**WEEK-3**

**Spring Core and Maven**

**Superset ID: 6419740**

**Exercise 1: Configuring a Basic Spring Application**

**Code:**

**BookRepository.java:**

package com.library.repository;

import org.springframework.stereotype.Repository;

@Repository

public class BookRepository {

    public void save() {

        System.out.println("BookRepository: Book saved.");

    }

}

**BookService.java:**

package com.library.service;

import com.library.repository.BookRepository;

import org.springframework.stereotype.Service;

@Service

public class BookService {

    private final BookRepository bookRepository;

    public BookService(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

    }

    public void addBook() {

        System.out.println("BookService: Adding book...");

        bookRepository.save();

    }

}

**MainApp.java:**

package com.library;

import com.library.service.BookService;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class MainApp implements CommandLineRunner {

    @Autowired

    private BookService bookService;

    public static void main(String[] args) {

        SpringApplication.run(MainApp.class, args);

    }

    @Override

    public void run(String... args) {

        bookService.addBook();

    }

}

**LibraryManagementApplicationTests.java:**

package com.library.LibraryManagement;

import org.junit.jupiter.api.Test;

import org.springframework.boot.test.context.SpringBootTest;

import com.library.MainApp;

@SpringBootTest(classes = MainApp.class)

public class LibraryManagementApplicationTests {

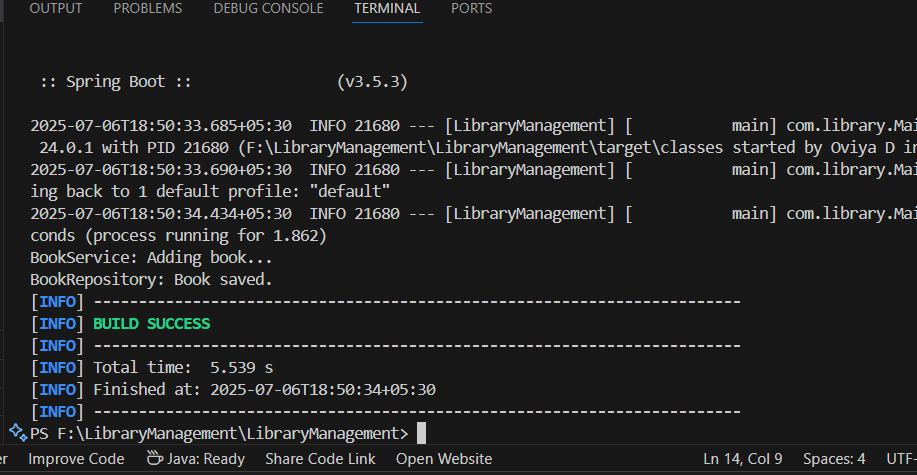
    @Test

    void contextLoads() {

    }

}

**Output:**

****

**Exercise 2: Implementing Dependency Injection**

**Code:**

**BookService.java:**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

    private BookRepository bookRepository;

    // Setter for dependency injection

    public void setBookRepository(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

    }

    public void addBook() {

        System.out.println("Adding book...");

        bookRepository.save();

    }

}

**BookRepository.java:**

package com.library.repository;

public class BookRepository {

    public void save() {

        System.out.println("Book saved to database.");

    }

}

**MainApp.java:**

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class MainApp {

    public static void main(String[] args) {

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

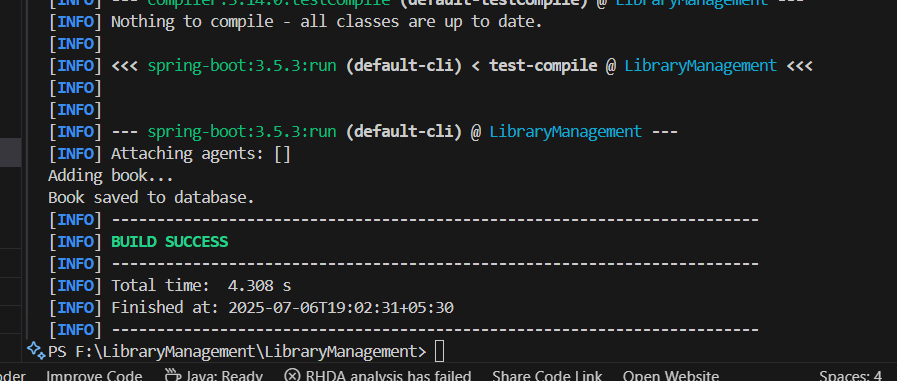
        BookService bookService = context.getBean("bookService", BookService.class);

        bookService.addBook();

