**Guided LAB 305.5.3B - Demonstration of @ManytoMany Relationship and Mapping**

## 

## **Lab Overview:**

* We will continue from the previous **LAB 305.5.2**
* In a many-to-many association, the source entity has a field that stores a collection of target entities. The @ManyToMany annotation is used to link the source entity with the target entity.
* A many-to-many association always uses an **intermediate** **table** called the **Join table** to store the association that joins two entities. The @JoinTable annotation can be used to specify the table name via the name attribute, as well as to specify the names of the Foreign Key columns. Otherwise, **Join tables** will be created by default name.
* In this lab, we will only implement unidirectional entity mapping using JPA and Hibernate. We can define the @ManytoMany annotation either in the **Class (Cohort)** class or **Teacher** class.

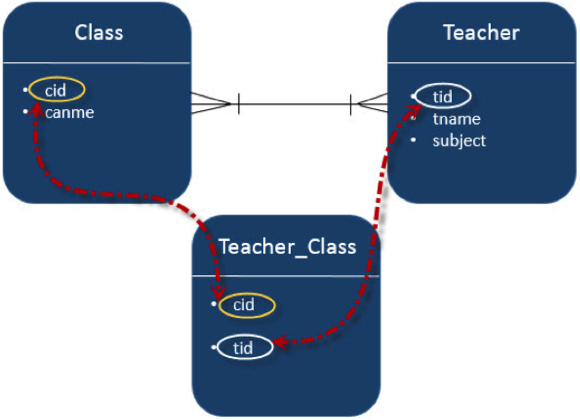
## **Learning Objectives:**

* By the end of this lab, learners will be able to use @ManytoMany relationship mapping.

## **Scenario:**

Let us consider an example of the relationship between **Class (Cohort)** and **Teacher** entities. A **Cohort** can have many **Teachers,** and vice-versa, a **Teacher** can belong to many **Cohorts**.

A join table (**teacher\_cohort**) is required to connect both sides, as shown in the diagram below.



## **Step 1: Create the Persistence class (Model class or Pojo).**

Create an entity class named **“Cohort.java”** in the model package.

* + ***src/main/java/model/Cohort.java***
* Here is the initial code of the **Cohort.java** class:

| package model;  import jakarta.persistence.\*;  @Entity  @Table(name="cohort")  public class Cohort {  @Id  @GeneratedValue( strategy= GenerationType.*IDENTITY* )  private int cohortId;  private String cohortName;  private String duration;  public int getCohortId() {  return cohortId;  }  public void setCohortId(int cohortId) {  this.cohortId = cohortId;  }  public String getCohortName() {  return cohortName;  }  public void setCohortName(String cohortName) {  this.cohortName = cohortName;  }  public String getDuration() {  return duration;  }  public void setDuration(String duration) {  this.duration = duration;  }  } |
| --- |

* Update the **Teacher.java** class:

**Teacher.java**

| package org.example.model;  import jakarta.persistence.\*;  import java.io.Serial;  import java.io.Serializable;  import java.util.Set;  @Entity  @Table(name="Teacher")  public class Teacher implements Serializable {  @Serial  private static final long *serialVersionUID* = 1L;  @Id  @GeneratedValue( strategy=GenerationType.*IDENTITY* )  private int teacherId;  private String salary;  private String teacherName;  // @OneToOne(cascade = CascadeType.*ALL*)  // private Address address;  @ManyToMany(targetEntity = Cohort.class)  private Set<Cohort> cohort;  public Set<Cohort> getCohort() {  return cohort;  }  public void setCohort(Set<Cohort> cohort) {  this.cohort = cohort;  }  //public Address getAddress() {  // return address;  // }  // public void setAddress(Address address) {  // this.address = address;  // }  public Teacher(String salary, String teacherName, Set<Cohort> cohort) {  this.salary = salary;  this.teacherName = teacherName;  }  public Teacher(String salary, String teacherName) {  super();  this.salary = salary;  this.teacherName = teacherName; }  public Teacher() {}  public Teacher(String salary, String teacherName, Department department) {  this.salary = salary;  this.teacherName = teacherName;  }  public Teacher(String salary, String teacherName) {  this.salary = salary;  this.teacherName = teacherName;  }  public int getTeacherId() {  return teacherId;  }  public void setTeacherId(int teacherId) {  this.teacherId = teacherId;  }  public String getSalary() {  return salary;  }  public void setSalary(String salary) {  this.salary = salary;  }  public String getTeacherName() {  return teacherName;  }  public void setTeacherName(String teacherName) {  this.teacherName = teacherName; }  } |
| --- |

### **This association is unidirectional. Here, the Teacher is on the owner's side, and the Cohort is on the other side.**

We can see that the Teacher class has a collection (Set<E>) of elements, because when you map a many-to-many association, you should use a Set instead of a List as the attribute type. Otherwise, Hibernate will take a very inefficient approach to removing entities from the association. It will remove all records from the association table and re-insert the remaining ones. You can avoid this by using a Set instead of a List as the attribute type.

## **Step 2: The Hibernate Configuration File (hibernate.cfg.xml)**

Add the Chort mapping to the configuration file.

| *<?*xml version="1.0" encoding="UTF-8"*?>*  <!DOCTYPE hibernate-configuration PUBLIC  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd"*>*  <hibernate-configuration>  <session-factory>  *<!-- Drop and re-create the database on startup -->*  <property name="hibernate.hbm2ddl.auto"> create-drop </property>  *<!-- Database connection settings -->*  <property name="connection.driver\_class">com.mysql.cj.jdbc.Driver</property>  <property name="connection.url">jdbc:mysql://localhost:3306/usersDb</property>  <property name="connection.username">*<!-- TODO username -->*</property>  <property name="connection.password"> *<!-- TODO Password -->* </property>  *<!-- MySQL DB dialect -->*  <property name="dialect">org.hibernate.dialect.MySQLDialect</property>  *<!-- print all executed SQL on console -->*  <property name="hibernate.show\_sql" >true </property>  <property name="hibernate.format\_sql" >true </property>  *<!-- Mapping entity file -->*  <mapping class="model.Teacher"/>  <mapping class="model.Department"/>  <mapping class="model.Address"/>  <mapping class="model.Cohort"/>  </session-factory>  </hibernate-configuration> |
| --- |

## **Step 3: App.java (main class)**

Add the following code to the App.java.

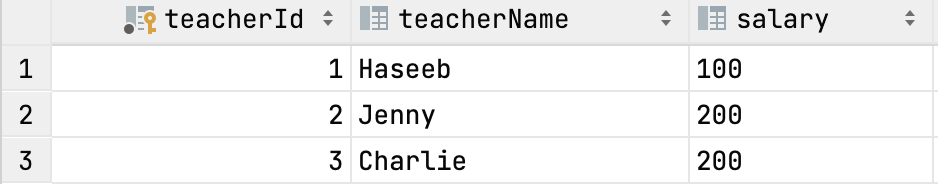
App.java

| import model.Address;  import model.Cohort;  import model.Department;  import model.Teacher;  import org.hibernate.Session;  import org.hibernate.SessionFactory;  import org.hibernate.Transaction;  import org.hibernate.cfg.Configuration;  import java.util.ArrayList;  import java.util.HashSet;  import java.util.Set;  public class App {  public static void main(String[] args) {  *manyToMany*();  }  public static void manyToOne(){  SessionFactory factory = new Configuration().configure().buildSessionFactory();  Session session = factory.openSession();  Transaction transaction = session.beginTransaction();  *//creating departments*  Department dept1 = new Department("IT");  Department dept2 = new Department("HR");  *//creating teacher*  Teacher t1 = new Teacher("1000","MHaseeb",dept1);  Teacher t2 = new Teacher("2220","Shahparan",dept1);  Teacher t3 = new Teacher("3000","James",dept1);  Teacher t4 = new Teacher("40000","Joseph",dept2);  *//Storing Departments in database*  session.persist(dept1);  session.persist(dept2);  *//Storing teachers in database*  session.persist(t1);  session.persist(t2);  session.persist(t3);  session.persist(t4);  transaction.commit(); }  public static void oneToMany(){  SessionFactory factory = new Configuration().configure().buildSessionFactory();  Session session = factory.openSession();  Transaction t = session.beginTransaction();  *//creating teacher*  Teacher t1 = new Teacher("1000","MHaseeb");  Teacher t2 = new Teacher("2220","Shahparan");  Teacher t3 = new Teacher("3000","James");  Teacher t4 = new Teacher("40000","Joseph");  Teacher t5 = new Teacher("200","Ali");  *//Add teacher entity object to the list*  ArrayList<Teacher> teachersList = new ArrayList<>();  teachersList.add(t1);  teachersList.add(t2);  teachersList.add(t3);  teachersList.add(t4);  teachersList.add(t5);  session.persist(t1);  session.persist(t2);  session.persist(t3);  session.persist(t4);  session.persist(t5);  *//Creating Department*  Department department = new Department();  department.setDeptName("Development");  department.setTeacherList(teachersList);  *//Storing Department*  session.persist(department);  t.commit(); }  public static void oneToOne(){  System.*out*.println("Maven + Hibernate + SQL One to One Mapping Annotations");  SessionFactory factory = new Configuration().configure().buildSessionFactory();  Session session = factory.openSession();  Transaction t = session.beginTransaction();  Address a1 = new Address("27th street","NYC","NY",11103);  Address a2 = new Address("28th street","Buffalo","NY",15803);  Teacher t1 = new Teacher("1000","MHaseeb");  Teacher t2 = new Teacher("2220","Shahparan");  // t1.setAddress(a1);  // t2.setAddress(a2);  session.persist(a1);  session.persist(a2);  session.persist(t1);  session.persist(t2);  t.commit();  }  public static void manyToMany(){  SessionFactory factory = new Configuration().configure().buildSessionFactory();  Session session = factory.openSession();  Transaction t = session.beginTransaction();  *//----Create Cohort/class Entity set one----*  Cohort Class1 = new Cohort("Java Developer", "14 weeks");  Cohort Class2 = new Cohort("FullStack Developer", "7 Weeks");  Cohort Class3 = new Cohort("Python Developer", "12 Weeks");  *//------ Store Cohort / Class --------*  session.persist(Class1);  session.persist(Class2);  session.persist(Class3);  *//-----Create Cohort one / Class one --------*  Set<Cohort> ClassSet1 = new HashSet<Cohort>();  ClassSet1.add(Class1);  ClassSet1.add(Class2);  ClassSet1.add(Class3);  *//-----Create Cohort two / Class two --------*  Set<Cohort> ClassSet2 = new HashSet<Cohort>();  ClassSet2.add(Class2);  ClassSet2.add(Class3);  ClassSet2.add(Class1);  *//-----Create Cohort Three / Class Three --------*  Set<Cohort> ClassSet3 = new HashSet<Cohort>();  ClassSet3.add(Class3);  ClassSet3.add(Class1);  ClassSet3.add(Class2);  Teacher t1 = new Teacher("100", "Haseeb", ClassSet1);  Teacher t2 = new Teacher("200", "Jenny", ClassSet2);  Teacher t3 = new Teacher("200", "Charlie", ClassSet3);  session.persist(t1);  session.persist(t2);  session.persist(t3);  t.commit();  }  } |
| --- |

