# Week - 3 Spring Core

## Exercise - 1: Configuring a Basic Spring Application

**Step- 1: Set Up a Spring Project**

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</version>

<packaging>jar</packaging>

<dependencies>

<!-- Spring Core -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.36</version>

</dependency>

</dependencies>

</project>

**Step - 2: Configure the Application Context:**

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">

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

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

</beans>

**Step - 3: Define Service Repository Classes:**

BookRepository.java

package com.library.repository;

public class BookRepository {

public void getBooks() {

System.out.println("Fetching books from repository.");

}

}

BookService.java

package com.library.service;

public class BookService {

public void listBooks() {

System.out.println("Listing books...");

}

}

**Step - 4: Run the Application:**

LibraryManagementApplication.java

package com.library;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryManagementApplication {

public static void main(String[] args) {

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

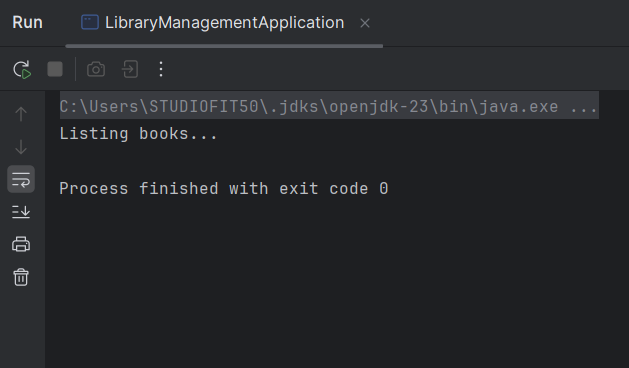
BookService service = (BookService) context.getBean("bookService");

service.listBooks();

}

}

**Output:**



**Exercise 2: Implementing Dependency Injection**

**Step- 1: Modify the XML Configuration**

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">

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

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

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

</bean>

</beans>

**Step- 2: Update the BookService class**

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 listBooks() {

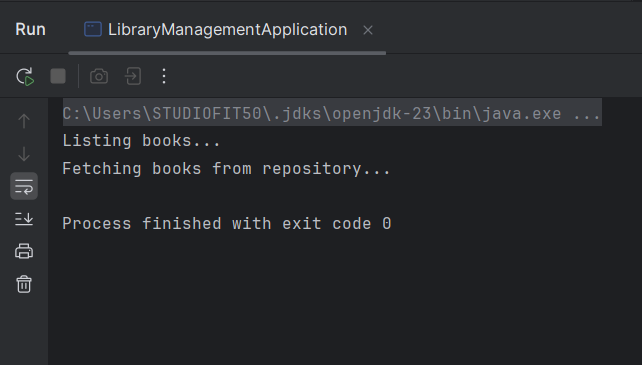
System.out.println("BookService: Listing books...");

bookRepository.getBooks();

}

}

**Step- 3: Test the configuration**



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

**Step- 1: Add Spring AOP Dependency**

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</version>

<packaging>jar</packaging>

<dependencies>

<!-- Spring Core -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.36</version>

</dependency>

<!-- AOP Dependency --!>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.36</version>

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.21</version>

</dependency>

</dependencies>

</project>

**Step- 2: Create an aspect for Logging**

LoggingAspect.java

package com.library.aspect;

import org.aspectj.lang.ProceedingJoinPoint;

public class LoggingAspect {

public Object logExecutionTime(ProceedingJoinPoint joinPoint) throws Throwable {

long start = System.currentTimeMillis();

Object result = joinPoint.proceed(); // run target method

long end = System.currentTimeMillis();

System.out.println("Execution time of " + joinPoint.getSignature() + ": " + (end - start) + "ms");

return result;

}

}

**Step- 3: Enable AspectJ Support**

applicationContext.xml

<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">

<!-- AOP Support -->

<aop:config>

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

<aop:around method="logExecutionTime" pointcut="execution(\* com.library.service.\*.\*(..))" />

