Explain the difference between Java Persistence API, Hibernate and Spring Data JPA

**1. JPA (Java Persistence API)**

* A specification (like an interface) provided by Java for object-relational mapping (ORM).
  + Defines how Java objects should be persisted in a relational database.
  + Does not provide implementation.
  + APIs: EntityManager, @Entity, @Id, @OneToMany, etc.
  + Needs a provider like Hibernate to work.

### ****2. Hibernate****

* A **popular implementation** of JPA and also a **standalone ORM framework.**
  + Provides the actual **code that persists Java objects** to the database.
  + Offers **additional features** beyond JPA (e.g., caching, custom annotations).
  + Can be used **with or without** JPA.

### 3. ****Spring Data JPA****

* A **Spring framework abstraction** on top of JPA (usually Hibernate underneath) to **reduce boilerplate.**
  + **Automatically generates** DAO/repository code (like findByName()).
  + Uses **JpaRepository, CrudRepository**, etc.
  + Integrated with **Spring Boot** and **Spring context**.

Example:

1.Create Maven Project:

2. Add Spring Boot Dependencies as-

<parent>

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

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

<version>3.2.0</version>

</parent>

<dependencies>

<!-- Spring Boot Starter JPA -->

<dependency>

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

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

</dependency>

<!-- H2 In-Memory Database -->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<!-- Spring Boot Starter -->

<dependency>

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

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

</dependency>

</dependencies>

<properties>

<java.version>17</java.version>

</properties>

3. Create Entity Class as-

package com.example.demo.model;

import jakarta.persistence.\*;

@Entity

Pu,blic class Student {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

public Student() {}

public Student(String name) { this.name = name; }

}

4. Create Repository Interface as –

package com.example.demo.repository;

import com.example.demo.model.Student;

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

public interface StudentRepository extends JpaRepository<Student, Long> {

}

5. Create Data Loader as –

package com.example.demo.runner;

import com.example.demo.model.Student;

import com.example.demo.repository.StudentRepository;

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

@Component

public class DataLoader implements CommandLineRunner {

private final StudentRepository repo;

public DataLoader(StudentRepository repo) {

this.repo = repo;

}

@Override

public void run(String... args) {

repo.save(new Student("Alice"));

repo.findAll().forEach(s -> System.out.println("Student: " + s.getName()));

}

}

6. Create Main Class as –

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.run(DemoApplication.class, args);

}

}

7. Configure Application as –

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driver-class-name=org.h2.Driver

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

OUTPUT :

