Java Persistence Architecture (JPA) Association Mappings

Session-2

Association Mappings

The mapping of associations between entity classes and the relationships between tables is the soul of ORM.

Following are the **four** ways in which the cardinality of the relationship between the objects can be expressed. An association mapping can be unidirectional as well as bidirectional.

Mapping type	Description
One-to-One	Mapping one-to-one relationship using Hibernate
One-to-Many	Mapping one-to-many relationship using Hibernate
Many-to-One	Mapping many-to-one relationship using Hibernate
Many-to-Many	Mapping many-to-many relationship using Hibernate

OneToMany Mapping

One-To-Many Relationship: A relationship in which each row in one table is linked to multiple rows in another table.

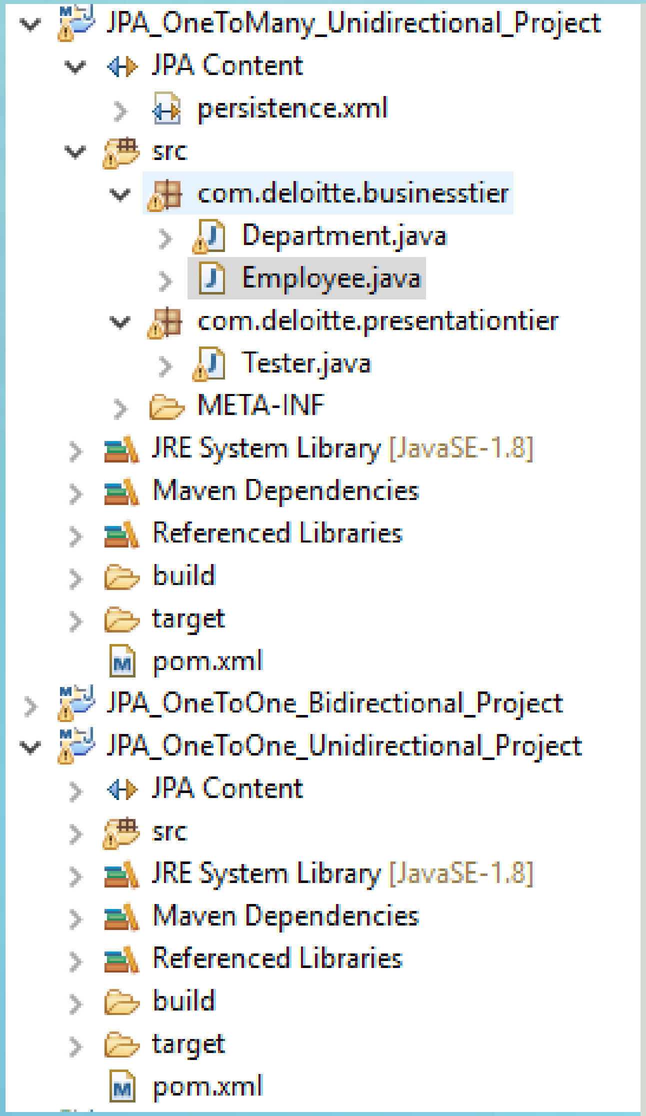
The directionality could be uni-directional (from one entity to another) or bi-directional.

In one-to-many unidirectional relationship from Department to Employee, then if one has an instance of Department object then one can traverse from Department object to Employee objects but not vice-versa.

However, In one-to-many bi-directional relationship between Department and Employee, it is possible to navigate from Department object to Employee object and vice versa.

Let us build one-to-many relationship between a Department entity and Employee entity with uni-directional navigation between Department to Employee entities.

