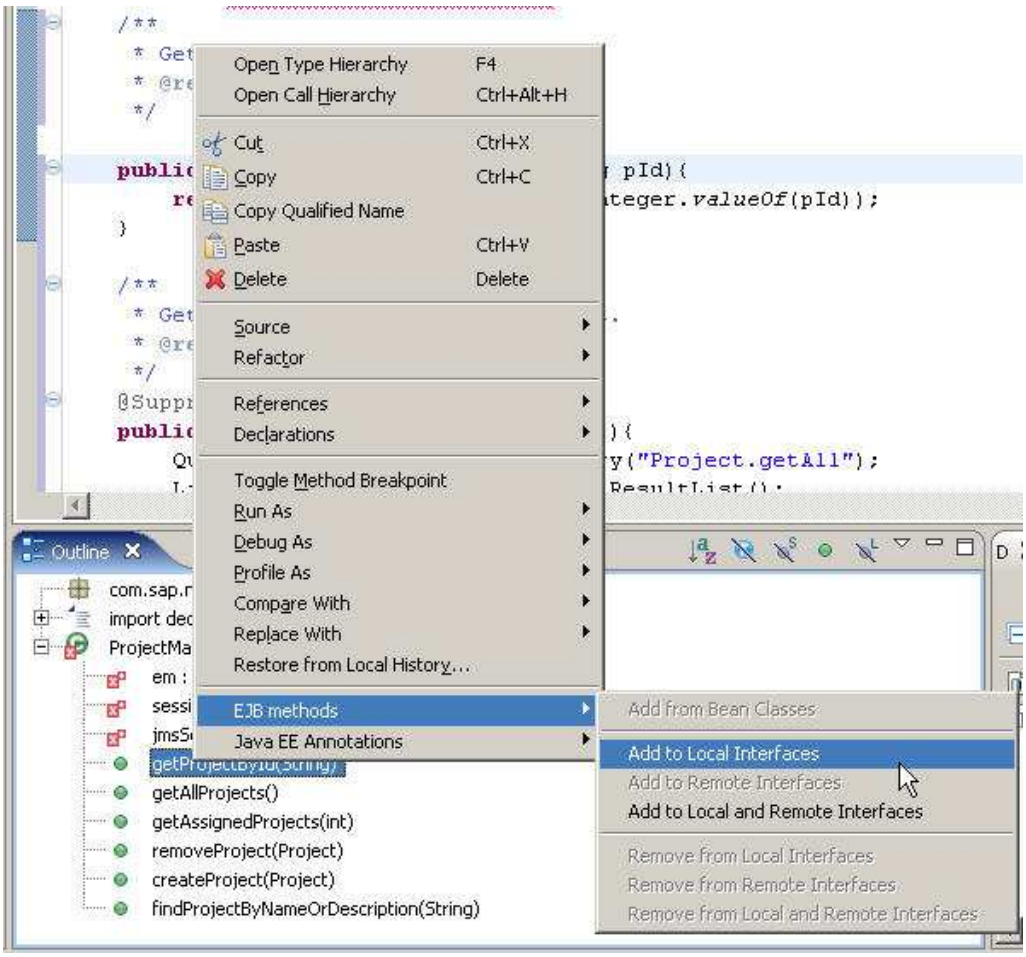
```
      return result;
   }

   /**
    * Get a list of projects by its name or description.
    * @param Name or description fragment
    * @return List of projects
    */
   @SuppressWarnings("unchecked")
   public List<Project> findProjectByNameOrDescription(String
pNameFragment){
      Query query =
em.createNamedQuery("Project.findByNameOrDescPart");
      String parameter = pNameFragment.toLowerCase();
      query.setParameter("namepart", "%"+parameter+"%");
      query.setParameter("namepart1", "%"+parameter+"%");
      List<Project> result = query.getResultList();
      return result;
   }
```

3. Exposing Methods to the Local Business Interface

1. Select, for example, the **`getProjectById()`** method of the bean in the *Outline* view.
2. To add it to the local business interface, choose *EJB Methods* → *Add to Local Interfaces*.

   This adds the method's definition to the business interface of the bean, thereby making it visible to other enterprise beans in the application



3. Repeat the same steps to expose all other methods of the ProjectManagementService bean to its business interface.

Result

The implementation of the bean should now look like the one shown in ProjectManagementService Source Code [Page 89].

Developing the Message Driven Beans

Use

In this part of the tutorial you learn how to implement the ProjectChangeReceiver message-driven bean that is used for consuming Java Messaging Service (JMS) messages and updating the project management history asynchronously. The application contains an additional message-driven bean – the ApplicationResetReceiver, which you can implement by copying its source code from the Reference [Page 44].

Procedure

To develop the bean, you go through the following steps:

1. Creating the ProjectChangeReceiver Bean [Page 38]

   Using the SAP NetWeaver Developer Studio tools, you create a message-driven bean in the EJB 3.0 project that you created.

2. Defining the Message Destination for the Bean [Page 39]

   You register the bean to a destination where the JMS messages are posted.

3. Implementing the ProjectChangeReceiver Bean [Page 40]

   a. You enable the bean to consume messages by making it implement the `javax.jms.MessageListener` interface.

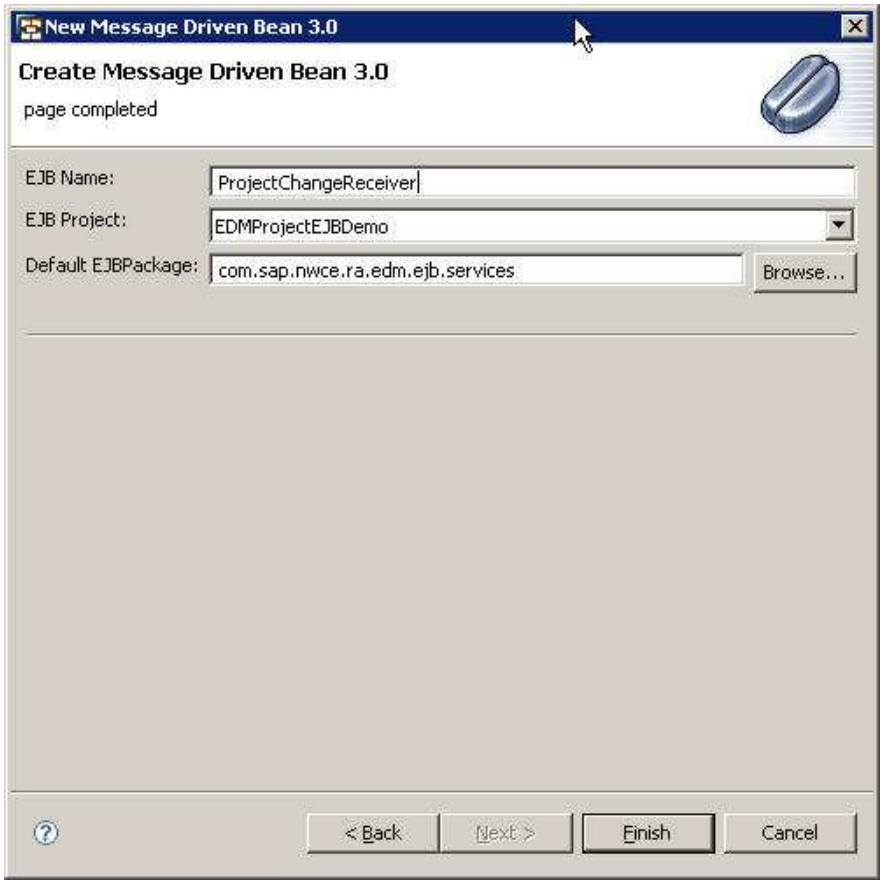
   b. You inject the required context variables.

   c. You implement the bean's `onMessage` method.

Creating the ProjectChangeReceiver Bean

Procedure

1. Choose *File* → *New* → *Other* from the main menu.
2. Choose *EJB* → *Message Driven Bean 3.0* on the *Select a wizard* screen
3. On the *New Message Driven Bean 3.0* screen, proceed as follows:
   a. Enter **`ProjectChangeReceiver`** in *EJB Name*.
   b. Enter **`EDMProjectEJBDemo`** in *EJB Project*.
   c. Enter **`com.sap.nwce.ra.edm.ejb.services`** in *Default EJBPackage*.



