COGNIZANT

Digital Nurture 4.0

Deep Skilling - Java FSE

WEEK-3 HANDS ON

By Kaviya P

**SPRING CORE\_MAVEN**

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

**Scenario:**

Your company is developing a web application for managing a library. You need to use the Spring Framework to handle the backend operations.

**Steps:**

1. **Set Up a Spring Project:**
   * Create a Maven project named **LibraryManagement**.
   * Add Spring Core dependencies in the **pom.xml** file.
2. **Configure the Application Context:**
   * Create an XML configuration file named **applicationContext.xml** in the **src/main/resources** directory.
   * Define beans for **BookService** and **BookRepository** in the XML file.
3. **Define Service and Repository Classes:**
   * Create a package **com.library.service** and add a class **BookService**.
   * Create a package **com.library.repository** and add a class **BookRepository**.
4. **Run the Application:**
   * Create a main class to load the Spring context and test the configuration.

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

<modelVersion>4.0.0</modelVersion>

<parent>

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

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

<version>3.5.3</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.week3</groupId>

<artifactId>SpringAndMaven</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>SpringAndMaven</name>

<description>Demo project for Spring Boot</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**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 id="bookRepository" class="com.week3.SpringAndMaven.repository.BookRepository" />

<bean id="bookService" class="com.week3.SpringAndMaven.service.BookService">

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

</bean>

</beans>

**BookRepository.java**

**package** com.week3.SpringAndMaven.repository;

**import** org.springframework.stereotype.Repository;

@Repository

**public** **class** BookRepository {

**public** **void** saveBook(String title) {

System.***out***.println("BookRepository: Book \"" + title + "\" saved to the database.");

}

}

**BookService.java**

**package** com.week3.SpringAndMaven.service;

**import** com.week3.SpringAndMaven.repository.BookRepository;

**public** **class** BookService {

**private** BookRepository bookRepository;

// Setter for dependency injection

**public** **void** setBookRepository(BookRepository bookRepository) {

**this**.bookRepository = bookRepository;

}

**public** **void** addBook(String title) {

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

bookRepository.saveBook(title);

}

}

**SpringAndMavenApplication.java**

**package** com.week3.SpringAndMaven;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** com.week3.SpringAndMaven.service.BookService;

@SpringBootApplication

**public** **class** SpringAndMavenApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(SpringAndMavenApplication.**class**, args);

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

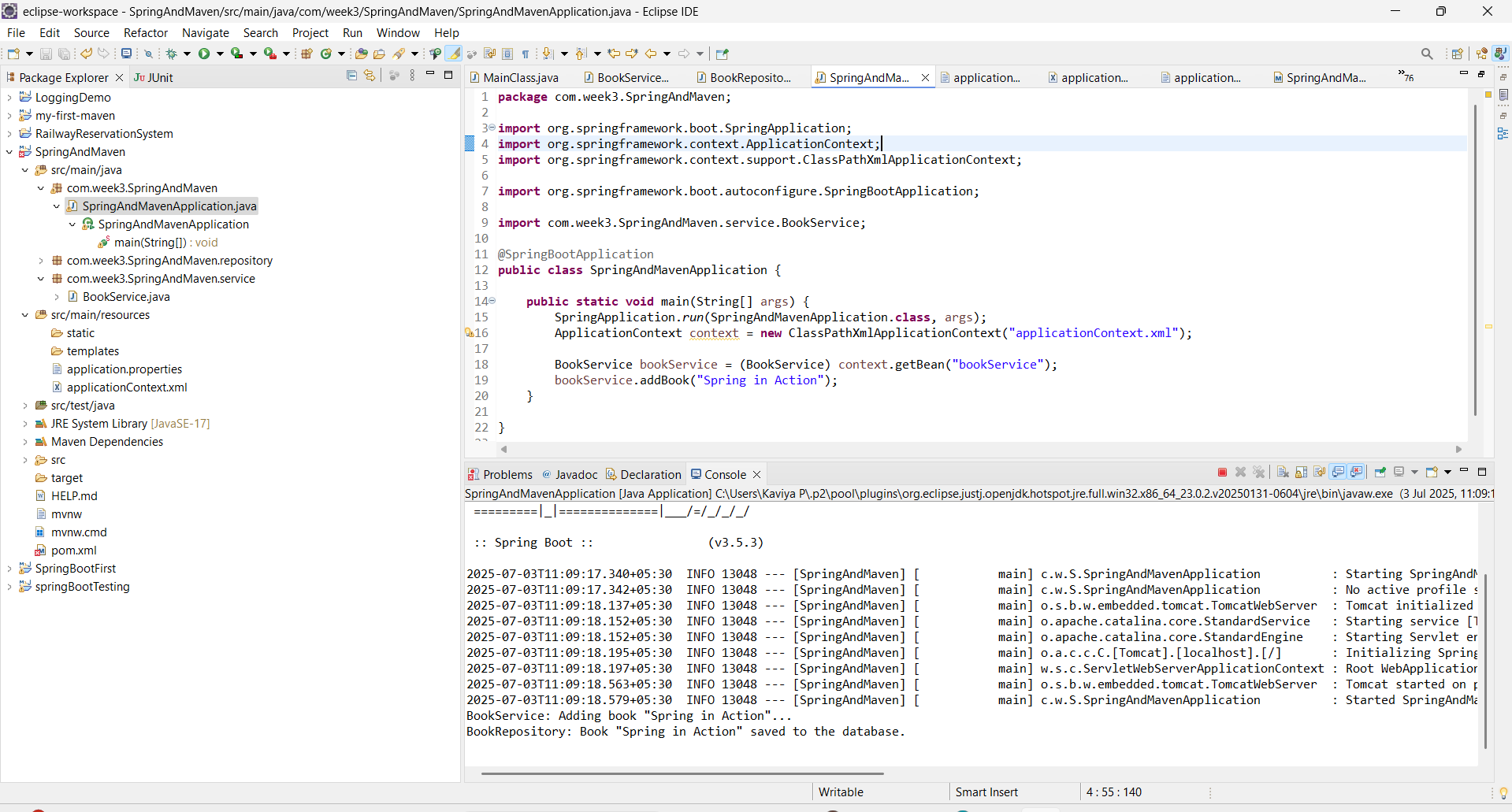
BookService bookService = (BookService) context.getBean("bookService");

bookService.addBook("Spring in Action");

}

}

OUTPUT

****

**Exercise 2: Implementing Dependency Injection**

**Scenario:**

In the library management application, you need to manage the dependencies between the BookService and BookRepository classes using Spring's IoC and DI.

**Steps:**

1. **Modify the XML Configuration:**
   * Update **applicationContext.xml** to wire **BookRepository** into **BookService**.
2. **Update the BookService Class:**
   * Ensure that **BookService** class has a setter method for **BookRepository**.
3. **Test the Configuration:**
   * Run the **LibraryManagementApplication** main class to verify the 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">

<!-- BookRepository Bean -->

<bean id="bookRepository" class="com.week3.SpringAndMaven.repository.BookRepository" />

<!-- BookService Bean with Dependency Injection -->

<bean id="bookService" class="com.week3.SpringAndMaven.service.BookService">

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

</bean>

</beans>

**BookRepository.java**

**package** com.week3.SpringAndMaven.repository;

**import** org.springframework.stereotype.Repository;

@Repository

**public** **class** BookRepository {

**public** **void** saveBook(String title) {

System.***out***.println("BookRepository: Book \"" + title + "\" saved to the database.");

}

}

**BookService.java**

**package** com.week3.SpringAndMaven.service;

**import** com.week3.SpringAndMaven.repository.BookRepository;

**public** **class** BookService {

**private** BookRepository bookRepository;

// Setter for dependency injection

**public** **void** setBookRepository(BookRepository bookRepository) {

**this**.bookRepository = bookRepository;

}

**public** **void** addBook(String title) {

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

bookRepository.saveBook(title);

}

}

**SpringAndMavenApplication.java**

**package** com.week3.SpringAndMaven;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** com.week3.SpringAndMaven.service.BookService;

@SpringBootApplication

**public** **class** SpringAndMavenApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(SpringAndMavenApplication.**class**, args);

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

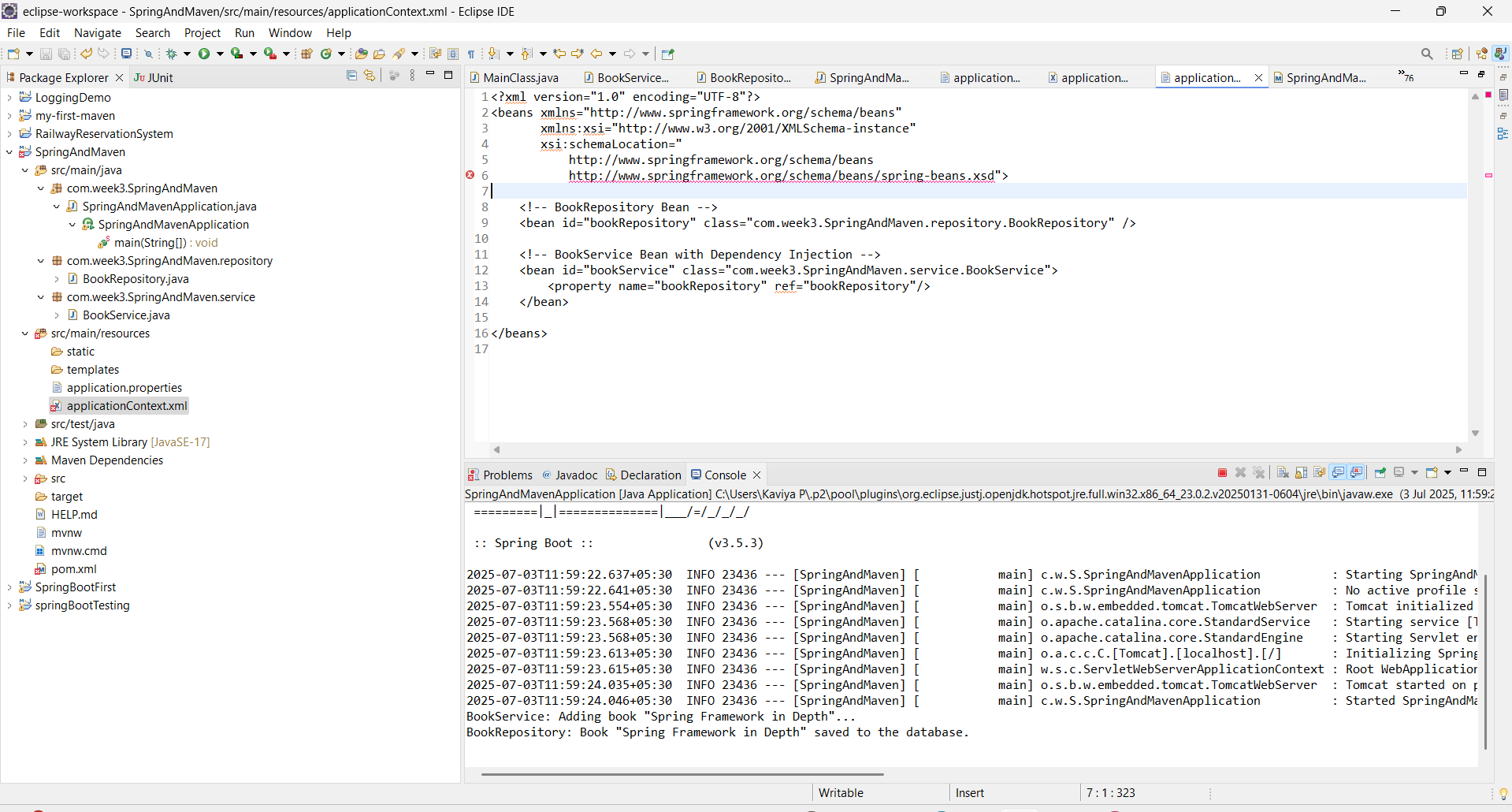
BookService bookService = (BookService) context.getBean("bookService");

bookService.addBook("Spring Framework in Depth");

}

}

**OUTPUT**

****

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

**Scenario:**

The library management application requires logging capabilities to track method execution times.

**Steps:**

1. **Add Spring AOP Dependency:**
   * Update **pom.xml** to include Spring AOP dependency.
2. **Create an Aspect for Logging:**
   * Create a package **com.library.aspect** and add a class **LoggingAspect** with a method to log execution times.
3. **Enable AspectJ Support:**
   * Update **applicationContext.xml** to enable **AspectJ** support and register the aspect.
4. **Test the Aspect:**
   * Run the **LibraryManagementApplication** main class and observe the console for log messages indicating method execution times.

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

<modelVersion>4.0.0</modelVersion>

<parent>

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

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

<version>3.5.3</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.week3</groupId>

<artifactId>SpringAndMaven</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>SpringAndMaven</name>

<description>Demo project for Spring Boot</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.21</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**SpringAndMavenApplication.java**

**package** com.week3.SpringAndMaven;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** com.week3.SpringAndMaven.service.BookService;

@SpringBootApplication

**public** **class** SpringAndMavenApplication {

**public** **static** **void** main(String[] args) {

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

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

service.addBook("Spring in Action");

((ClassPathXmlApplicationContext) context).close(); }

}

**LoggingAspect.java**

**package** com.week3.SpringAndMaven.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.week3.SpringAndMaven.service.\*.\*(..))")

