**Difference between JPA, Hibernate and Spring Data JPA**

Java Persistence API (JPA)

* JSR 338 Specification for persisting, reading and managing data from Java objects
* Does not contain concrete implementation of the specification
* Hibernate is one of the implementation of JPA

Hibernate

* ORM Tool that implements JPA

Spring Data JPA

* Does not have JPA implementation, but reduces boiler plate code
* This is another level of abstraction over JPA implementation provider like Hibernate
* Manages transactions

**Hibernate Project in Eclipse:**

**pom.xml:-**

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example.hibernate</groupId>

<artifactId>hibernate-demo</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- Hibernate ORM (Jakarta Compatible) -->

<dependency>

<groupId>org.hibernate.orm</groupId>

<artifactId>hibernate-core</artifactId>

<version>6.4.4.Final</version>

</dependency>

<!-- Jakarta Persistence API -->

<dependency>

<groupId>jakarta.persistence</groupId>

<artifactId>jakarta.persistence-api</artifactId>

<version>3.1.0</version>

</dependency>

<!-- H2 Database -->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<version>2.1.214</version>

<scope>runtime</scope>

</dependency>

<!-- Logging backend for Hibernate -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-simple</artifactId>

<version>2.0.9</version>

</dependency>

</dependencies>

</project>

**hibernate.cfg.xml:**

<!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>

<property name="hibernate.connection.driver\_class">org.h2.Driver</property>

<property name="hibernate.connection.url">jdbc:h2:mem:testdb</property>

<property name="hibernate.connection.username">sa</property>

<property name="hibernate.connection.password"></property>

<property name="hibernate.hbm2ddl.auto">update</property>

<property name="show\_sql">true</property>

<mapping class="com.example.hibernate.Employee"/>

</session-factory>

</hibernate-configuration>

**Employee.java**

package com.example.hibernate;

import jakarta.persistence.\*;

@Entity

@Table(name = "employees")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.*IDENTITY*)

private int id;

private String name;

private String department;

public Employee() {}

public Employee(String name, String department) {

this.name = name;

this.department = department;

}

// Getters and setters

public int getId() { return id; }

public void setId(int id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public String getDepartment() { return department; }

public void setDepartment(String department) { this.department = department; }

}

**HibernateUtil.java**

package com.example.hibernate;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

public class HibernateUtil {

private static final SessionFactory sessionFactory = buildSessionFactory();

private static SessionFactory buildSessionFactory() {

try {

return new Configuration().configure().buildSessionFactory();

} catch (Throwable ex) {

throw new ExceptionInInitializerError(ex);

}

}

public static SessionFactory getSessionFactory() {

return sessionFactory;

}

}

**MainApp.java**

package com.example.hibernate;

import org.hibernate.Session;

import org.hibernate.Transaction;

public class MainApp {

public static void main(String[] args) {

Session session = HibernateUtil.getSessionFactory().openSession();

Transaction tx = session.beginTransaction();

Employee emp = new Employee("John Doe", "IT");

session.save(emp);

tx.commit();

session.close();

System.out.println("Employee saved!");

}

}

Output:

