WEEK 3:

**Spring Data JPA - Quick Example**

**Library Management:**

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.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<!-- Spring Core -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

<!-- Spring Context for core DI container -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

<!-- Spring AOP for Aspect-Oriented Programming -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.34</version>

</dependency>

<!-- Spring Web MVC for building web applications -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.34</version>

</dependency>

</dependencies>

</project>

ApplicationContext.xml

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

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="

http://www.springframework.org/schema/beans

https://www.springframework.org/schema/beans/spring-beans.xsd">

<!-- Define the BookRepository bean -->

<bean id="bookRepository" class="com.library.repository.BookRepository" />

<!-- Define the BookService bean and inject BookRepository -->

<bean id="bookService" class="com.library.service.BookService">

<property name="bookRepository" ref="bookRepository" />

</bean>

</beans>

BookRepository.java

package com.library.repository;

public class BookRepository {

public void getBookList() {

System.***out***.println("Fetching book list from the repository...");

}

}

BookService.java

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// ✅ Setter method for DI

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void showBooks() {

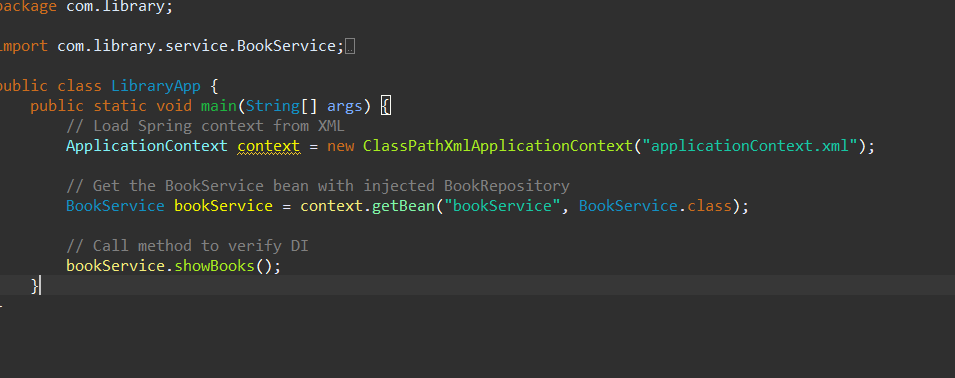
System.***out***.println("Service: Displaying books...");

bookRepository.getBookList();