4. Choose *Finish* to create the bean.

Defining the Message Destination for the Bean

Procedure

1. Open the ProjectChangeReceiver bean in the Java editor.
2. To define the message driven bean properties that refer to configuring its environment, add the respective configuration properties to the `activationConfig` element of the `@MessageDriven` annotation:

```
@MessageDriven(activationConfig = {
      @ActivationConfigProperty(propertyName = "destinationType",
propertyValue = "javax.jms.Queue"),
      @ActivationConfigProperty(propertyName = "destination",
propertyValue = "prjChangeQueue"),
})
```

You complete the definition of the JMS destination used by the ProjectChangeReceiver bean later on using the *jms-resource.xml* file in the Enterprise Application project for the Project Management and Employee Services application.

Implementing the ProjectChangeReceiver Bean

Procedure

1. Specify that ProjectChangeReceiver implements the `javax.jms.MessageListener` interface:

```
public class ProjectChangeReceiver implements MessageListener {
```

2. Inject a *MessageDrivenContext* and a *PersistenceContext*:

```
@Resource
    public MessageDrivenContext mdc;
@PersistenceContext(unitName = "ems/EJB3_EDM_DEMO_PU")
```

3. Implement the bean's onMessage method as follows:

```
public void onMessage(Message inMessage) {
        TextMessage msg = null;

        try {
            if (inMessage instanceof TextMessage) {
                msg = (TextMessage) inMessage;
                Projectchanges chPrj = new Projectchanges();
                chPrj.setProjectId(msg.getIntProperty("ProjectId"));

chPrj.setEndDate(java.sql.Date.valueOf(msg.getStringProperty("EndDate")));

chPrj.setStartDate(java.sql.Date.valueOf(msg.getStringProperty("StartDate")));
                chPrj.setTitle(msg.getStringProperty("Title"));
                chPrj.setDescription(msg.getStringProperty("Description"));
                chPrj.setLeader(msg.getIntProperty("LeadId"));
                chPrj.setChangeDate(new
java.sql.Timestamp(msg.getLongProperty("ChangeDate")));
                chPrj.setStatus(msg.getIntProperty("Status"));
                chPrj.setChangerId(msg.getStringProperty("Changer"));
                em.merge(chPrj);
                logger.info("MESSAGE BEAN: Message received: " +
chPrj.getProjectId());
            } else {
                logger.warning(
                        "Message of wrong type: "
                        + inMessage.getClass().getName());
            }
        } catch (JMSException e) {
            logger.severe(
                    "MessageBean.onMessage: JMSException: " + e.toString());
            e.printStackTrace();
            mdc.setRollbackOnly();
        } catch (Throwable te) {
            logger.severe("MessageBean.onMessage: Exception: " +
te.toString());
            te.printStackTrace();
        }
    }
```

Result

The implementation of the bean should now look like the one shown in ProjectChangeReceiver Source Code [Page 82].

Deploying and Running the Application

Use

In the previous steps of the tutorial you have implemented the persistence and business logic layers of the Project Management and Employee Services application. To view the application at runtime, you need to import the prepackaged projects, deploy them to the application server, and access the Web layer using a browser.

Procedure

1. In the SAP NetWeaver Developer Studio, open Help → *Welcome.*

   This opens the Developer Studio welcome page.

2. Choose *Samples*.

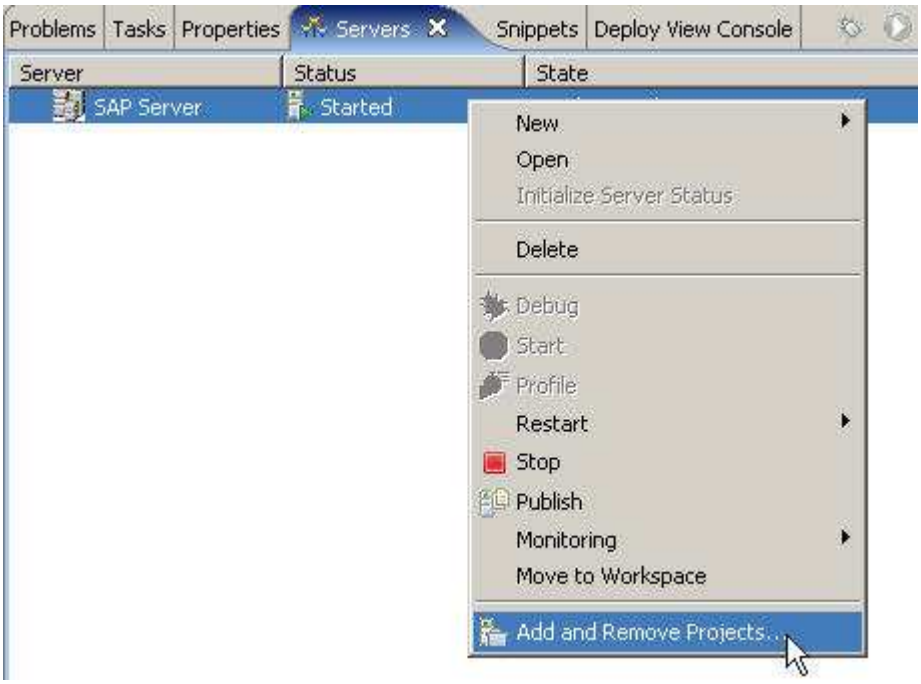
3. Click the Project Management and Employee Services icon.

   The application projects are automatically imported and displayed in the workbench.

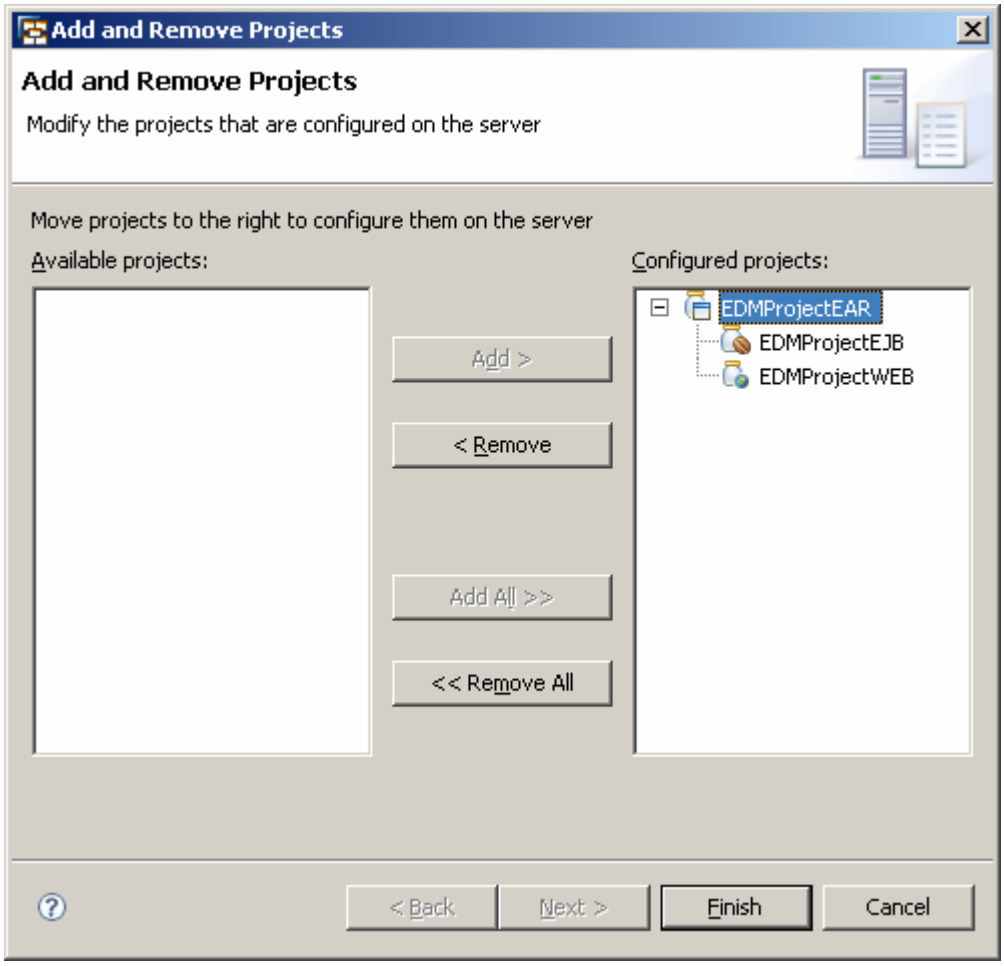
4. To open the *Servers* view, choose *Window* → *Show View* → *Servers* from the main menu.

   Make sure the server status is *Started*.

5. To add the project to the deployment list, choose *Add and Remove Projects* from the *SAP Server* context menu.



6. Select the *EDMProjectEAR* from the *Available projects* list and choose *Add* to add it to the *Configured projects* list.

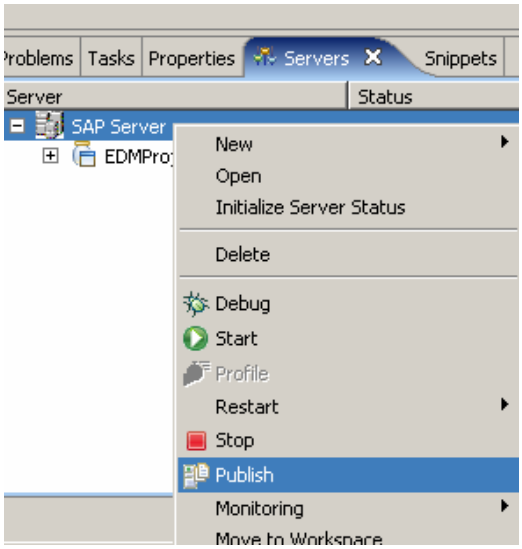


7. Choose *Finish*.

   The application deployment starts automatically and you might be requested to enter credentials for authenticating to the server. Enter the Administrator username and password.

   You get a confirmation message if the deployment is successful.

8. If you need to redeploy the application to the server, choose *Publish* from the *SAP Server* context menu.



9. To run the application, double click the *EDMProjectWEB* alias in the *SAP Microsoft Management Console*. For more information, see Launching Deployed Applications from the SAP MMC [External].

Reference

In this section you can find the source codes of all EJB 3.0 session and entity beans used in the reference aplication:

- Entity beans:
  - Department [Page 45]
  - Employee [Page 48]
  - Navigation [Page 54]
  - Project [Page 57]
  - ProjectChanges [Page 59]
  - Skill [Page 63]
  - UserGroup [Page 67]
- Session beans:
  - DataResetService [Page 76]
  - EmployeeManagementService [Page 80]
  - ProjectChangeSender [Page 90]
  - ProjectManagementService [Page 89]
- Message-driven beans:
  - ProjectChangeReceiver [Page 82]
  - ApplicationResetReceiver [Page 72]

Department Source Code

```
/*
 * Copyright (c) 2006 SAP AG, Germany. All rights reserved.
 * SAP C42 Reference Implementation. Use is subject to license terms.
 */
package com.sap.nwce.ra.edm.ejb.entity;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToMany;
import javax.persistence.OneToOne;
import javax.persistence.Table;
import javax.persistence.Version;

@NamedQueries({
    @NamedQuery(name="Department.getAll", query="SELECT d FROM Department d")
})
@Entity
@Table(name="RI_EDM_DEPARTMENT")
public class Department implements Serializable {
    @Id
    @Column(name="DEPARTMENT_ID")
    private String departmentId;

    private String name;

    @Version
```

```
    private int version;

    @OneToOne(cascade=CascadeType.ALL)
    @JoinColumn(name="MANAGER_ID", updatable = false)
    private Employee manager;

    @OneToMany(fetch=FetchType.LAZY, cascade = CascadeType.ALL, mappedBy =
"employeeDepartment")
    private List<Employee> employees = new ArrayList<Employee>();

    private static final long serialVersionUID = 1L;

    public Department() {
        super();
    }

    public String getDepartmentId() {
        return this.departmentId;
    }

    public void setDepartmentId(String departmentId) {
        this.departmentId = departmentId;
    }

    public String getName() {
        return this.name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getVersion() {
        return this.version;
    }

    public void setVersion(int version) {
        this.version = version;
    }
```

```
    public Employee getManager() {
        return manager;
    }

    public void setManager(Employee manager) {
        this.manager = manager;
    }

    public List<Employee> getEmployees() {
        return employees;
    }

    public void setEmployees(List<Employee> employees) {
        this.employees = employees;
    }

  public void addEmployee(Employee e) {
     employees.add(e);
     e.setEmployeeDepartment(this);
  }

  public void removeEmployee(Employee e) {
     if (employees.remove(e)) {
        e.setEmployeeDepartment(null);
     }
  }
}
```

Employee Source Code

```
/*
 * Copyright (c) 2006 SAP AG, Germany. All rights reserved.
 * SAP C42 Reference Implementation. Use is subject to license terms.
 */
package com.sap.nwce.ra.edm.ejb.entity;

import java.io.Serializable;
import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.List;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.ManyToOne;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToMany;
import javax.persistence.OneToOne;
import javax.persistence.Table;
import javax.persistence.TableGenerator;
import javax.persistence.Version;

@NamedQueries({
    @NamedQuery(name="Employee.getAll", query="SELECT e FROM Employee e"),
    @NamedQuery(name="Employee.findByNamePart", query="SELECT e FROM
Employee e WHERE lower(e.lastName) LIKE :namepart OR lower(e.firstName) LIKE
:namepart1"),
    @NamedQuery(name="Employee.findByLogin", query="SELECT e FROM Employee e
WHERE e.login = :login")
})
```

```
@Entity
@Table(name="RI_EDM_EMPLOYEE")
public class Employee implements Serializable {
    @Id
    @Column(name="EMPLOYEE_ID")
    @GeneratedValue(strategy=GenerationType.TABLE, generator = "IdGenerator")
    @TableGenerator(name="IdGenerator", table = "RI_EDM_GENERATOR",
pkColumnName = "BEAN_NAME", valueColumnName = "MAX_ID")
    private int employeeId;

    private String email;

    private String currency;

    private BigDecimal salary;

  @ManyToOne(fetch=FetchType.LAZY)
    @JoinColumn(name="DEPARTMENT_ID")
    private Department employeeDepartment;

  @OneToOne(mappedBy="manager")
    private Department deptManager;

  private String salutation;

  @Version
    private int version;

    @Column(name="LAST_NAME")
    private String lastName;

    private String login;

    @Column(name="FIRST_NAME")
    private String firstName;

  @OneToMany(mappedBy="leader", fetch = FetchType.LAZY)
    private List<Project> projectsLeadByEmployee = new ArrayList<Project>();

  @ManyToMany(mappedBy="employees", fetch = FetchType.LAZY)
```

```
    private List<Project> projectsByParticipant = new ArrayList<Project>();


  @ManyToMany(fetch=FetchType.EAGER)
    @JoinTable(name="RI_EDM_EMP_SKILL", joinColumns = @JoinColumn(name =
"EMPLOYEE_ID", referencedColumnName = "EMPLOYEE_ID"), inverseJoinColumns =
@JoinColumn(name = "SKILL_ID", referencedColumnName = "SKILL_ID"))
    private List<Skill> skills = new ArrayList<Skill>();


  @ManyToMany(fetch=FetchType.LAZY)
    @JoinTable(name="RI_EDM_EMP_UGR", joinColumns = @JoinColumn(name =
"EMPLOYEE_ID", referencedColumnName = "EMPLOYEE_ID"), inverseJoinColumns =
@JoinColumn(name = "GROUP_ID", referencedColumnName = "GROUP_ID"))
    private List<UserGroup> groups = new ArrayList<UserGroup>();


    private static final long serialVersionUID = 1L;


    public Employee() {
        super();
    }


    public int getEmployeeId() {
        return this.employeeId;
    }


    public void setEmployeeId(int employeeId) {
        this.employeeId = employeeId;
    }


    public String getEmail() {
        return this.email;
    }


    public void setEmail(String email) {
        this.email = email;
    }


    public String getCurrency() {
        return this.currency;
    }


    public void setCurrency(String currency) {
```

```
        this.currency = currency;
    }

    public BigDecimal getSalary() {
        return this.salary;
    }

    public void setSalary(BigDecimal salary) {
        this.salary = salary;
    }

    public String getSalutation() {
        return this.salutation;
    }

    public void setSalutation(String salutation) {
        this.salutation = salutation;
    }

    public int getVersion() {
        return this.version;
    }

    public void setVersion(int version) {
        this.version = version;
    }

    public String getLastName() {
        return this.lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getLogin() {
        return this.login;
    }
```

```
    public void setLogin(String login) {
        this.login = login;
    }

    public String getFirstName() {
        return this.firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public Department getDeptManager() {
        return deptManager;
    }

    public void setDeptManager(Department deptManager) {
        this.deptManager = deptManager;
    }

    public Department getEmployeeDepartment() {
        return employeeDepartment;
    }

    public void setEmployeeDepartment(Department employeeDepartment) {
        this.employeeDepartment = employeeDepartment;
    }

    public List<UserGroup> getGroups() {
        return groups;
    }

    public void setGroups(List<UserGroup> groups) {
        this.groups = groups;
    }

    public List<Project> getProjectsByParticipant() {
        return projectsByParticipant;
    }
```

```
    public void setProjectsByParticipant(List<Project> projectsByParticipant) {
        this.projectsByParticipant = projectsByParticipant;
    }

    public List<Project> getProjectsLeadByEmployee() {
        return projectsLeadByEmployee;
    }

    public void setProjectsLeadByEmployee(List<Project> projectsLeadByEmployee) {
        this.projectsLeadByEmployee = projectsLeadByEmployee;
    }

    public List<Skill> getSkills() {
        return skills;
    }

    public void setSkills(List<Skill> skills) {
        this.skills = skills;
    }

}
```

