**Spring Core Maven**

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

<?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 http://www.springframework.org/schema/beans/spring-beans.xsd">

<!-- bean definitions here -->

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

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

</bean>

</beans>

package com.library.repository;

public class BookRepository {

public void saveBook(String title) {

System.***out***.println("Saving book: " + title);

}

}

package com.library.service;

import com.library.repository.BookRepository;

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

public void addBook(String title) {

System.***out***.println("Adding book:"+title);

}

}

package com.cognizant.LibraryManagement;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class App {

public static void main(String[] args) {

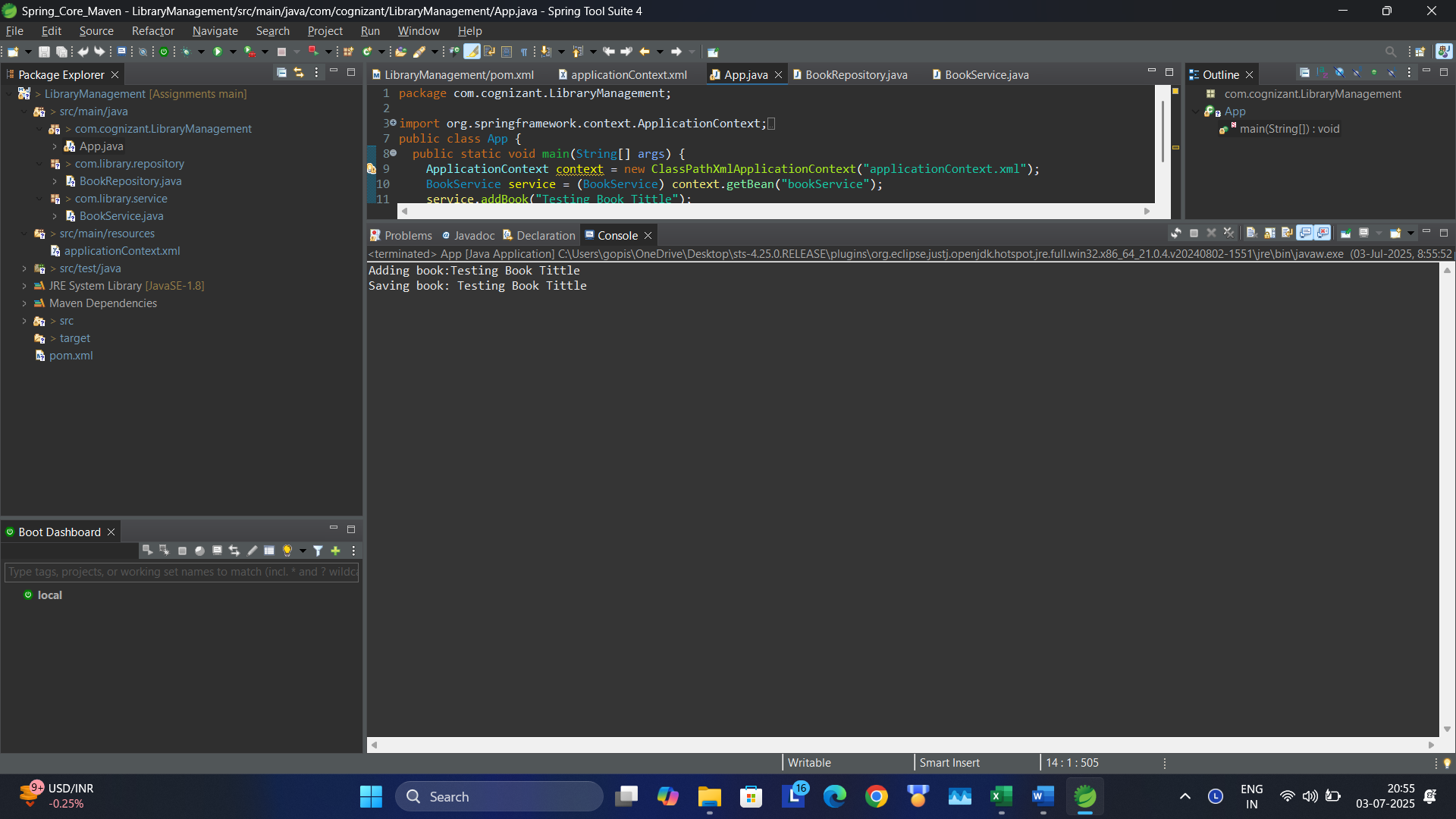
ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

BookService service = (BookService) context.getBean("bookService");

service.addBook("Testing Book Tittle");

}

}



**Exercise 2: Implementing Dependency Injection**

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

<!-- bean definitions here -->

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

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

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

</bean>

</beans>

**Book Service Class with the setter:**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook(String title) {

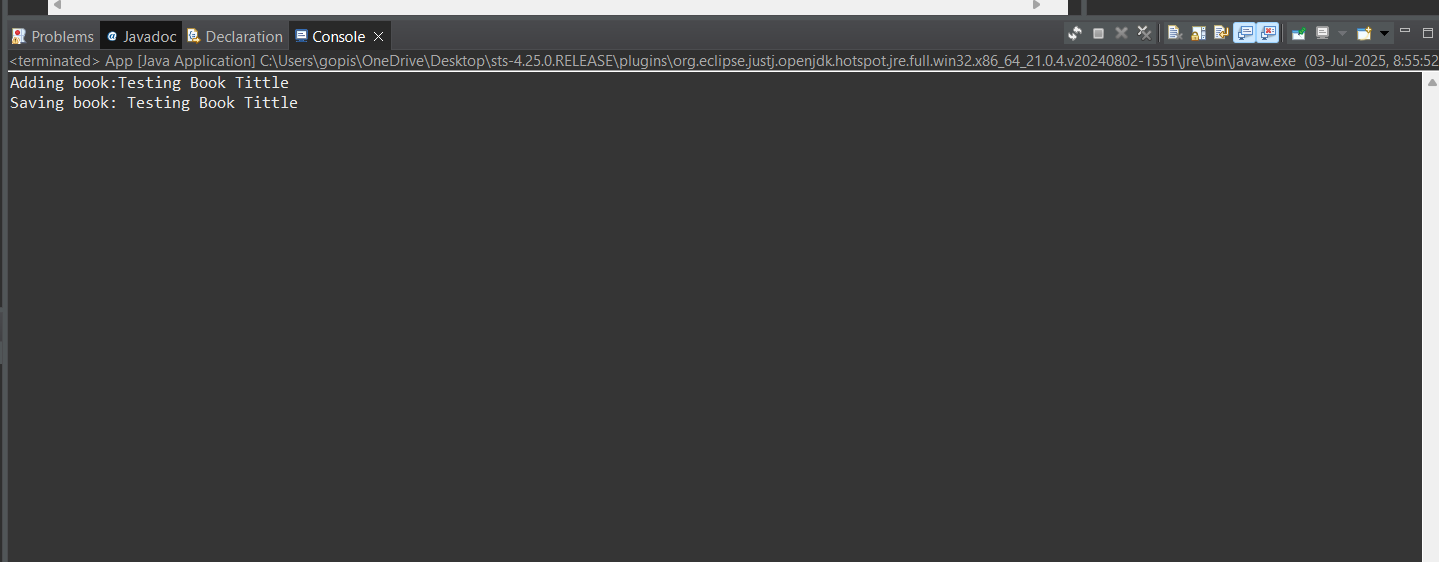
System.***out***.println("Adding book:"+title);

bookRepository.saveBook(title);

