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

**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.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0.0</version>

<packaging>jar</packaging>

<parent>

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

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

<version>2.7.18</version>

</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>

<properties>

<java.version>1.8</java.version>

</properties>

</project>

**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.show-sql=true

spring.jpa.hibernate.ddl-auto=update

**Book.java**

package com.library.entity;

import javax.persistence.\*;

@Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

public Book() {

}

public Book(String title) {

this.title = title;

}

public Long getId() {

return id;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

}

**BookRepository.java**

package com.library.repository;

import com.library.entity.Book;

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

public interface BookRepository extends JpaRepository<Book, Long> {

}

**BookController.java**

package com.library.controller;

import com.library.entity.Book;

import com.library.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 repository;

@GetMapping

public List<Book> getAllBooks() {

return repository.findAll();

}

@PostMapping

public Book addBook(@RequestBody Book book) {

return repository.save(book);

}

@PutMapping("/{id}")

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

Book book = repository.findById(id)

.orElseThrow(() -> new RuntimeException("Book not found with id: " + id));

book.setTitle(bookDetails.getTitle());

return repository.save(book);

}

@DeleteMapping("/{id}")

public void deleteBook(@PathVariable Long id) {

Book book = repository.findById(id)

.orElseThrow(() -> new RuntimeException("Book not found with id: " + id));

repository.delete(book);

}

}

**LibraryManagementApplication.java**

package com.library;

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);

}

}

**BookService.java**

package com.library.service;

import com.library.entity.Book;

import com.library.repository.BookRepository;

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

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class BookService {

@Autowired

private BookRepository bookRepository;

public Book addBook(String title) {

return bookRepository.save(new Book(title));

}

public List<Book> getAllBooks() {

return bookRepository.findAll();

}

public Book updateBook(Long id, String newTitle) {

Book book = bookRepository.findById(id)

.orElseThrow(() -> new RuntimeException("Book not found with id: " + id));

book.setTitle(newTitle);

return bookRepository.save(book);

}

public void deleteBook(Long id) {

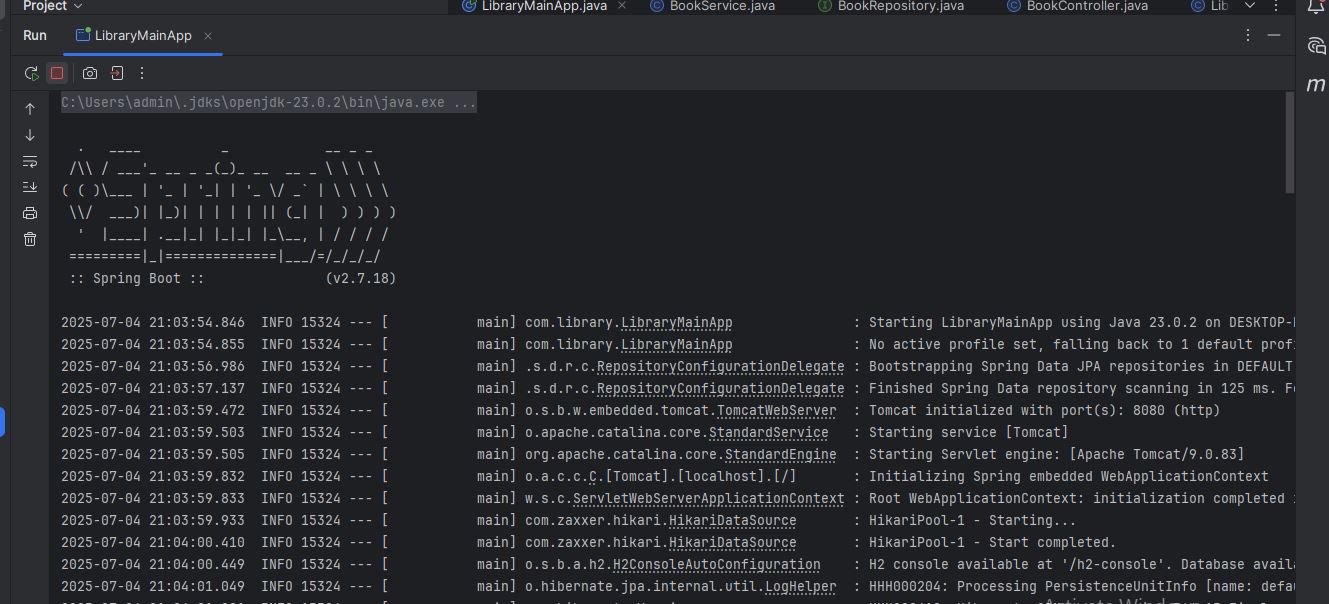
Book book = bookRepository.findById(id)