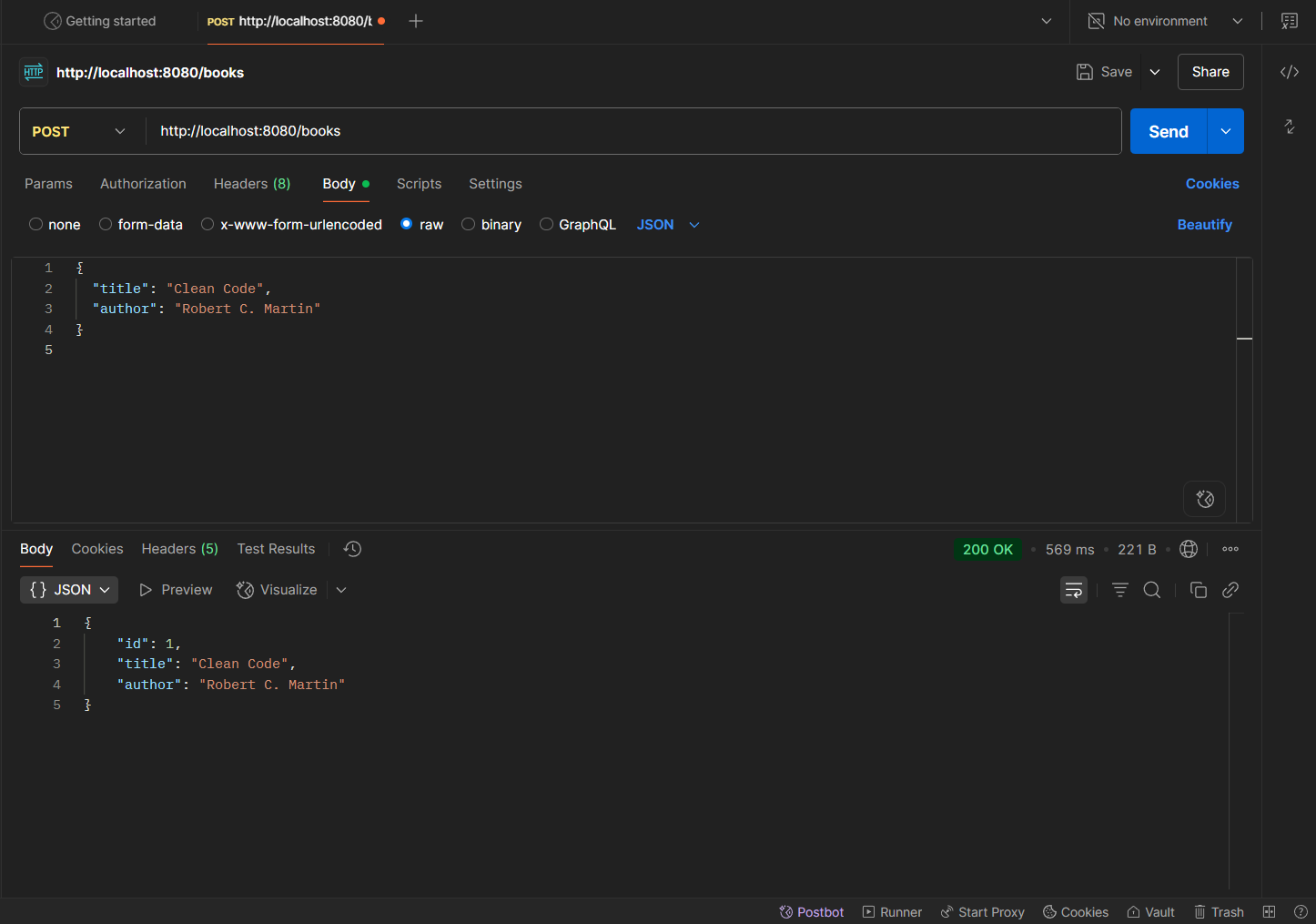
}

}

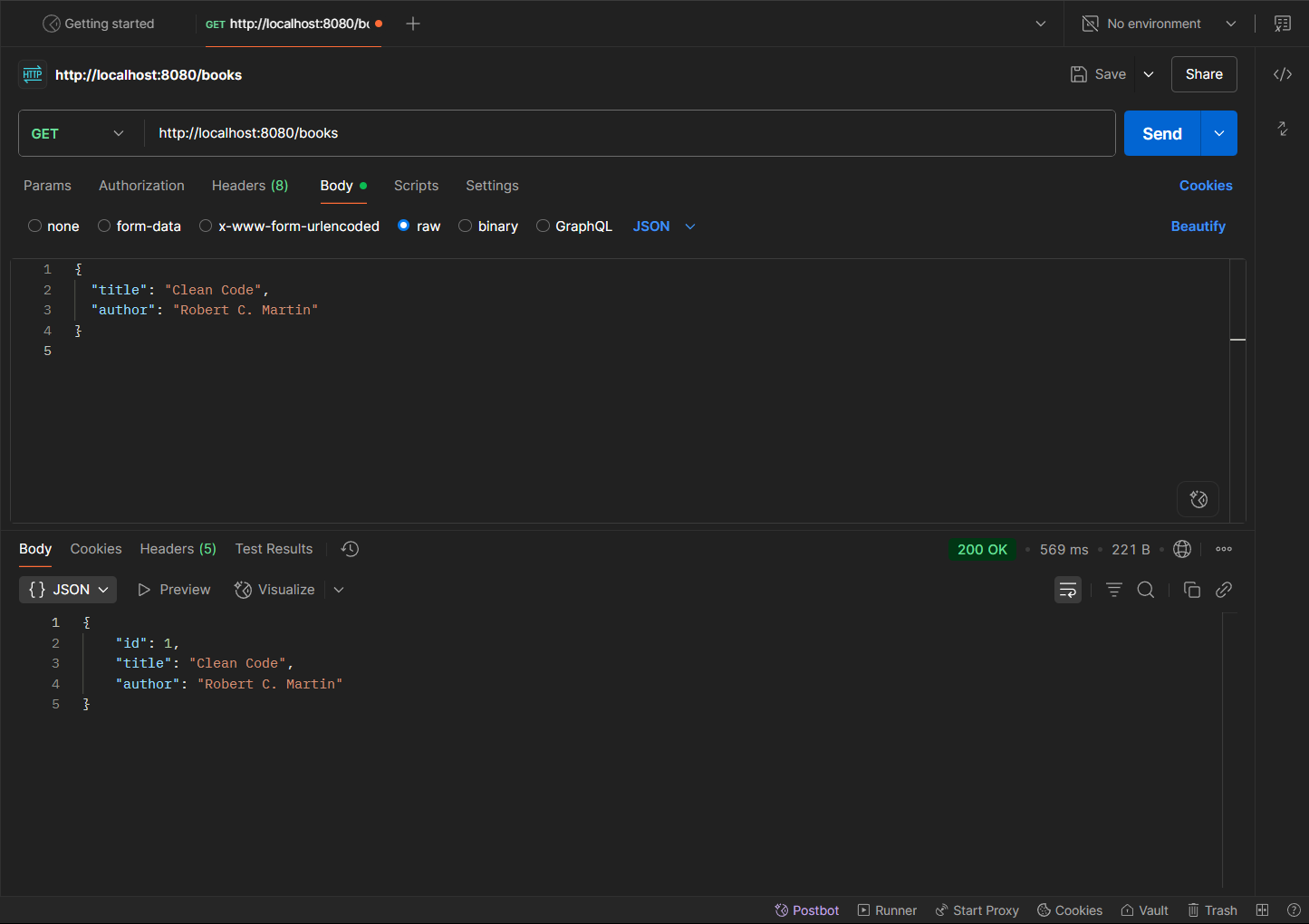
LibraryApp.java



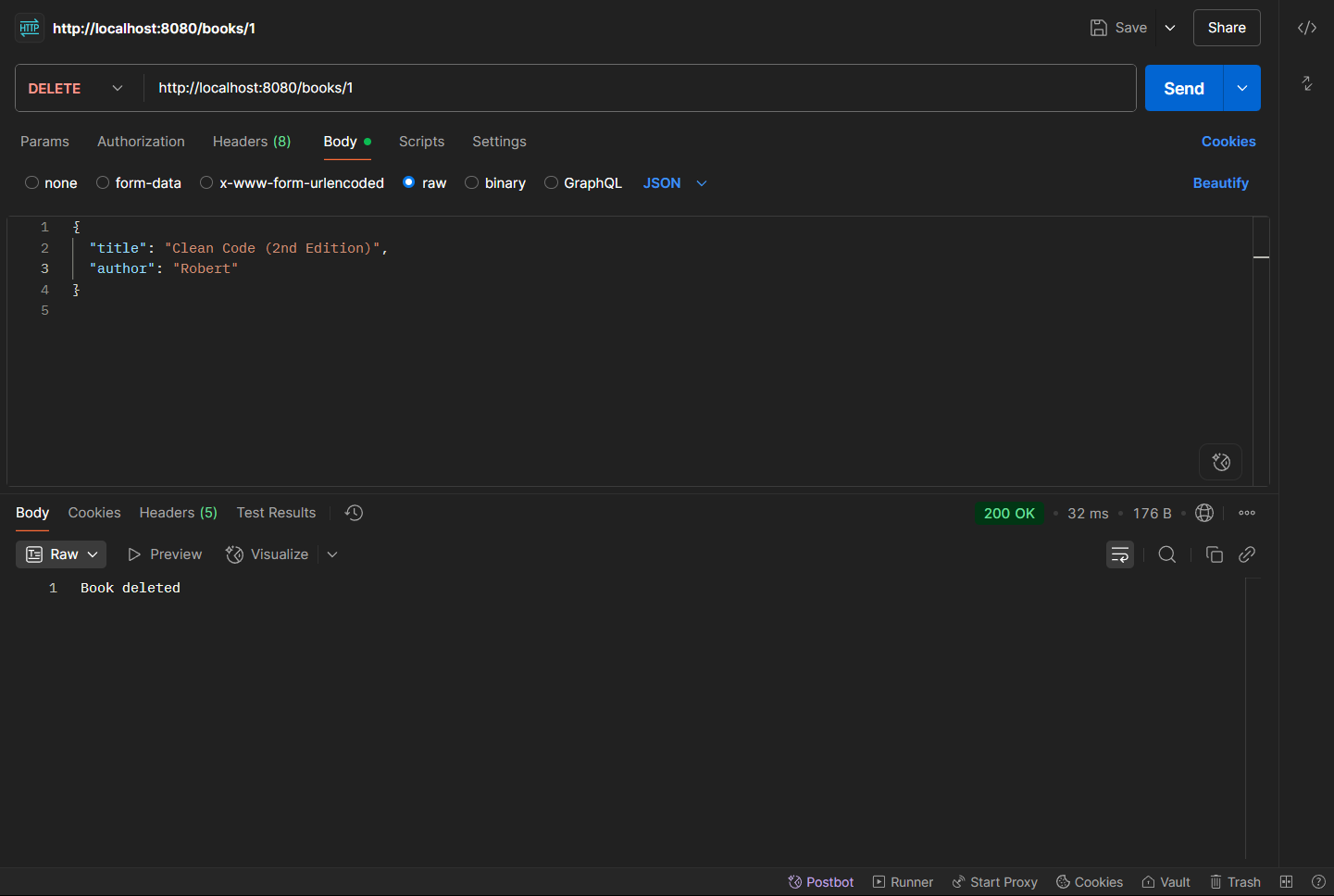
POST:



GET



DELETE:



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

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature / Aspect** | **JPA (Java Persistence API)** | **Hibernate** | **Spring Data JPA** |
| **Type** | Specification (interface) | Implementation of JPA + additional features | Abstraction on top of JPA and ORM tools like Hibernate |
| **Defined By** | Java (part of Java EE / Jakarta EE) | Red Hat / Hibernate team | Spring Framework |
| **Purpose** | Define standard APIs for ORM in Java | Provide actual ORM implementation | Simplify repository/DAO layer development |
| **Included in Java SE?** | No | No | No |
| **Provides Implementation?** | No (needs provider like Hibernate) | Yes | No (uses JPA providers like Hibernate underneath) |
| **Boilerplate Code** | Moderate | Moderate | Minimal (auto-generates query methods) |
| **Ease of Use** | Lower | Medium | High (less code, more abstraction) |
| **Query Language** | JPQL | HQL (JPQL + extras) | JPQL, Native SQL, Method Name Queries |
| **Transaction Management** | Not provided directly | Provided via JTA or manual | Integrated with Spring’s @Transactional |
| **Integration with Spring** | Needs manual configuration | Integrated via Spring ORM or JPA | Seamless integration with Spring Boot |
| **Custom Queries** | Manual via EntityManager | Manual via Session | Easy using method naming or @Query annotation |
| **Best For** | Standardized ORM layer across Java apps | Full-featured ORM with advanced control | Rapid Spring-based app development with minimal config |