OneToMany Bi-directional Mapping Application


JPA_OneToMany_Unidirectional_Project
JPA Content
persistence.xml
src
com.deloitte.businesstier
Department.java
Employee.java
com.deloitte.presentationtier
Tester.java
META-INF
JRE System Library [JavaSE-1.8]
Maven Dependencies
Referenced Libraries
build
target
pom.xml
JPA_OneToOne_Bidirectional_Project
JPA_OneToOne_Unidirectional_Project
JPA Content
src
JRE System Library [JavaSE-1.8]
Maven Dependencies
Referenced Libraries
build
target
pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
<persistence-unit name="JPA_OneToMany_Unidirectional_Project">
<class>com.deloitte.businesstier.Department</class>
<class>com.deloitte.businesstier.Employee</class>

<properties>
<property name="hibernate.dialect"
value="org.hibernate.dialect.Oracle10gDialect"/>
<property name="hibernate.hbm2ddl.auto" value="update"/>
<property name="hibernate.connection.driver_class"
value="oracle.jdbc.OracleDriver"/>
<property name="hibernate.connection.username" value="scott"/>
<property name="hibernate.connection.password" value="tiger"/>
<property name="show_sql" value="true"/>
<property name="hibernate.connection.url"
value="jdbc:oracle:thin:@//localhost:1521/orcl"/>
</properties>
</persistence-unit>
</persistence>
```

OneToMany Bi-directional Mapping Application

```
@Entity
@Table(name="department_onetomany_d")
public class Department {

@Id
@SequenceGenerator(name="DEPT_SEQ",sequenceName="DEPT_ONETOMANY_SEQ_D",
allocationSize=1,initialValue=1)
@GeneratedValue(strategy=GenerationType.SEQUENCE,generator="DEPT_SEQ")
private Integer deptno;
private String dname;
private String loc;

@OneToMany(mappedBy="department",fetch=FetchType.LAZY, cascade=CascadeType.ALL)
private Set<Employee> employeeSet;
//getter and setter methods
//override toString() method
}
```

```
@Entity
@Table(name="employee_onetomany_d")
public class Employee {
@Id
@SequenceGenerator(name="EMP_SEQ",sequenceName="EMP_ONETOMANY_SEQ_D",
allocationSize=1,initialValue=1001)
@GeneratedValue(strategy=GenerationType.SEQUENCE,
generator="EMP_SEQ")
private Long empno;
private String ename;
private Date birthdate;
private String job;
private Double sal;
@ManyToOne(cascade=CascadeType.ALL)
@JoinColumn(name="DEPT_DEPTNO",referencedColumnName="deptno")
private Department department;
//getter and setter methods
//override toString() method
}
```

OneToMany Bi-directional Mapping Application

```
public class Tester {
static EntityManagerFactory emf =
Persistence.createEntityManagerFactory("JPA_OneToMany_Unidirectional_Project");
static EntityManager em = emf.createEntityManager();

public static void main(String[] args) {
 populateTables();
 showTables();

}
```

Contd…

Contd…

OneToMany Bi-directional Mapping Application

```java
private static void populateTables() {
em.getTransaction().begin();

 Department d=new Department();
 d.setDname("IT");
 d.setLoc("Hyderabad");

 Employee e1=new Employee();
 e1.setEname("Swamy");
 e1.setBirthdate(new Date());
 e1.setJob("Manager");
 e1.setSal(96789.00);

 Employee e2=new Employee();
 e2.setEname("Laxmi");
 e2.setBirthdate(new Date());
 e2.setJob("Developer");
 e2.setSal(74566.00);

em.persist(e1);
 em.persist(e2);

 Set<Employee> employeeSet=new HashSet<>();
 employeeSet.add(e1);
 employeeSet.add(e2);

 e1.setDepartment(d);
 e2.setDepartment(d);

 d.setEmployeeSet(employeeSet);

 em.flush();

 em.getTransaction().commit();
}

}
```

Contd…

Contd…

OneToMany Bi-directional Mapping Application

```
private static void showTables() {
em.getTransaction().begin();
String sql1="Select D from Department D";