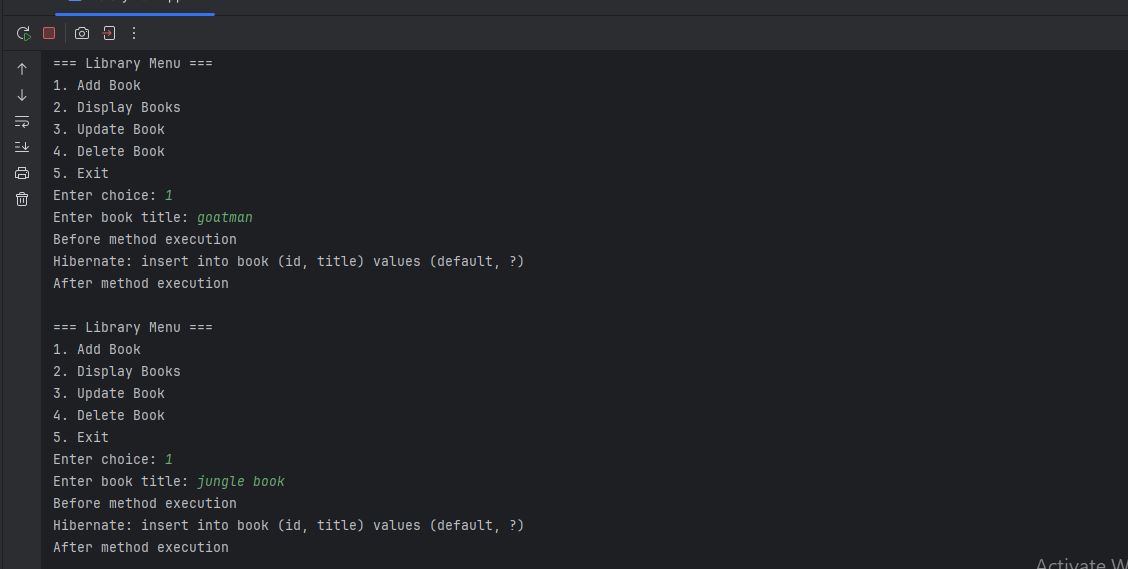
.orElseThrow(() -> new RuntimeException("Book not found with id: " + id));

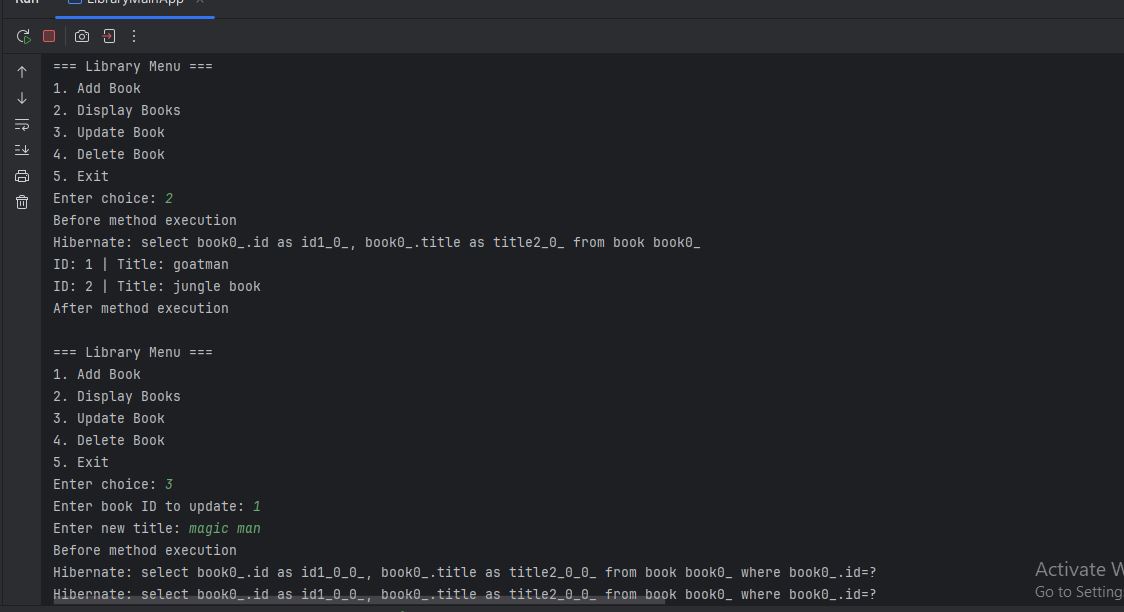
bookRepository.delete(book);