}

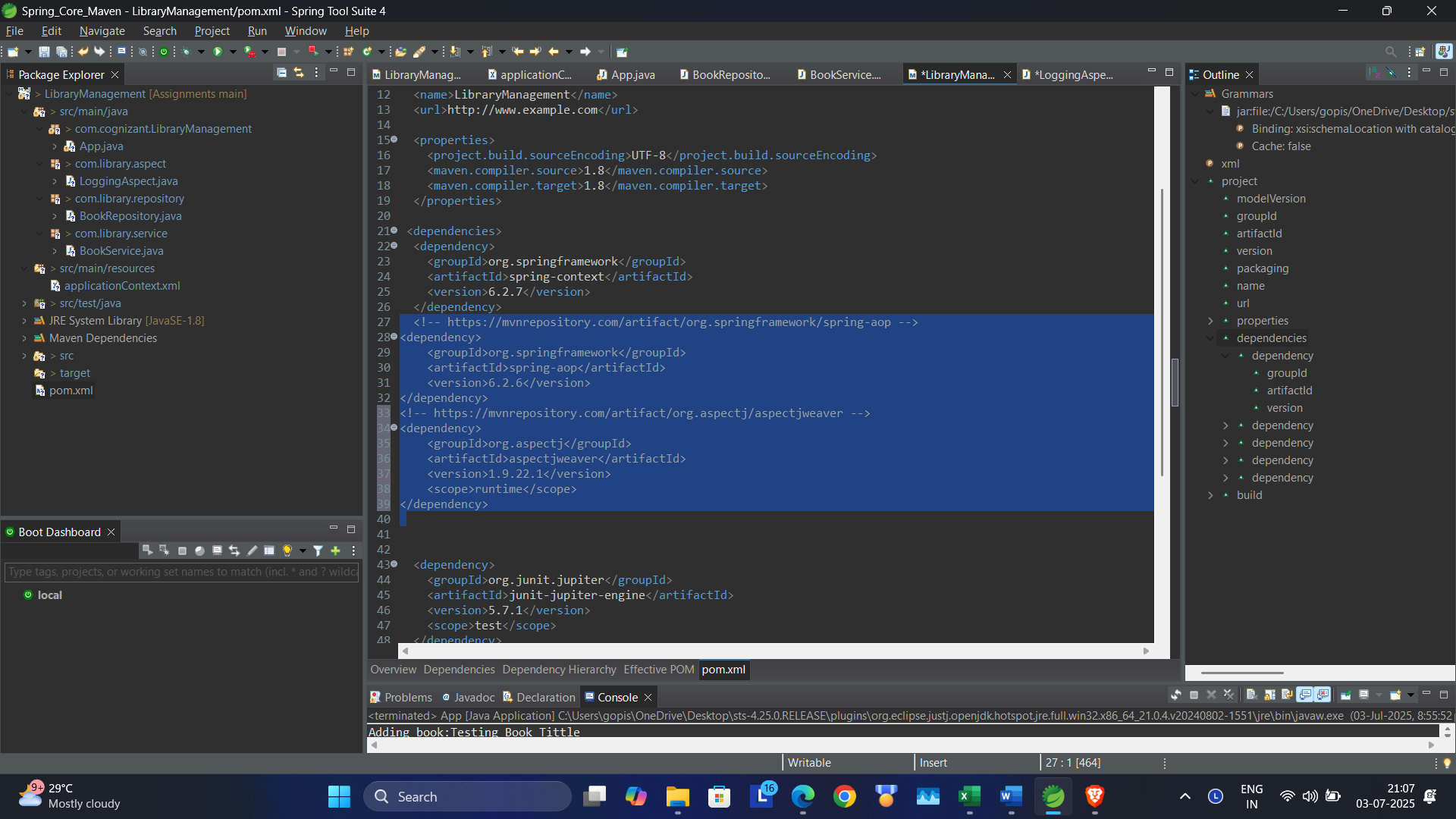
}

Output:



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

Updated pom.xml



<!-- https://mvnrepository.com/artifact/org.springframework/spring-aop -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>6.2.6</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.aspectj/aspectjweaver -->

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.22.1</version>

<scope>runtime</scope>

</dependency>

Code:

<?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

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

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

http://www.springframework.org/schema/aop/spring-aop-3.0.xsd ">

<aop:aspectj-autoproxy />

<!-- bean definitions here -->

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

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

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

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

</bean>

</beans>

package com.library.aspect;

import org.aspectj.lang.ProceedingJoinPoint;

import org.aspectj.lang.annotation.Around;

import org.aspectj.lang.annotation.Aspect;

import org.springframework.stereotype.Component;

*@Aspect*

*@Component*

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*();

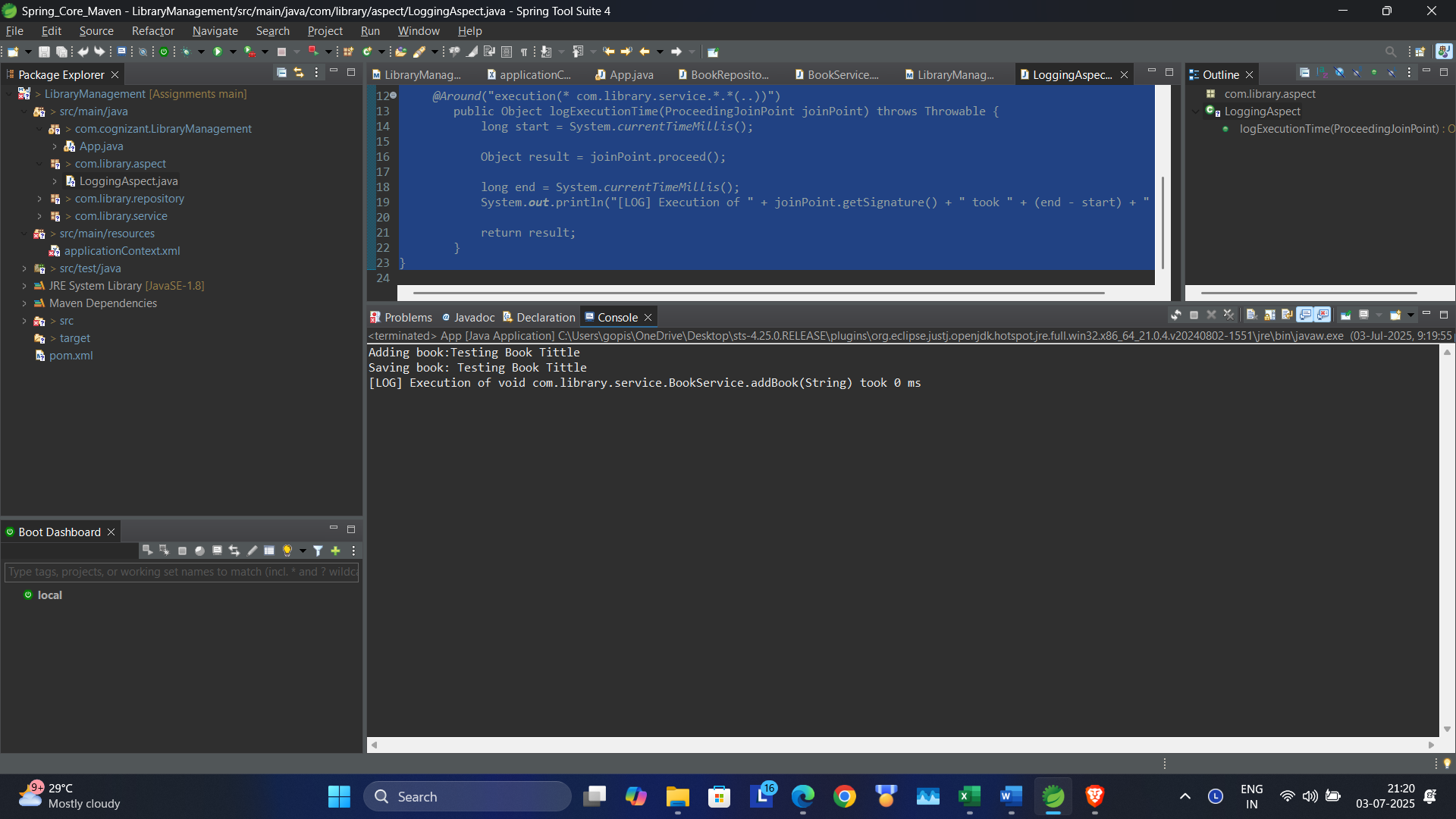
System.***out***.println("[LOG] Execution of " + joinPoint.getSignature() + " took " + (end - start) + " ms");

return result;

}

}

Output:



**Exercise 4: Creating and Configuring a Maven Project**

**Created a new Maven Project with dependencies Spring Context, Spring AOP, and Spring WebMVC**

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

<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>LibraryManagement2</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>LibraryManagement2</name>

<url>http://www.example.com</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

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

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

</properties>

<dependencies>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-engine</artifactId>

<version>5.7.1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-params</artifactId>

<version>5.7.1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>6.2.7</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-aop -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aspects</artifactId>

<version>6.2.7</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>6.2.6</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-surefire-plugin</artifactId>