    }

}

**Output:**

****

**Exercise 3: Implementing Logging with Spring AOP**

**Code:**

**BookService.java:**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

    private BookRepository bookRepository;

    // Setter for dependency injection

    public void setBookRepository(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

    }

    public void addBook() {

        System.out.println("Adding book...");

        bookRepository.save();

    }

}

**BookRepository.java:**

package com.library.repository;

public class BookRepository {

    public void save() {

        System.out.println("Book saved to database.");

    }

}

**MainApp.java:**

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class MainApp {

    public static void main(String[] args) {

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

        BookService bookService = context.getBean("bookService", BookService.class);

        bookService.addBook();

    }

}

**LoggingAspect.java:**

package com.library.aspect;

import org.aspectj.lang.ProceedingJoinPoint;

import org.aspectj.lang.annotation.Around;

import org.aspectj.lang.annotation.Aspect;

@Aspect

public class LoggingAspect {

    @Around("execution(\* com.library.service.\*.\*(..))")

    public Object logExecutionTime(ProceedingJoinPoint joinPoint) throws Throwable {

        long start = System.currentTimeMillis();

        Object result = joinPoint.proceed();

        long end = System.currentTimeMillis();

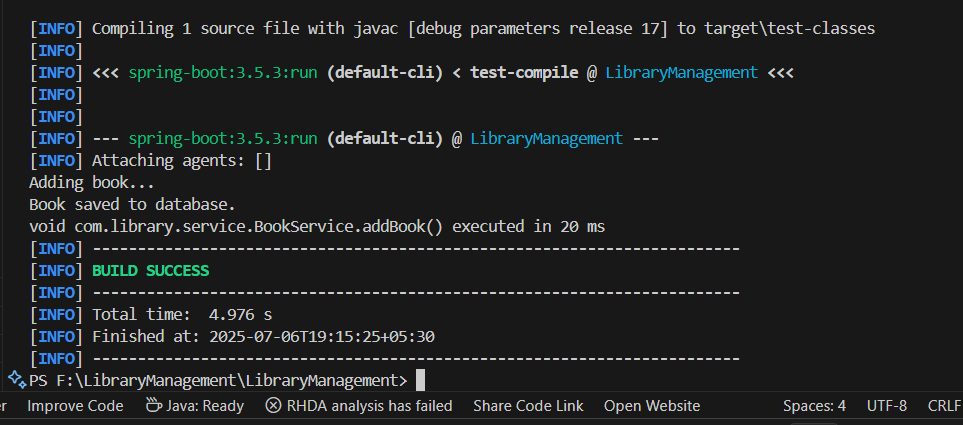
        System.out.println(joinPoint.getSignature() + " executed in " + (end - start) + " ms");

        return result;

    }

}

**Output:**

****

**Exercise 4: 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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.34</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aspects</artifactId>

<version>5.3.34</version>

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.21</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

**BookService.java:**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

    private BookRepository bookRepository;

    // Setter for dependency injection

    public void setBookRepository(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

    }

    public void addBook() {

        System.out.println("Adding book...");

        bookRepository.save();

    }

}

**BookRepository.java:**

package com.library.repository;

public class BookRepository {

    public void save() {

        System.out.println("Book saved to database.");

    }

}

**MainApp.java:**

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class MainApp {

    public static void main(String[] args) {

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

        BookService bookService = context.getBean("bookService", BookService.class);

        bookService.addBook();

    }

}

**LoggingAspect.java:**

package com.library.aspect;

import org.aspectj.lang.ProceedingJoinPoint;

import org.aspectj.lang.annotation.Around;

import org.aspectj.lang.annotation.Aspect;

@Aspect

public class LoggingAspect {

    @Around("execution(\* com.library.service.\*.\*(..))")

    public Object logExecutionTime(ProceedingJoinPoint joinPoint) throws Throwable {

        long start = System.currentTimeMillis();

        Object result = joinPoint.proceed();

        long end = System.currentTimeMillis();

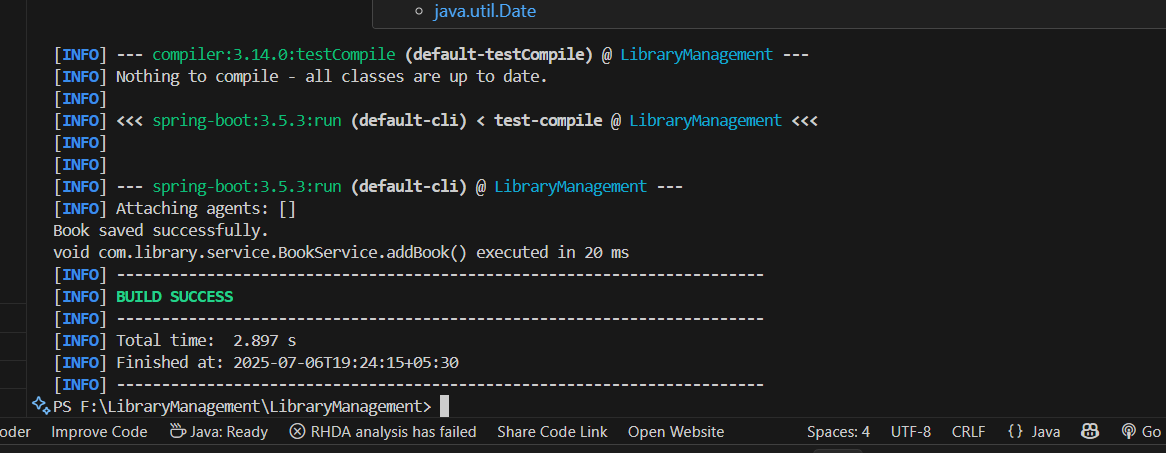
        System.out.println(joinPoint.getSignature() + " executed in " + (end - start) + " ms");

        return result;

    }

}

**Output:**

****

**Exercise 5: Configuring the Spring IoC Container**

**Code:**

**BookService.java:**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

    private BookRepository bookRepository;

    // Setter for dependency injection

    public void setBookRepository(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

    }

    public void addBook() {

        System.out.println("Adding book...");

        bookRepository.save();

    }

}

**BookRepository.java:**

package com.library.repository;

public class BookRepository {

    public void save() {

        System.out.println("Book saved successfully.");

    }

}

**MainApp.java:**

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class MainApp {

    public static void main(String[] args) {

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

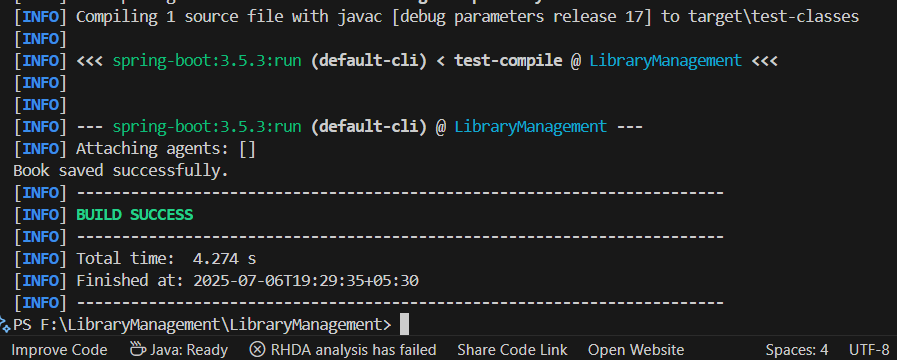
        BookService bookService = context.getBean("bookService", BookService.class);

        bookService.addBook();

    }

}

**Output:**

****

**Exercise 6: Configuring Beans with Annotations**

**Code:**

**MainApp.java:**

package com.library;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

    public static void main(String[] args) {

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

        BookService bookService = context.getBean("bookService", BookService.class);

        bookService.addBook();

    }

}

**BookService.java:**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

    private BookRepository bookRepository;

    public void setBookRepository(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

    }

    public void addBook() {

        bookRepository.save();

    }

}

**BookRepository.java:**

package com.library.repository;

import org.springframework.stereotype.Repository;