**public** Object logExecutionTime(ProceedingJoinPoint joinPoint) **throws** Throwable {

**long** start = System.*currentTimeMillis*();

Object proceed = joinPoint.proceed();

**long** end = System.*currentTimeMillis*();

System.***out***.println("[LOG] Method " + joinPoint.getSignature() + " executed in " + (end - start) + " ms");

**return** proceed;

}

}

**applicationContext.xml**

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

<beans xmlns="http://www.springframework.org/schema/beans"

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

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

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

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

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

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

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

<!-- Enable component scanning -->

<context:component-scan base-package="com.week3.SpringAndMaven"/>

<!-- Enable AOP support -->

<aop:aspectj-autoproxy />

<!-- Beans -->

<bean id="bookRepository" class="com.week3.SpringAndMaven.repository.BookRepository" />

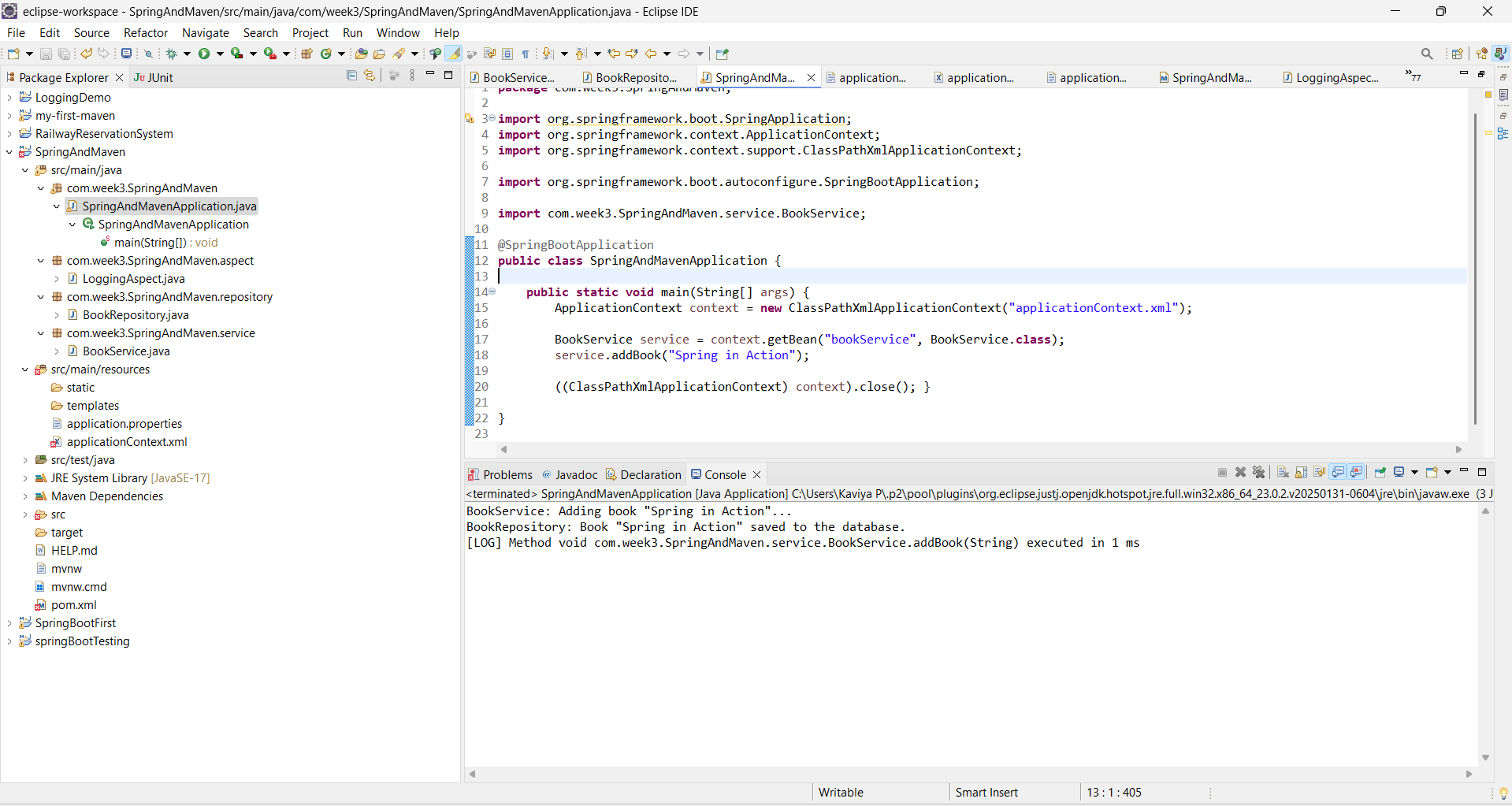
<bean id="bookService" class="com.week3.SpringAndMaven.service.BookService">

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

</bean>

</beans>

**OUTPUT :**

****

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

**Scenario:**

You need to set up a new Maven project for the library management application and add Spring dependencies.

**Steps:**

1. **Create a New Maven Project:**
   * Create a new Maven project named **LibraryManagement**.
2. **Add Spring Dependencies in pom.xml:**
   * Include dependencies for Spring Context, Spring AOP, and Spring WebMVC.
3. **Configure Maven Plugins:**
   * Configure the Maven Compiler Plugin for Java version 1.8 in the pom.xml file.

**SpringAndMavenApplication.java**

**package** com.week3.SpringAndMaven;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** com.week3.SpringAndMaven.service.BookService;

@SpringBootApplication

**public** **class** SpringAndMavenApplication {

**public** **static** **void** main(String[] args) {

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

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

service.addBook("Spring in Action");

((ClassPathXmlApplicationContext) context).close(); }

}

**LoggingAspect.java**

**package** com.week3.SpringAndMaven.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.week3.SpringAndMaven.service.\*.\*(..))")

**public** Object logExecutionTime(ProceedingJoinPoint joinPoint) **throws** Throwable {

**long** start = System.*currentTimeMillis*();

Object proceed = joinPoint.proceed();

**long** end = System.*currentTimeMillis*();

System.***out***.println("[LOG] Method " + joinPoint.getSignature() + " executed in " + (end - start) + " ms");

**return** proceed;

}

}

**applicationContext.xml**

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

<beans xmlns="http://www.springframework.org/schema/beans"

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

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

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

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

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

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

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

<!-- Enable component scanning -->

<context:component-scan base-package="com.week3.SpringAndMaven"/>

<!-- Enable AOP support -->

<aop:aspectj-autoproxy />

<!-- Beans -->

<bean id="bookRepository" class="com.week3.SpringAndMaven.repository.BookRepository" />

<bean id="bookService" class="com.week3.SpringAndMaven.service.BookService">

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

</bean>

</beans>

**BookRepository.java**

**package** com.week3.SpringAndMaven.repository;

**import** org.springframework.stereotype.Repository;

@Repository

**public** **class** BookRepository {

**public** **void** saveBook(String title) {

System.***out***.println("BookRepository: Book \"" + title + "\" saved to the database.");

}

}

**BookService.java**

**package** com.week3.SpringAndMaven.service;

**import** com.week3.SpringAndMaven.repository.BookRepository;

**public** **class** BookService {

**private** BookRepository bookRepository;

// Setter for dependency injection

**public** **void** setBookRepository(BookRepository bookRepository) {

**this**.bookRepository = bookRepository;

}

**public** **void** addBook(String title) {

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

bookRepository.saveBook(title);

