WEEK 3  
Topic 1 Spring Core And Maven

Exercise 1 Configuring a Basic Spring Application

Code   
  
//ShoppingMain.java

package org.shopping;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class ShoppingMain {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext-shopping.xml");

ShoppingApp app = context.getBean("shoppingApp", ShoppingApp.class);

app.start();

}

}

//ProductApp.java

package org.shopping;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

@SpringBootApplication

public class ProductApp {

public static void main(String[] args) {

SpringApplication.run(ProductApp.class, args);

}

@Bean

CommandLineRunner run(ProductRepository repository) {

return args -> {

Product p = new Product();

p.setId(1L);

p.setName("Laptop");

p.setPrice(50000);

repository.save(p);

repository.findAll().forEach(product ->

System.out.println(product.getName() + " - ₹" + product.getPrice()));

};

}

}

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

<modelVersion>4.0.0</modelVersion>

<groupId>org.example</groupId>

<artifactId>spring-full-demo</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.22</version>

</dependency>

<dependency>

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

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

<version>2.7.10</version>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<version>1.4.200</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>javax.xml.bind</groupId>

<artifactId>jaxb-api</artifactId>

<version>2.3.1</version>

</dependency>

<dependency>

<groupId>org.glassfish.jaxb</groupId>

<artifactId>jaxb-runtime</artifactId>

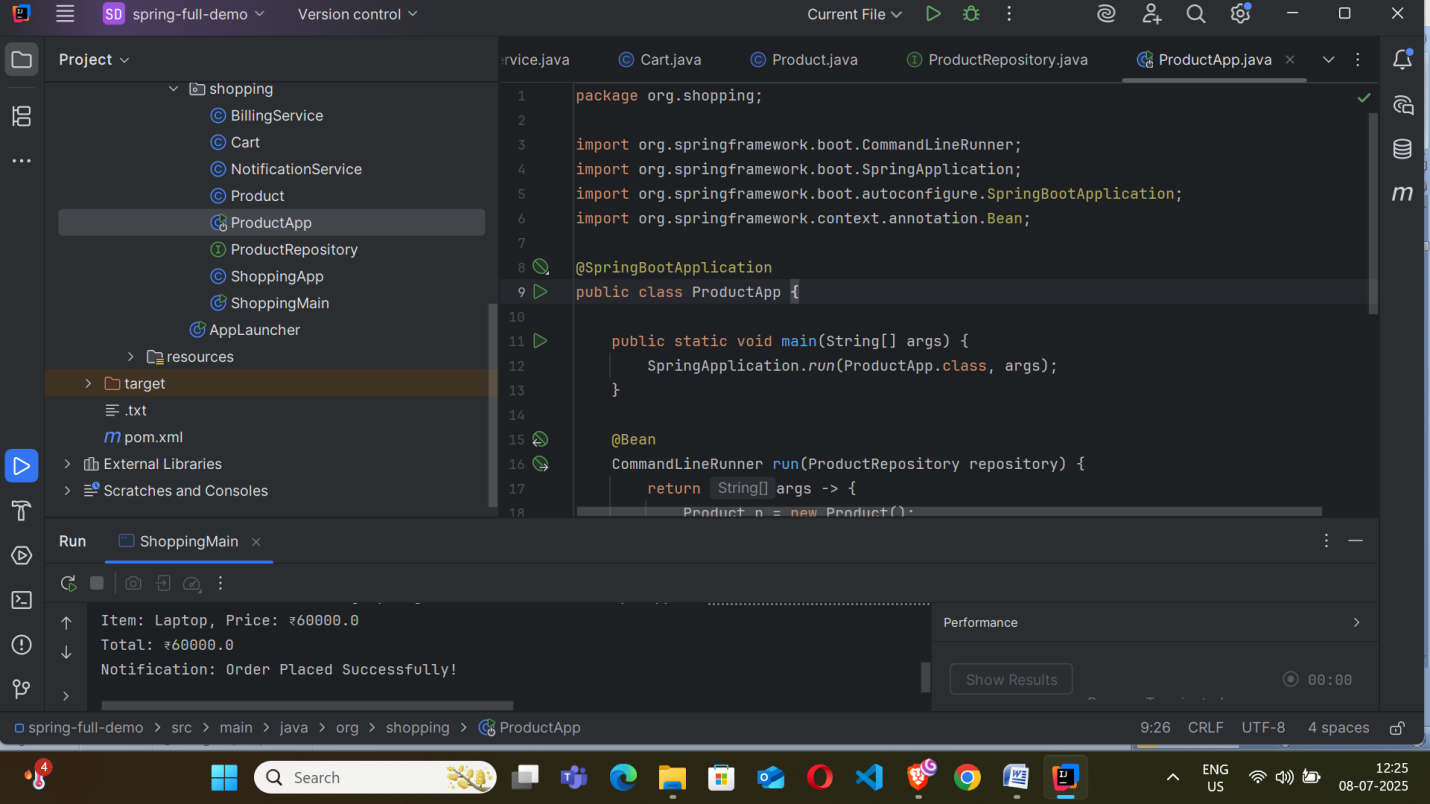
<version>2.3.1</version>

</dependency>

</dependencies>

</project>

O/P



Exercise 2: Implementing Dependency Injection  
Code :  
//ProductApp.java

package org.shopping;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

@SpringBootApplication

public class ProductApp {

public static void main(String[] args) {

SpringApplication.run(ProductApp.class, args);

}

@Bean

CommandLineRunner run(ProductRepository repository) {

return args -> {

Product p = new Product();

p.setId(1L);

p.setName("Laptop");

p.setPrice(50000);

repository.save(p);