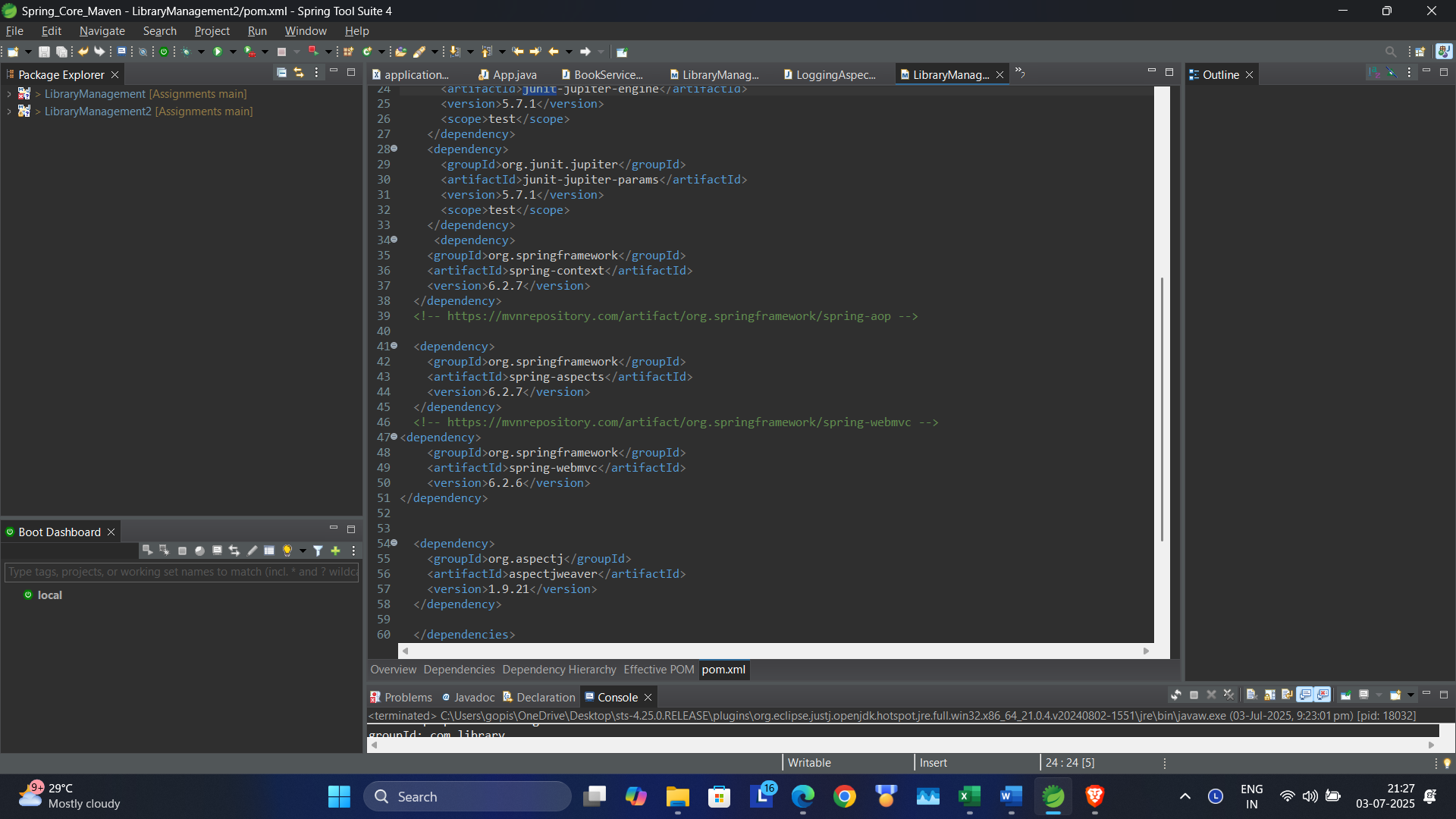
<version>3.0.0-M5</version>

</plugin>

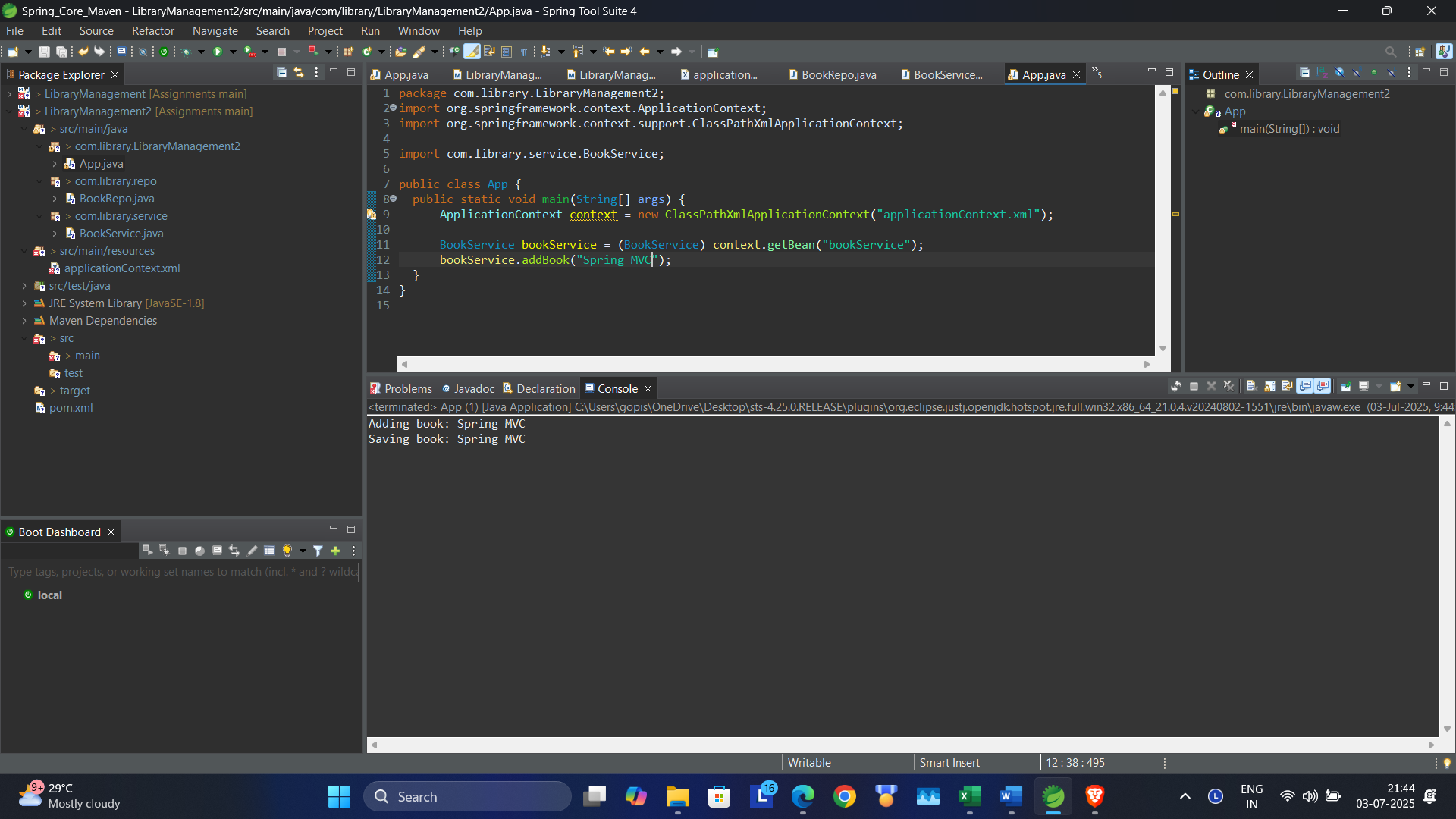
</plugins>

</build>

</project>



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



**Exercise 6: Configuring Beans with Annotations**

package com.library.LibraryManagement2;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class App {

public static void main(String[] args) {

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

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

bookService.addBook("Spring MVC");

}

}

package com.library.repo;

import org.springframework.stereotype.Repository;

*@Repository*

public class BookRepo {

public void saveBook(String title) {

System.***out***.println("Saving book: " + title);

}

}

package com.library.service;

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

import org.springframework.stereotype.Service;

import com.library.repo.BookRepo;

*@Service*

public class BookService {

*@Autowired*

private BookRepo bookRepository;

//public void setBookRepository(BookRepo bookRepository) {

// this.bookRepository = bookRepository;

//}

public void addBook(String title) {

System.***out***.println("Adding book: " + title);

bookRepository.saveBook(title);

}

}

<?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:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="

