WEEK3

Spring Core and Maven

Exercise 1: Configuring a Basic Spring Application

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(String bookName) {

bookRepository.saveBook(bookName);

}

}

BookRepository.java

**package** com.library.repository;

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

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

System.***out***.println("Book saved: " + bookName);

}

}

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

service.addBook("Effective Java");

}

}

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" />

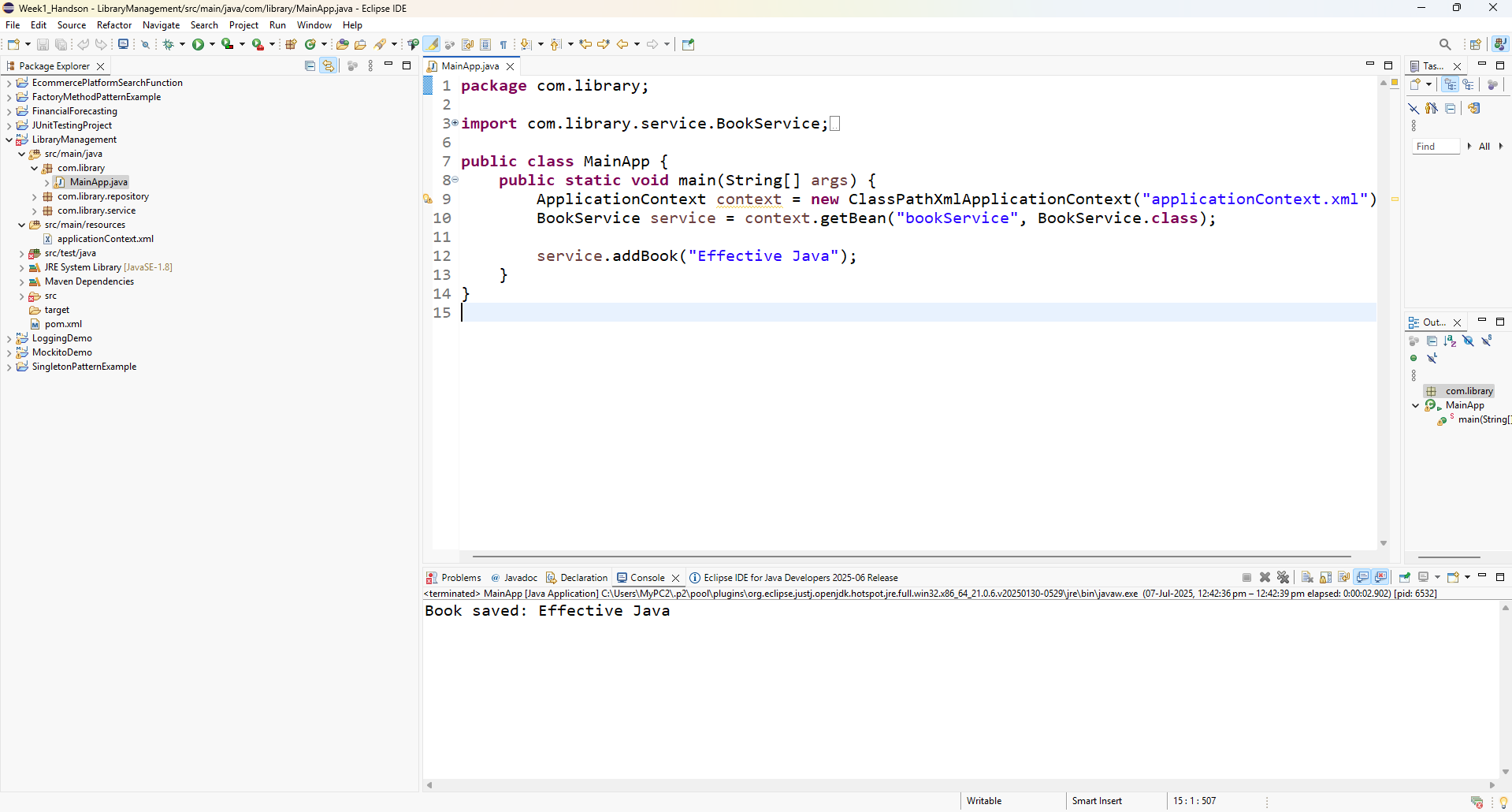
<bean id="bookService" class="com.library.service.BookService">

