**WEEK-3**

**SPRING CORE AND MAVEN**

**Exercise 9: Creating a Spring Boot Application**

**1.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>com.example</groupId>

    <artifactId>librarymanagement</artifactId>

    <version>1.0.0</version>

    <packaging>jar</packaging>

    <name>LibraryManagement</name>

    <description>Spring Boot Library Management System</description>

    <parent>

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

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

        <version>3.2.4</version>

        <relativePath/>

    </parent>

    <dependencies>

        <dependency>

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

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

        </dependency>

        <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>

    <build>

        <plugins>

            <plugin>

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

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

            </plugin>

        </plugins>

    </build>

</project>

**2.application.properties:**

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

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true

spring.jpa.hibernate.ddl-auto=update

**3.LibraryManagementApplication.java:**

package com.example.librarymanagement;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class LibraryManagementApplication {

    public static void main(String[] args) {

        SpringApplication.run(LibraryManagementApplication.class, args);

    }

}

**4.BookRepository.java:**

package com.example.librarymanagement.repository;

import com.example.librarymanagement.entity.Book;

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

public interface BookRepository extends JpaRepository<Book, Long> {

}

**5.Book.java:**

package com.example.librarymanagement.entity;

import jakarta.persstence.\*;

@Entity

public class Book {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    private String title;

    private String author;

    public Long getId() { return id; }

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

    public String getTitle() { return title; }

    public void setTitle(String title) { this.title = title; }

    public String getAuthor() { return author; }

    public void setAuthor(String author) { this.author = author; }

}

**6.BookController:**

package com.example.librarymanagement.controller;

import com.example.librarymanagement.entity.Book;

import com.example.librarymanagement.repository.BookRepository;

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

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/books")

public class BookController {

    @Autowired

    private BookRepository bookRepository;

    @GetMapping

    public List<Book> getAllBooks() {

        return bookRepository.findAll();

    }

    @GetMapping("/{id}")

    public Book getBookById(@PathVariable Long id) {

        return bookRepository.findById(id).orElse(null);

    }

    @PostMapping

    public Book addBook(@RequestBody Book book) {

        return bookRepository.save(book);

    }

    @PutMapping("/{id}")

    public Book updateBook(@PathVariable Long id, @RequestBody Book bookDetails) {

        Book book = bookRepository.findById(id).orElse(null);

        if (book != null) {

            book.setTitle(bookDetails.getTitle());

            book.setAuthor(bookDetails.getAuthor());

            return bookRepository.save(book);

        }

        return null;

    }

    @DeleteMapping("/{id}")

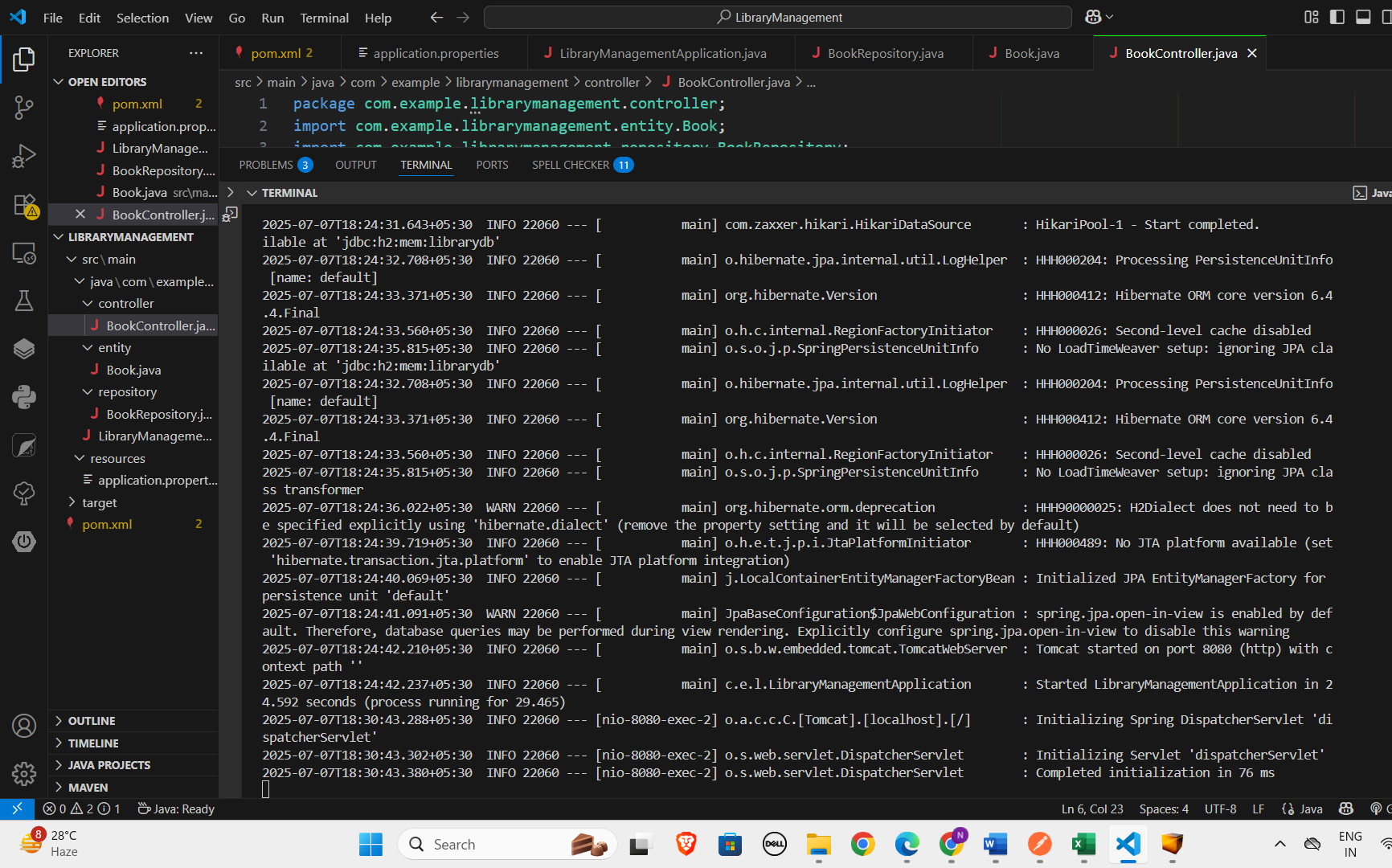
    public void deleteBook(@PathVariable Long id) {