Navigation Source Code

```
/*
 * Copyright (c) 2006 SAP AG, Germany. All rights reserved.
 * SAP C42 Reference Implementation. Use is subject to license terms.
 */
package com.sap.nwce.ra.edm.ejb.entity;

import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.Version;

@Entity
@Table(name="RI_EDM_NAVIGATION")
public class Navigation implements Serializable {
    @Id
    @Column(name="NAVIGATION_ID")
    private String navigationId;

    private int position;

    @Version
    private int version;

    private String label;

    private String href;

    private static final long serialVersionUID = 1L;

    public Navigation() {
        super();
    }

    public String getNavigationId() {
```

```
        return this.navigationId;
    }

    public void setNavigationId(String navigationId) {
        this.navigationId = navigationId;
    }

    public int getPosition() {
        return this.position;
    }

    public void setPosition(int position) {
        this.position = position;
    }

    public int getVersion() {
        return this.version;
    }

    public void setVersion(int version) {
        this.version = version;
    }

    public String getLabel() {
        return this.label;
    }

    public void setLabel(String label) {
        this.label = label;
    }

    public String getHref() {
        return this.href;
    }

    public void setHref(String href) {
        this.href = href;
    }
```

```
}
```

Project Source Code

```
/*
 * Copyright (c) 2006 SAP AG, Germany. All rights reserved.
 * SAP C42 Reference Implementation. Use is subject to license terms.
 */
package com.sap.nwce.ra.edm.ejb.entity;

import java.io.Serializable;
import java.sql.Date;
import java.util.ArrayList;
import java.util.List;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.ManyToOne;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;
import javax.persistence.TableGenerator;
import javax.persistence.Version;

@NamedQueries({
    @NamedQuery(name="Project.getAll", query="SELECT p FROM Project p"),
    @NamedQuery(name="Project.getAllForEmployee", query="SELECT p FROM Project
p where p.leader.employeeId = :employeeId"),
    @NamedQuery(name="Project.findByNameOrDescPart", query="SELECT p FROM
Project p WHERE lower(p.title) LIKE :namepart OR lower(p.description) LIKE :namepart1")
})
@Entity
@Table(name="RI_EDM_PROJECT")
```

```
public class Project implements Serializable {
    @Id
    @Column(name="PROJECT_ID")
    @GeneratedValue(strategy=GenerationType.TABLE, generator = "IdGenerator")
    @TableGenerator(name="IdGenerator", table = "RI_EDM_GENERATOR",
pkColumnName = "BEAN_NAME", valueColumnName = "MAX_ID")
    private int projectId;

  @ManyToOne
    @JoinColumn(name="LEAD_ID")
    private Employee leader;

    private int status;

    private String description;

    @Column(name="START_DATE")
    private Date startDate;

    private String title;

  @ManyToMany(fetch=FetchType.EAGER)
    @JoinTable(name="RI_EDM_EMP_PRJ", joinColumns = @JoinColumn(name =
"PROJECT_ID", referencedColumnName = "PROJECT_ID"), inverseJoinColumns =
@JoinColumn(name = "EMPLOYEE_ID", referencedColumnName = "EMPLOYEE_ID"))
    private List<Employee> employees = new ArrayList<Employee>();

  @ManyToMany(fetch=FetchType.EAGER)
    @JoinTable(name="RI_EDM_PRJ_SKILL", joinColumns = @JoinColumn(name =
"PROJECT_ID", referencedColumnName = "PROJECT_ID"), inverseJoinColumns =
@JoinColumn(name = "SKILL_ID", referencedColumnName = "SKILL_ID"))
    private List<Skill> skills = new ArrayList<Skill>();

  @Version
    private int version;

    @Column(name="END_DATE")
    private Date endDate;

    private static final long serialVersionUID = 1L;
```

```
    public Project() {
        super();
    }

    public int getProjectId() {
        return this.projectId;
    }

    public void setProjectId(int projectId) {
        this.projectId = projectId;
    }

    public int getStatus() {
        return this.status;
    }

    public void setStatus(int status) {
        this.status = status;
    }

    public String getDescription() {
        return this.description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public Date getStartDate() {
        return this.startDate;
    }

    public void setStartDate(Date startDate) {
        this.startDate = startDate;
    }

    public String getTitle() {
        return this.title;
    }
```

```
public void setTitle(String title) {
    this.title = title;
}

public int getVersion() {
    return this.version;
}

public void setVersion(int version) {
    this.version = version;
}

public Date getEndDate() {
    return this.endDate;
}

public void setEndDate(Date endDate) {
    this.endDate = endDate;
}

public List<Employee> getEmployees() {
    return employees;
}

public void setEmployees(List<Employee> employees) {
    this.employees = employees;
}

public Employee getLeader() {
    return leader;
}

public void setLeader(Employee leader) {
    this.leader = leader;
}

public List<Skill> getSkills() {
    return skills;
```

```
    }

    public void setSkills(List<Skill> skills) {
        this.skills = skills;
    }

}
```

