Exercise: ORM Implementation with Hibernate - XML and Annotation Configuration

# 1. Objective

To implement Object Relational Mapping (ORM) using Hibernate in two ways:  
1. XML Configuration  
2. Annotation Configuration

# 2. Problem Statement / Scenario

We aim to map a Java class `Employee` to a relational database table using Hibernate ORM. This will be achieved in two ways:  
- By using XML-based configuration files.  
- By using annotations in the persistence class.  
  
The application will demonstrate CRUD operations by configuring Hibernate, creating SessionFactory, and interacting with the database.

# 3. Implementation

## 3.1 XML Configuration

Step 1: Persistence Class - Employee.java

public class Employee {  
 private int id;  
 private String firstName;  
 private String lastName;  
 private int salary;  
  
 // Getters and Setters  
}

Step 2: Mapping File - Employee.hbm.xml

<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"  
 "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">  
<hibernate-mapping>  
 <class name="Employee" table="EMPLOYEE">  
 <id name="id" column="ID"/>  
 <property name="firstName" column="FIRST\_NAME"/>  
 <property name="lastName" column="LAST\_NAME"/>  
 <property name="salary" column="SALARY"/>  
 </class>  
</hibernate-mapping>

Step 3: Hibernate Configuration File - hibernate.cfg.xml

<!DOCTYPE hibernate-configuration PUBLIC  
 "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  
 "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">  
<hibernate-configuration>  
 <session-factory>  
 <property name="hibernate.dialect">org.hibernate.dialect.H2Dialect</property>  
 <property name="hibernate.connection.driver\_class">org.h2.Driver</property>  
 <property name="hibernate.connection.url">jdbc:h2:~/test</property>  
 <property name="hibernate.connection.username">sa</property>  
 <property name="hibernate.connection.password"/>  
 <property name="hibernate.hbm2ddl.auto">update</property>  
 <mapping resource="Employee.hbm.xml"/>  
 </session-factory>  
</hibernate-configuration>

Step 4: Database Operation - MainApp.java

public class MainApp {  
 public static void main(String[] args) {  
 SessionFactory factory = new Configuration().configure().buildSessionFactory();  
 Session session = factory.openSession();  
 Transaction tx = session.beginTransaction();  
  
 Employee emp = new Employee();  
 emp.setId(1);  
 emp.setFirstName("John");  
 emp.setLastName("Doe");  
 emp.setSalary(5000);  
  
 session.save(emp);  
 tx.commit();  
 session.close();  
 }  
}

Output

Hibernate: insert into EMPLOYEE (FIRST\_NAME, LAST\_NAME, SALARY, ID) values (?, ?, ?, ?)

## 3.2 Annotation Configuration

Step 1: Persistence Class with Annotations - Employee.java

import jakarta.persistence.\*;  
  
@Entity  
@Table(name = "EMPLOYEE")  
public class Employee {  
  
 @Id  
 private int id;  
  
 @Column(name = "FIRST\_NAME")  
 private String firstName;  
  
 @Column(name = "LAST\_NAME")  
 private String lastName;  
  
 @Column(name = "SALARY")  
 private int salary;  
  
 // Getters and Setters  
}

Step 2: Hibernate Configuration File - hibernate.cfg.xml

<hibernate-configuration>  
 <session-factory>  
 <property name="hibernate.dialect">org.hibernate.dialect.H2Dialect</property>  
 <property name="hibernate.connection.driver\_class">org.h2.Driver</property>  
 <property name="hibernate.connection.url">jdbc:h2:~/test</property>  
 <property name="hibernate.connection.username">sa</property>  
 <property name="hibernate.connection.password"/>  
 <property name="hibernate.hbm2ddl.auto">update</property>  
 <mapping class="Employee"/>  
 </session-factory>  
</hibernate-configuration>

Step 3: Database Operation - MainApp.java

public class MainApp {  
 public static void main(String[] args) {  
 SessionFactory factory = new Configuration().configure().buildSessionFactory();  
 Session session = factory.openSession();  
 Transaction tx = session.beginTransaction();  
  
 Employee emp = new Employee();  
 emp.setId(2);  
 emp.setFirstName("Jane");  
 emp.setLastName("Smith");  
 emp.setSalary(6000);  
  
 session.save(emp);  
 tx.commit();  
 session.close();  
 }  
}

Output

Hibernate: insert into EMPLOYEE (FIRST\_NAME, LAST\_NAME, SALARY, ID) values (?, ?, ?, ?)

# 4. Conclusion

Both XML and Annotation-based configurations achieve the same result using Hibernate ORM. Annotation-based configuration is often preferred in modern development due to its simplicity and inline definition. However, XML configuration offers more flexibility in large enterprise applications where separation of concerns is desired.  
  
This exercise demonstrates the setup, configuration, and execution of ORM using both methods effectively.