        bookRepository.deleteById(id);

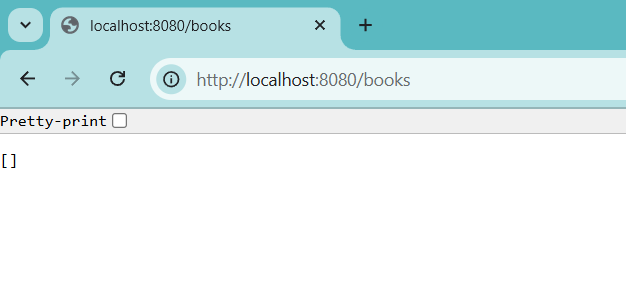
    }

}

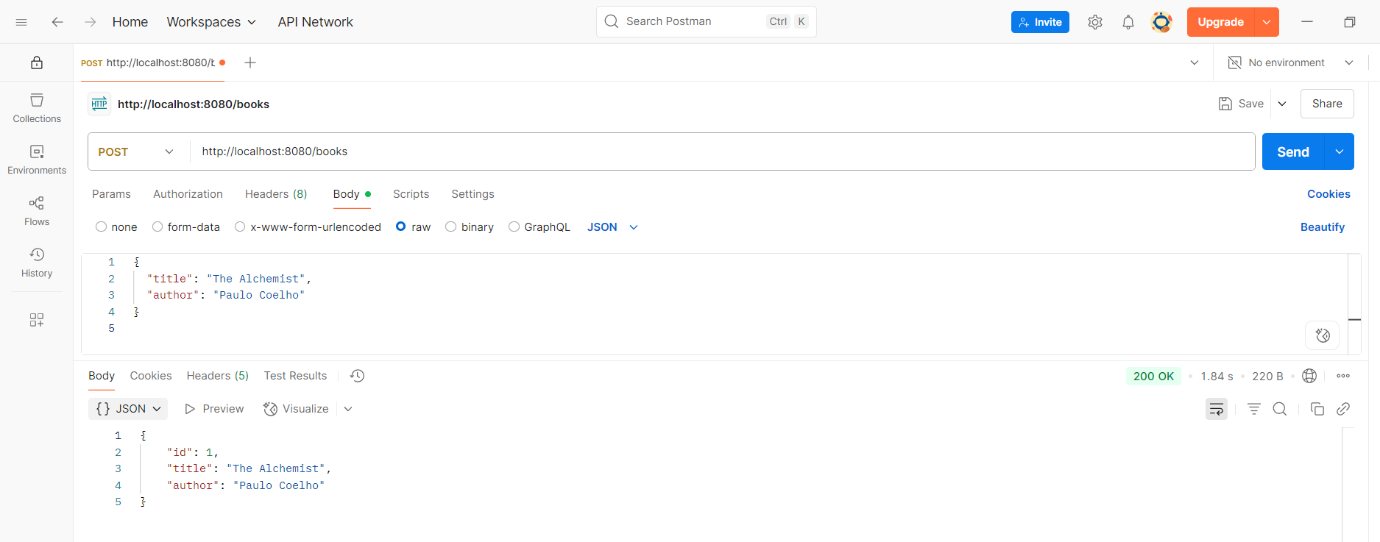
**Output:**



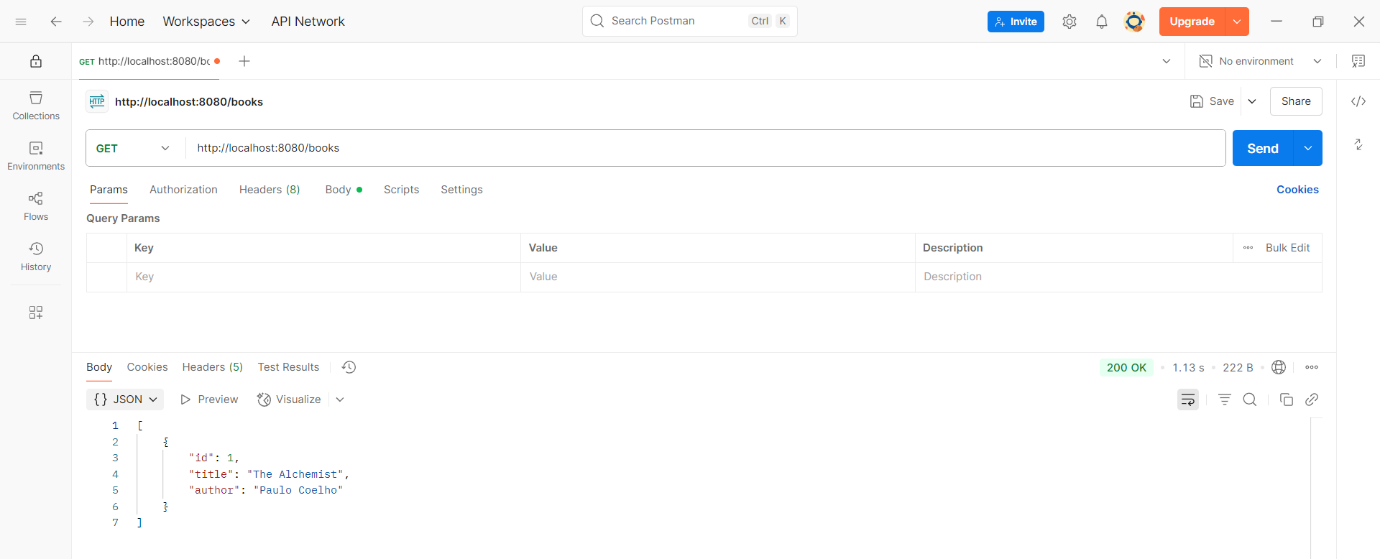