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

</bean>

</beans>

OUTPUT:



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

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

<!-- Bean for BookRepository -->

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

<!-- Bean for BookService with dependency injection -->

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

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

</bean>

</beans>

BookService.java

**package** com.library.service;

**import** com.library.repository.BookRepository;

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

**private** BookRepository bookRepository;

// Setter for Dependency Injection (DI)

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

**this**.bookRepository = bookRepository;

}

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

bookRepository.saveBook(bookName);

}

}

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**);

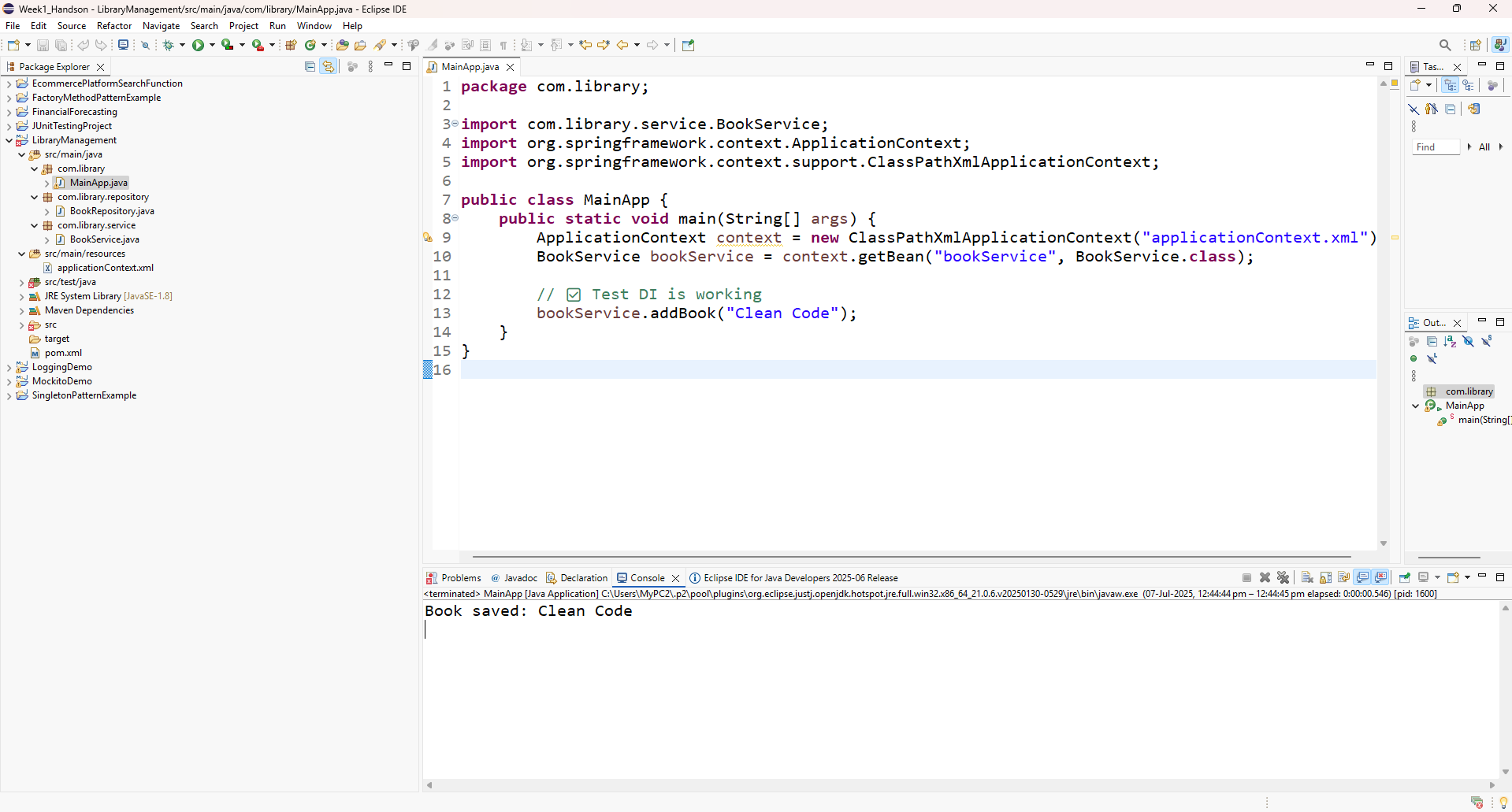
// ✅ Test DI is working

bookService.addBook("Clean Code");

}

}

OUTPUT:



Exercise 3: Implementing Logging with Spring AOP

src/main/java/com/library/aspect/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(); // Proceed with the actual method

long end = System.currentTimeMillis();

String methodName = joinPoint.getSignature().toShortString();

System.out.println("🔍 Method " + methodName + " executed in " + (end - start) + "ms");

return result;

}

}

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");

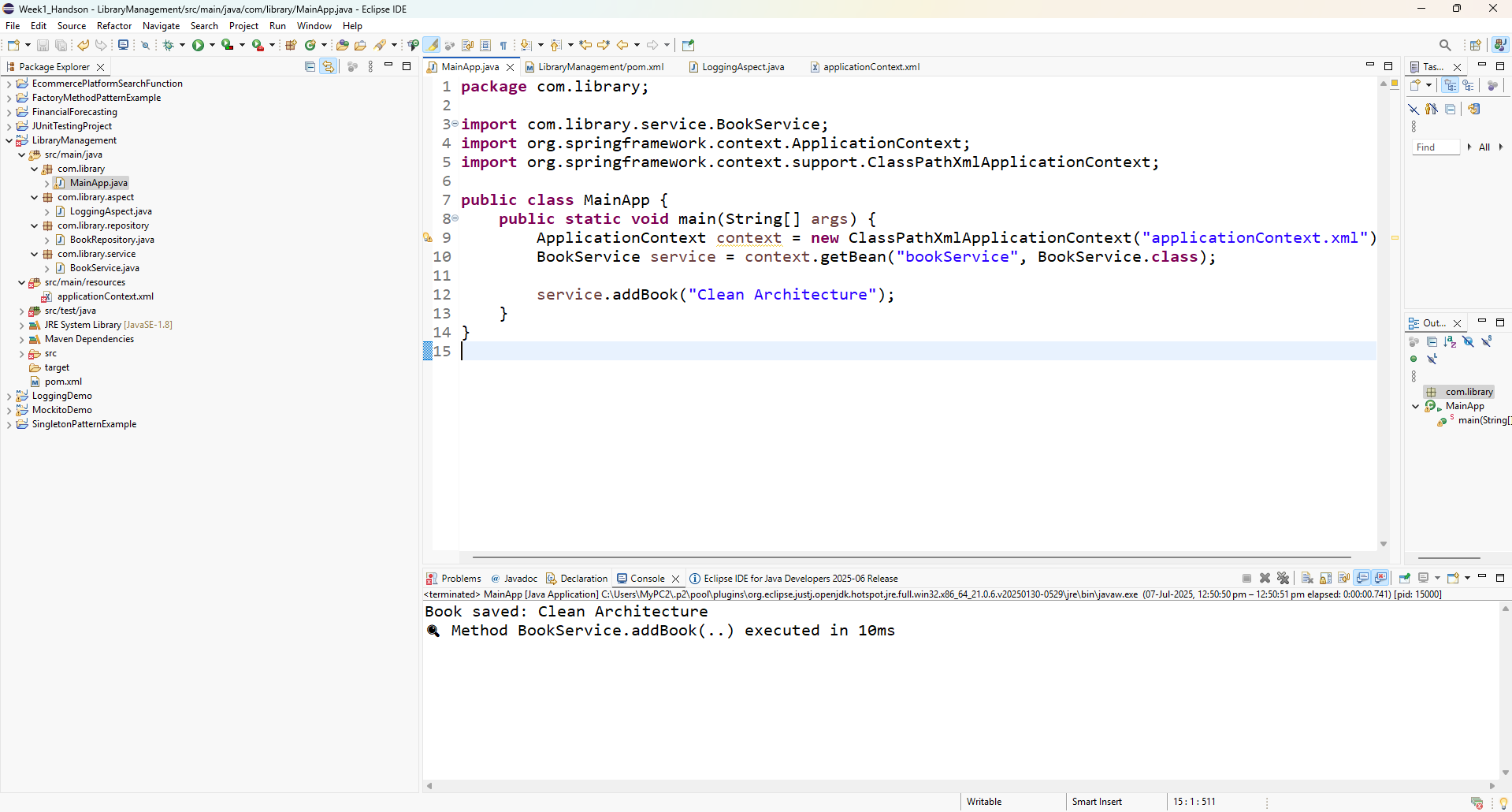
BookService service = context.getBean("bookService", BookService.**class**);

