**Guided LAB - 305.5.2A - Demonstration of @OneToOne Relationship and Mapping**

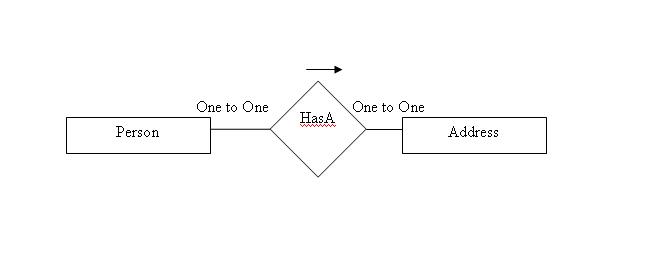
**Lab Overview:**

* 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 (i.e. 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



## This demonstration consists of the following files:

## **Model classes: Person.java and Address.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 **SQL databases:** Add the following **jar dependencies** to the **pom.xml** file under the **</dependencies> tag** of your maven project

| <dependencies>  *<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->*  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>8.0.25</version>  </dependency>    *<!-- 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>   </dependencies> |
| --- |

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

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

**Address.java class**

| import java.io.Serializable; import javax.persistence.Entity; import javax.persistence.GeneratedValue; import javax.persistence.GenerationType; import javax.persistence.Id; import javax.persistence.OneToOne; import javax.persistence.Table; @Entity @Table public class Address implements Serializable {  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;  }} |
| --- |

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

| package com.perscholas.model; import javax.persistence.CascadeType; import javax.persistence.Entity; import javax.persistence.GeneratedValue; import javax.persistence.GenerationType; import javax.persistence.Id; import javax.persistence.OneToOne; import javax.persistence.Table; @Entity @Table public class Person {  @Id  @GeneratedValue( strategy=GenerationType.IDENTITY )  private int PersonId;   private String name;  private String email;  private int age;    **@OneToOne(cascade = CascadeType.ALL)**  private Address adress;  public Person(int personId, String name, String email, int age) {  super();  PersonId = personId;  this.name = name;  this.email = email;  this.age = age;  }    public Person()  {  }  public Address getAdress() {  return adress;  }  public void setAdress(Address adress) {  this.adress = adress;  }  public int getPersonId() {  return PersonId;  }  public void setPersonId(int personId) {  PersonId = personId;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getEmail() {  return email;  }  public void setEmail(String email) {  this.email = email;  }  public int getAge() {  return age;  }  public void setAge(int age) {  this.age = age;  } } |
| --- |

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

* **For Eclipse IDE**: To create the configuration file, To do so, 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**.
* **For the IntelliJ IDE:** Create a configuration file named **hibernate.cfg.xml** under the resources folder and write the following code in it.

*Note: In this lab, 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>   <session-factory>  <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.Person"/>   <mapping class="com.perscholas.model.Address"/>    </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**

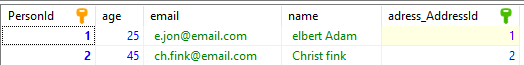
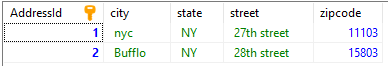
| package com.perscholas.model; import org.hibernate.Session; import org.hibernate.SessionFactory; import org.hibernate.Transaction; import org.hibernate.cfg.Configuration;  public class App  {  public static void main( String[] args )  {  System.out.println("Maven + Hibernate + SQL One to One Mapping Annotations");   SessionFactory factory = new Configuration().configure().buildSessionFactory();  Session session = factory.openSession();    Transaction t = session.beginTransaction();     Address a1 = new Address();  a1.setCity("nyc");  a1.setState("NY");  a1.setStreet("27th street");  a1.setZipcode(11103);    Address a2 = new Address();  a2.setCity("Bufflo");  a2.setState("NY");  a2.setStreet("28th street");  a2.setZipcode(15803);    Person p1 = new Person();  p1.setAge(25);  p1.setEmail("e.jon@email.com");  p1.setName("elbert Adam");  p1.setAdress(a1);    Person p2 = new Person();  p2.setAge(45);  p2.setEmail("ch.fink@email.com");  p2.setName("Christ fink");  p2.setAdress(a2);    session.persist(p1);  session.persist(p2);  session.persist(a1);  session.persist(a2);  t.commit();   } } |
| --- |

## 

## **Step 5: 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.

****

* + **adress\_AddressId is the Foreign Key in the Teacher Table.**
  + **AddressId is the PRIMARY KEY in the Person 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.

## 