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

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

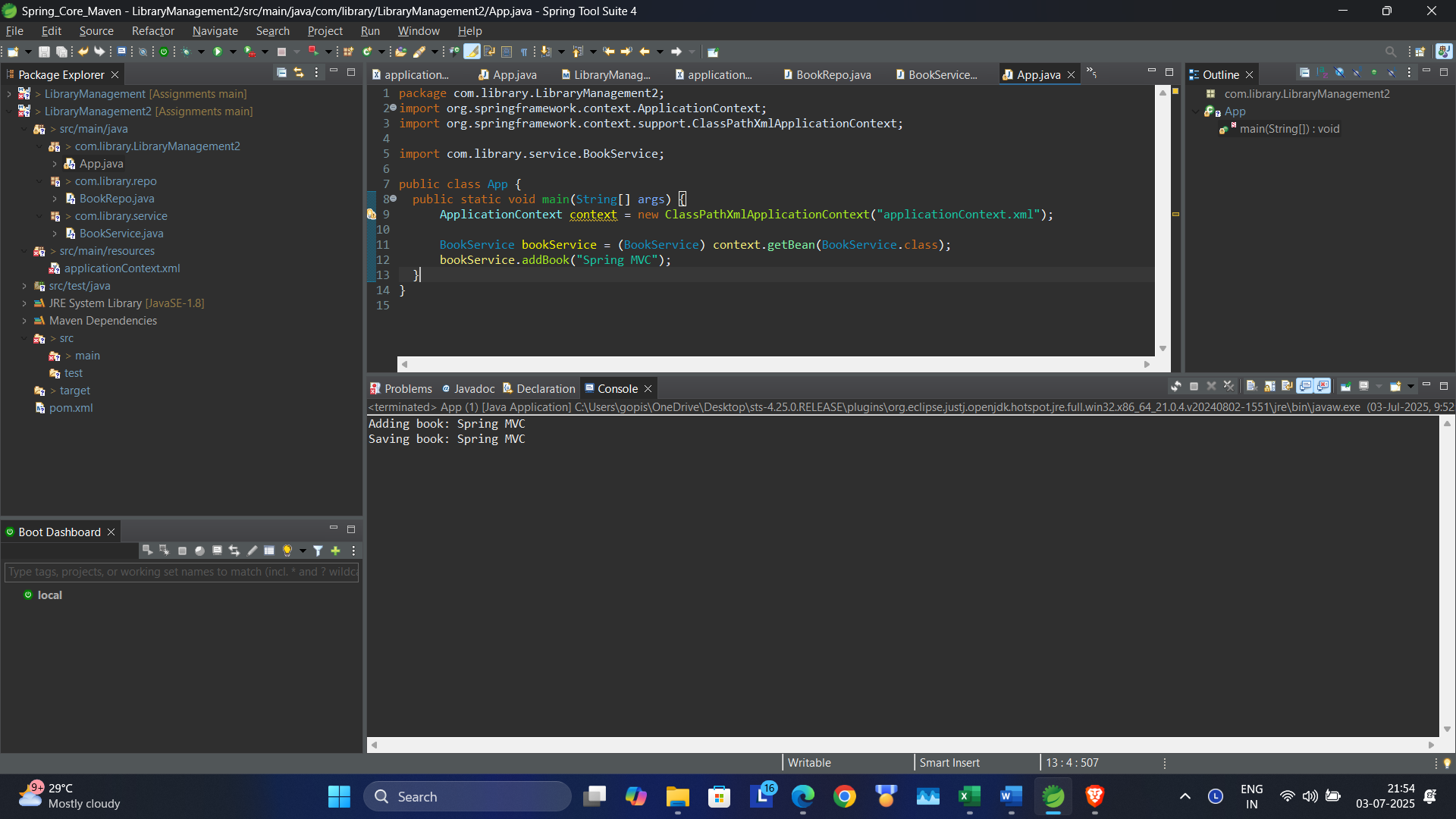
http://www.springframework.org/schema/context

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

<context:component-scan base-package="com.library" />

</beans>

Output:



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

Code:

package com.library.service;

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

import org.springframework.stereotype.Service;

import com.library.repo.BookRepo;

*@Service*

public class BookService {

//@Autowired

private BookRepo bookRepository;

private String message;

public void setBookRepository(BookRepo bookRepository) {

this.bookRepository = bookRepository;

}

public BookService(String message) {

this.message = message;

}

public void addBook(String title) {

System.***out***.println(message + ": Adding book: " + title);

if (bookRepository != null) {

bookRepository.saveBook(title);

} else {

System.***out***.println("BookRepository not injected!");

}

}

}

Pom.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:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="

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

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

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

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

<!--<context:component-scan base-package="com.library" /> -->

<bean id="bookRepository" class="com.library.repo.BookRepo" />

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

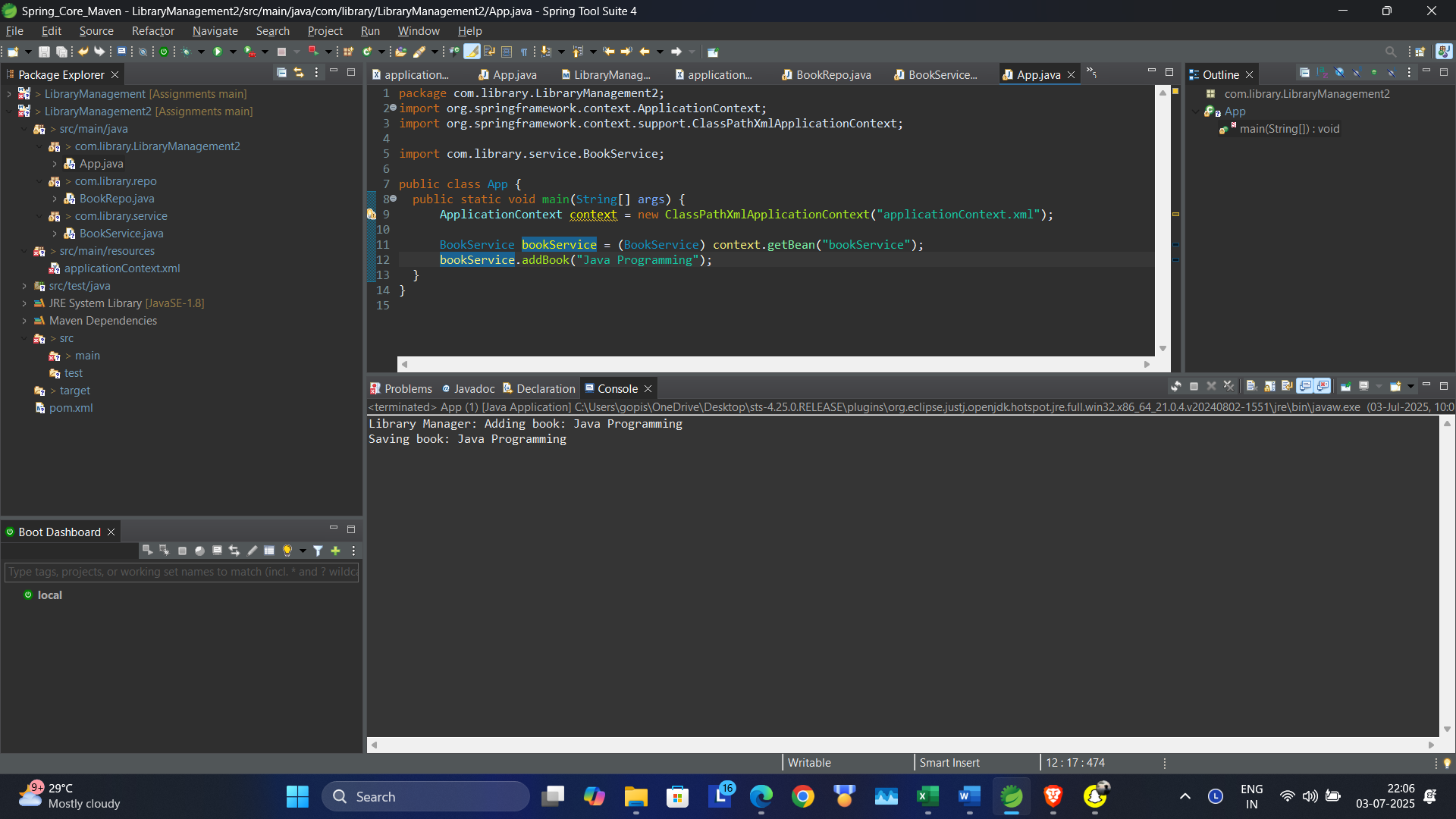
<constructor-arg value="Library Manager"/>

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

</bean>

</beans>

Output:



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

Code:

package com.library.aspect;

import org.aspectj.lang.JoinPoint;

public class LoggingAspect {

public void logBefore(JoinPoint joinPoint) {

System.***out***.println("[LOG BEFORE] Method called: " + joinPoint.getSignature().getName());

}

public void logAfter(JoinPoint joinPoint) {

System.***out***.println("[LOG AFTER] Method finished: " + joinPoint.getSignature().getName());

}

}

Pom.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

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

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

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

<aop:aspectj-autoproxy />

<bean id="bookRepository" class="com.library.repo.BookRepo" />

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

<constructor-arg value="Library Logger"/>

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

</bean>

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

<aop:config>

<aop:aspect id="logAspect" ref="loggingAspect">

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

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

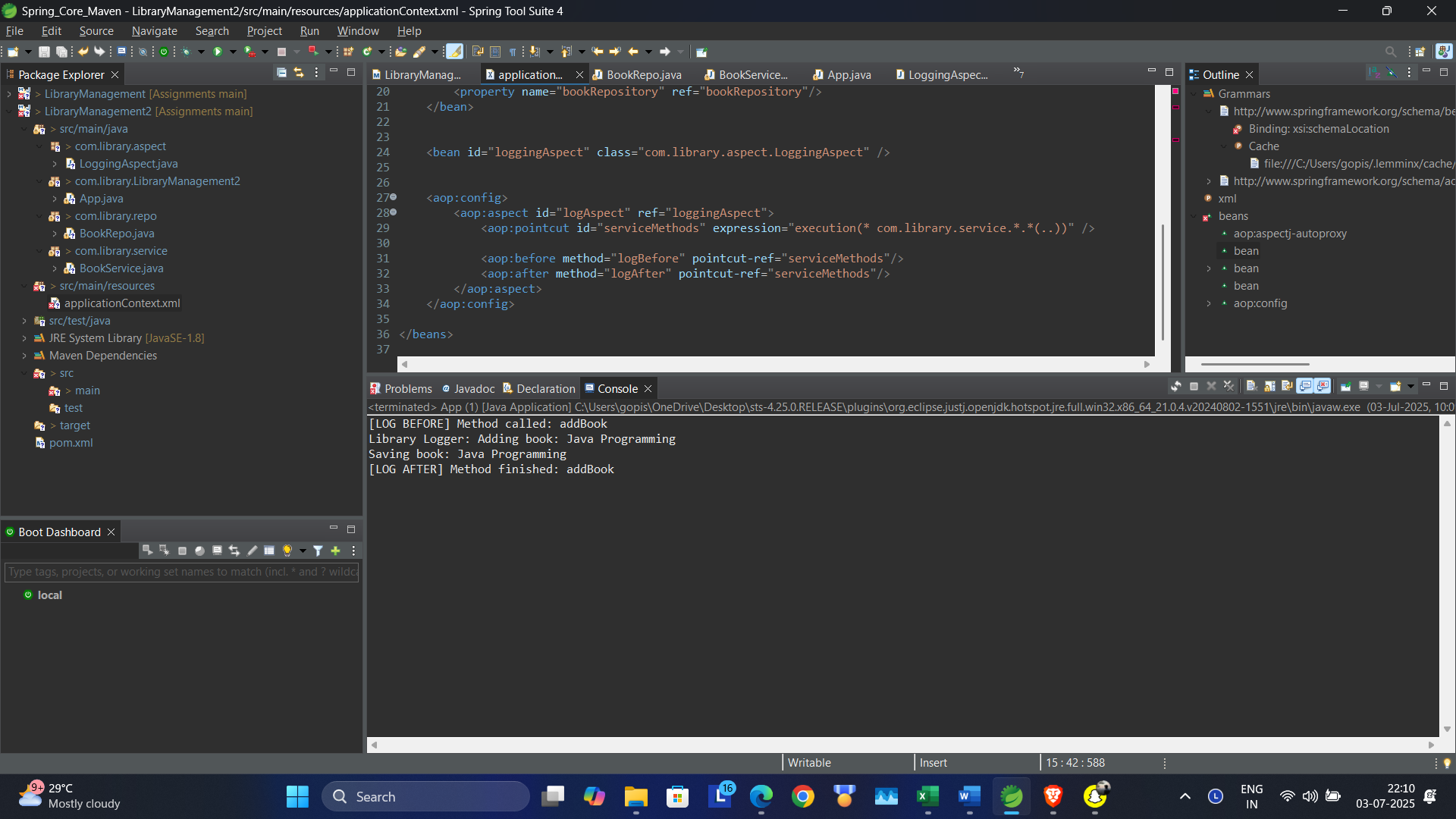
<aop:after method="logAfter" pointcut-ref="serviceMethods"/>

</aop:aspect>

</aop:config>

</beans>

Output:



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

Application.properties:

spring.application.name=LibraryManagement3

server.port = 8080

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

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=admin

spring.datasource.password=admin

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

spring.jpa.hibernate.ddl-auto=update

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console

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 createBook(@RequestBody Book book) {

return bookRepository.save(book);

}

@GetMapping

public List<Book> getAllBooks() {

return bookRepository.findAll();

}

@GetMapping("/{id}")

public Book getBookById(@PathVariable("id") Long id) {

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

}

@PutMapping("/{id}")

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

Book b = new Book();

b = bookRepository.findByid(id);

b.setTitle(updatedBook.getTitle());

b.setAuthor(updatedBook.getAuthor());

bookRepository.save(b);

return b;

}

@DeleteMapping("/{id}")

public void deleteBook(@PathVariable("id") Long id) {

bookRepository.deleteById(id);

}

}

Book.java

package com.library.entity;

import jakarta.persistence.\*;

*@Entity*

public class Book {

*@Id*

*@GeneratedValue*(strategy = *GenerationType*.***IDENTITY***)

private Long id;

private String title;

private String author;

public Book() {}

public Book(String title, String author) {

this.title = title;

this.author = 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; }

}

BookRepository.java

package com.library.repository;

import com.library.entity.Book;

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

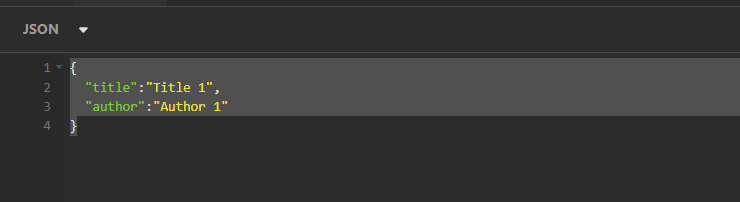
public interface BookRepository extends JpaRepository<Book, Long> {