}

}

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

<modelVersion>4.0.0</modelVersion>

<parent>

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

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

<version>3.5.3</version>

<relativePath/>

</parent>

<groupId>com.week3</groupId>

<artifactId>SpringAndMaven</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>SpringAndMaven</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<!-- Spring Web -->

<dependency>

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

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

</dependency>

<!-- Spring AOP -->

<dependency>

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

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

</dependency>

<!-- AspectJ (if needed explicitly) -->

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.21</version>

</dependency>

<!-- Spring Test -->

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

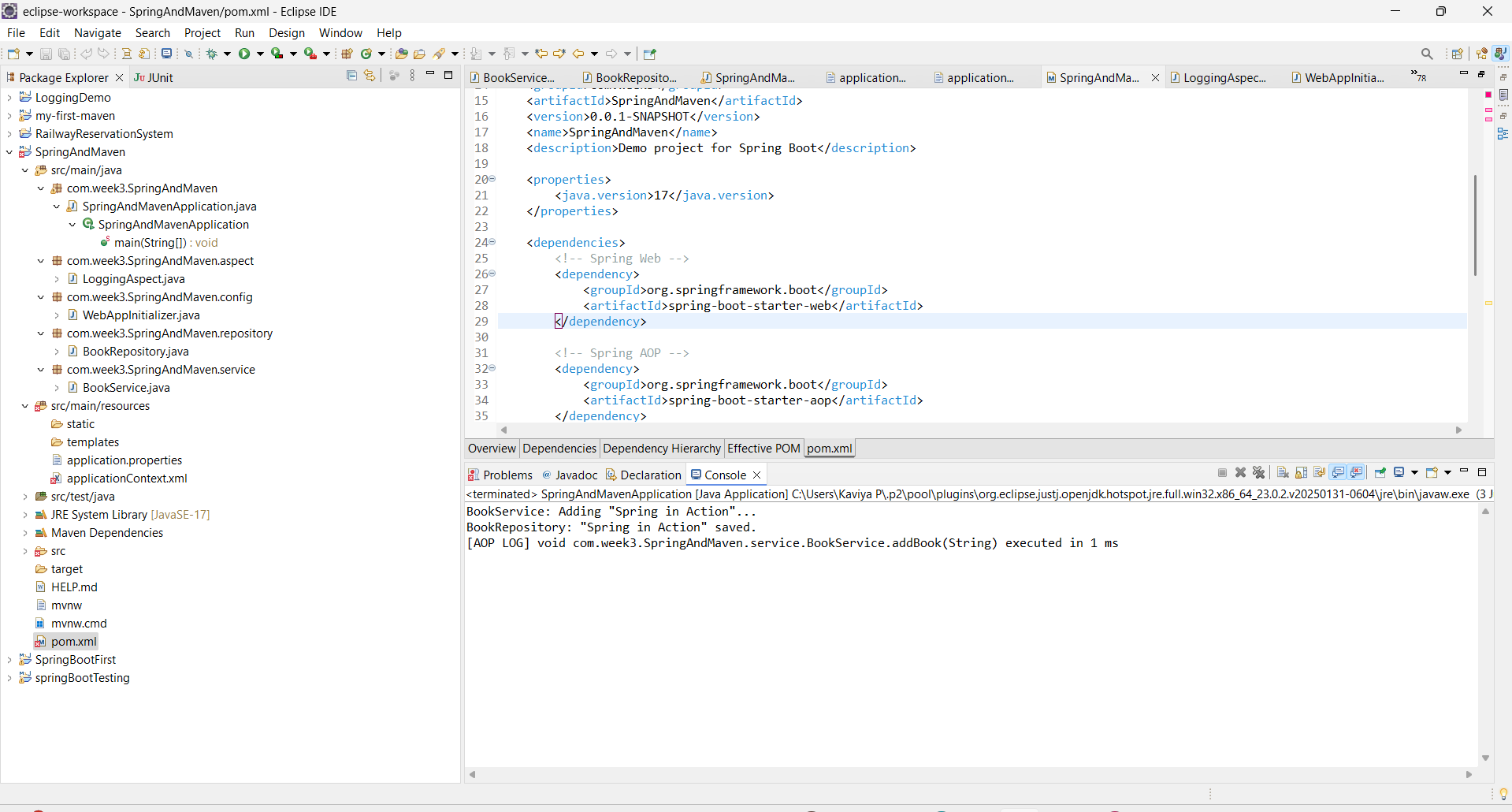
</plugin>

</plugins>

</build>

</project>

**OUTPUT:**

****

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

**Scenario:**

The library management application requires a central configuration for beans and dependencies.

**Steps:**

1. **Create Spring Configuration File:**
   * Create an XML configuration file named **applicationContext.xml** in the **src/main/resources** directory.
   * Define beans for **BookService** and **BookRepository** in the XML file.
2. **Update the BookService Class:**
   * Ensure that the **BookService** class has a setter method for **BookRepository**.
3. **Run the Application:**
   * Create a main class to load the Spring context and test the configuration.

**BookRepository.java**

**package** com.week3.SpringAndMaven.repository;

**import** org.springframework.stereotype.Repository;

@Repository

**public** **class** BookRepository {

**public** **void** saveBook(String title) {

System.***out***.println("BookRepository: Book \"" + title + "\" saved to the database.");

}

}

**BookService.java**

**package** com.week3.SpringAndMaven.service;

**import** com.week3.SpringAndMaven.repository.BookRepository;

**public** **class** BookService {

**private** BookRepository bookRepository;

// Setter for dependency injection

**public** **void** setBookRepository(BookRepository bookRepository) {

**this**.bookRepository = bookRepository;

}

**public** **void** addBook(String title) {

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

bookRepository.saveBook(title);

}

}

**SpringAndMavenApplication.java**

**package** com.week3.SpringAndMaven;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** com.week3.SpringAndMaven.service.BookService;

@SpringBootApplication

**public** **class** SpringAndMavenApplication {

**public** **static** **void** main(String[] args) {

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

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

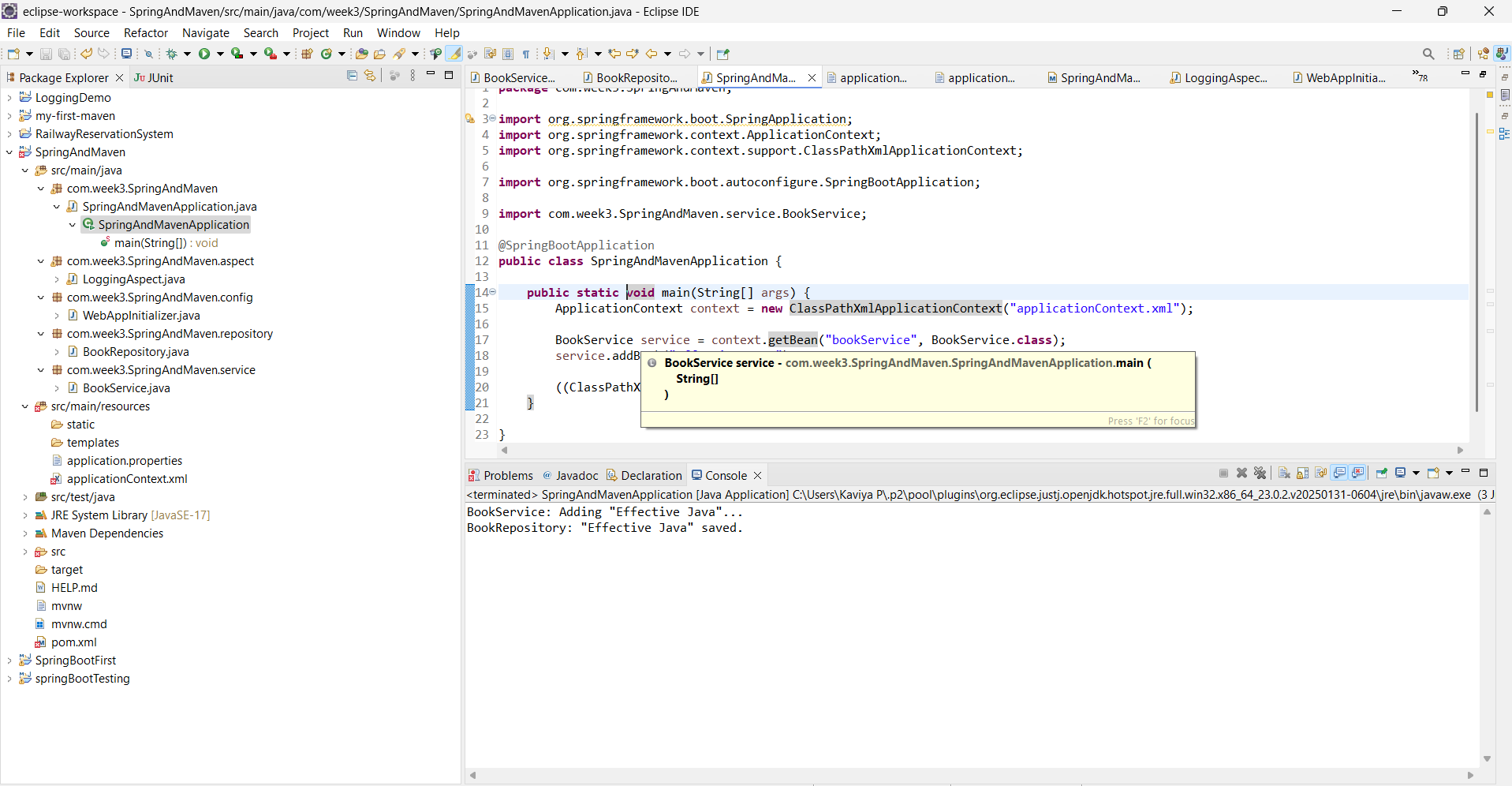
service.addBook("Effective Java");

((ClassPathXmlApplicationContext) context).close();

}

}

**OUTPUT:**

****

**Exercise 6: Configuring Beans with Annotations**

**Scenario:**

You need to simplify the configuration of beans in the library management application using annotations.

**Steps:**

1. **Enable Component Scanning:**
   * Update **applicationContext.xml** to include component scanning for the **com.library** package.
2. **Annotate Classes:**
   * Use **@Service** annotation for the **BookService** class.
   * Use **@Repository** annotation for the **BookRepository** class.
3. **Test the Configuration:**
   * Run the **LibraryManagementApplication** main class to verify the annotation-based configuration.

**SpringAndMavenApplication.java**

**package** com.week3.SpringAndMaven;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** com.week3.SpringAndMaven.service.BookService;

@SpringBootApplication

**public** **class** SpringAndMavenApplication {

**public** **static** **void** main(String[] args) {

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

BookService service = context.getBean(BookService.**class**);

service.addBook("Spring Boot in Practice");

((ClassPathXmlApplicationContext) context).close();

}

}

**BookRepository.java**

**package** com.week3.SpringAndMaven.repository;

**import** org.springframework.stereotype.Repository;

@Repository

**public** **class** BookRepository {

**public** **void** saveBook(String title) {

System.***out***.println("BookRepository: \"" + title + "\" saved.");

}

}

**BookService.java**

**package** com.week3.SpringAndMaven.service;

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

**import** org.springframework.stereotype.Service;

**import** com.week3.SpringAndMaven.repository.BookRepository;

@Service

**public** **class** BookService {

@Autowired

**private** BookRepository bookRepository;

**public** **void** addBook(String title) {

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

bookRepository.saveBook(title);

}

}

**applicationContext.xml**

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

<beans xmlns="http://www.springframework.org/schema/beans"

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

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

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

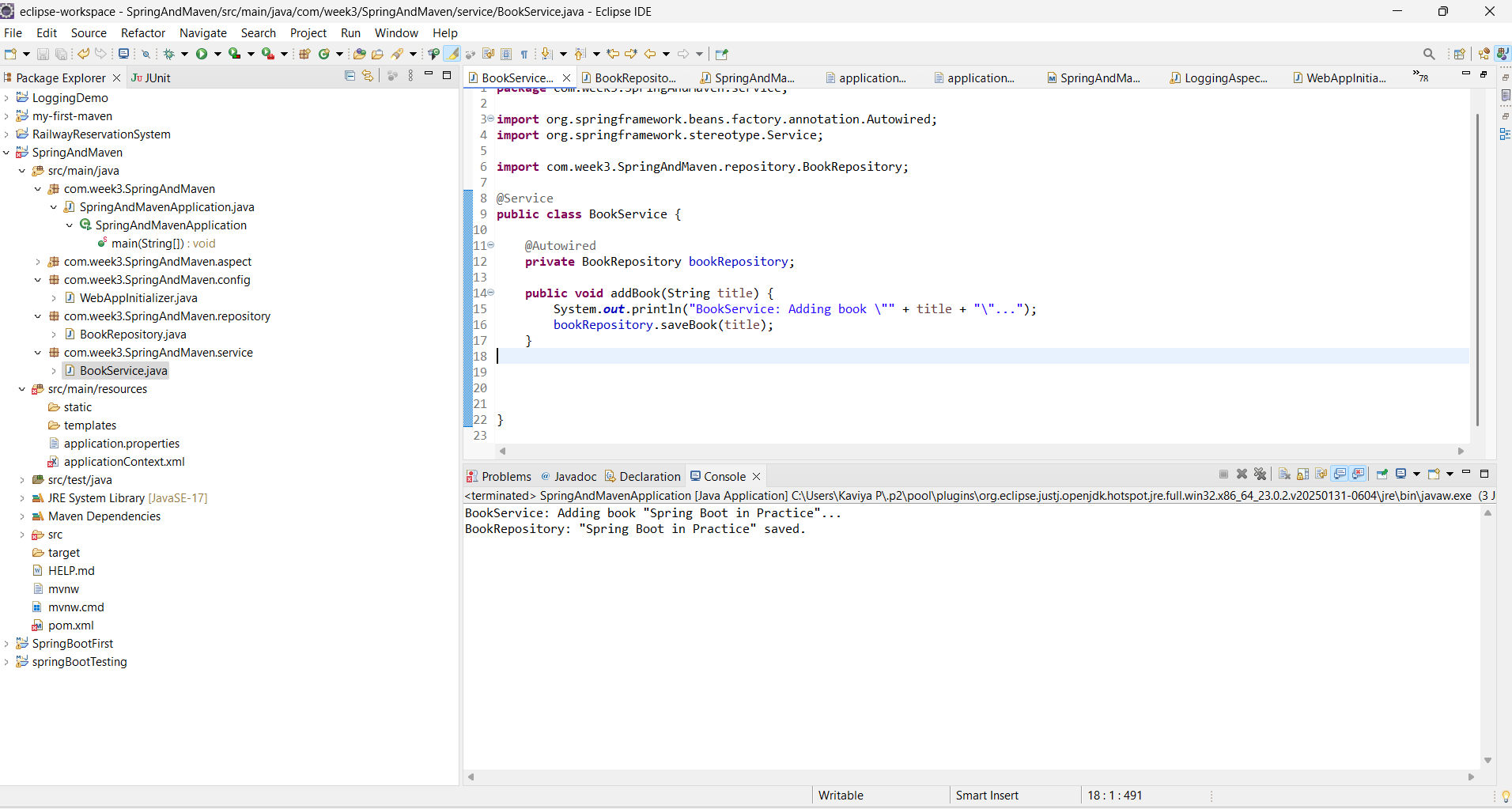
http://www.springframework.org/schema/context/spring-context.xsd">

