### **Guided Lab - Hibernate With H2 Database**

**Objectives:**

* Let's start developing step-by-step Hibernate applications using Maven as a project management and build tool.
* After this lab, learners will have demonstrated the ability to create Hibernate application with H2 Database

## Technologies and Tools Used

* Hibernate 5.3.7.Final
* IDE - Eclipse or Intellij (your choice)
* Maven 3.5.3
* JavaSE 1.8
* H2 In-Memory

## Development Steps

1. Create a Simple Maven Project.
2. Set up Project Directory Structure.
3. Add Jar Dependencies in pom.xml.
4. Create the Entity Classes (Persistent class).
5. Create a Hibernate configuration file - hibernate.cfg.xml.
6. Create the Main Class and run an Application.

## Setup Java Maven Project and Add Jar Dependencies

* Create a new Maven Project - **left click on File Menu - New- Maven Project.**
* Add the following **jar dependencies** in the **pom.xml** under **</dependencies> tag** file of your maven project for **H2 Memory Database** and **Hibernate.**

## **pom.xml file:**

* ***Note: <artifactId> must be according to your created project name, Do not forget to change the <artifactId>***

| <!-------------------------------- hibernate Dependency------- ------------------- > <dependency>  <groupId>org.hibernate</groupId>  <artifactId>hibernate-core</artifactId>  <version>5.5.7.Final</version> </dependency>  <!---------------- https://mvnrepository.com/artifact/com.h2database/h2 --------> <dependency>  <groupId>com.h2database</groupId>  <artifactId>h2</artifactId>  <scope>runtime</scope> </dependency>  <dependency>  <groupId>com.h2database</groupId>  <artifactId>h2</artifactId>  <version>1.4.194</version> </dependency> <dependency>  <groupId>org.hibernate</groupId>  <artifactId>hibernate-annotations</artifactId>  <version>3.5.5-Final</version> </dependency> |
| --- |

## Create the Persistence class (Model class or Pojo).

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

| import javax.persistence.Column; import javax.persistence.Entity; import javax.persistence.GeneratedValue; import javax.persistence.GenerationType; import javax.persistence.Id; import javax.persistence.Table; @Entity @Table(name = "student") public class Student {  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  @Column(name = "id")  private int id;   @Column(name = "first\_name")  private String firstName;   @Column(name = "last\_name")  private String lastName;   @Column(name = "email")  private String email;   public Student() {   }  public Student(String firstName, String lastName, String email) {  this.firstName = firstName;  this.lastName = lastName;  this.email = email;  }   public int getId() {  return id;  }   public void setId(int id) {  this.id = id;  }   public String getFirstName() {  return firstName;  }   public void setFirstName(String firstName) {  this.firstName = firstName;  }   public String getLastName() {  return lastName;  }   public void setLastName(String lastName) {  this.lastName = lastName;  }   public String getEmail() {  return email;  }  public void setEmail(String email) {  this.email = email;  } } |
| --- |

### 

## Create the Hibernate Configuration File (hibernate.cfg.xml)

* *Note: In this exercise, we will use the default database named* ***“test,”*** *but if you want to use another database, you have to change the database name in below code:*
* **For Eclipse IDE**: 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**.
* **For intellij IDE:** Create a configuration file named hibernate.cfg.xml under the resources folder and write the following code in it:

| <?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>  <!------------------ JDBC Database connection settings --------------->  <property name="connection.driver\_class">org.h2.Driver</property> <property name="connection.url">jdbc:h2:tcp://localhost/~/test</property>  <property name="connection.username">sa</property>  <property name="connection.password"></property>  <!------ JDBC connection pool settings using built-in test pool ---->  <property name="connection.pool\_size">1</property>  <!-- Select our SQL dialect -->  <property name="dialect">org.hibernate.dialect.H2Dialect</property>  <!-- Echo the SQL to stdout -->  <property name="show\_sql">true</property>  <!-- Set the current session context -->  <property name="current\_session\_context\_class">thread</property>  <!-- Drop and re-create the database schema on startup -->  <property name="hbm2ddl.auto">create-drop</property>  <!------------ Mapping entity file ------------->  <mapping class="com.perscholas.Hib\_H2memory.Student"/>  </session-factory> </hibernate-configuration> |
| --- |

If you get “Error for DriverManagerConnection,” you can use the statement below in connection.url.

**<property name="connection.url">jdbc:h2:~/test</property>**

In the above configuration, notice that Hibernate has a dialect for the H2 database (org.hibernate.dialect.H2Dialect). By default, the Java application connects to an **H2 in-memory** store with the username **sa** and an **empty** password.

## Run an Application

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

Now, let us execute the code we created above. Let’s run the App class.

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

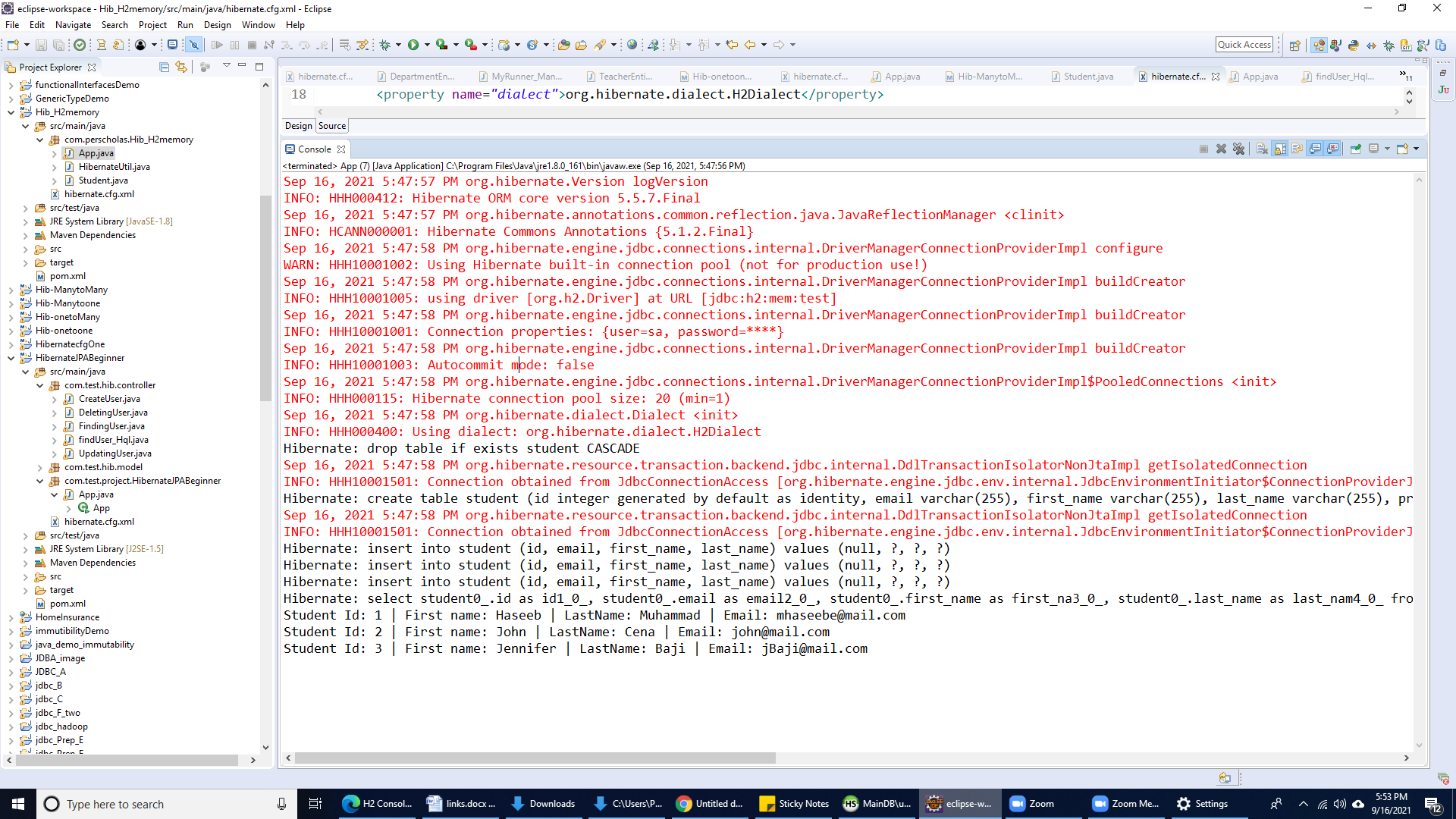
Note: If you are using **IntelliJ IDE,** you have to create a Main class(App class); if you are using **Eclipse IDE,** **“App class”** is created automatically.

**App.java**

| import java.util.List; import org.hibernate.Query; 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();  Student student = new Student("Haseeb", "Muhammad", "mhaseebe@mail.com");  Student student1 = new Student("John", "Cena", "john@mail.com");  Student student2 = new Student("Jennifer", "Bai", "jBaji@mail.com");    // save the student objects  session.save(student);  session.save(student1);  session.save(student2);  // commit transaction  t.commit();  // ------------------ retrieve data from H2 database ----------------------   String hql = "SELECT s FROM Student s";   Query query = session.createQuery(hql);  List<Student> list = query.getResultList();    for (Student stu : list) {  System.out.println("Student Id: " +stu.getId() + " | First name: " + stu.getFirstName() +" | LastName: "+ stu.getLastName() +" | Email: " + stu.getEmail() );  }  } } |
| --- |

In most Hibernate applications, the SessionFactory should be instantiated once during application initialization. The single instance should then be used by all code in a particular process, and any Session should be created using this single SessionFactory.

The SessionFactory is thread-safe and can be shared; a Session is a single-threaded object.



**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.

**CANVAS STAFF USE ONLY: Canvas Submission Guideline:**

| **Instructions for Canvas Assignment Creation** |
| --- |
| **Assignment Name: GLAB - 305.6.2 - Demonstration of @ManytoMany Relationship and Mapping**  **Points:** **100**  **Assignment Group: Module 305 - JDBC, ORM, and Hibernate - (Not Graded)**  **Display Grade As: Completed/Incomplete**  **Do not count this assignment towards the final grade: Checked**  **Submission Types: Files Uploads**  **Allowed Attempts: Unlimited**  **Everything else is the default.** |

## 