**Guided LAB - 305.5.2B - Demonstration of OneToOne Relationship and Mapping with IntelliJ Ultimate**

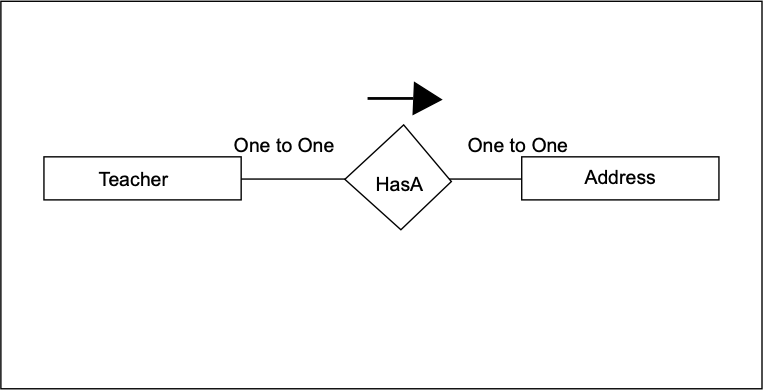
**Lab Overview:**

* We will continue from the **previous LAB 305.5.1**
* The @OneToOne annotation is used to map the source entity with the target entity.
* The one-to-one association can be either unidirectional or bidirectional.
  + In **unidirectional association**, the source entity has a relationship field that refers to the target entity, and the source entity’s table contains the foreign key.
  + In a **bidirectional association**, each entity (e.g., source and target) has a relationship field that refers to each other, and the target entity’s table contains the foreign key. The source entity must use the mappedBy attribute to define the bidirectional one-to-one mapping.
* In this lab, we will implement only **unidirectional entity mapping using** **Hibernate**.

**Scenario:**

* Let us consider an example of a relationship between a ***Person*** and an ***Address***  entity. A person can have one address and that address belongs to one person only. That is a typical example of a **one-to-one** relationship or association. We will model this in a database, and we will need to store the primary key of the ***Address*** record as a foreign key in the ***Person*** table.

The database schema should look like this:



## 

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

* Create an entity class named **“Address.java”** under the model package.
  + ***src/main/java/model/Address.java***
* Here is the initial code of the **Address.java** class:

| package model;  import jakarta.persistence.\*;  import java.io.Serial;  import java.io.Serializable;  @Entity  @Table  public class Address implements Serializable {  @Serial  private static final long *serialVersionUID* = 1L;  @Id  @GeneratedValue( strategy= GenerationType.*IDENTITY* )  private int AddressId;  private String street;  private String city;  private String state;  private int zipCode;  public Address() {  }  public Address(String street, String city, String state, int zipCode) {  this.street = street;  this.city = city;  this.state = state;  this.zipCode = zipCode;  }  public int getAddressId() {  return AddressId;  }  public void setAddressId(int addressId) {  AddressId = addressId;  }  public String getStreet() {  return street;  }  public void setStreet(String street) {  this.street = street;  }  public String getCity() {  return city;  }  public void setCity(String city) {  this.city = city;  }  public String getState() {  return state;  }  public void setState(String state) {  this.state = state;  }  public int getZipCode() {  return zipCode;  }  public void setZipCode(int zipCode) {  this.zipCode = zipCode;  }} |
| --- |

* *Add the following code to the Teacher.java class.*

| package model;  import jakarta.persistence.\*;  import model.Department;  import java.io.Serial;  import java.io.Serializable;  @Entity  @Table  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;  public Address getAddress() {  return address;  }  public void setAddress(Address address) {  this.address = address;  }  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 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; }  } |
| --- |

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

* Add the new mapping class to the hibernate.cfg.xml 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"/>  </session-factory>  </hibernate-configuration> |
| --- |

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

Add the following code to it App.java:

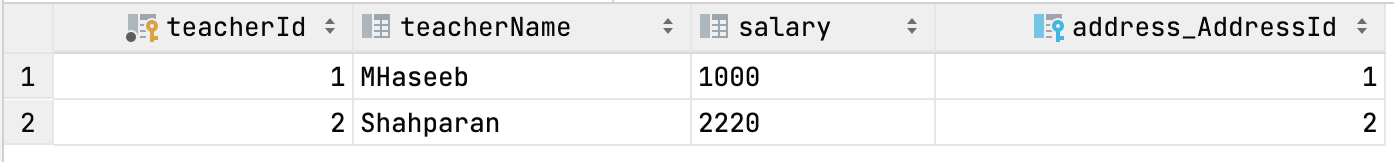
| import model.Address;  import model.Department;  import model.Teacher;  import org.hibernate.Session;  import org.hibernate.SessionFactory;  import org.hibernate.Transaction;  import org.hibernate.cfg.Configuration;  import java.util.ArrayList;  public class App {  public static void main(String[] args) {  *oneToOne*();  }  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();  }  } |
| --- |

## **Step 4: Run an Application**

Finally, we will run our application class(App.java) with the main() method.

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

Is

****

* + **address\_AddressId is the Foreign Key in the Teacher Table.**

**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.

## 