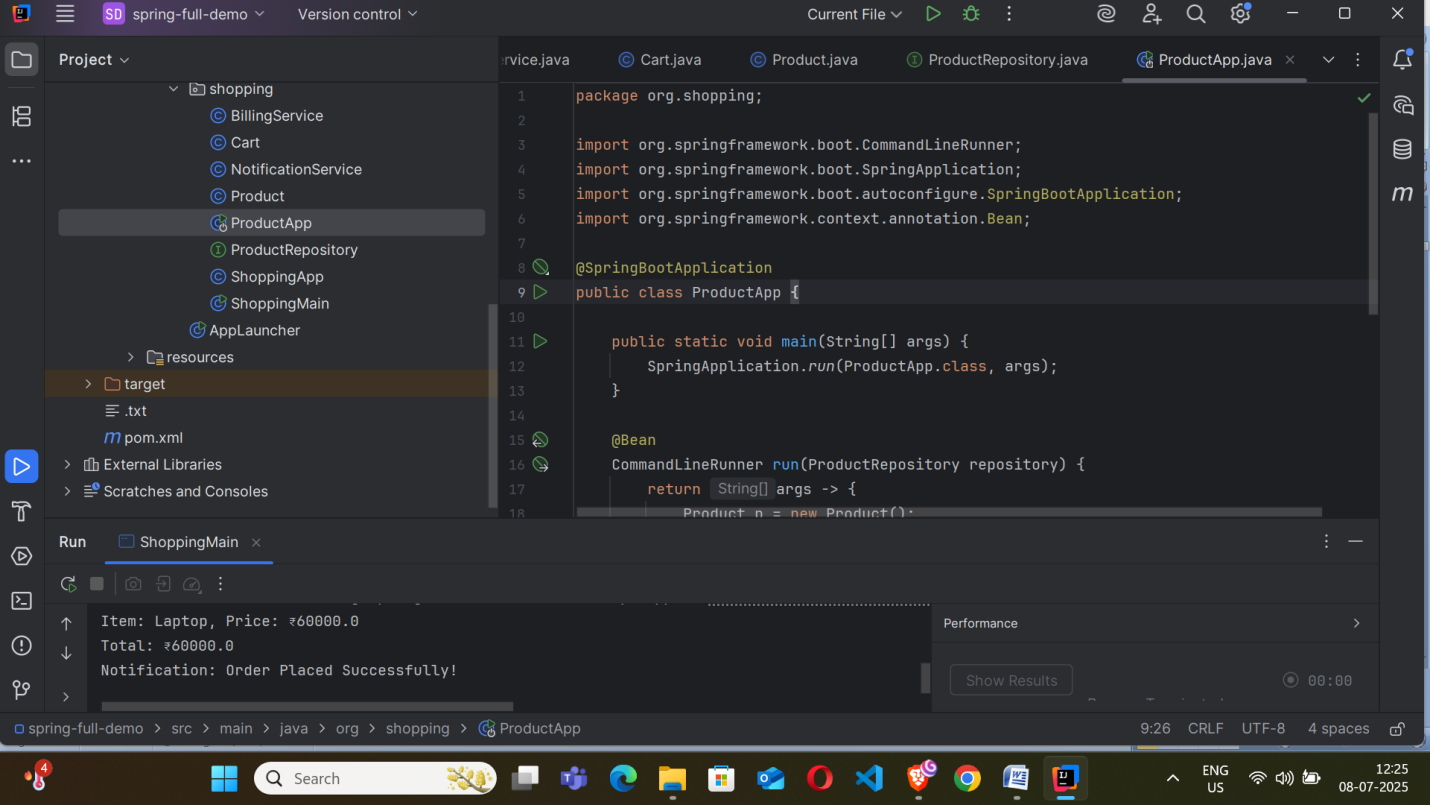
repository.findAll().forEach(product ->

System.out.println(product.getName() + " - ₹" + product.getPrice()));

};

}

}

O/P:  
  


Exercise 3: Creating and Configuring a Maven Project  
Code :  
//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 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>org.example</groupId>

<artifactId>spring-full-demo</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.22</version>

</dependency>

<dependency>

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

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

<version>2.7.10</version>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<version>1.4.200</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>javax.xml.bind</groupId>