@Repository

public class BookRepository {

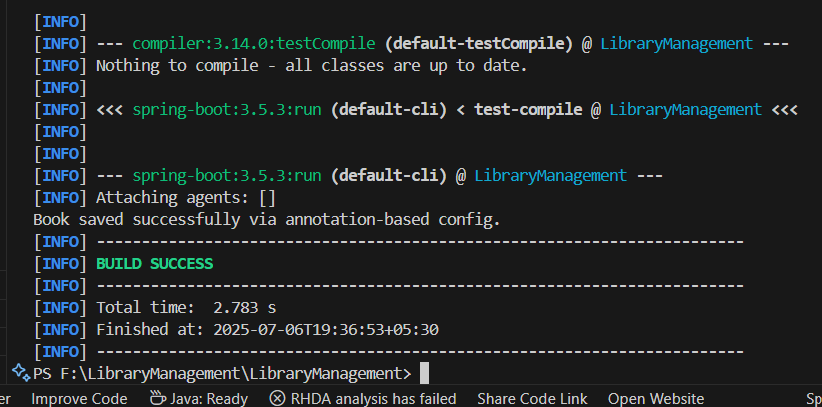
    public void save() {

        System.out.println("Book saved successfully via annotation-based config.");

    }

}

**Output:**

****

**Exercise 7: Implementing Constructor and Setter Injection**

**Code:**

**MainApp.java:**

package com.library;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

    public static void main(String[] args) {

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

        BookService bookService = context.getBean("bookService", BookService.class);

        bookService.addBook();

    }

}

**BookService.java:**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

public BookService(BookRepository bookRepository) {

this.bookRepository = bookRepository;

System.out.println("Constructor injection called");

}

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

System.out.println("Setter injection called");

}

public void addBook() {

bookRepository.save();

}

}

**BookRepository.java:**

package com.library.repository;

import org.springframework.stereotype.Repository;

@Repository

public class BookRepository {

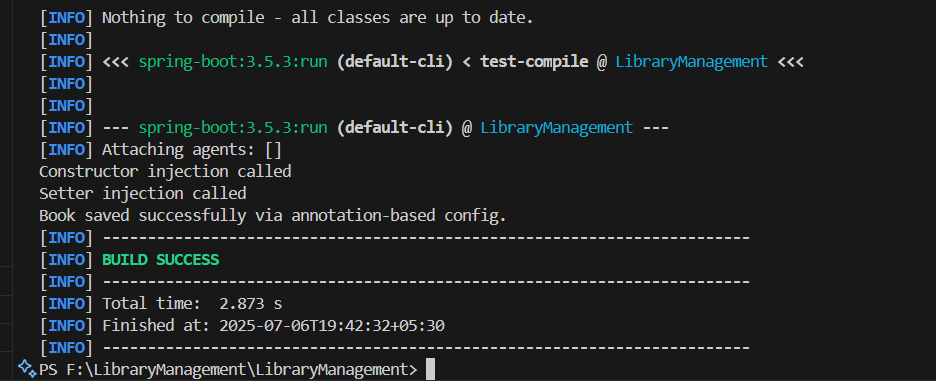
    public void save() {

        System.out.println("Book saved successfully via annotation-based config.");

    }

}

**Output:**

****

**Exercise 8: Implementing Basic AOP with Spring**

**Code:**

**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"

       xmlns:aop="http://www.springframework.org/schema/aop"

       xsi:schemaLocation="

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

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

    <!-- Enable AspectJ -->

    <aop:aspectj-autoproxy/>

    <!-- Define beans -->

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

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

        <constructor-arg ref="bookRepository"/>

    </bean>

    <!-- Define aspect bean -->

    <bean id="loggingAspect" class="com.library.aspect.LoggingAspect"/>

    <!-- AOP Config -->

    <aop:config>

        <aop:aspect ref="loggingAspect">

            <aop:pointcut id="bookServiceMethods" expression="execution(\* com.library.service.\*.\*(..))"/>

            <aop:before pointcut-ref="bookServiceMethods" method="logBefore"/>

            <aop:after pointcut-ref="bookServiceMethods" method="logAfter"/>

        </aop:aspect>

    </aop:config>

</beans>

**MainApp.java:**

package com.library;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

public static void main(String[] args) {

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

BookService bookService = context.getBean("bookService", BookService.class);

bookService.addBook();