System.out.println("Department Details...");
Query query1=em.createQuery(sql1);
List<Department> departmentList= query1.getResultList();
Iterator<Department> iterator1=departmentList.iterator();
while(iterator1.hasNext()){
System.out.println(iterator1.next());
}
System.out.println("------------------------------");
String sql2="Select E from Employee E";
Query query2=em.createQuery(sql2);
System.out.println("Employee Details...");
List<Employee> employeeList= query2.getResultList();
Iterator<Employee> iterator2=employeeList.iterator();
while(iterator2.hasNext()){
System.out.println(iterator2.next());
}
em.getTransaction().commit();
}
```

Contd…

One to many bi-directional web application

OrderProcessingSystemJPARESTV6.zip

ManyToMany Mapping

A ManyToMany relationship in Java is where the source object has an attribute that stores a collection of target objects and (if) those target objects had the inverse relationship back to the source object it would also be a ManyToMany relationship

Every many-to-many association has two sides, the owning side and the non-owning, or inverse, side. The join table is specified on the owning side. If the association is bidirectional, either side may be designated as the owning side.

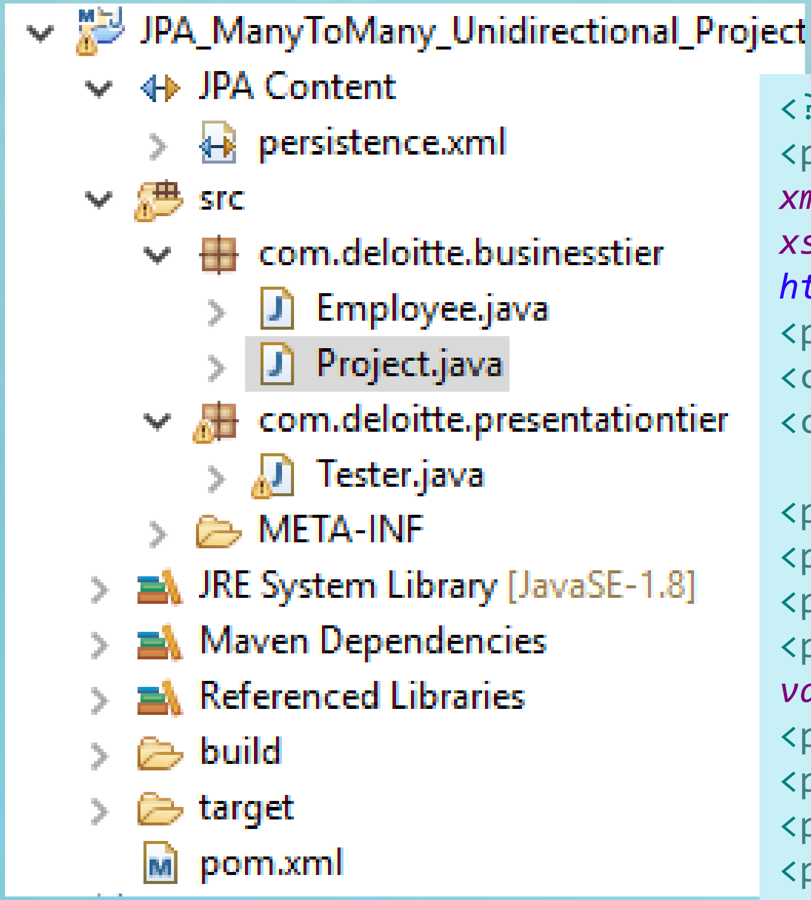
If the relationship is bidirectional, the non-owning side must use the *mappedBy* element of the ManyToMany annotation to specify the relationship field or property of the owning side.

All ManyToMany relationships require a JoinTable.

The JoinTable is defined using the @JoinTable annotation . The JoinTable defines a foreign key to the source object's primary key (joinColumns), and a foreign key to the target object's primary key (inverseJoinColumns).

Normally the primary key of the JoinTable is the composite primary key which is combination of both foreign keys.

ManyToMany Bi-directional Mapping


JPA_ManyToMany_Unidirectional_Project
JPA Content
persistence.xml
src
com.deloitte.businesstier
Employee.java
Project.java
com.deloitte.presentationtier
Tester.java
META-INF
JRE System Library [JavaSE-1.8]
Maven Dependencies
Referenced Libraries
build
target
pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
<persistence-unit name="JPA_ManyToMany_Unidirectional_Project">
<class>com.deloitte.businesstier.Employee</class>
<class>com.deloitte.businesstier.Project</class>

<properties>
<property name="hibernate.dialect" value="org.hibernate.dialect.Oracle10gDialect"/>
<property name="hibernate.hbm2ddl.auto" value="update"/>
<property name="hibernate.connection.driver_class"
value="oracle.jdbc.OracleDriver"/>
<property name="hibernate.connection.username" value="scott"/>
<property name="hibernate.connection.password" value="tiger"/>
<property name="show_sql" value="true"/>
<property name="hibernate.connection.url"
value="jdbc:oracle:thin:@//localhost:1521/orcl"/>
</properties>

</persistence-unit>
</persistence>
```