<artifactId>jaxb-api</artifactId>

<version>2.3.1</version>

</dependency>

<dependency>

<groupId>org.glassfish.jaxb</groupId>

<artifactId>jaxb-runtime</artifactId>

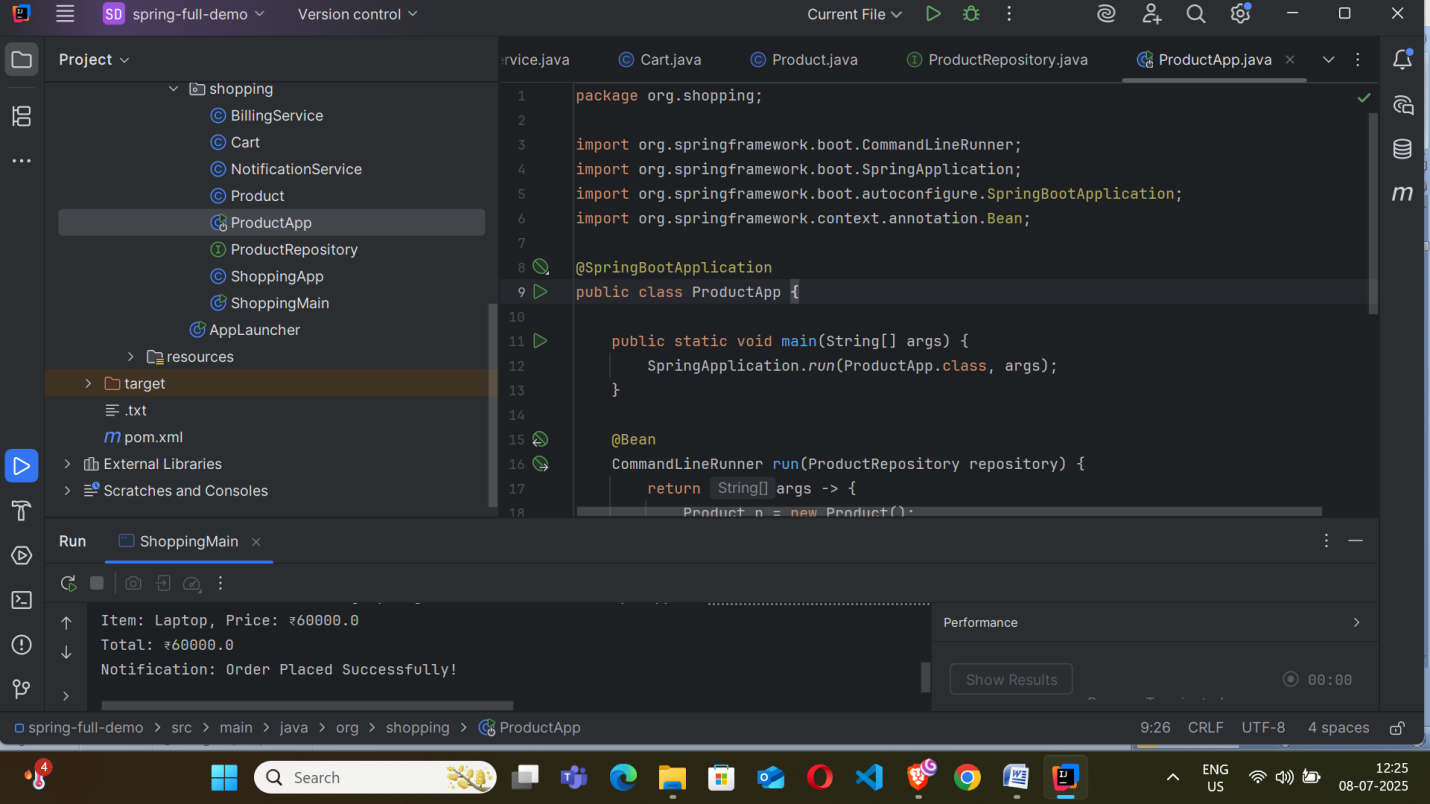
<version>2.3.1</version>

</dependency>

</dependencies>

</project>

O/P



` Topic 2 Spring\_Data\_JPA

Exercise 1 Spring Data JPA - Quick Example  
  
Code :

//Product.java

package org.shopping;

import javax.persistence.Entity;

import javax.persistence.Id;

@Entity

public class Product {

@Id

private Long id;

private String name;

private double price;

public Long getId() { return id; }

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

public String getName() { return name; }

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

public double getPrice() { return price; }

public void setPrice(double price) { this.price = price; }

}

**//**ProductApp.java

package org.shopping;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

@SpringBootApplication

public class ProductApp {

public static void main(String[] args) {

SpringApplication.run(ProductApp.class, args);

}

@Bean

CommandLineRunner run(ProductRepository repository) {

return args -> {

Product p = new Product();

p.setId(1L);

p.setName("Laptop");

p.setPrice(50000);

repository.save(p);

repository.findAll().forEach(product ->

System.out.println(product.getName() + " - ₹" + product.getPrice()));

};

}

}

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

<modelVersion>4.0.0</modelVersion>

<groupId>org.example</groupId>

<artifactId>spring-full-demo</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.22</version>

</dependency>

<dependency>

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

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

<version>2.7.10</version>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<version>1.4.200</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>javax.xml.bind</groupId>