ProjectChanges Source Code

```
package com.sap.nwce.ra.edm.ejb.entity;

import java.io.Serializable;
import java.sql.Date;
import java.sql.Timestamp;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.TableGenerator;
import javax.persistence.Version;

@Entity
@Table(name="RI_EDM_PRJ_HISTORY")
public class Projectchanges implements Serializable {
    @Id
    @TableGenerator(name="IdGenerator", table="RI_EDM_GENERATOR",
pkColumnName="BEAN_NAME", valueColumnName="MAX_ID")
  @GeneratedValue(strategy=GenerationType.TABLE, generator="IdGenerator")
    private int id;

    @Column(name="END_DATE")
    private Date endDate;

    @Version
    private int version;

    private String description;

    private String title;

    private Timestamp changeDate;
```

```
    private int status;

    @Column(name="LEAD_ID")
    private int leader;

    @Column(name="PROJECT_ID")
    private int projectId;

    @Column(name="CHANGER_ID")
    private String changerId;

    @Column(name="START_DATE")
    private Date startDate;

    private static final long serialVersionUID = 1L;

    public Projectchanges() {
        super();
    }

    public int getId() {
        return this.id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public Date getEndDate() {
        return this.endDate;
    }

    public void setEndDate(Date endDate) {
        this.endDate = endDate;
    }

    public int getVersion() {
        return this.version;
    }
```

```
public void setVersion(int version) {
    this.version = version;
}

public String getDescription() {
    return this.description;
}

public void setDescription(String description) {
    this.description = description;
}

public String getTitle() {
    return this.title;
}

public void setTitle(String title) {
    this.title = title;
}

public Timestamp getChangeDate() {
    return this.changeDate;
}

public void setChangeDate(Timestamp changeDate) {
    this.changeDate = changeDate;
}

public int getStatus() {
    return this.status;
}

public void setStatus(int status) {
    this.status = status;
}

public int getProjectId() {
    return this.projectId;
```

```
    }

    public void setProjectId(int projectId) {
        this.projectId = projectId;
    }

    public String getChangerId() {
        return this.changerId;
    }

    public void setChangerId(String changerId) {
        this.changerId = changerId;
    }

    public Date getStartDate() {
        return this.startDate;
    }

    public void setStartDate(Date startDate) {
        this.startDate = startDate;
    }

    public int getLeader() {
        return leader;
    }

    public void setLeader(int leader) {
        this.leader = leader;
    }

}
```

Skill Source Code

```
/*
 * Copyright (c) 2006 SAP AG, Germany. All rights reserved.
 * SAP C42 Reference Implementation. Use is subject to license terms.
 */
package com.sap.nwce.ra.edm.ejb.entity;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;
import javax.persistence.TableGenerator;
import javax.persistence.Version;

@NamedQueries({
    @NamedQuery(name="Skill.getAll", query="SELECT s FROM Skill s"),
    @NamedQuery(name="Skill.findByNamePart", query="SELECT b FROM Skill b
WHERE lower(b.name) LIKE :namepart")
})
@Entity
@Table(name="RI_EDM_SKILL")
public class Skill implements Serializable {
    @Id
    @Column(name="SKILL_ID")
    @GeneratedValue(strategy=GenerationType.TABLE, generator = "IdGenerator")
    @TableGenerator(name="IdGenerator", table = "RI_EDM_GENERATOR",
pkColumnName = "BEAN_NAME", valueColumnName = "MAX_ID")
    private int skillId;

    @Version
    private int version;
```

```
@Column(name="LANGUAGE_ID")
private String language;

private String name;

private static final long serialVersionUID = 1L;

public Skill() {
    super();
}

public int getSkillId() {
    return this.skillId;
}

public void setSkillId(int skillId) {
    this.skillId = skillId;
}

public int getVersion() {
    return this.version;
}

public void setVersion(int version) {
    this.version = version;
}

public String getName() {
    return this.name;
}

public void setName(String name) {
    this.name = name;
}

public String getLanguage() {
    return language;
}
```

```
    public void setLanguage(String language) {
        this.language = language;
    }

}
```

UserGroup Source Code

```
/*
 * Copyright (c) 2006 SAP AG, Germany. All rights reserved.
 * SAP C42 Reference Implementation. Use is subject to license terms.
 */
package com.sap.nwce.ra.edm.ejb.entity;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.ManyToMany;
import javax.persistence.FetchType;
import javax.persistence.JoinTable;
import javax.persistence.JoinColumn;
import javax.persistence.Version;

@Entity
@Table(name="RI_EDM_USERGROUP")
public class UserGroup implements Serializable {
    @Id
    @Column(name="GROUP_ID")
    private String groupId;

    @Version
    private int version;

    private String groupname;

  @ManyToMany(fetch=FetchType.EAGER)
    @JoinTable(name="RI_EDM_UGR_NAV", joinColumns = @JoinColumn(name =
"GROUP_ID", referencedColumnName = "GROUP_ID"), inverseJoinColumns =
@JoinColumn(name = "NAVIGATION_ID", referencedColumnName = "NAVIGATION_ID"))
```

```
    private List<Navigation> navigations = new ArrayList<Navigation>();

    private static final long serialVersionUID = 1L;

    public UserGroup() {
        super();
    }

    public String getGroupId() {
        return this.groupId;
    }

    public void setGroupId(String groupId) {
        this.groupId = groupId;
    }

    public int getVersion() {
        return this.version;
    }

    public void setVersion(int version) {
        this.version = version;
    }

    public String getGroupname() {
        return this.groupname;
    }

    public void setGroupname(String groupname) {
        this.groupname = groupname;
    }

    public List<Navigation> getNavigations() {
        return navigations;
    }

    public void setNavigations(List<Navigation> navigations) {
        this.navigations = navigations;
    }
```

```
}
```

ApplicationResetReceiver Source Code

```
package com.sap.nwce.ra.edm.ejb.services;

/*
 * Copyright (c) 2006 Sun Microsystems, Inc.  All rights reserved.  U.S.
 * Government Rights - Commercial software.  Government users are subject
 * to the Sun Microsystems, Inc. standard license agreement and
 * applicable provisions of the FAR and its supplements.  Use is subject
 * to license terms.
 *
 * This distribution may include materials developed by third parties.
 * Sun, Sun Microsystems, the Sun logo, Java and J2EE are trademarks
 * or registered trademarks of Sun Microsystems, Inc. in the U.S. and
 * other countries.
 *
 * Copyright (c) 2006 Sun Microsystems, Inc. Tous droits reserves.
 *
 * Droits du gouvernement americain, utilisateurs gouvernementaux - logiciel
 * commercial. Les utilisateurs gouvernementaux sont soumis au contrat de
 * licence standard de Sun Microsystems, Inc., ainsi qu'aux dispositions
 * en vigueur de la FAR (Federal Acquisition Regulations) et des
 * supplements a celles-ci.  Distribue par des licences qui en
 * restreignent l'utilisation.
 *
 * Cette distribution peut comprendre des composants developpes par des
 * tierces parties. Sun, Sun Microsystems, le logo Sun, Java et J2EE
 * sont des marques de fabrique ou des marques deposees de Sun
 * Microsystems, Inc. aux Etats-Unis et dans d'autres pays.
 */

import javax.annotation.Resource;
import javax.ejb.ActivationConfigProperty;
import javax.ejb.EJB;
import javax.ejb.EJBException;
import javax.ejb.MessageDriven;
import javax.ejb.MessageDrivenContext;
import javax.jms.Connection;
```

```
import javax.jms.ConnectionFactory;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageListener;
import javax.jms.MessageProducer;
import javax.jms.ObjectMessage;
import javax.jms.Session;
import javax.jms.TemporaryQueue;
import javax.jms.TextMessage;

import com.sap.nwce.ra.edm.util.DataResetResult;
import com.sap.nwce.ra.edm.util.ReferenceApplicationEJBLogger;



/**
 * The MessageBean class is a message-driven bean.  It implements
 * the javax.jms.MessageListener interface. It is defined as public
 * (but not final or abstract).
 */
@MessageDriven(
          activationConfig =  {
          @ActivationConfigProperty(propertyName = "destinationType", propertyValue =
"javax.jms.Queue"),
        @ActivationConfigProperty(propertyName = "destination", propertyValue =
"appResetQueue"),
        @ActivationConfigProperty(propertyName = "connectionFactoryName",
propertyValue = "EDMQueueConnectionFactory")
})

public class ApplicationResetReceiverBean implements MessageListener {


   @Resource
   public MessageDrivenContext mdc;


   @EJB private DataResetServiceLocal dataServiceFacade;


     @Resource(name="EDMQueueConnectionFactory")
     private ConnectionFactory queueConnectionFactory;
```

```
    /**
     * Constructor, which is public and takes no arguments.
     */
    public ApplicationResetReceiverBean() {
    }


    public void setMessageDrivenContext(MessageDrivenContext ctx) throws
EJBException{
      this.mdc = ctx;
    }


    /**
     * onMessage method, declared as public (but not final or
     * static), with a return type of void, and with one argument
     * of type javax.jms.Message.
     *
     * Casts the incoming Message to a TextMessage and displays
     * the text.
     *
     * @param inMessage    the incoming message
     */
    public void onMessage(Message inMessage) {
        TextMessage msg = null;
        DataResetResult dataResetResult;
        try {
            if (inMessage instanceof TextMessage) {
                msg = (TextMessage) inMessage;
                ReferenceApplicationEJBLogger.logInfo("MESSAGE BEAN: Message received: "
+ msg.getText());
                if (msg.getText().equals("reset")){
                   dataResetResult = dataServiceFacade.resetContent();
                   processNotification((TemporaryQueue) msg.getJMSReplyTo(),
dataResetResult);
                }else{
                   ReferenceApplicationEJBLogger.logInfo("MESSAGE BEAN: Message received:
" + msg.getText() + " but nothing to do!");
                }
            } else {
             ReferenceApplicationEJBLogger.logInfo("Message of wrong type: " +
inMessage.getClass().getName());
            }
```

```
      } catch (JMSException jmsExc) {
          ReferenceApplicationEJBLogger.logException(jmsExc);
         mdc.setRollbackOnly();
      } catch (Throwable te) {
          ReferenceApplicationEJBLogger.logThrowable(te);
      }
   }



    private void processNotification(TemporaryQueue queue, DataResetResult result)
throws JMSException{
          Session jmsSession = null;
          MessageProducer queueSender = null;
          Connection connection = null;


          try {
               connection = queueConnectionFactory.createConnection();
               jmsSession = connection.createSession(false,
Session.AUTO_ACKNOWLEDGE);
               queueSender = jmsSession.createProducer(queue);
               ObjectMessage msg = jmsSession.createObjectMessage();
               msg.setObject(result);
               queueSender.send(msg);
          } finally {
        if (connection != null) {
          connection.close();
        }
          }
     }
}
```