service.addBook("Clean Architecture");

}

}

OUTPUT:



Exercise 4: Creating and Configuring a Maven Project

In pom.xml adding this dependency

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.30</version>

</dependency>

Adding Maven Compiler Plugin

<build>

<plugins>

<plugin>

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

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

<version>3.10.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

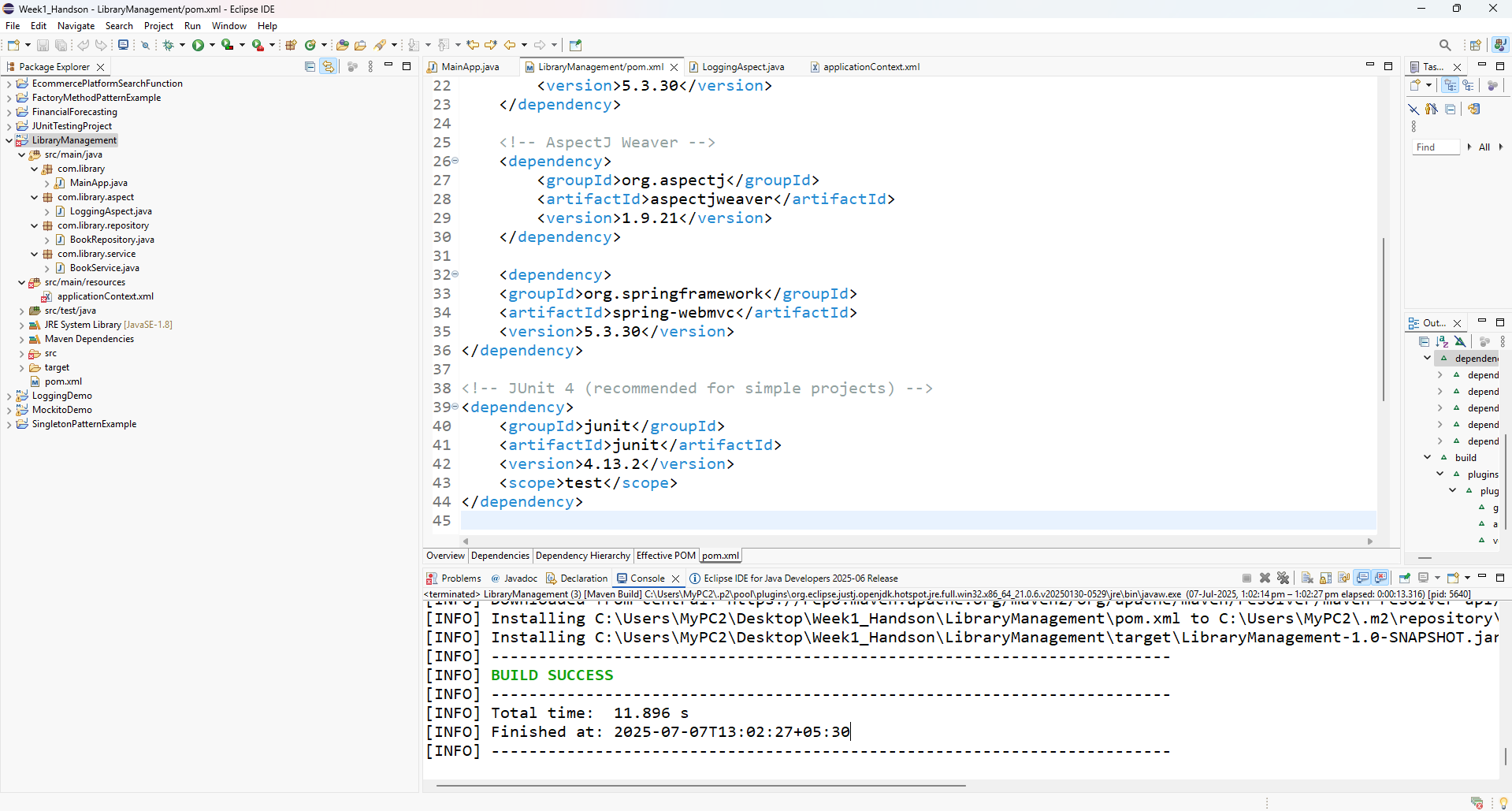
</configuration>

</plugin>

</plugins>

</build>

OUTPUT:



Exercise 5: Configuring the Spring IoC Container

src/main/resources/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">

<!-- Repository Bean -->

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

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

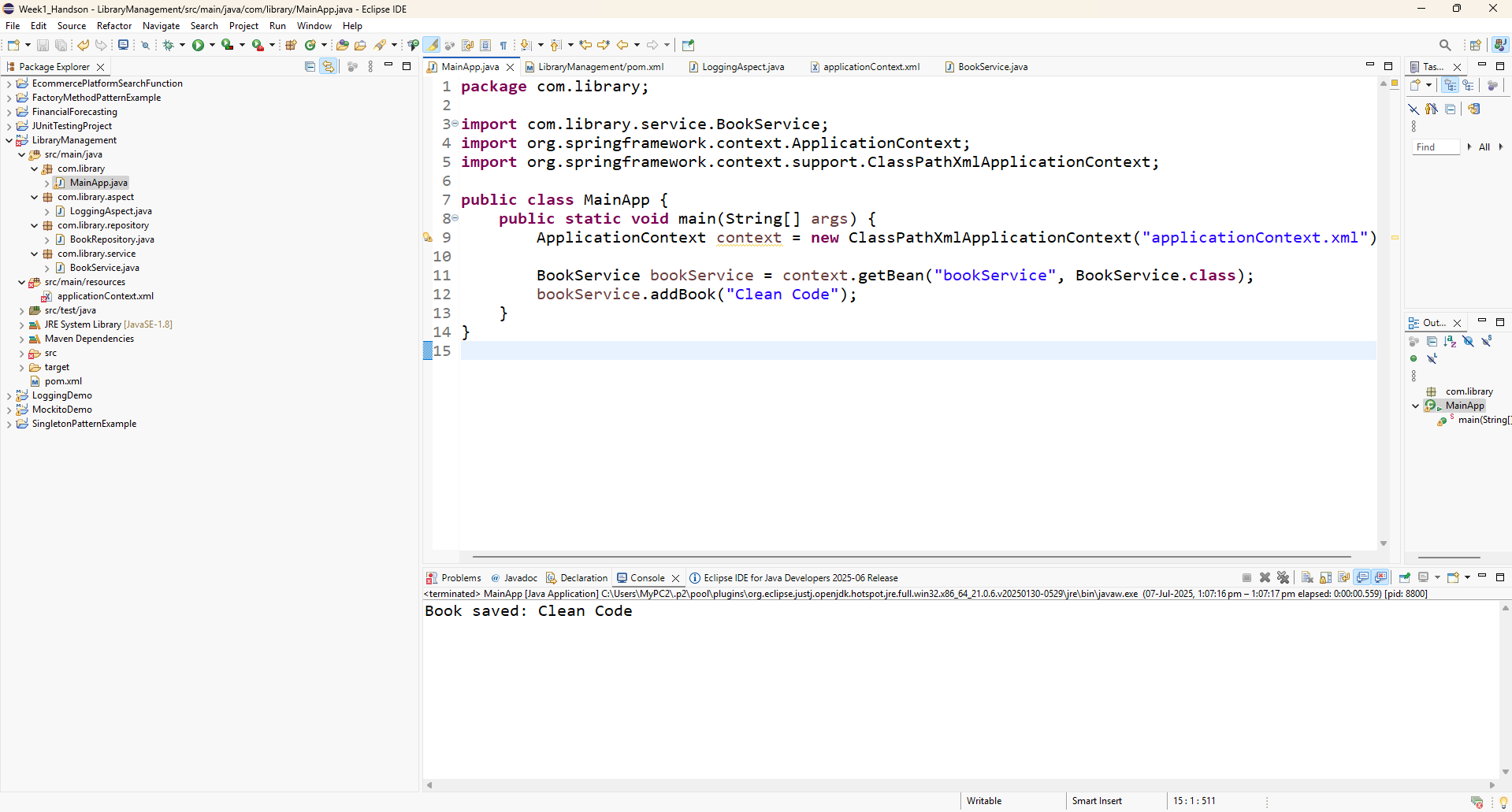
<bean id="bookService" class="com.library.service.BookService">