</aop:aspect>

</aop:config>

<!-- Beans -->

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

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

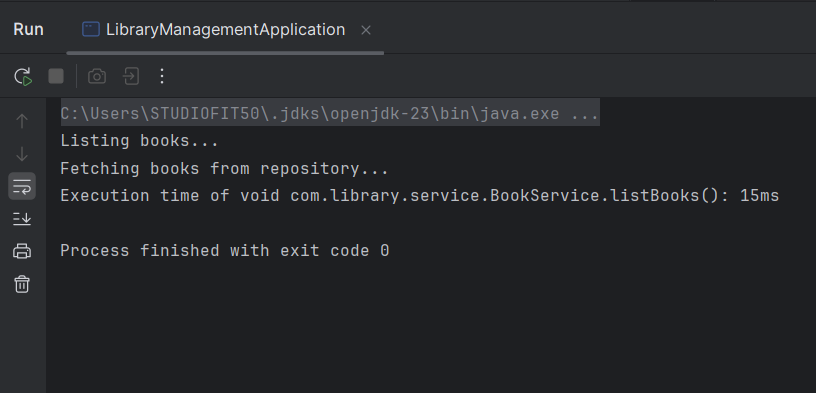
<property name="bookRepository" ref="bookRepository" />

</bean>

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

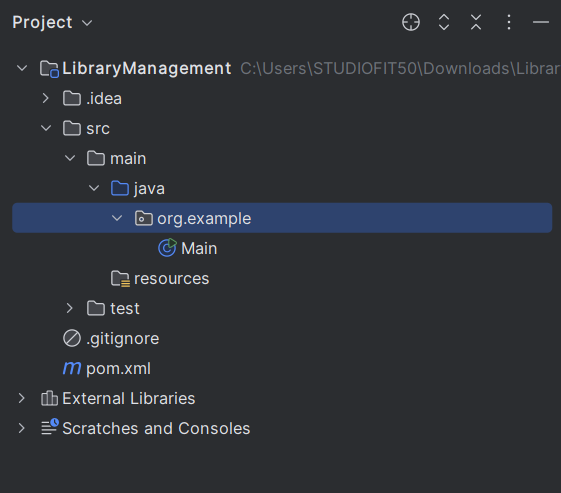
</beans>

**Step 4: Test the Aspect**



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

**Step 1: Create new Maven Project**



**Step 2: Add Spring Dependencies in pom.xml**

pom.xml

<?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>org.example</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>23</maven.compiler.source>

<maven.compiler.target>23</maven.compiler.target>

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

</properties>

<dependencies>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.36</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.36</version>

</dependency>

<!-- Spring WebMVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.36</version>

</dependency>

</dependencies>

</project>

**Step 3: Configure Maven Plugins**

pom.xml

<?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>org.example</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>23</maven.compiler.source>

<maven.compiler.target>23</maven.compiler.target>

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

</properties>

<dependencies>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.36</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.36</version>

</dependency>

<!-- Spring WebMVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.36</version>

</dependency>

</dependencies>

<!-- Maven Plugins --!>

<build>

<plugins>

<plugin>

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

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

<version>3.11.0</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

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

**Step 1: Create Spring Configuration File**

application.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">

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

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

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

</bean>

</beans>

**Step 2: Update the BookService Class**

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 listBooks() {

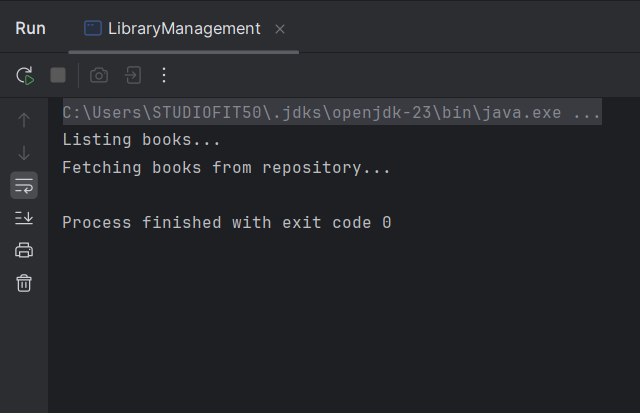
System.out.println("Listing books...");

bookRepository.getBooks();

}

}

**Step - 3: Run the Application**



**Exercise 6: Configuring Beans with Annotations**

**Step 1: Enable Component Scanning**

applicationContext.xml

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

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

<!-- Enable annotation-based config -->

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

</beans>

**Step 2: Annotate Classes**

BookService.java

package com.library.service;  
  
import com.library.repository.BookRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
@Service  
public class BookService {  
 private BookRepository bookRepository;  
  
 @Autowired  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void listBooks() {  
 System.*out*.println("Listing books...");  
 bookRepository.getBooks();  
 }  
}

BookRepository.java

package com.library.repository;

import org.springframework.stereotype.Repository;

@Repository

public class BookRepository {

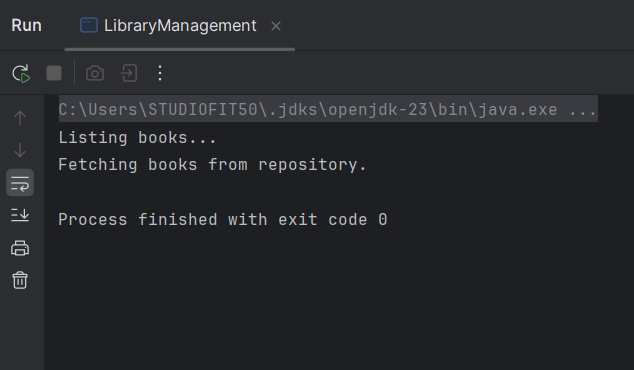
public void getBooks() {

System.out.println("Fetching books from repository.");

}

}

**Step 3: Test the Configuration**



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

**Step-1: Configure Constructor 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"  
 xmlns:context="http://www.springframework.org/schema/context"  
 xsi:schemaLocation="  
 http://www.springframework.org/schema/beans https://www.springframework.org/schema/beans/spring-beans.xsd  
 http://www.springframework.org/schema/context https://www.springframework.org/schema/context/spring-context.xsd">  
  
 <bean id="bookRepository" class="com.library.repository.BookRepository" />  
  
 <bean id="bookService" class="com.library.service.BookService">  
 <constructor-arg ref="bookRepository" />  
 </bean>  
  
</beans>

**Step 2: Configure Setter Injection**

BookService.java

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Constructor injection

public BookService(BookRepository bookRepository) {

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

this.bookRepository = bookRepository;

}

// Setter injection

public void setBookRepository(BookRepository bookRepository) {

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

this.bookRepository = bookRepository;

}

public void listBooks() {

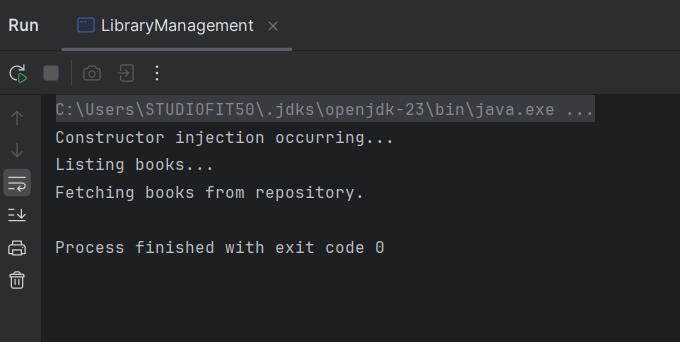
System.out.println("Listing books...");

bookRepository.getBooks();

}

}

**Step 3: Test the Injection**



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

**Step 1 & 2: Define an Aspect and create advice methods**

LoggingAspect.java

package com.library.aspect;

public class LoggingAspect {

public void beforeMethod() {

System.out.println("--Before method execution--");