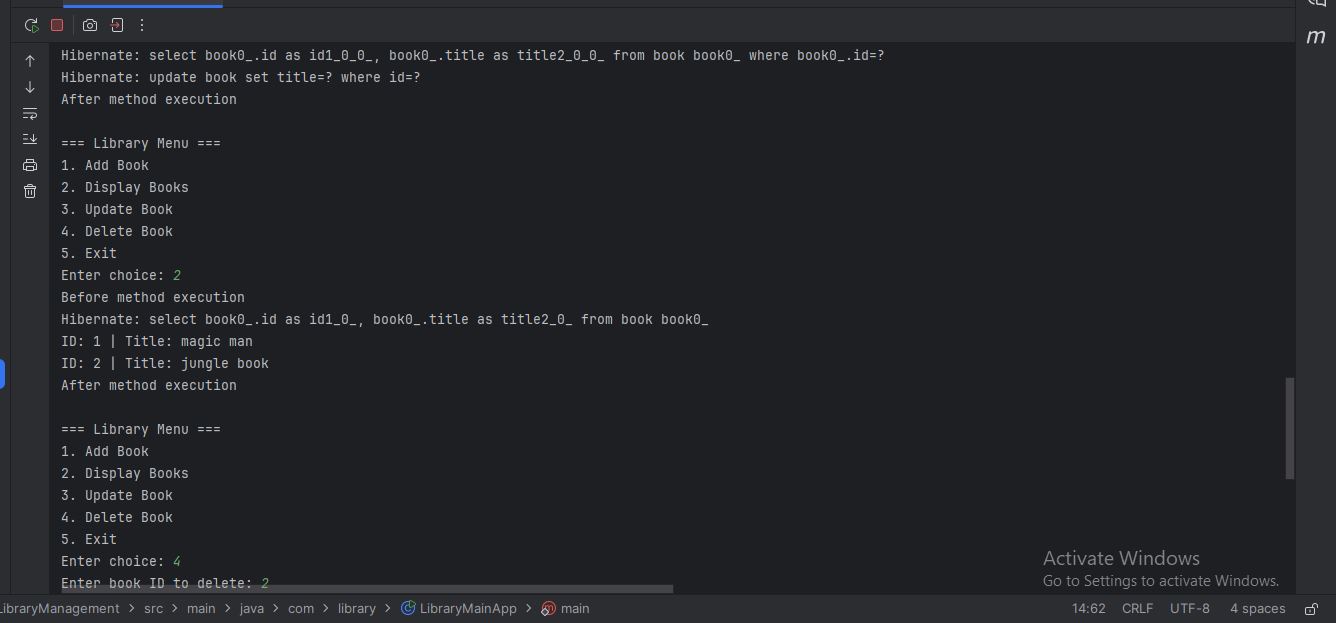
}

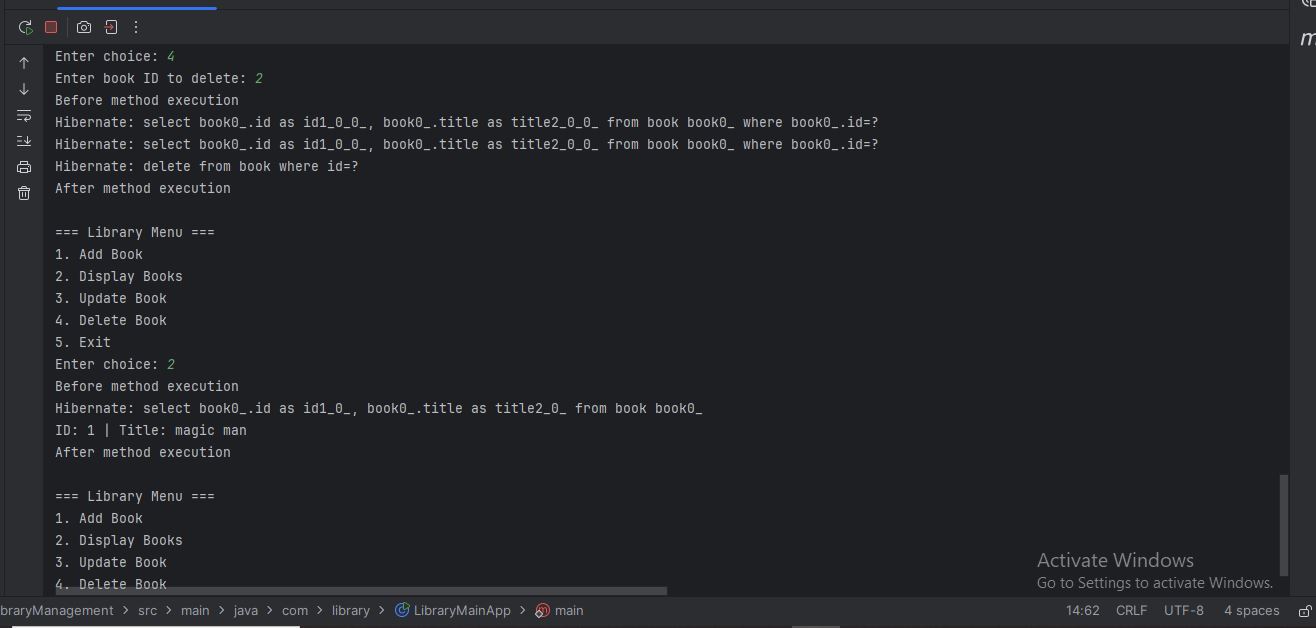
}









s9f.JPG