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

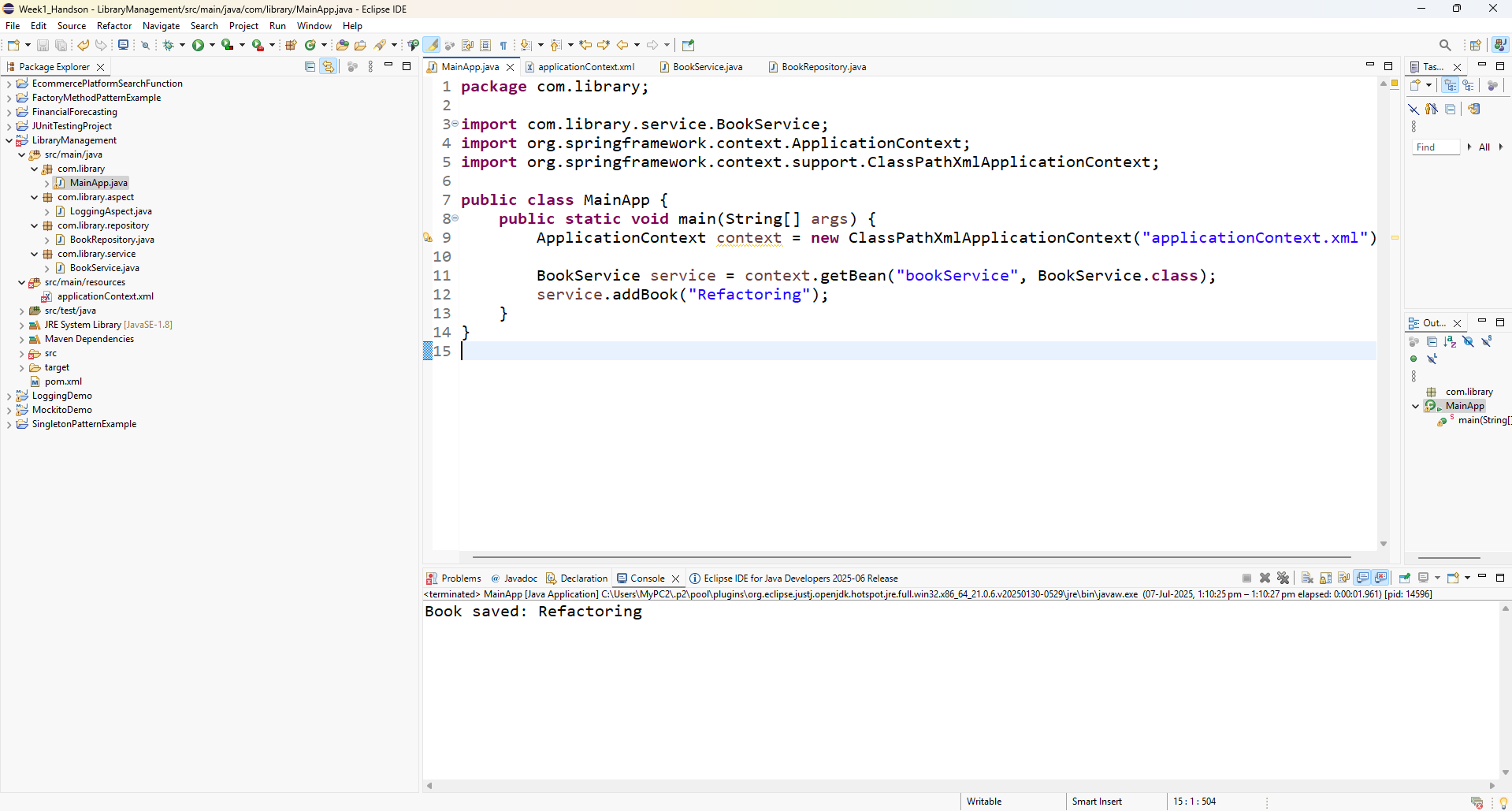
</bean>

</beans>

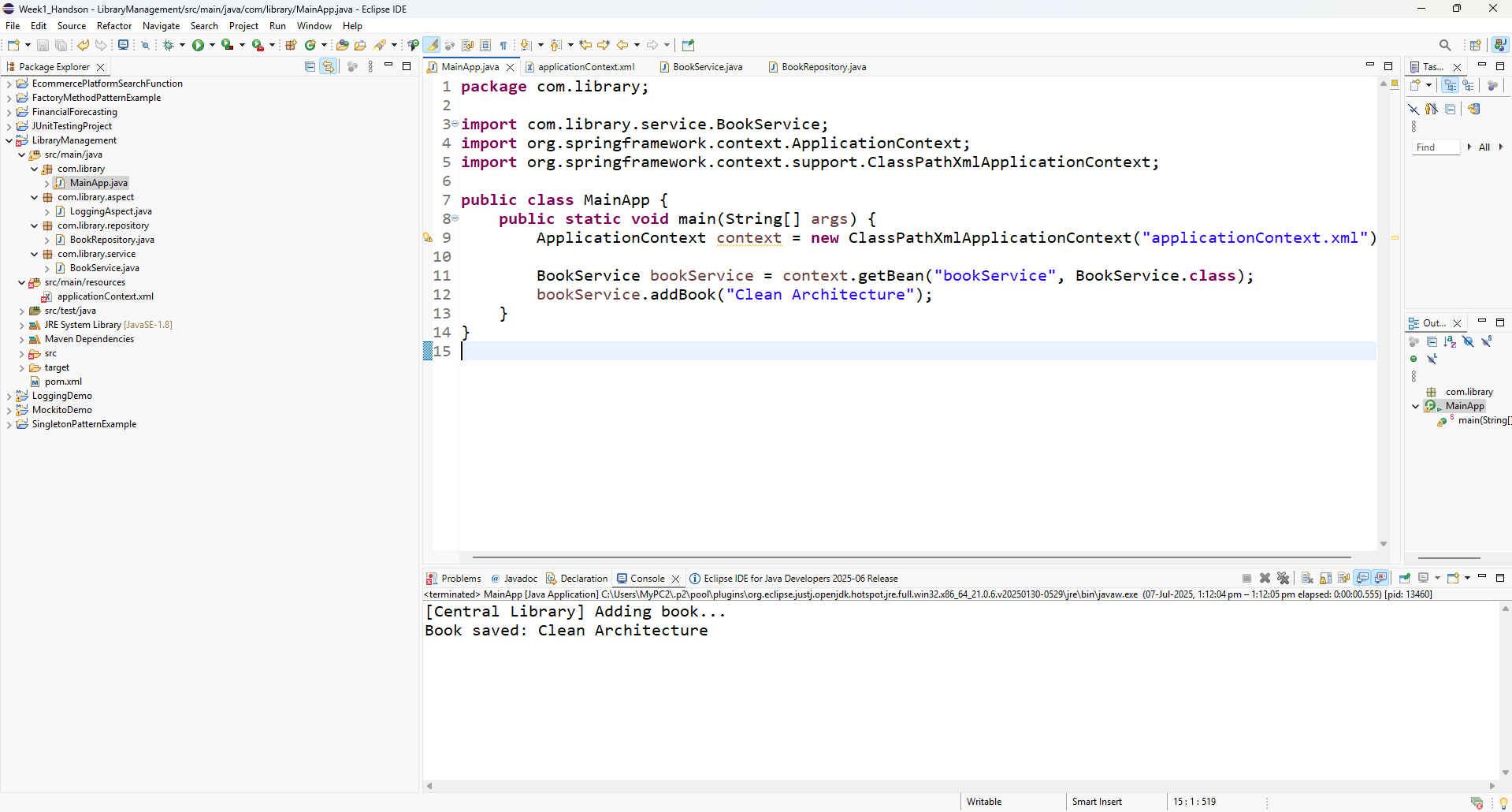
OUTPUT:



Exercise 6: Configuring Beans with Annotations



Exercise 7: Implementing Constructor and Setter Injection



Exercise 8: Implementing Basic AOP with Spring

src/main/java/com/library/aspect/LoggingAspect.java

package com.library.aspect;

import org.aspectj.lang.JoinPoint;

public class LoggingAspect {

// ✅ Before advice

public void logBefore(JoinPoint joinPoint) {

System.out.println("➡️ [LOG] Before method: " + joinPoint.getSignature().getName());

}

// ✅ After advice

public void logAfter(JoinPoint joinPoint) {

System.out.println("⬅️ [LOG] After method: " + joinPoint.getSignature().getName());

}

}

src/main/resources/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: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

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

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

<!-- ✅ AOP support -->

<aop:aspectj-autoproxy />

<!-- ✅ Register the aspect bean -->

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

<!-- ✅ AOP Configuration -->

<aop:config>

<aop:aspect ref="loggingAspect">

<!-- Before advice on all service methods -->

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

<!-- After advice on all service methods -->

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

</aop:aspect>

</aop:config>

<!-- ✅ Repository -->

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

<!-- ✅ Service with constructor injection -->

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

<constructor-arg ref="bookRepository" />

<property name="libraryName" value="Central Library" />

</bean>

</beans>

src/main/java/com/library/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 service = context.getBean("bookService", BookService.class);

service.addBook("Design Patterns");

}

}