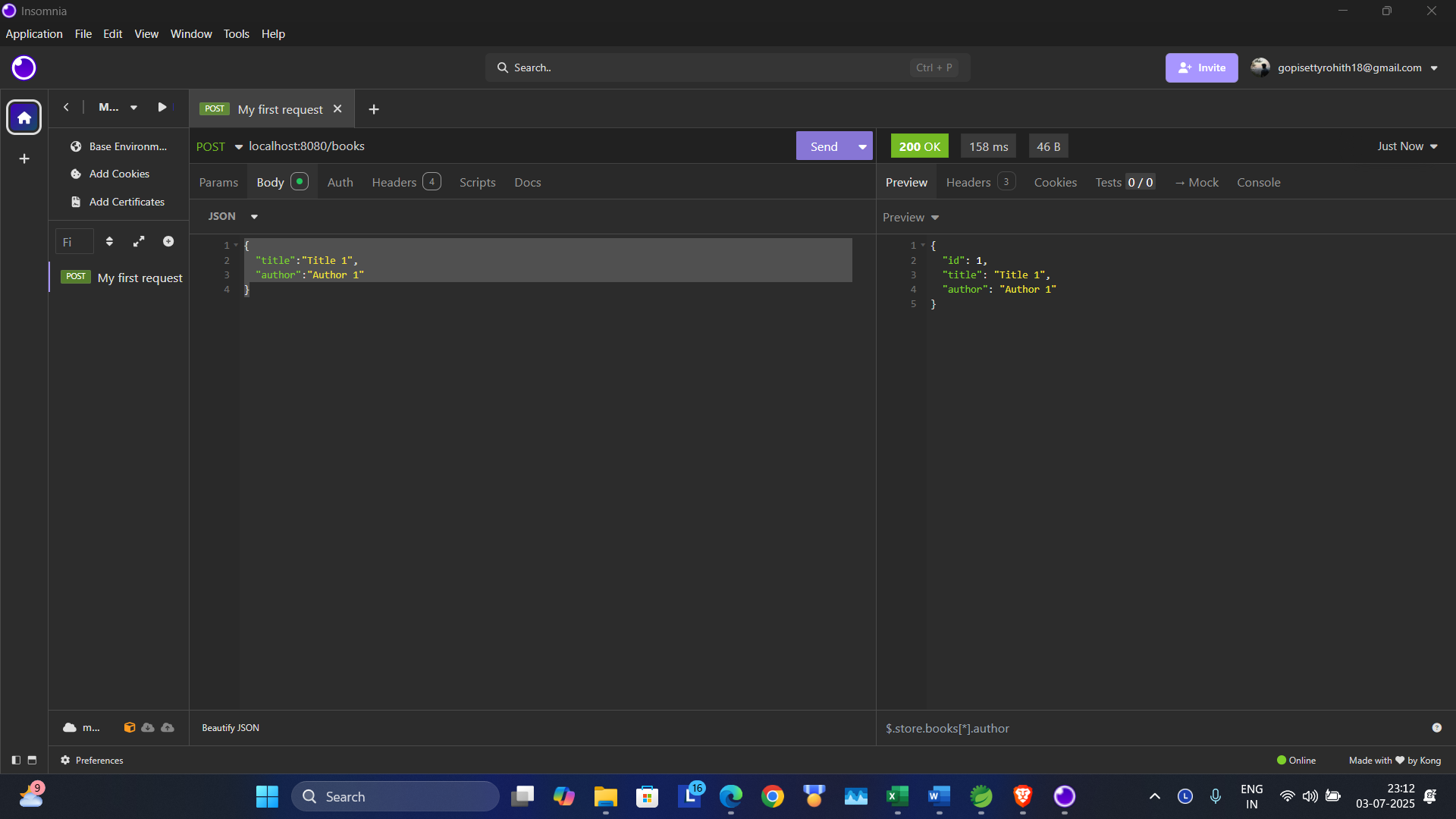
}

**Testing:**

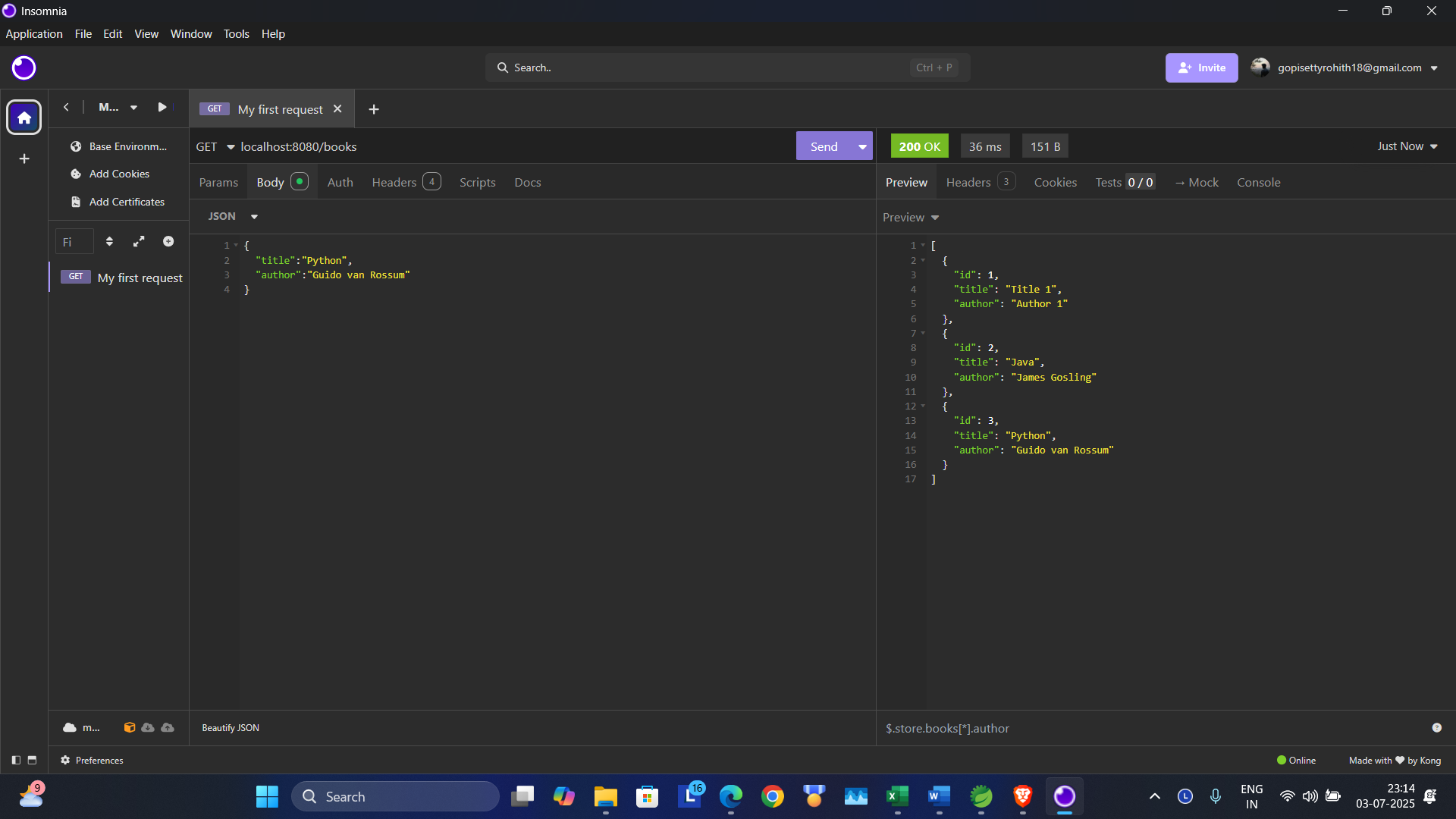
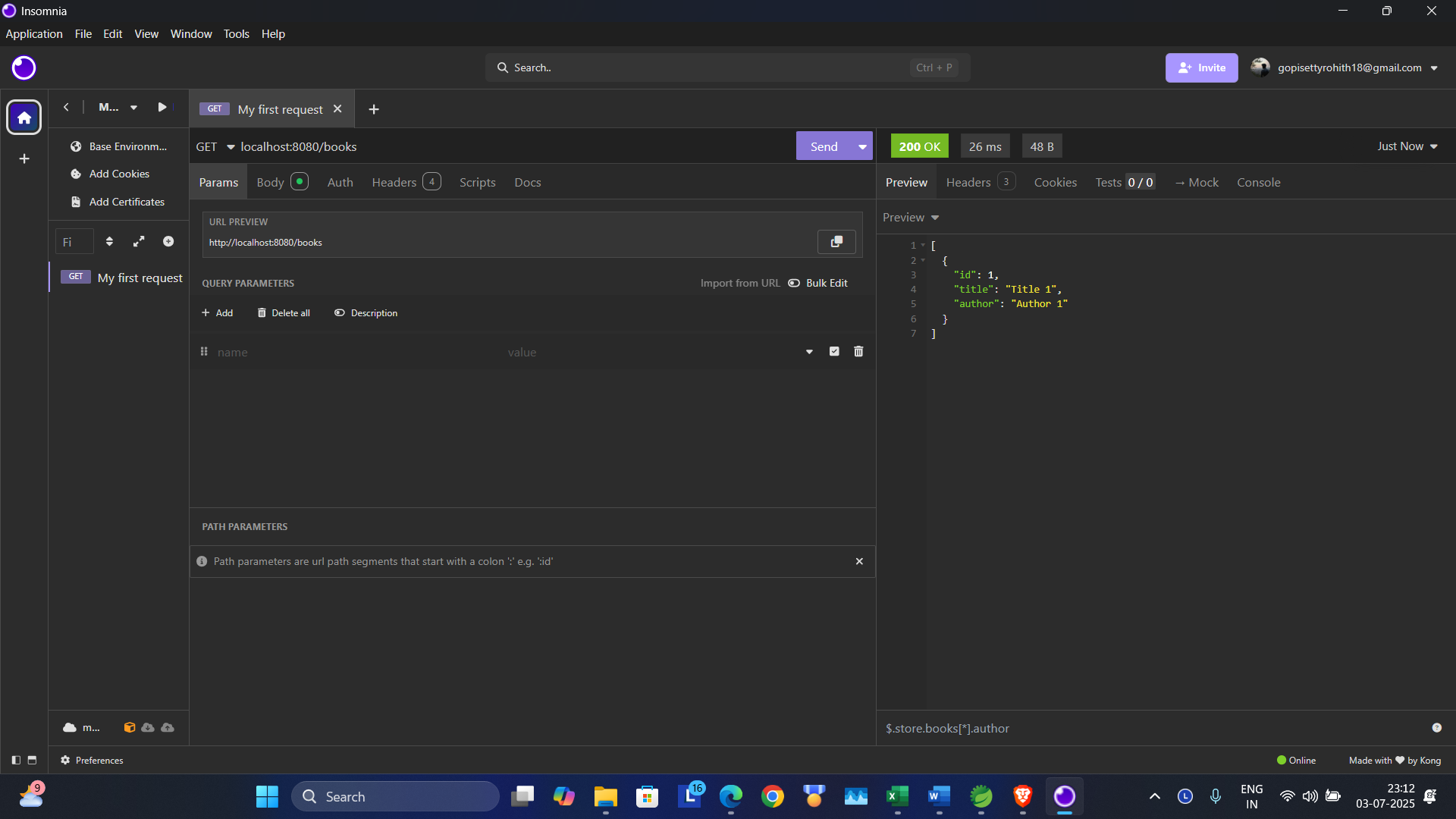
Post method: <http://localhost:8080/books>