Applying GET:



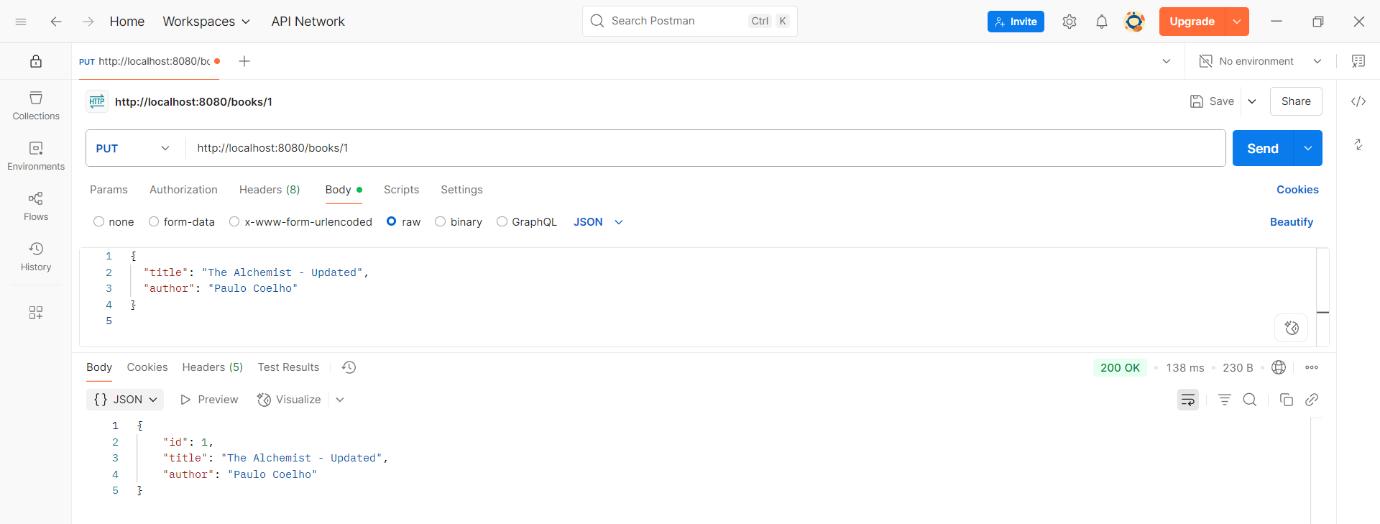
Add a Book: POST /books:



View all books: GET /books:



Update Book: PUT /books/1:



Delete Book:

