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

## 

## **Lab Overview:**

* 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.

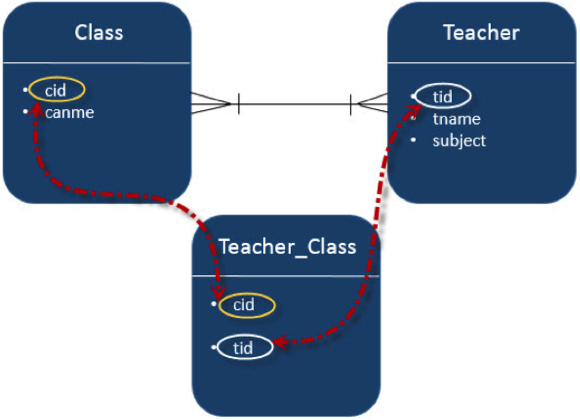
## **Learning Objectives:**

* 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.



## 

## 

## This demonstration consists of the following files:

## **Model classes: Cohort.java and Teacher.java**

## **Hibernate XML configuration file: hibernate.cfg.xml**

## **For Main class and Run an Application: App.java**

## **Maven project: pom.xml**

## 

## 

## 

## 

## 

## 

## **Step 1: Setup the Java Maven Project and Add Jar Dependencies**

* For **Hibernate** and **mySQL Database:** Add the following **jar dependencies** in the **pom.xml** file under **</dependencies> tag** of your Maven project.

| <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core --> <dependency>  <groupId>org.hibernate</groupId>  <artifactId>hibernate-core</artifactId>  <version>5.5.7.Final</version> </dependency>   <dependency>  <groupId>org.hibernate</groupId>  <artifactId>hibernate-annotations</artifactId>  <version>3.5.5-Final</version> </dependency>  *<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->*  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>8.0.25</version>  </dependency> |
| --- |

## **Step 2: Create the Persistence class (Model class or Pojo).**

* Create a package named **“com.perscholas.model.”**
* Create an entity class named **“Cohort.java”** under the above package.
  + ***src\main\java\com\perscholas\model\Cohort.java***
* Here is the initial code of the **Cohort.java** class:

**Cohort.java**

| package com.perscholas.model; import java.util.Set; import javax.persistence.Entity; import javax.persistence.GeneratedValue; import javax.persistence.GenerationType; import javax.persistence.Id; import javax.persistence.ManyToMany; import javax.persistence.Table; @Entity @Table(name="cohort") public class Cohort{  @Id  @GeneratedValue( strategy=GenerationType.IDENTITY )  private int cid;  private String CName;  private String duration;  public Cohort() {  }   public Cohort( String cName, String duration) {  this.CName = cName;  this.duration = duration;  }  public int getCid() {  return cid;  }  public void setCid(int cid) {  this.cid = cid;  }  public String getCName() {  return CName;  }  public void setCName(String cName) {  CName = cName;  }  public String getDuration() {  return duration;  }  public void setDuration(String duration) {  this.duration = duration;  } } |
| --- |

* Create a second entity class named **“Teacher.java”** under the same package.
  + ***src\main\java\com\perscholas\model\Teacher.java***
* Here is the code of the **Teacher.java** class:

**Teacher.java**

| package com.perscholas.model; import java.util.List; import java.util.Set; import javax.persistence.CascadeType; import javax.persistence.Entity; import javax.persistence.GeneratedValue; import javax.persistence.GenerationType; import javax.persistence.Id; import javax.persistence.ManyToMany; import javax.persistence.OneToOne; import javax.persistence.Table; @Entity @Table(name="Teacher") public class Teacher{  @Id  @GeneratedValue( strategy=GenerationType.IDENTITY )  private int tit;  private String salary;  private String Teachername;    @ManyToMany(targetEntity = Cohort.class)  private Set CohortSet;  public Teacher(String salary, String teachername, Set CohortSet) {  this.salary = salary;  this.Teachername = teachername;  this.CohortSet = CohortSet;    }  public Teacher() {  super();  }    public Set getCohortSet() {  return CohortSet;  }  public void setCohortSet(Set cohortSet) {  CohortSet = cohortSet;  }  public int getTit() {  return tit;  }  public void setTit(int tit) {  this.tit = tit;  }  public String getSalary() {  return salary;  }  public void setSalary(String salary) {  this.salary = salary;  }  public String getTeachername() {  return Teachername;  }  public void setTeachername(String teachername) {  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 that by using a Set instead of a List as the attribute type.

## **Step 3: Create the Hibernate Configuration File (hibernate.cfg.xml)**

### To create the configuration file, right-click on **src/main/java** → **New → Other - search files from search panel → click on File → specify the file nam**e **“hibernate.cfg.xml”** → **Finish**.

* *Note: In this exercise, we will use the* ***“usersdb”*** *database, but if you want to use another database for that, you have to change the database name in the below code.*
* Open newly created file and paste the following XML code:

| <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE hibernate-configuration PUBLIC   "-//Hibernate/Hibernate Configuration DTD 5.3//EN"   "http://www.hibernate.org/dtd/hibernate-configuration-5.3.dtd">  <hibernate-configuration>   <property name="hibernate.hbm2ddl.auto"> update </property>   <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">root</property>  <property name="connection.password">password</property>  <property name="dialect">org.hibernate.dialect.MySQL5Dialect</property>   <property name="hibernate.show\_sql" >true </property>  <property name="hibernate.format\_sql" >true </property>   <!-- Mapping entity file -->  <mapping class="com.perscholas.model.Cohort"/>   <mapping class="com.perscholas.model.Teacher"/>    </session-factory>  </hibernate-configuration> |
| --- |

## 

## 

## **Step 4: Create App.java (main class)**

Note: If you are using the **IntelliJ IDE,** you have to create an App.java class. However, if you are using the **Eclipse IDE, the** “**App.java class**” is created automatically.

Add the following code to it App.java.

App.java

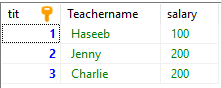
| package com.perscholas.model; import java.util.ArrayList; import java.util.HashSet; import java.util.List; import java.util.Set; import org.hibernate.Session; import org.hibernate.SessionFactory; import org.hibernate.Transaction; import org.hibernate.cfg.Configuration;  public class App  {  public static void main( String[] args )  {  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.**

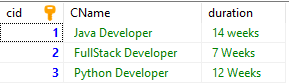
**Right click on App.java → Run As → Java Application.**

At the start of each thread, a database schema will be created, and the following result can be seen in theDatabase

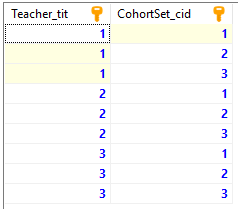
**teacher Table**

****

**cohort Table**

****

**teacher\_cohort Table**

****

* Both **Cohort** and **Teacher** have Many-To-Many relationships. That means 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 (like 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.

## 