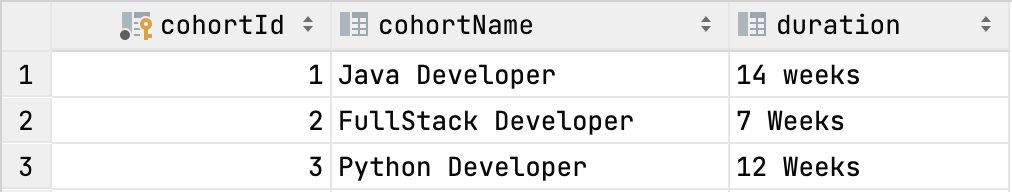
## **Step 5: Run an Application.**

At the start of each thread, a database schema will be created, and the following result can be seen in the Database. Run the **app.java** file to view the following results in your database.

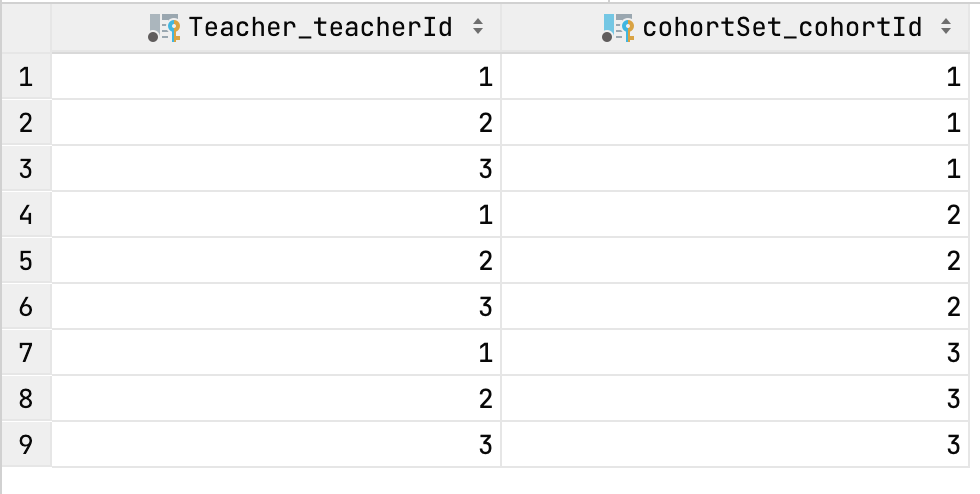
**teacher Table**

****

**cohort Table**

****

**teacher\_cohort Table**

****

* Both **Cohort** and **Teacher** have Many-To-Many relationships. That means that each record of **Cohort** is referred to by the Teacher set (Teacher\_tit), which should be the primary keys in the Teacher table and stored in the **Teacher\_Cohort** table. The intermediate (join) table, which was created, is named **teacher\_cohort** and contains two foreign keys, one of them refers to the Teacher’s primary key and the other refers to the Cohort’s primary Key.
* There is no way to avoid the join table solution for achieving a Many-To-Many association because you could achieve it through any other technique such as adding new columns or comma-separated values).

**Submission Instructions:**

Include the following deliverables in your submission:

* + Submit your source code or screenshot using the Start Assignment button in the top-right corner of the assignment page in Canvas.

## 