<artifactId>jaxb-api</artifactId>

<version>2.3.1</version>

</dependency>

<dependency>

<groupId>org.glassfish.jaxb</groupId>

<artifactId>jaxb-runtime</artifactId>

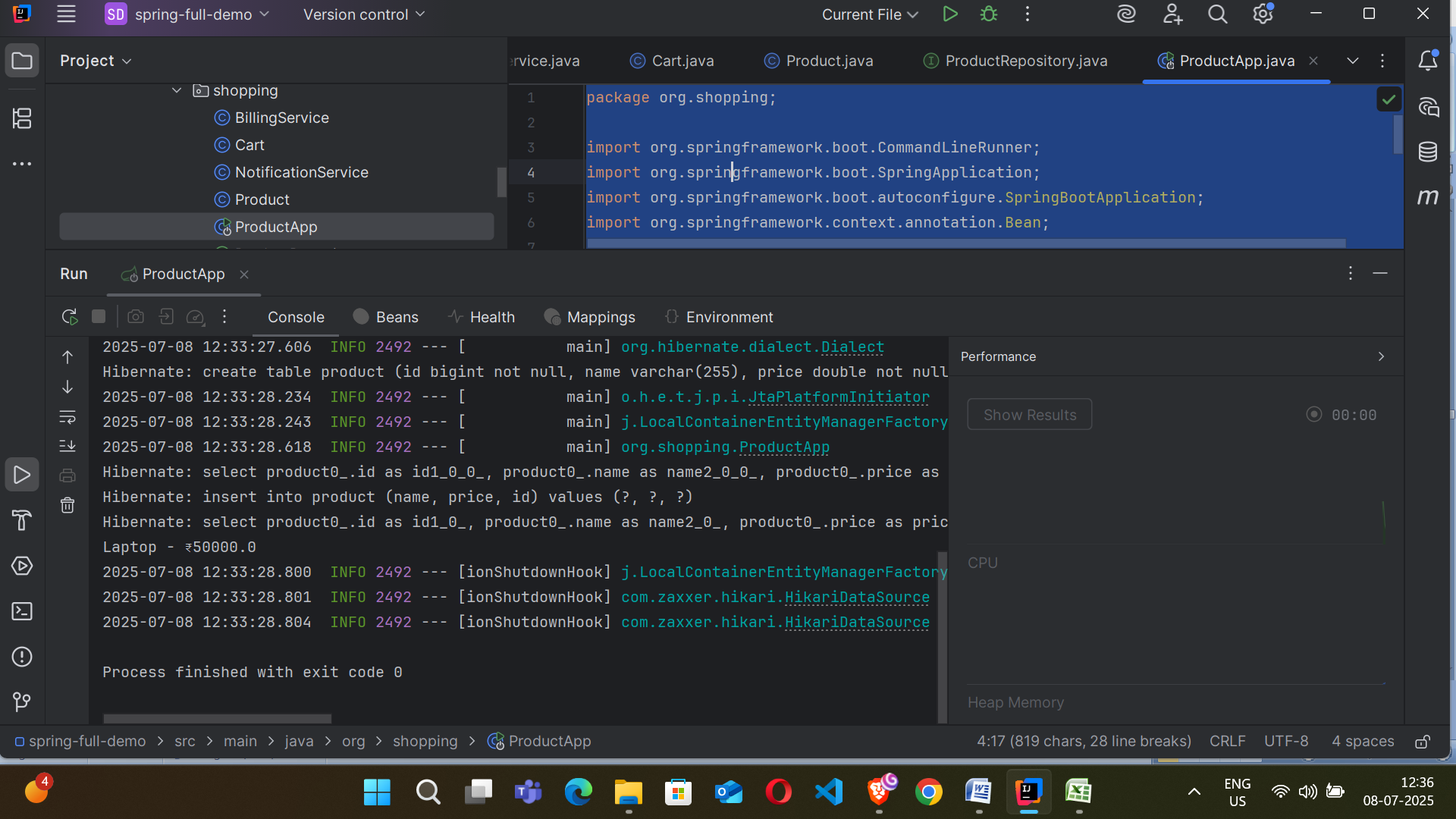
<version>2.3.1</version>

</dependency>

</dependencies>

</project>

O/P:

  
  
Exercise 2:

Difference between JPA, Hibernate and Spring Data JPA

**//.txt**

1. JPA (Java Persistence API):

* JPA is a specification, not an implementation.
* It defines a standard for object-relational mapping (ORM) in Java.
* It allows developers to map Java objects to database tables using annotations like @Entity, @Id, etc.
* It requires more manual configuration, such as setting up EntityManager, transactions, and writing queries using JPQL.
* It does not provide any persistence logic by itself.

2. Hibernate:

* Hibernate is an implementation of JPA with additional features.
* It provides an ORM framework to map Java classes to database tables.
* It supports lazy loading, caching, dirty checking, and more.
* Developers can use JPQL, native SQL, and Hibernate Criteria API to write queries.
* It reduces some boilerplate code required in plain JPA.
* It manages sessions using SessionFactory.

3. Spring Data JPA:

* Spring Data JPA is a framework provided by Spring that builds on top of JPA and Hibernate.
* It simplifies data access by providing ready-to-use repository interfaces like JpaRepository or CrudRepository.
* It can automatically generate queries based on method names (e.g., findByName(String name)), reducing the need to write boilerplate code.
* It is tightly integrated with Spring Boot, and often requires no manual configuration for setting up the data layer.
* It still uses Hibernate or any JPA provider internally, but hides much of the complexity from the developer.

Summary:

1. JPA defines what to do (it's just an API/spec).
2. Hibernate is one of the tools that does it, with extra features.
3. Spring Data JPA makes the developer's job easier by automating and simplifying common tasks using JPA/Hibernate under the hood.