Json:



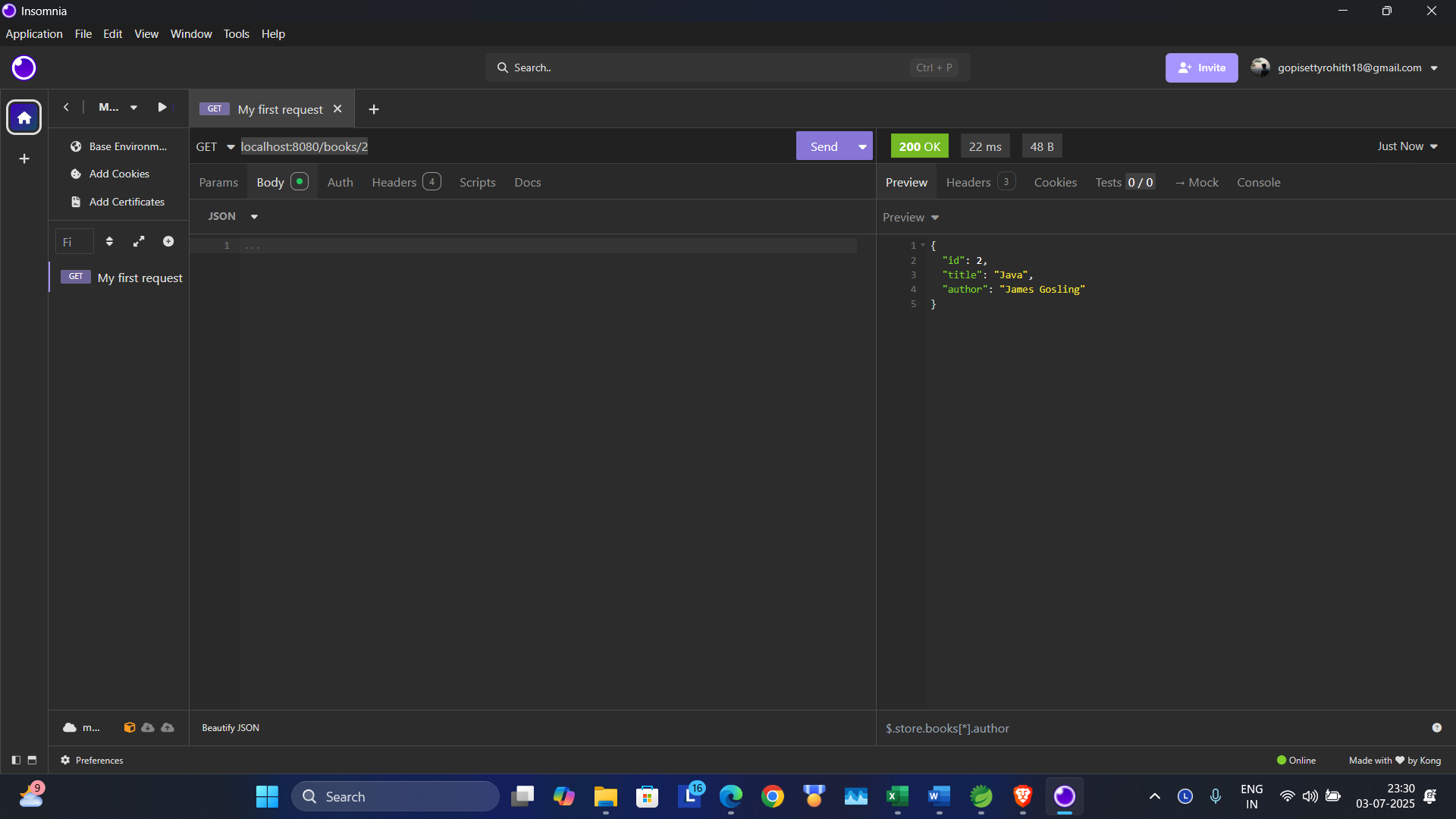


GET method: <http://localhost:8080/books>



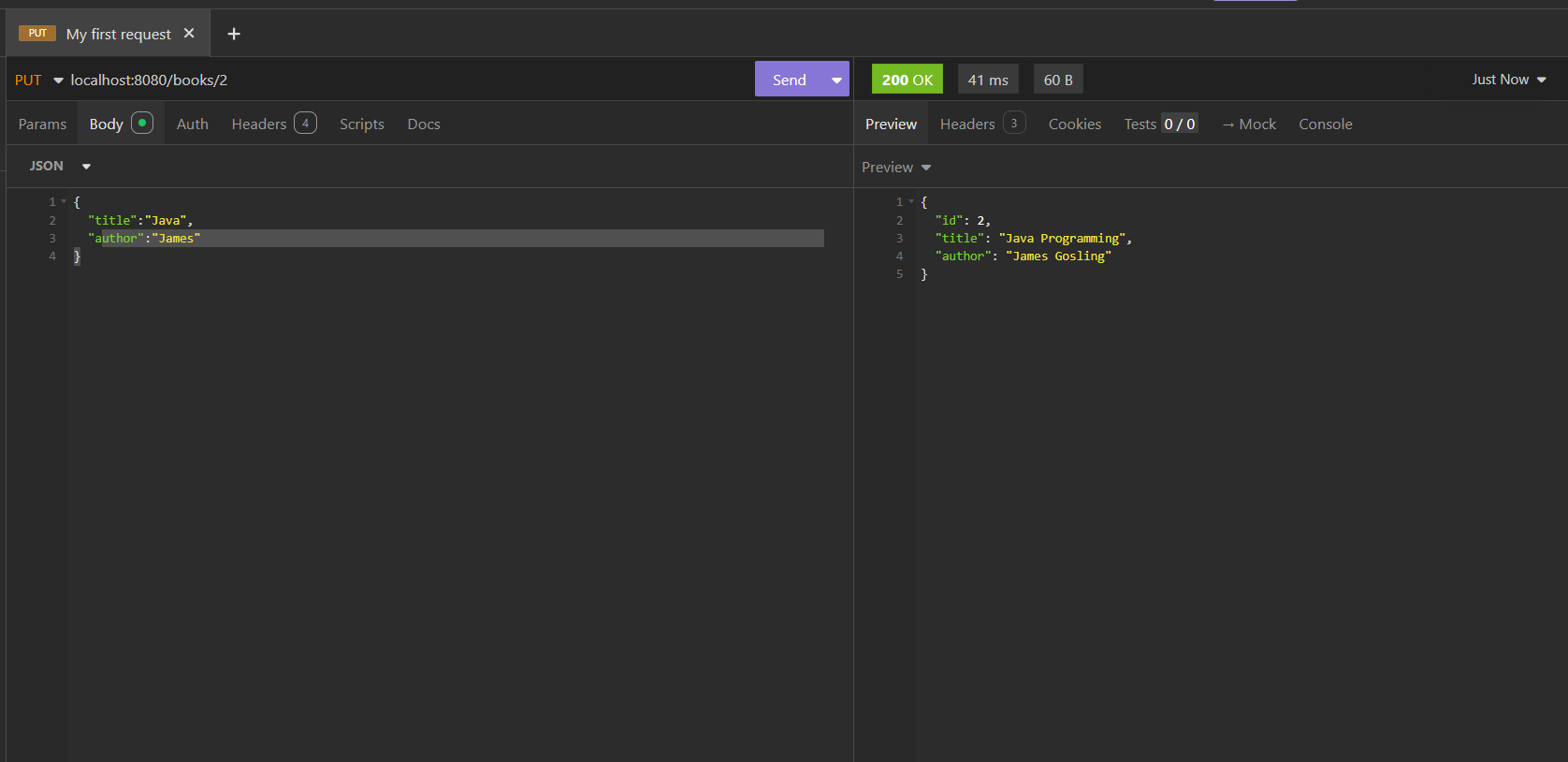
Get method

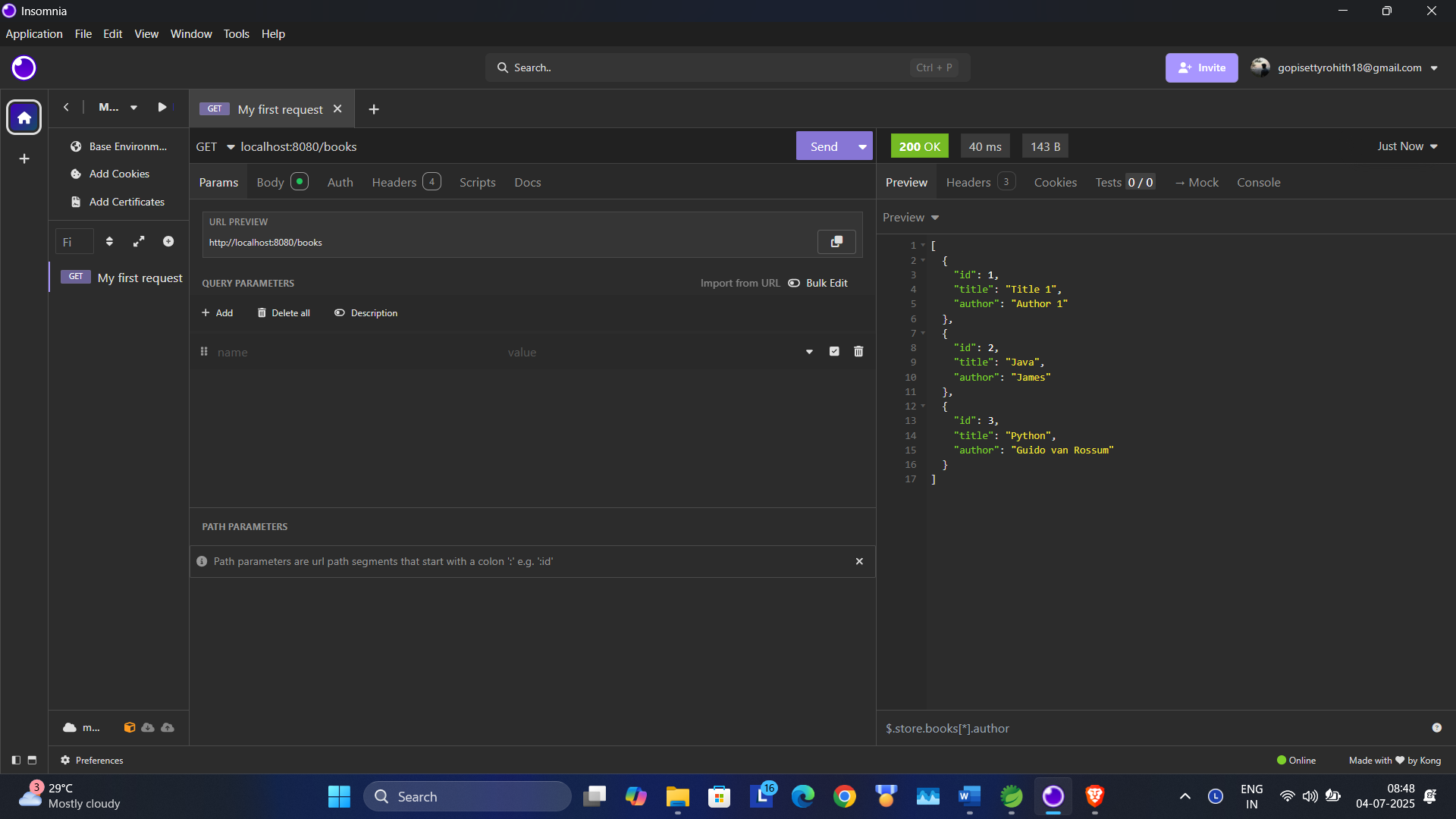