ManyToMany Bi-directional Mapping

```java
@Entity
@Table(name="employee_manytomany")
public class Employee {
@Id
@SequenceGenerator(name="EMP_MTOM_SEQ",
sequenceName="EMP_MANYTOMANY_SEQ",allocationSize=1,initialValue=1001)
@GeneratedValue(strategy=GenerationType.SEQUENCE,generator="EMP_MTOM_SEQ")
private Long empId;
private String ename;
private Date birthdate;
private String job;
private Double sal;

@ManyToMany
@JoinTable(
name="EMP_PROJECTS_MANYTOMANY",
joinColumns=@JoinColumn(name="EMP_ID",referencedColumnName="EMPID"),
inverseJoinColumns=@JoinColumn(name="PROJECT_ID",referencedColumnName="PROJECTID")
)
private List<Project> projectList;
//getter and setter methods
//override toString() method
}
```

ManyToMany Bi-directional Mapping

```
@Entity
@Table(name="project_manytomany")
public class Project {
@Id
@SequenceGenerator(name="PROJECT_MTOM_SEQ",sequenceName="PROJECT_MANYTOMANY_SEQ",
allocationSize=1,initialValue=1)
@GeneratedValue(strategy=GenerationType.SEQUENCE,generator="PROJECT_MTOM_SEQ")
private Long projectId;
private String projectName;
private Integer projectDuration;

@ManyToMany(mappedBy="projectList")
private List<Employee> employees;

//getter and setter methods
// override toString() method
}
```

ManyToMany Bi-directional Mapping

```
public class Tester {
static EntityManagerFactory emf =
Persistence.createEntityManagerFactory("JPA_ManyToMany_Unidirectional_Project");
static EntityManager em = emf.createEntityManager();

public static void main(String[] args) {
populateTables();
showTableContents();

}

private static void populateTables() {
try{
em.getTransaction().begin();
Employee e1= new Employee();Employee e2=new Employee();
Employee e3=new Employee();
e1.setEname("Madhav");e1.setJob("Developer");e1.setBirthdate(new Date());e1.setSal(45678.00);
e2.setEname("Lavanya");e2.setJob("Developer");e2.setBirthdate(new Date());e2.setSal(65678.00);
e3.setEname("Vinay");e3.setJob("Manager");e3.setBirthdate(new Date());e3.setSal(95678.00);
```

Contd…

ManyToMany Bi-directional Mapping

Contd…

```
Project p1=new Project();Project p2=new Project();Project p3=new Project();
p1.setProjectName("OrderProcessingSystem");
p1.setProjectDuration(10);p2.setProjectName("CarChaseMobileApp");
p2.setProjectDuration(15);
p3.setProjectName("ERP App");p3.setProjectDuration(24);

em.persist(p1);em.persist(p2);em.persist(p3);

List<Project> projectList1=new ArrayList<>();
List<Project> projectList2=new ArrayList<>();

projectList1.add(p1);projectList1.add(p3);projectList2.add(p2);projectList2.add(p3);
e1.setProjectList(projectList1);e2.setProjectList(projectList2);
e3.setProjectList(projectList1);

em.persist(e1);em.persist(e2);em.persist(e3);
em.getTransaction().commit();
} catch(PersistenceException e){
e.printStackTrace();
}
}
```

Contd…

ManyToMany Bi-directional Mapping

Contd…

```
private static void showTableContents() {
try{
em.getTransaction().begin();
System.out.println("Employee List...");
Query query1=em.createQuery("select E from Employee E");
List<Employee> employeeList=query1.getResultList();
System.out.println(employeeList);
System.out.println("Project List...");
Query query2=em.createQuery("select P from Project P");
List<Project> projectList=query2.getResultList();
System.out.println(projectList);
System.out.println("Employee-Project Allocation List...");

em.getTransaction().commit();

} catch(PersistenceException e){
e.printStackTrace();
}
}
}
```

Thank You!