<!-- Enable annotation-based configuration -->

<context:component-scan base-package="com.week3.SpringAndMaven" />

</beans>

**OUTPUT**

****

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

**Scenario:**

The library management application requires both constructor and setter injection for better control over bean initialization.

**Steps:**

1. **Configure Constructor Injection:**
   * Update applicationContext.**xml** to configure constructor injection for **BookService**.
2. **Configure Setter Injection:**
   * Ensure that the **BookService** class has a setter method for **BookRepository** and configure it in **applicationContext.xml**.
3. **Test the Injection:**
   * Run the **LibraryManagementApplication** main class to verify both constructor and setter injection.

**SpringAndMavenApplication.java**

**package** com.week3.SpringAndMaven;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** com.week3.SpringAndMaven.service.BookService;

@SpringBootApplication

**public** **class** SpringAndMavenApplication {

**public** **static** **void** main(String[] args) {

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

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

service.addBook("Spring in Action");

((ClassPathXmlApplicationContext) context).close();

}

}

**BookRepository.java**

**package** com.week3.SpringAndMaven.repository;

**import** org.springframework.stereotype.Repository;

@Repository

**public** **class** BookRepository {

**public** **void** saveBook(String title) {

System.***out***.println("BookRepository: Book \"" + title + "\" saved to the database.");

}

}

**BookService.java**

**package** com.week3.SpringAndMaven.service;

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

**import** org.springframework.stereotype.Service;

**import** com.week3.SpringAndMaven.repository.BookRepository;

**public** **class** BookService {

**private** BookRepository bookRepository;

// Constructor Injection

**public** BookService(BookRepository bookRepository) {

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

**this**.bookRepository = bookRepository;

}

// Setter Injection

**public** **void** setBookRepository(BookRepository bookRepository) {

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

**this**.bookRepository = bookRepository;

}

**public** **void** addBook(String title) {

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

bookRepository.saveBook(title);

}

}

**Application.xml**

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

<beans xmlns="http://www.springframework.org/schema/beans"

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

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

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

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

<!-- BookRepository Bean -->

<bean id="bookRepository" class="com.week3.SpringAndMaven.repository.BookRepository" />

<!-- BookService Bean with constructor and setter injection -->

<bean id="bookService" class="com.week3.SpringAndMaven.service.BookService">

<!-- Constructor Injection -->

<constructor-arg ref="bookRepository"/>

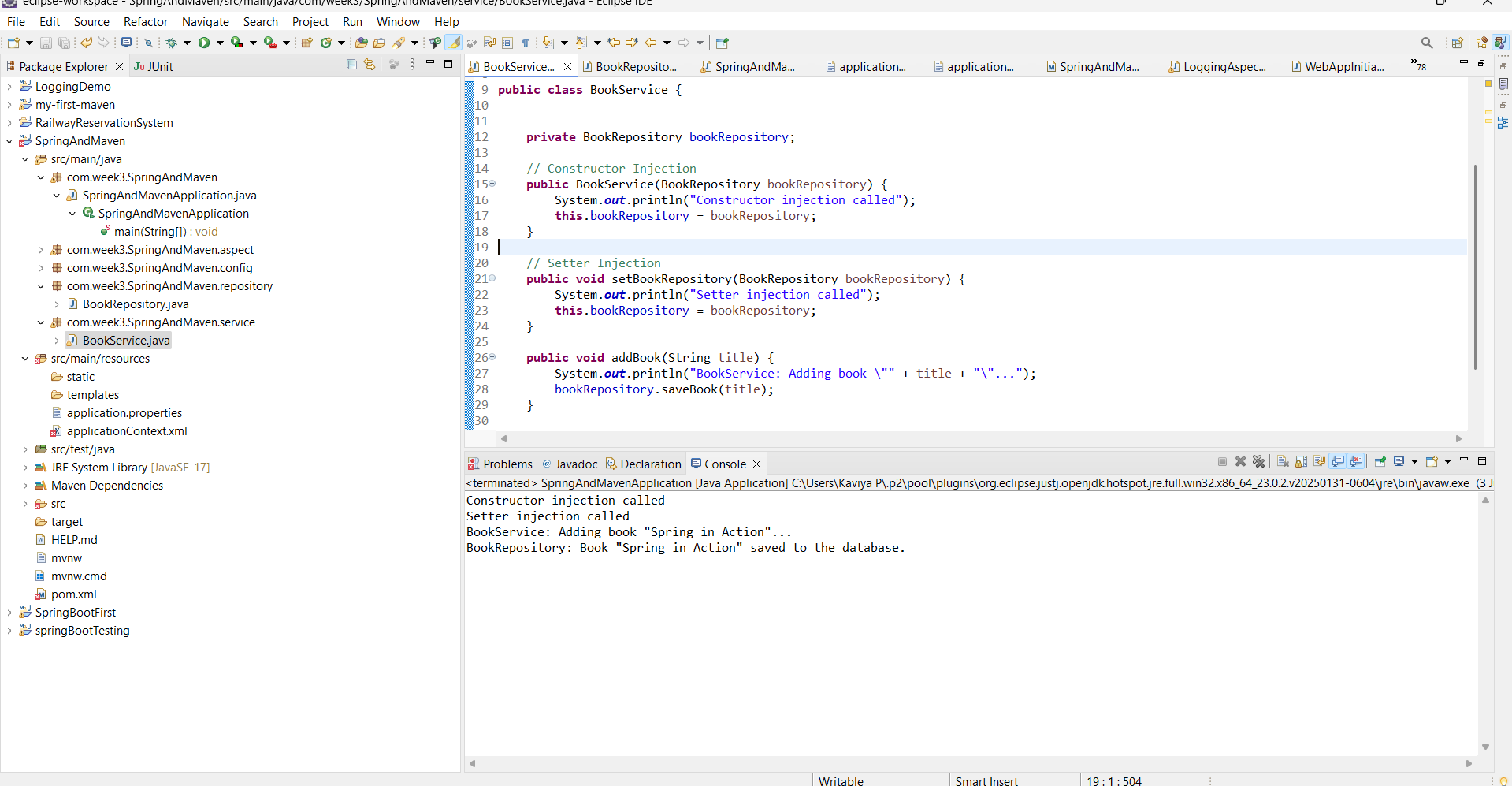
<!-- Setter Injection (will override constructor-set instance) -->

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

</bean>

</beans>

**OUTPUT**

****

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

**Scenario:**

The library management application requires basic AOP functionality to separate cross-cutting concerns like logging and transaction management.

**Steps:**

1. **Define an Aspect:**
   * Create a package **com.library.aspect** and add a class **LoggingAspect**.
2. **Create Advice Methods:**
   * Define advice methods in **LoggingAspect** for logging before and after method execution.
3. **Configure the Aspect:**
   * Update **applicationContext.xml** to register the aspect and enable **AspectJ** auto-proxying.
4. **Test the Aspect:**
   * Run the **LibraryManagementApplication** main class to verify the AOP functionality.

**SpringAndMavenApplication.java**

**package** com.week3.SpringAndMaven;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** com.week3.SpringAndMaven.service.BookService;

@SpringBootApplication

**public** **class** SpringAndMavenApplication {

**public** **static** **void** main(String[] args) {

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

BookService service = context.getBean(BookService.**class**);

service.addBook("Spring AOP Basics");

((ClassPathXmlApplicationContext) context).close();

}

}

**LoggingAspect.java**

**package** com.week3.SpringAndMaven.aspect;

**import** org.aspectj.lang.ProceedingJoinPoint;

**import** org.aspectj.lang.annotation.After;

**import** org.aspectj.lang.annotation.Around;

**import** org.aspectj.lang.annotation.Aspect;

**import** org.aspectj.lang.annotation.Before;

**import** org.springframework.stereotype.Component;

@Aspect

@Component

**public** **class** LoggingAspect {

@Before("execution(\* com.week3.SpringAndMaven.service.\*.\*(..))")

**public** **void** beforeMethod() {

System.***out***.println("[LOG] Before method execution.");

}

@After("execution(\* com.week3.SpringAndMaven.service.\*.\*(..))")

**public** **void** afterMethod() {

System.***out***.println("[LOG] After method execution.");

}

}

**BookRepository.java**

**package** com.week3.SpringAndMaven.repository;

**import** org.springframework.stereotype.Repository;

@Repository

**public** **class** BookRepository {

**public** **void** saveBook(String title) {

System.***out***.println("BookRepository: Book \"" + title + "\" saved to the database.");

}

}

**BookService.java**

**package** com.week3.SpringAndMaven.service;

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

**import** org.springframework.stereotype.Service;

**import** com.week3.SpringAndMaven.repository.BookRepository;

@Service

**public** **class** BookService {

**private** BookRepository bookRepository;

**public** BookService(BookRepository bookRepository) {

**this**.bookRepository = bookRepository;

}

**public** **void** addBook(String title) {

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

bookRepository.saveBook(title);

}

}

**Application.xml**

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

<beans xmlns="http://www.springframework.org/schema/beans"

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

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

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

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

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

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

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

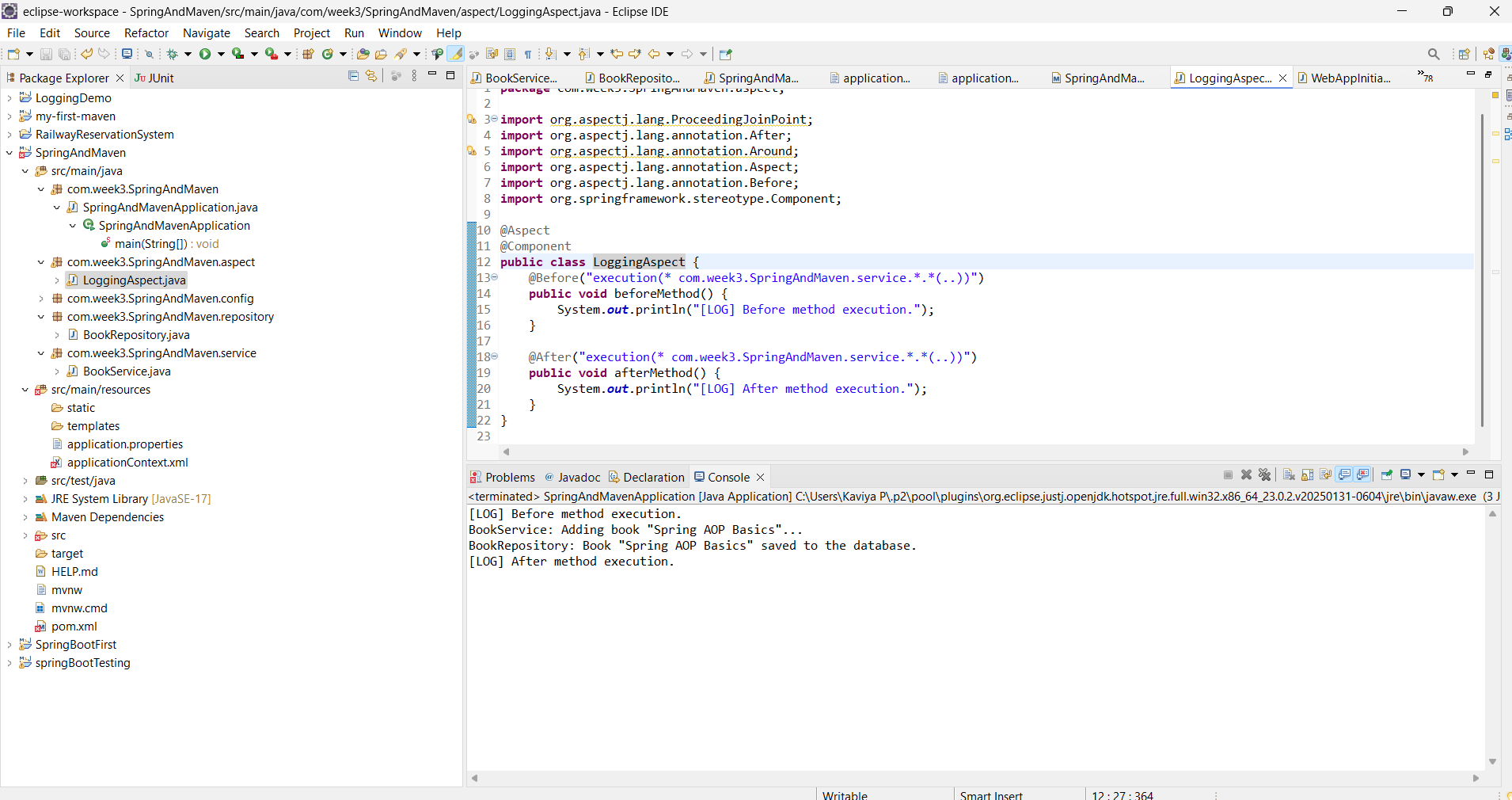
<!-- Enable annotation-based configuration -->

<context:component-scan base-package="com.week3.SpringAndMaven" />

<aop:aspectj-autoproxy />

</beans>

**OUTPUT**

****

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

**Scenario:**

You need to create a Spring Boot application for the library management system to simplify configuration and deployment.

