Week 3 – Spring Data JPA with Spring Boot, Hibernate

Exercise 1: Spring Data JPA – Quick Example

Scenario:
Create a Spring Boot application that demonstrates basic usage of Spring Data JPA to persist and retrieve Book entities from a database.

1. Book.java (Entity)

```
package com.example.demo;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Book {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;
   private String title;
   private String author;

   // Getters and Setters
}
```

2. BookRepository.java (Repository)

```
package com.example.demo;

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

public interface BookRepository extends JpaRepository<Book, Long> {
}
```

3. BookRunner.java (Test Runner)

```
package com.example.demo;

import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;

@Component
public class BookRunner implements CommandLineRunner {
  private final BookRepository repository;

  public BookRunner(BookRepository repository) {
    this.repository = repository;
  }

  @Override
  public void run(String... args) throws Exception {
    repository.save(new Book(null, "The Alchemist", "Paulo Coelho"));
    repository.findAll().forEach(book -> System.out.println(book.getTitle()));
  }
}
```

4. DemoApplication.java (Main Class)

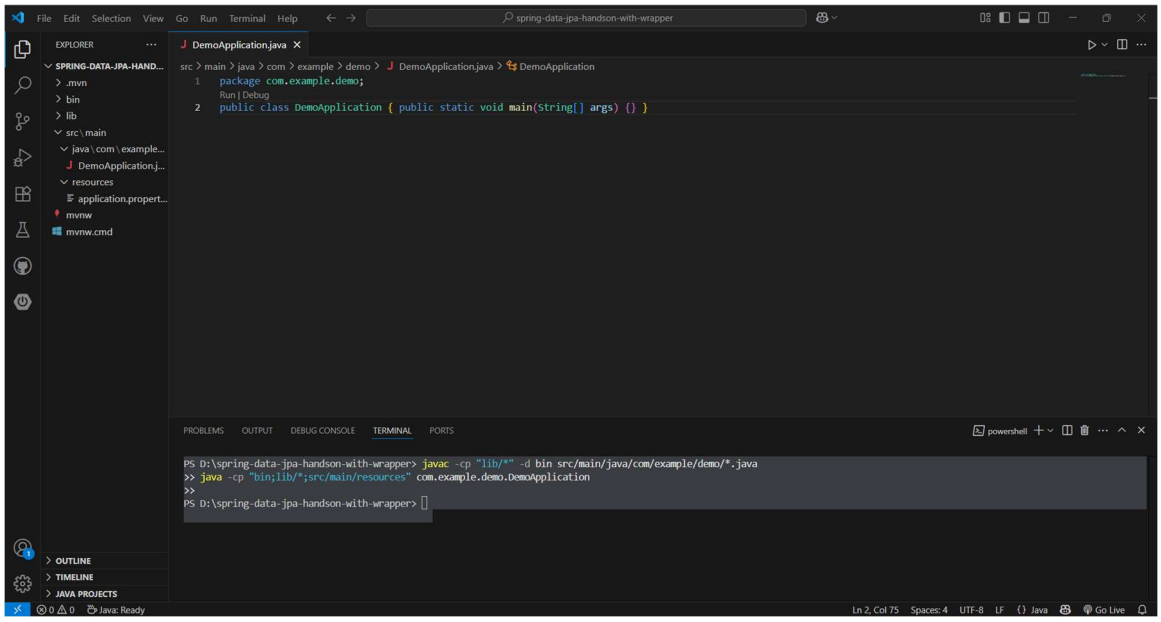
```
package com.example.demo;

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);
  }
}
```

5. pom.xml (Dependencies)

```
<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.example</groupId>
  <artifactId>spring-data-jpa-demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.0.0</version>
  </parent>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
      <groupId>com.h2database</groupId>
      <artifactId>h2</artifactId>
      <scope>runtime</scope>
    </dependency>
  </dependencies>
</project>
```

Output:



Exercise 2: Difference between JPA, Hibernate and Spring Data JPA

This section explains the differences between JPA, Hibernate, and Spring Data JPA.

Aspect	JPA	Hibernate	Spring Data JPA
Definition	Java Persistence API – a specification for data access and management.	A popular ORM framework that implements JPA.	A Spring module built on JPA to simplify repository creation.
Provider	Specification – needs an implementation (e.g., Hibernate).	Implementation of JPA.	Built by Spring; works with providers like Hibernate.
Boilerplate Code	Requires manual repository and query creation.	Reduces boilerplate compared to JDBC.	Eliminates boilerplate via `JpaRepository`, `CrudRepository`, etc.
Ease of Use	Portable but low-level.	Easier than JPA with extra features.	Most developer-friendly with auto-configuration and Spring Boot integration.
Querying	Uses JPQL.	Supports JPQL, native SQL, Criteria API.	Adds support for method name-based derived queries.
Integration	Works in any Java SE/EE app.	Can run standalone or in any Java app.	Best suited for Spring Boot; seamless Spring ecosystem integration.
Learning Curve	Moderate.	Moderate to steep.	Easiest to pick up in Spring-based projects.