}

}

**LoggingAspect.java:**

package com.library.aspect;

import org.aspectj.lang.JoinPoint;

public class LoggingAspect {

public void logBefore(JoinPoint joinPoint) {

System.out.println(">> Before method: " + joinPoint.getSignature().getName());

}

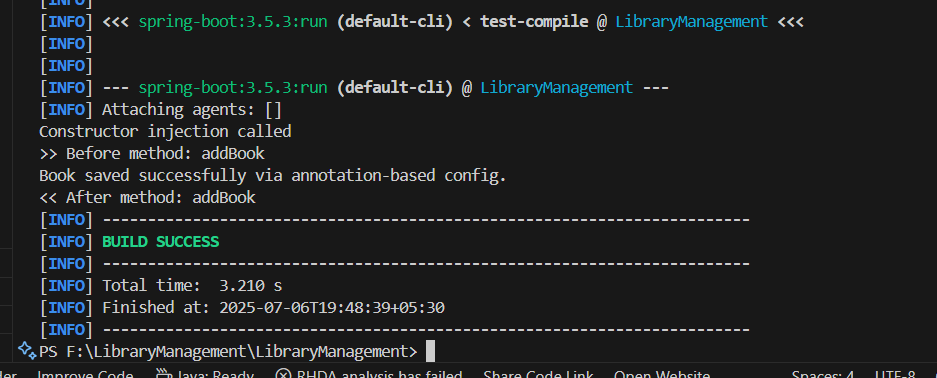
public void logAfter(JoinPoint joinPoint) {

System.out.println("<< After method: " + joinPoint.getSignature().getName());

}

}

**Output:**



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

**Code:**

**Book.java:**

package com.library.entity;

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 & Setters

    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;

    }

}

**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 bookRepository;

    @PostMapping

    public Book addBook(@RequestBody Book book) {

        return bookRepository.save(book);

    }

    @GetMapping

    public List<Book> getAllBooks() {

        return bookRepository.findAll();

    }

    @GetMapping("/{id}")

    public Book getBookById(@PathVariable Long id) {

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

    }

    @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);

    }

}

**BookRepository.java:**

package com.library.repository;

import com.library.entity.Book;

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

import org.springframework.stereotype.Repository;

@Repository

public interface BookRepository extends JpaRepository<Book, Long> {

}

**BookService.java:**

package com.library.service;

import com.library.entity.Book;

import com.library.repository.BookRepository;

import org.springframework.stereotype.Service;

@Service

public class BookService {

    private BookRepository bookRepository;

    public BookService(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

        System.out.println("Constructor injection called");

    }

    public void setBookRepository(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

        System.out.println("Setter injection called");

    }

    public void addBook() {

        Book book = new Book();

        book.setTitle("Spring in Action");

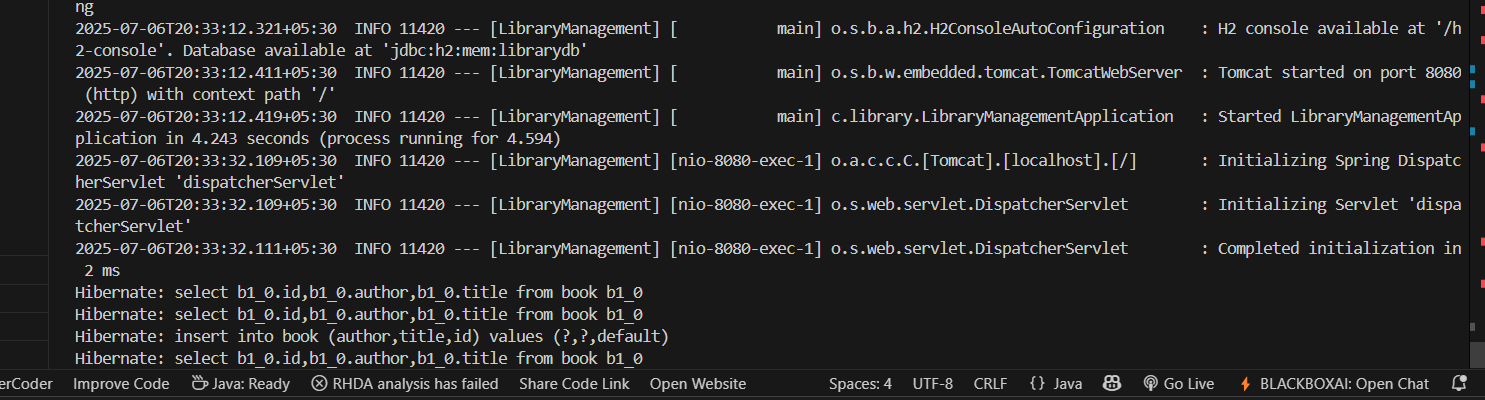
        book.setAuthor("Craig Walls");

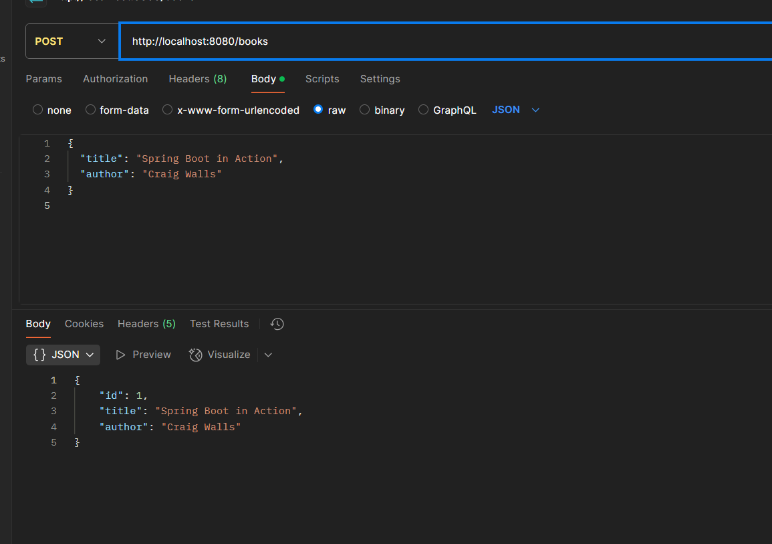
        bookRepository.save(book);

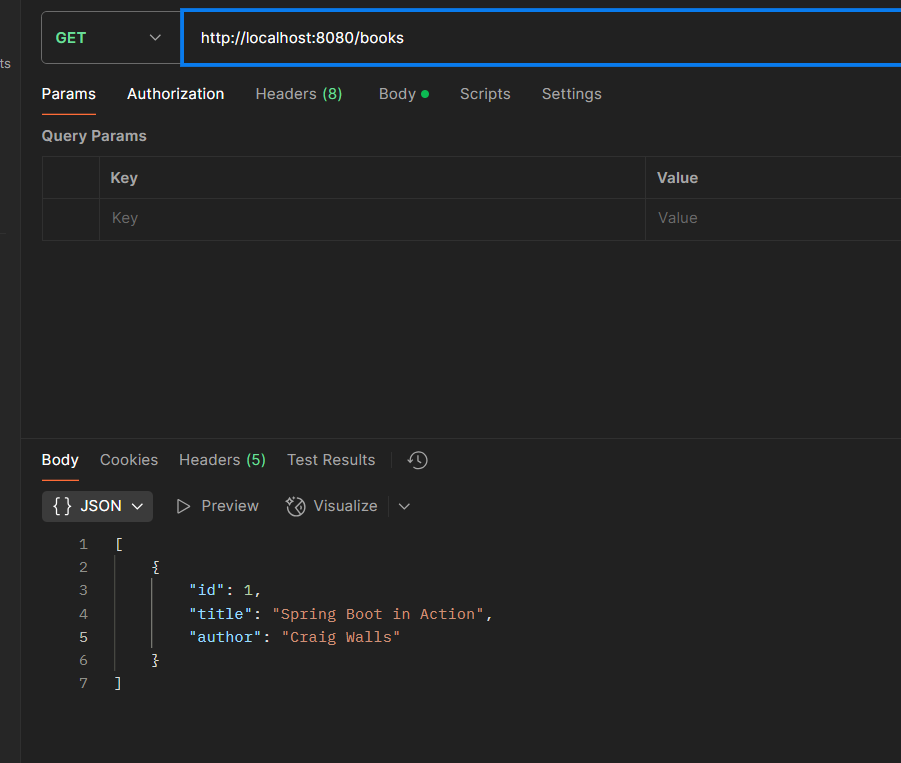
    }

}

**Output:**

****

****

****