}

public void afterMethod() {

System.out.println("--After method execution--");

}

}

**Step 3: Configure the Aspect**

applicationContext.xml

<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 AOP -->

<aop:config>

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

<aop:before method="beforeMethod" pointcut="execution(\* com.library.service.\*.\*(..))" />

<aop:after method="afterMethod" pointcut="execution(\* com.library.service.\*.\*(..))" />

</aop:aspect>

</aop:config>

<!-- Beans -->

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

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

<constructor-arg ref="bookRepository" />

</bean>

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

</beans>

pom.xml

<?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>org.example</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>23</maven.compiler.source>

<maven.compiler.target>23</maven.compiler.target>

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

</properties>

<dependencies>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.36</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.36</version>

</dependency>

<!-- Spring WebMVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.36</version>

</dependency>

<!-- AspectJ 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>3.11.0</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

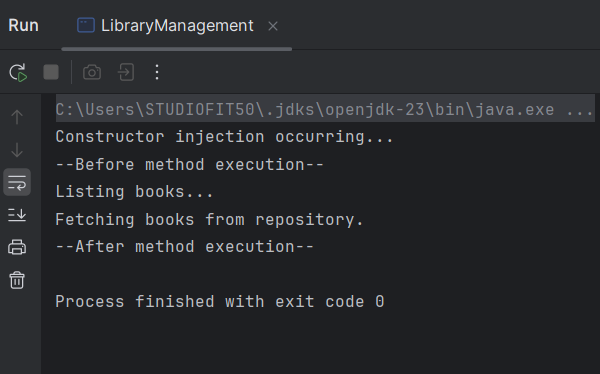
</plugin>

</plugins>

</build>

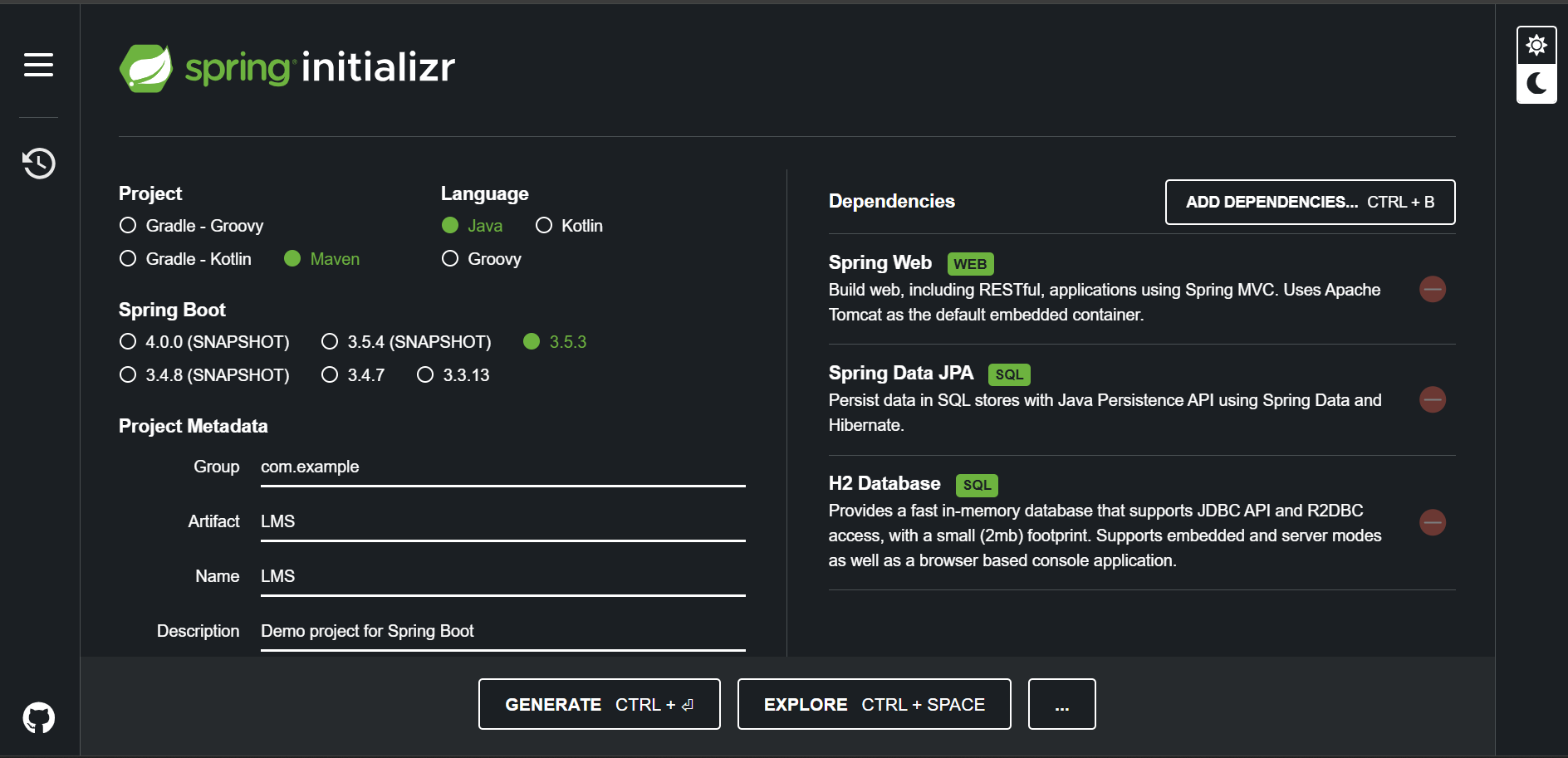
</project>

**Step 4: Test the Aspect**



**Exercise 9: Implementing Basic AOP with Spring**

**Step 1 & 2: Create a Spring Boot Project and Add dependencies**



**Step-3: Create application properties**

spring.application.name=LMS

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.jpa.hibernate.ddl-auto=update

spring.h2.console.enabled=true

**Step 4: Define Entities and Repositories**

Book.java

package com.LMS.model;

import jakarta.persistence.\*;

@Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

private String author;

public void setId(Long id) {

this.id = id;

}

public Long getId() {

return 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.LMS.repository;

import com.LMS.model.Book;

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

public interface BookRepository extends JpaRepository<Book, Long> {

}

**Step 5: Create a REST Controller**

BookController.java

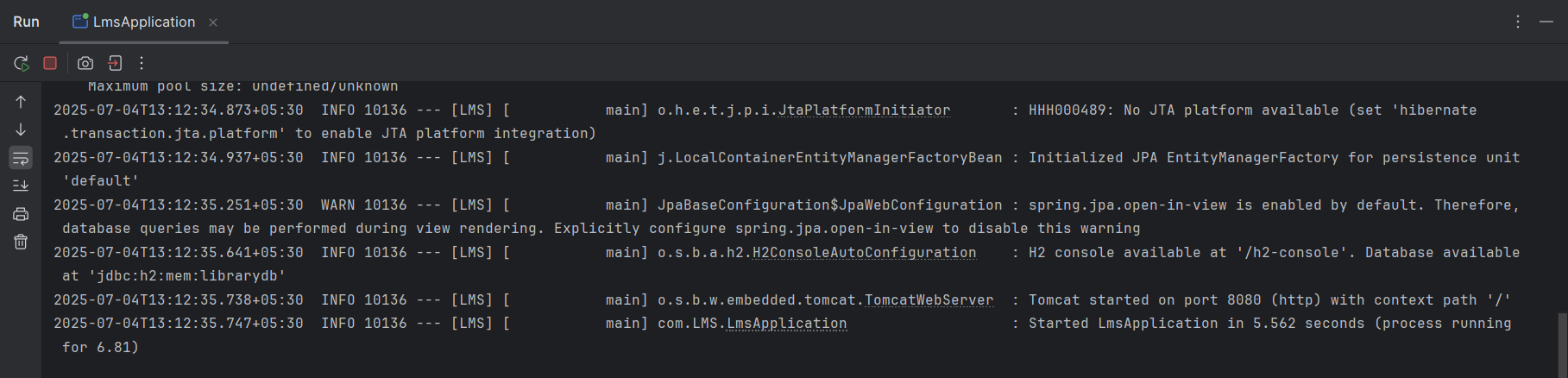
package com.LMS.controller;  
  
import com.LMS.model.Book;  
import com.LMS.repository.BookRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/books")  
public class BookController {  
  
 @Autowired  
 private BookRepository bookRepository;  
  
 @GetMapping  
 public List<Book> getAllBooks() {  
 return bookRepository.findAll();  
 }  
  
 @PostMapping  
 public Book addBook(@RequestBody Book book) {  
 return bookRepository.save(book);  
 }  
  
 @GetMapping("/{id}")  
 public Book getBookById(@PathVariable Long id) {  
 return bookRepository.findById(id).orElse(null);  
 }  
  
 @PutMapping("/{id}")  
 public Book updateBook(@PathVariable Long id, @RequestBody Book book) {  
 book.setId(id);  
 return bookRepository.save(book);  
 }  
  
 @DeleteMapping("/{id}")  
 public void deleteBook(@PathVariable Long id) {  
 bookRepository.deleteById(id);  
 }  
}

**Step 6: Run the Application**

LmsApplication.java

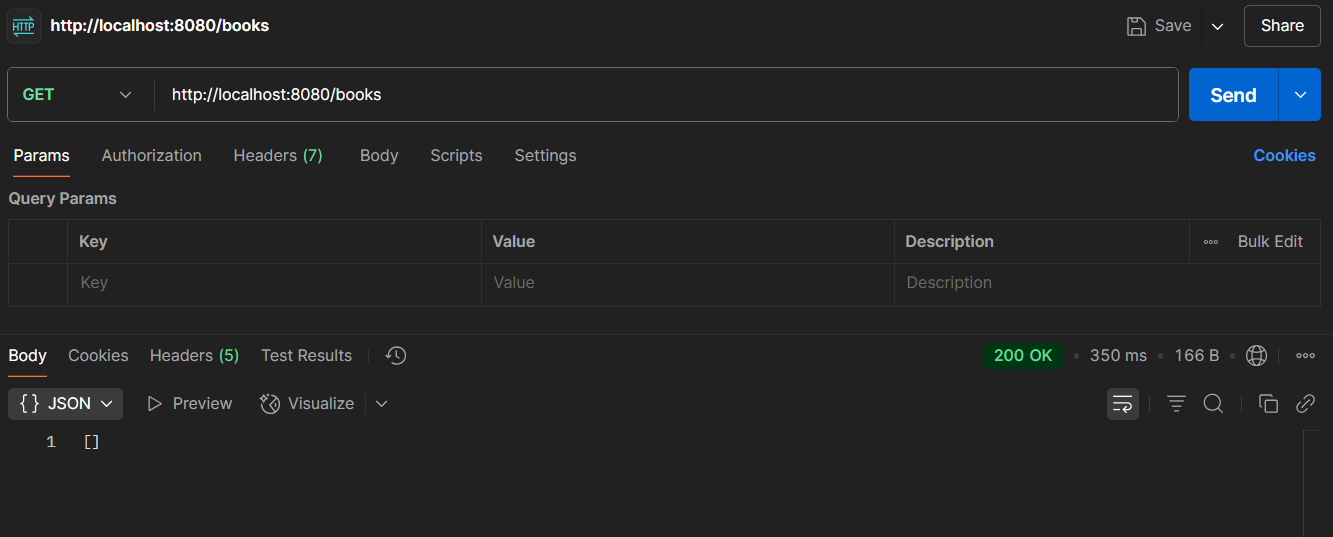
package com.LMS;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class LmsApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(LmsApplication.class, args);  
 }  
  
}

Output:

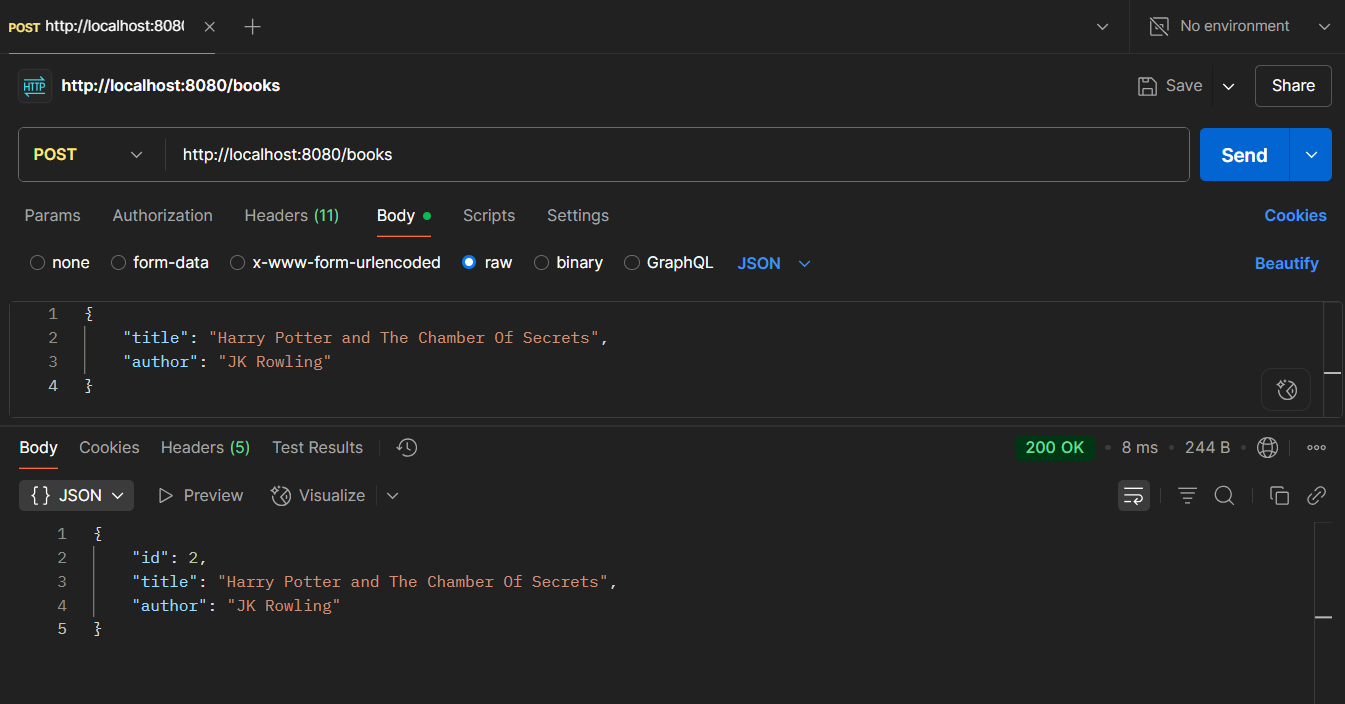
**Terminal (Server running success) :**

**Testing using Postman:**

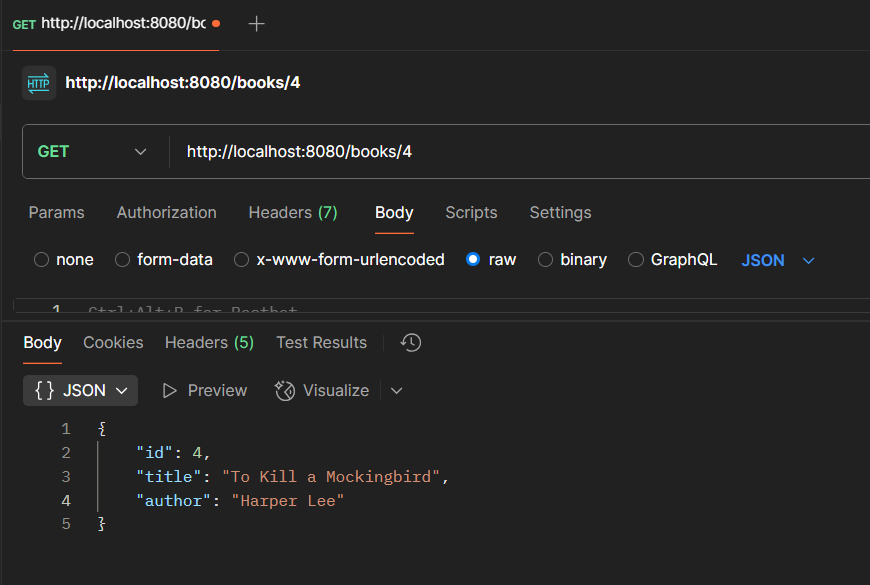
GET /books (Empty book list)



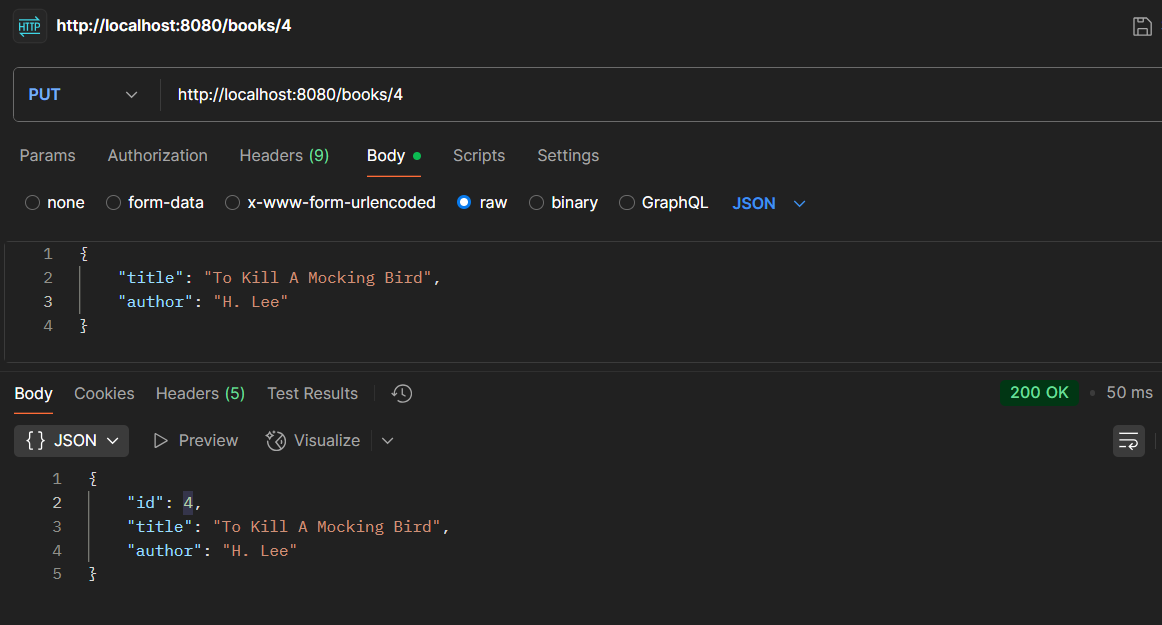
POST /books



GET /books/id



PUT /books/id



DELETE books/{id}