**Steps:**

1. **Create a Spring Boot Project:**
   * Use **Spring Initializr** to create a new Spring Boot project named **LibraryManagement**.
2. **Add Dependencies:**
   * Include dependencies for **Spring Web, Spring Data JPA, and H2 Database**.
3. **Create Application Properties:**
   * Configure database connection properties in **application.properties**.
4. **Define Entities and Repositories:**
   * Create **Book** entity and **BookRepository** interface.
5. **Create a REST Controller:**
   * Create a **BookController** class to handle CRUD operations.
6. **Run the Application:**
   * Run the Spring Boot application and test the REST endpoints.

**SpringAndMavenApplication.java**

**package** com.week3.SpringAndMaven;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** com.week3.SpringAndMaven.service.BookService;

@SpringBootApplication

**public** **class** SpringAndMavenApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(SpringAndMavenApplication.**class**, args);

}

}

**BookController.java**

**package** com.week3.SpringAndMaven.controller;

**import** java.util.List;

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

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

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

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

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

**import** com.week3.SpringAndMaven.model.Book;

**import** com.week3.SpringAndMaven.repository.BookRepository;

@RestController

@RequestMapping("/books")

**public** **class** BookController {

@Autowired

**private** BookRepository bookRepository;

// Create

@PostMapping

**public** Book createBook(@RequestBody Book book) {

**return** bookRepository.save(book);

}

// Read all

@GetMapping

**public** List<Book> getAllBooks() {

**return** bookRepository.findAll();

}

// Read by ID

@GetMapping("/{id}")

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

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

}

// Update

@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**;

}

// Delete

@DeleteMapping("/{id}")

**public** String deleteBook(@PathVariable Long id) {

bookRepository.deleteById(id);

**return** "Book deleted with ID: " + id;

}

}

**Book.java**

**package** com.week3.SpringAndMaven.model;

**import** jakarta.persistence.\*;

@Entity

**public** **class** Book {

@Id

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

**private** Long id;

**private** String title;

**private** String author;

// Constructors

**public** Book() {}

**public** Book(String title, String author) {

**this**.title = title;

**this**.author = 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; }

}

**BookRepository.java**

**package** com.week3.SpringAndMaven.repository;

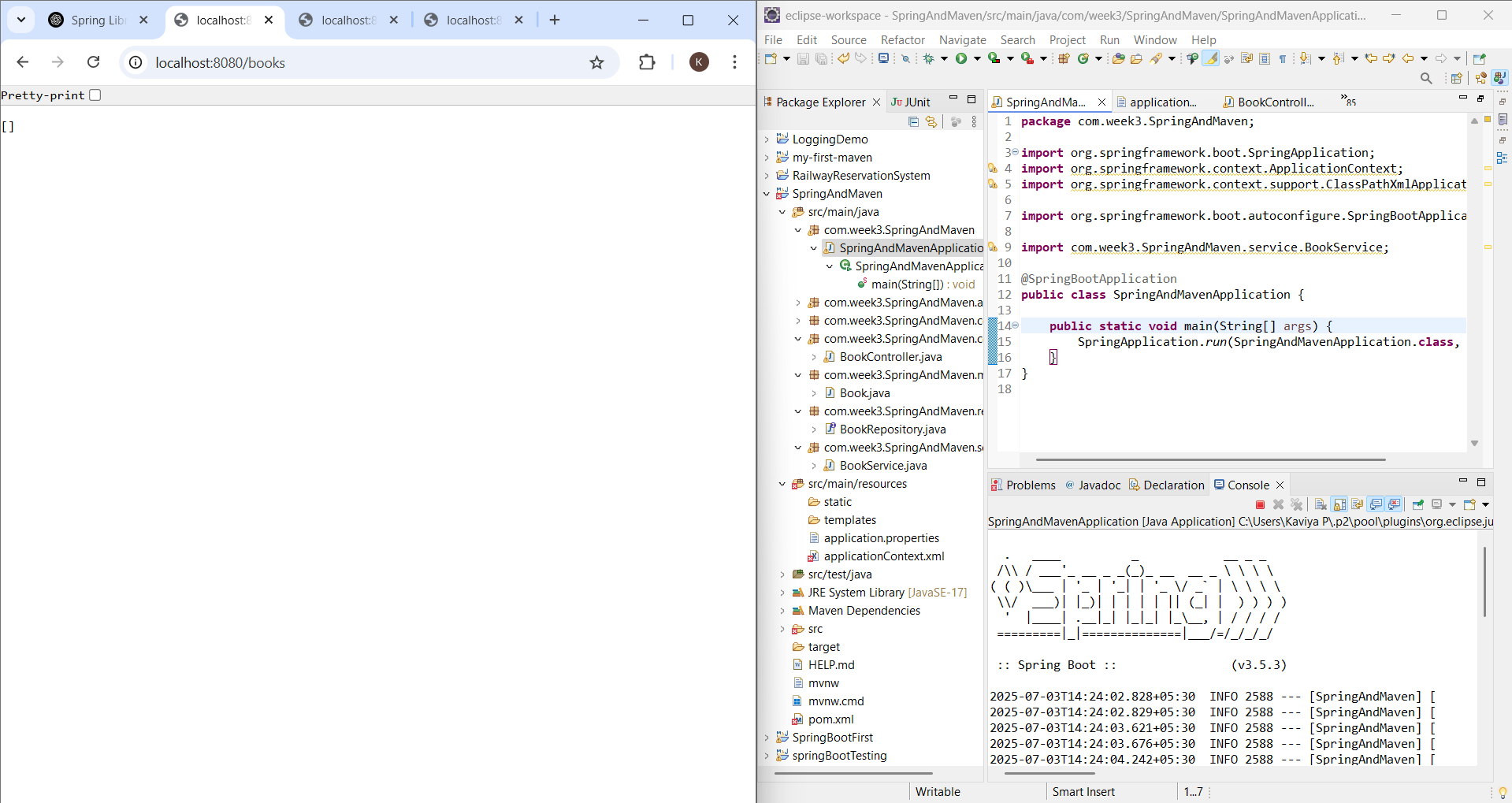
**import** org.springframework.data.jpa.repository.JpaRepository;

**import** com.week3.SpringAndMaven.model.Book;

**public** **interface** BookRepository **extends** JpaRepository<Book, Long> {

}

**OUTPUT**

****