DataResetService Source Code

```
/**
 *
 */
package com.sap.nwce.ra.edm.ejb.services;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;

import javax.annotation.Resource;
import javax.ejb.EJBException;
import javax.ejb.SessionContext;
import javax.ejb.Stateless;
import javax.ejb.TransactionAttribute;
import javax.ejb.TransactionAttributeType;
import javax.ejb.TransactionManagement;
import javax.ejb.TransactionManagementType;
import javax.sql.DataSource;

import com.sap.nwce.ra.edm.util.DataResetResult;
import com.sap.nwce.ra.edm.util.PackageResolver;
import com.sap.nwce.ra.edm.util.ReferenceApplicationEJBLogger;

/**
 * @author D042633
 *
 */
@Stateless
@TransactionManagement(value=TransactionManagementType.CONTAINER)
@TransactionAttribute(TransactionAttributeType.NOT_SUPPORTED)
public class DataResetServiceBean implements DataResetServiceLocal {
```

```
  @Resource
  public SessionContext sc;

  @Resource(name="SAPDEMO_DS")
  private DataSource _dataSource;

  private static final String INSERT_SQLFILE = "update_application_data.sql";
  private static final String DELETE_SQLFILE = "delete_application_data.sql";


  private Connection getConnection() throws SQLException {
      return _dataSource.getConnection();
  }

public void setSessionContext(SessionContext ctx) throws EJBException{
  this.sc = ctx;
}


  @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW)
  @SuppressWarnings("finally")
  public DataResetResult resetContent() {

      Connection connection = null;
      DataResetResult dataresult = new DataResetResult();
      try {
          connection = this.getConnection();
          deleteAllContent(connection);
          importContent(connection);
          dataresult.setResultStatus(true);
      } catch (SQLException sqlExc){
          ReferenceApplicationEJBLogger.logException(sqlExc);
          dataresult.setResultStatus(false);
          dataresult.setErrorMessage(prepearErrorStatement(sqlExc.getMessage()));
          sc.setRollbackOnly();
      } catch (IOException ioExc) {
          ReferenceApplicationEJBLogger.logException(ioExc);
          dataresult.setResultStatus(false);
          dataresult.setErrorMessage(prepearErrorStatement(ioExc.getMessage()));
```

```
            sc.setRollbackOnly();
        } finally {
            try{
                connection.close();
            }catch (SQLException ioExc){
                ReferenceApplicationEJBLogger.logException(ioExc);
            }
            return dataresult;
        }
    }

    private void deleteAllContent(Connection connection) throws SQLException,
IOException {

        Statement statement = connection.createStatement();
        final InputStream in =
this.getClass().getClassLoader().getResourceAsStream(PackageResolver.resolvePackage(
this, DELETE_SQLFILE));

        if(in != null) {
            BufferedReader reader = new BufferedReader ( new InputStreamReader ( in ));
            String line;
            while(((line=reader.readLine()) != null)) {
                String readedString = line.toString();
                if(readedString.toUpperCase().indexOf("DELETE") == 0){
                    statement.execute(readedString);
                }
            }
            reader.close();
        }else{
            throw new IOException("The requested delete file is not avaible. Please check
name and location");
        }
        statement.close();
        in.close();

    }

    private void importContent(Connection connection) throws SQLException,
IOException {
        Statement statement = connection.createStatement();
```

```
      final InputStream in =
this.getClass().getClassLoader().getResourceAsStream(PackageResolver.resolvePackage(
this, INSERT_SQLFILE));


          if(in != null) {
               BufferedReader reader = new BufferedReader ( new InputStreamReader ( in ));
               String line;
               while(((line=reader.readLine()) != null)) {
                    String readedString = line.toString();
                    if(readedString.toUpperCase().indexOf("INSERT") == 0 ||
readedString.toUpperCase().indexOf("UPDATE") == 0){
                         statement.execute(readedString);
                    }
               }
               reader.close();
          }else{
               throw new IOException("The requested insert/update file is not avaible. Please
check name and location");
          }
          statement.close();
          in.close();


     }


     private String prepearErrorStatement(String exceptionMessage){
          final StringBuffer errorMessage = new StringBuffer();
          errorMessage.append("Error during data reset. Contact your administrator to
handle the exception. Error detail message is: \n\r " );
          errorMessage.append(exceptionMessage);


          return errorMessage.toString();
     }
}
```