<http://localhost:8080/books/2>



PUT Method:

Previos:

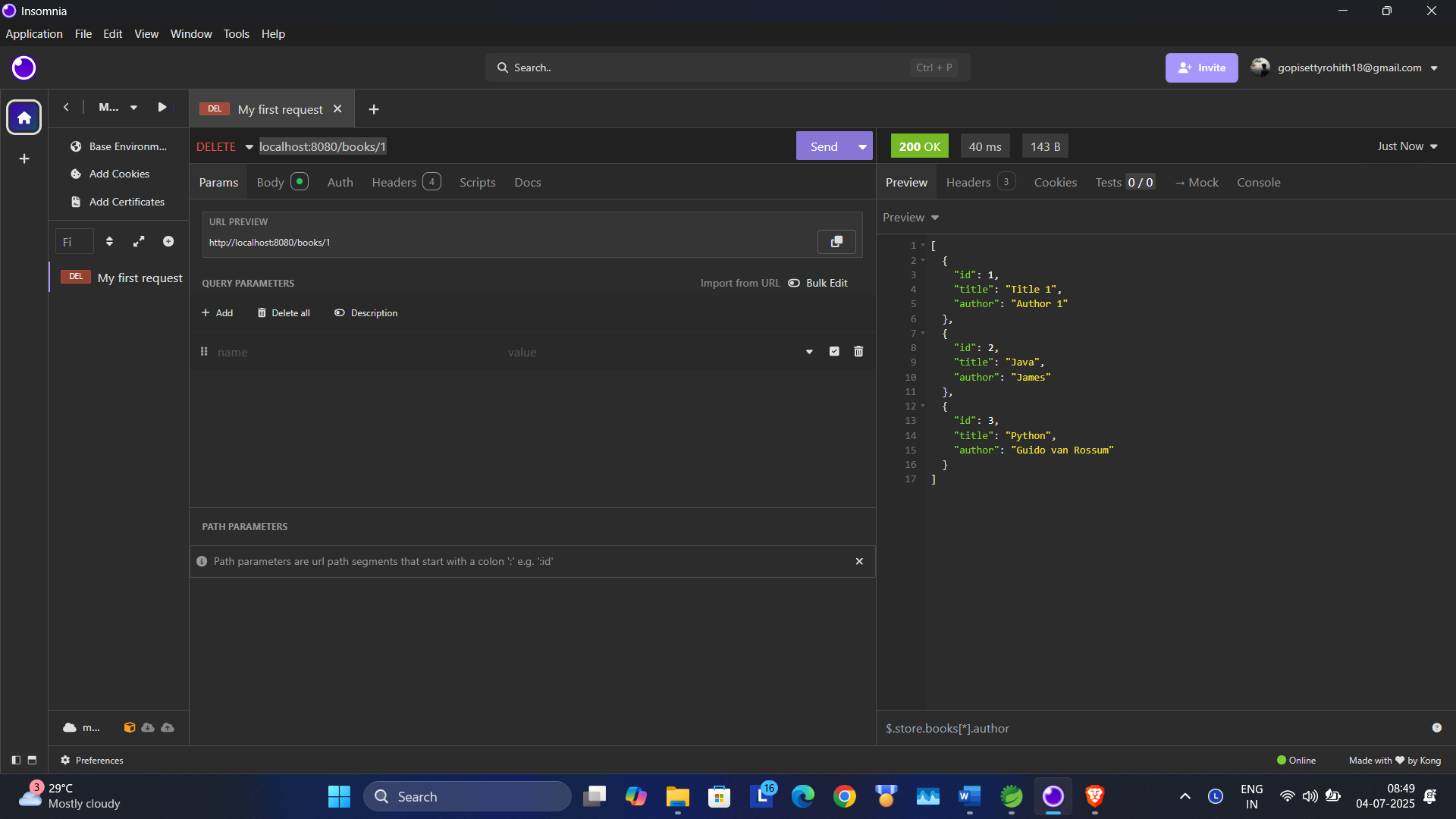




Delete Method:

url: <http://localhost:8080/books/1>

previous



After Deleting:

