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

**Java Persistence API (JPA)**

* JPA is a Java specification (JSR 338) for object-relational mapping (ORM).
* It defines standard APIs to persist, retrieve, update, and delete data using Java objects.
* JPA itself does not provide any implementation — it’s just an interface (a set of rules).
* It is vendor-neutral, allowing developers to switch between ORM providers.
* Popular implementations of JPA: Hibernate, EclipseLink, OpenJPA.

**Hibernate**

* Hibernate is a concrete ORM tool that implements the JPA specification.
* It also provides extra features beyond JPA like caching, dirty checking, etc.
* Hibernate allows both:
  + Using JPA annotations (standard)
  + Using Hibernate-native APIs (proprietary features)
* Requires more manual setup (session factory, transactions) if used standalone.

**Spring Data JPA**

* Spring Data JPA is a Spring project that sits on top of JPA and simplifies its usage.
* It does not implement JPA but instead relies on a JPA provider (e.g., Hibernate).
* It removes boilerplate code by providing repository interfaces like JpaRepository.
* Automatically manages:
  + Transactions
  + EntityManager
  + Common queries like save(), findById(), etc.
* Promotes rapid development with minimal configuration**.**

**Code Comparison**

**application.properties**

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn

spring.datasource.username=root

spring.datasource.password=Deepu\_123\*

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

**Pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant</groupId>

<artifactId>orm-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>orm-learn</name>

<description>Spring Boot Hibernate JPA Example</description>

<!-- Parent spring-boot -->

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.5.3</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<!-- Java 21 compatibility -->

<properties>

<java.version>21</java.version>

</properties>

<!-- Dependencies -->

<dependencies>

<!-- Spring Boot Starter Web -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- Spring Boot Starter Data JPA -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<!-- MySQL Connector -->

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</depndency>

<!-- Spring Boot DevTools (optional for hot reload) -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

<!-- Spring Boot Starter Test (for JUnit testing) -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<!-- Plugin for compiling -->

<build>

<plugins>

<!-- Spring Boot Maven Plugin -->

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**Spring Data JPA**

**1.Employee.java**

package com.cognizant.ormlearn.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "employee")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.*IDENTITY*)

private int id;

private String name;

private double salary;

private boolean permanent;

// 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 double getSalary() { return salary; }

public void setSalary(double salary) { this.salary = salary; }

public boolean isPermanent() { return permanent; }

public void setPermanent(boolean permanent) { this.permanent = permanent; }

}

**2.EmployeeRepository.java**

package com.cognizant.ormlearn.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.cognizant.ormlearn.model.Employee;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**3.EmployeeService.java**

package com.cognizant.ormlearn.service;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.cognizant.ormlearn.model.Employee;

import com.cognizant.ormlearn.repository.EmployeeRepository;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}

**4.App.java**

package com.cognizant.ormlearn;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.beans.factory.annotation.Autowired;

import com.cognizant.ormlearn.model.Employee;

import com.cognizant.ormlearn.service.EmployeeService;

@SpringBootApplication

public class App implements CommandLineRunner {

@Autowired

private EmployeeService employeeService;

public static void main(String[] args) {

SpringApplication.run(App.class, args);

}

@Override

public void run(String... args) {

Employee emp = new Employee();

emp.setName("Deepika");

emp.setSalary(50000.0);

emp.setPermanent(true);

employeeService.addEmployee(emp);

System.out.println("Employee added successfully!");

}

}

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**HIBERNATE**

**1.hibernate.cfg.xml**

<hibernate-configuration>

<session-factory>

<property name=*"hibernate.connection.driver\_class"*>com.mysql.cj.jdbc.Driver</property>

<property name=*"hibernate.connection.url"*>jdbc:mysql://localhost:3306/ormlearn</property>

<property name=*"hibernate.connection.username"*>root</property>

<property name=*"hibernate.connection.password"*>Deepu\_123\*</property>

<property name=*"hibernate.dialect"*>org.hibernate.dialect.MySQLDialect</property>

<property name=*"hibernate.show\_sql"*>true</property>

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

</session-factory>

</hibernate-configuration>

**2.Employee2.java**

package com.cognizant.ormlearn;

import jakarta.persistence.\*;

@Entity

@Table(name = "employee2")

public class Employee2 {

@Id

@GeneratedValue(strategy = GenerationType.*IDENTITY*)

private int id;

private String name;

private double salary;

private boolean permanent;

// 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 double getSalary() {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

public boolean isPermanent() {

return permanent;

}

public void setPermanent(boolean permanent) {

this.permanent = permanent;

}

}

**2.HibernateUtil.java**

package com.cognizant.ormlearn;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

public class HibernateUtil {

private static final SessionFactory sessionFactory;

static {

try {

Configuration configuration = new Configuration();

configuration.configure("hibernate.cfg.xml");

configuration.addAnnotatedClass(Employee2.class);

sessionFactory = configuration.buildSessionFactory();

} catch (Throwable ex) {

System.err.println("Initial SessionFactory creation failed." + ex);

throw new ExceptionInInitializerError(ex);

}

}

public static SessionFactory getSessionFactory() {

return sessionFactory;

}

}

**3.HibernateEmployeeDemo.java**

package com.cognizant.ormlearn;

import org.hibernate.Session;

import org.hibernate.Transaction;

public class HibernateEmployeeDemo {

public static void main(String[] args) {

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

Transaction tx = null;

try {

tx = session.beginTransaction();

Employee2 emp = new Employee2();

emp.setName("Manjula");

emp.setSalary(70000.0);

emp.setPermanent(true);

session.save(emp);

tx.commit();

System.out.println("Hibernate: Employee2 added successfully!");

} catch (Exception e) {

if (tx != null) tx.rollback();

e.printStackTrace();

} finally {

session.close();

}

}

}

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.