EmployeeManagementService Source Code

```
/*
 * Copyright (c) 2006 SAP AG, Germany. All rights reserved.
 * SAP C42 Reference Implementation. Use is subject to license terms.
 */
package com.sap.nwce.ra.edm.ejb.services;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

import javax.annotation.Resource;
import javax.ejb.Local;
import javax.ejb.SessionContext;
import javax.ejb.Stateless;
import javax.ejb.TransactionAttribute;
import javax.ejb.TransactionAttributeType;
import javax.ejb.TransactionManagement;
import javax.ejb.TransactionManagementType;
import javax.persistence.EntityManager;
import javax.persistence.NoResultException;
import javax.persistence.PersistenceContext;
import javax.persistence.Query;

import com.sap.nwce.ra.edm.ejb.entity.Department;
import com.sap.nwce.ra.edm.ejb.entity.Employee;
import com.sap.nwce.ra.edm.ejb.entity.Navigation;
import com.sap.nwce.ra.edm.ejb.entity.Skill;
import com.sap.nwce.ra.edm.ejb.entity.UserGroup;
import com.sap.nwce.ra.edm.util.ReferenceApplicationEJBLogger;



/**
 * Employee management session facade. provides operations on employees and related objects.
 */
@Stateless
```

```
@Local ({EmployeeManagementServiceLocal.class})
@SuppressWarnings("boxing")
@TransactionManagement(value=TransactionManagementType.CONTAINER)
public class EmployeeManagementServiceBean implements
EmployeeManagementServiceLocal {

    @Resource
    private SessionContext sessionCtx;

    @PersistenceContext(unitName = "ems/EJB3_EDM_DEMO_PU")
  private EntityManager em;

    /**
     * Get all employees in a single list.
     * @return list of all employees
     */
    @SuppressWarnings("unchecked")
    public List<Employee> getAllEmployees(){
        Query query = em.createNamedQuery("Employee.getAll");
        List<Employee> result = query.getResultList();
     return result;
    }

    /**
     * Get an employee by its primary key.
     * @return employee entity
     */
    public Employee getEmployeeByID(String pId){
        ReferenceApplicationEJBLogger.logInfo("search for employee by Id: " + pId);
        return em.find(Employee.class, Integer.valueOf(pId));
    }

    /**
     * Get a list of employees with matching lastname.
     * @return list of employees
     */
    @SuppressWarnings("unchecked")
    public List<Employee> findEmployeeByNamePart(String pNameFragment) {
        Query query = em.createNamedQuery("Employee.findByNamePart");
        String parameter = pNameFragment.toLowerCase();
```

```
        query.setParameter("namepart", "%"+parameter+"%");
        query.setParameter("namepart1", "%"+parameter+"%");
        List<Employee> result = query.getResultList();
        return result;
    }

    /**
     * Get all skills.
     * @return
     */
    @SuppressWarnings("unchecked")
    public List<Skill> getAllSkills() {
        Query query = em.createNamedQuery("Skill.getAll");
        List<Skill> result = query.getResultList();
     return result;
    }

    /**
     * Persist the state of an Employee.
     * @param pEmployee
     */
    @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW)
    public Employee saveEmployee(Employee pEmployee){
        ReferenceApplicationEJBLogger.logInfo("persist employee: " +
pEmployee.getEmployeeId());
        return em.merge(pEmployee);
    }

    /**
     * Persist the state of an Skill.
     * @param pEmployee
     */
    @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW)
    public Skill saveSkill(Skill pSkill){
        ReferenceApplicationEJBLogger.logInfo("persist skill: " + pSkill.getSkillId());
        return em.merge(pSkill);
    }

    public void removeSkill(Skill pSkill){
        em.remove(pSkill);
```

```
    }
    /**
     * Get an employee by its login.
     * @param login of employee
     * @return Employee or null if no employee was found
     */
    @SuppressWarnings("unchecked")
    public Employee findEmployeeByLogin(String pLogin) {
        ReferenceApplicationEJBLogger.logInfo("find employee for login name: " +
pLogin);
        Employee  result = null;
        Query query = em.createNamedQuery("Employee.findByLogin");
        query.setParameter("login", pLogin);
        try {
            result = (Employee)query.getSingleResult();
        } catch (NoResultException e) {
            ReferenceApplicationEJBLogger.logInfo("No Employee found for login name: "
+ pLogin);
            return null;
        }
        return result;
    }

    /**
     * Get employee associated with the current user by its login.
     */
    @SuppressWarnings("unchecked")
    public Employee findEmployeeByCurrentUserId() {
        return findEmployeeByLogin(sessionCtx.getCallerPrincipal().getName());
    }

    public Skill getSkillById(String sklId){
        return em.find(Skill.class, Integer.parseInt(sklId));
    }

    @SuppressWarnings("unchecked")
    public List<Skill> findSkillByName(String pNameFragment){
        Query query = em.createNamedQuery("Skill.findByNamePart");
        query.setParameter("namepart", "%"+pNameFragment.toLowerCase()+"%");
        List<Skill> result = query.getResultList();
```

```
        return result;
    }

    /**
     * Get all departments.
     * @return list of departments
     */
    @SuppressWarnings("unchecked")
    public List<Department> getAllDepartments(){
        Query query = em.createNamedQuery("Department.getAll");
        List<Department> result = query.getResultList();
     return result;
    }

    /**
     * Get a department by its id.
     * @param depId
     * @return
     */
    public Department getDepartmentById(String departmentId) {
        return em.find(Department.class, departmentId);
    }

  /**
   * Get the list of all navigations for the employee.
   * @param employee login name to get the allowed navigations for
   * @return ordered list of navigations, empty if login was unknown
   */
    public List<Navigation> getOrderedNavigations(String pLogin) {
        Employee emp = this.findEmployeeByLogin(pLogin);
        List<Navigation> result = new ArrayList<Navigation>();
        if(emp != null) {
            //if login was found return lsit, otherwise return empty list
            for(UserGroup ug: emp.getGroups()){
                for(Navigation nav: ug.getNavigations()){
                    if(!result.contains(nav)){
                        result.add(nav);
                    }
                }
```

```
            }
            Collections.sort(result, new NavigationComparator());
        }
        return result;
    }

  /**
   * This utility class allows for comparing Navigation entities by their position attributes.
   * @author D042764
   */
    private class NavigationComparator implements Comparator<Navigation>{

        public int compare(Navigation o1, Navigation o2) {
            return o1.getPosition() - o2.getPosition();
        }

  }
}
```

ProjectChangeReceiver Source Code

```
package com.sap.nwce.ra.edm.ejb.services;

/*
 * Copyright (c) 2006 Sun Microsystems, Inc.  All rights reserved.  U.S.
 * Government Rights - Commercial software.  Government users are subject
 * to the Sun Microsystems, Inc. standard license agreement and
 * applicable provisions of the FAR and its supplements.  Use is subject
 * to license terms.
 *
 * This distribution may include materials developed by third parties.
 * Sun, Sun Microsystems, the Sun logo, Java and J2EE are trademarks
 * or registered trademarks of Sun Microsystems, Inc. in the U.S. and
 * other countries.
 *
 * Copyright (c) 2006 Sun Microsystems, Inc. Tous droits reserves.
 *
 * Droits du gouvernement americain, utilisateurs gouvernementaux - logiciel
 * commercial. Les utilisateurs gouvernementaux sont soumis au contrat de
 * licence standard de Sun Microsystems, Inc., ainsi qu'aux dispositions
 * en vigueur de la FAR (Federal Acquisition Regulations) et des
 * supplements a celles-ci.  Distribue par des licences qui en
 * restreignent l'utilisation.
 *
 * Cette distribution peut comprendre des composants developpes par des
 * tierces parties. Sun, Sun Microsystems, le logo Sun, Java et J2EE
 * sont des marques de fabrique ou des marques deposees de Sun
 * Microsystems, Inc. aux Etats-Unis et dans d'autres pays.
 */

import javax.annotation.Resource;
import javax.ejb.ActivationConfigProperty;
import javax.ejb.MessageDriven;
import javax.ejb.MessageDrivenContext;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageListener;
```

```
import javax.jms.TextMessage;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

import com.sap.nwce.ra.edm.ejb.entity.Projectchanges;
import com.sap.nwce.ra.edm.util.ReferenceApplicationEJBLogger;



/**
 * The MessageBean class is a message-driven bean.  It implements
 * the javax.jms.MessageListener interface. It is defined as public
 * (but not final or abstract).
 *
 */
@MessageDriven(activationConfig = {
          @ActivationConfigProperty(propertyName = "destinationType", propertyValue =
"javax.jms.Queue"),
          @ActivationConfigProperty(propertyName = "destination", propertyValue =
"prjChangeQueue"),
        @ActivationConfigProperty(propertyName = "connectionFactoryName",
propertyValue = "EDMQueueConnectionFactory")


})

public class ProjectChangeReceiverBean implements MessageListener {

   @Resource
   public MessageDrivenContext mdc;
     @PersistenceContext(unitName = "ems/EJB3_EDM_DEMO_PU")
   private EntityManager em;

   /**
    * Constructor, which is public and takes no arguments.
    */
   public ProjectChangeReceiverBean() {
   }

   /**
    * onMessage method, declared as public (but not final or
    * static), with a return type of void, and with one argument
```

```
    * of type javax.jms.Message.
    *
    * Casts the incoming Message to a TextMessage and displays
    * the text.
    *
    * @param inMessage    the incoming message
    */
   public void onMessage(Message inMessage) {
      TextMessage msg = null;

      try {
         if (inMessage instanceof TextMessage) {
            msg = (TextMessage) inMessage;
            Projectchanges chPrj = new Projectchanges();
            chPrj.setProjectId(msg.getIntProperty("ProjectId"));
            chPrj.setEndDate(java.sql.Date.valueOf(msg.getStringProperty("EndDate")));
            chPrj.setStartDate(java.sql.Date.valueOf(msg.getStringProperty("StartDate")));
            chPrj.setTitle(msg.getStringProperty("Title"));
            chPrj.setDescription(msg.getStringProperty("Description"));
            chPrj.setLeader(msg.getIntProperty("LeadId"));
            chPrj.setChangeDate(new
java.sql.Timestamp(msg.getLongProperty("ChangeDate")));
            chPrj.setStatus(msg.getIntProperty("Status"));
            chPrj.setChangerId(msg.getStringProperty("Changer"));
            em.merge(chPrj);
            ReferenceApplicationEJBLogger.logInfo("MESSAGE BEAN: Message received: "
+ chPrj.getProjectId());
         } else {
          ReferenceApplicationEJBLogger.logWarn("Message of wrong type: "
                  + inMessage.getClass().getName());
         }
      } catch (JMSException e) {
          ReferenceApplicationEJBLogger.logException( e);
         mdc.setRollbackOnly();
      } catch (Throwable te) {
          ReferenceApplicationEJBLogger.logThrowable( te);
      }
   }
}
```

ProjectChangeSender Source Code

```
/*
 * Copyright (c) 2006 SAP AG, Germany. All rights reserved.
 * SAP C42 Reference Implementation. Use is subject to license terms.
 */
package com.sap.nwce.ra.edm.ejb.services;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import javax.annotation.Resource;
import javax.ejb.Local;
import javax.ejb.Stateless;
import javax.ejb.TransactionManagement;
import javax.ejb.TransactionManagementType;
import javax.jms.Connection;
import javax.jms.ConnectionFactory;
import javax.jms.JMSException;
import javax.jms.MessageProducer;
import javax.jms.Queue;
import javax.jms.Session;
import javax.jms.TextMessage;

import com.sap.nwce.ra.edm.ejb.entity.Project;
import com.sap.nwce.ra.edm.util.ReferenceApplicationEJBLogger;

/**
 * JMS SenderSession.
 */
@Stateless
@Local ({ProjectChangeSenderLocal.class})
@SuppressWarnings("boxing")
@TransactionManagement(value = TransactionManagementType.CONTAINER)
public class ProjectChangeSenderBean implements ProjectChangeSenderLocal {

    @Resource(name="EDMQueueConnectionFactory")
    private ConnectionFactory queueConnectionFactory;
```

```
    @Resource(name="prjChangeQueue")
    private Queue queue;

    private Connection queueConnection;

    public ProjectChangeSenderBean() {
    }

    @PostConstruct
    public void initialize() {
        try {
            queueConnection = queueConnectionFactory.createConnection();
            queueConnection.start();
        } catch (JMSException jmse) {
            ReferenceApplicationEJBLogger.logException(jmse);
            throw new RuntimeException(jmse);
        }
    }

    /**
     * Creates session, sender, message and send created message
     */
    public void sendMessage(Project pProject, String pCaller) {
        Session queueSession = null;
        MessageProducer queueSender = null;
        try {
            queueSession = queueConnection.createSession(false,
Session.AUTO_ACKNOWLEDGE);
            queueSender = queueSession.createProducer(queue);
            TextMessage textMsg = queueSession.createTextMessage();
            textMsg.setIntProperty("ProjectId", pProject.getProjectId());
            textMsg.setStringProperty("StartDate", pProject.getStartDate().toString());
            textMsg.setStringProperty("EndDate", pProject.getEndDate().toString());
            textMsg.setStringProperty("Title", pProject.getTitle());
            textMsg.setStringProperty("Description", pProject.getDescription());
            if(pProject.getLeader() != null) {
                textMsg.setIntProperty("LeadId",
pProject.getLeader().getEmployeeId());
            } else {
                textMsg.setIntProperty("LeadId", -1);
```

```
            }
            textMsg.setLongProperty("ChangeDate", System.currentTimeMillis());
            textMsg.setIntProperty("Status", pProject.getStatus());
            textMsg.setStringProperty("Changer", pCaller);
            queueSender.send(textMsg);
        } catch (JMSException jmse) {
            ReferenceApplicationEJBLogger.logException(jmse);
            throw new RuntimeException(jmse);
        } finally {
            if (queueSession != null) {
                try {
                    queueSession.close();
                } catch (JMSException jmse) {
                    ReferenceApplicationEJBLogger.logException(jmse);
                }
            }
        }
    }

    @PreDestroy
    public void stopSession() {

        if (queueConnection != null) {
            try {
                queueConnection.close();
            } catch (JMSException jmse) {
                ReferenceApplicationEJBLogger.logException(jmse);
                throw new RuntimeException(jmse);
            }
        }
    }

}
```

ProjectManagementService Source Code

```
/*
 * Copyright (c) 2006 SAP AG, Germany. All rights reserved.
 * SAP C42 Reference Implementation. Use is subject to license terms.
 */
package com.sap.nwce.ra.edm.ejb.services;

import java.util.List;

import javax.annotation.Resource;
import javax.ejb.EJB;
import javax.ejb.Local;
import javax.ejb.SessionContext;
import javax.ejb.Stateless;
import javax.ejb.TransactionAttribute;
import javax.ejb.TransactionAttributeType;
import javax.ejb.TransactionManagement;
import javax.ejb.TransactionManagementType;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.Query;

import com.sap.nwce.ra.edm.ejb.entity.Project;
import com.sap.nwce.ra.edm.util.ReferenceApplicationEJBLogger;

/**
 * Project session facade. provides operations on projects and related objects.
 */
@Stateless
@Local ({ProjectManagementServiceLocal.class})
@SuppressWarnings("boxing")
@TransactionManagement(value=TransactionManagementType.CONTAINER)
public class ProjectManagementServiceBean implements ProjectManagementServiceLocal{

    //@TODO change the entity manager
    @PersistenceContext(unitName = "ems/EJB3_EDM_DEMO_PU")
  private EntityManager em;
```

```
@Resource
private SessionContext sessionCtx;

@EJB(name="JMSSenderSessionLocal")
private ProjectChangeSenderLocal jmsSender;
/**
 * Get a project by its id.
 * @return project entity
 */

public Project getProjectById(String pId){
      ReferenceApplicationEJBLogger.logInfo("serach for a project by Id: " + pId);
      return em.find(Project.class, Integer.valueOf(pId));
}

/**
 * Get all projects in a single list.
 * @return list of all projects
 */
@SuppressWarnings("unchecked")
public List<Project> getAllProjects(){
      Query query = em.createNamedQuery("Project.getAll");
      List<Project> result = query.getResultList();
 return result;
}

/**
 * Get all assigned projects.
 * @return list of all assigned projects
 */
@SuppressWarnings("unchecked")
public List<Project> getAssignedProjects(int pEmployeeId){
      Query query = em.createNamedQuery("Project.getAllForEmployee");
      query.setParameter("employeeId", pEmployeeId);
      List<Project> result = query.getResultList();
      return result;
}

/**
```

```
     * Persists a project (create or update) and sends a message to
     * the projectchangequeue to notify the system that the project
     * was changed
     */
    @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW)
    public Project createProject(Project pProject){
        ReferenceApplicationEJBLogger.logInfo("persisting project: " + pProject.getProjectId());
        Project result = em.merge(pProject);
        String caller = sessionCtx.getCallerPrincipal().getName();
        jmsSender.sendMessage(result, caller);
        return result;
    }

    /**
     * Get a list of projects by its name or description.
     * @param Name or description fragment
     * @return List of projects
     */
    @SuppressWarnings("unchecked")
    public List<Project> findProjectByNameOrDescription(String pNameFragment){
        ReferenceApplicationEJBLogger.logInfo("searching projects for name fragement: " +
pNameFragment);
        Query query = em.createNamedQuery("Project.findByNameOrDescPart");
        String parameter = pNameFragment.toLowerCase();
        query.setParameter("namepart", "%"+parameter+"%");
        query.setParameter("namepart1", "%"+parameter+"%");
        List<Project> result = query.getResultList();
        return result;
    }

}
```