**WEEK 3**

**Spring Core and Maven Hands-On Exercises**

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

**LibraryManagement/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-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

</dependencies>

</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.library.repository.BookRepository"/>

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

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

</bean>

</beans>

**BookService**

**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(String title) {

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

bookRepository.saveBook(title);

}

}

**BookRepository**

**package** com.library.repository;

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

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

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

}

}

**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) {

// Load Spring container from XML configuration

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

// Retrieve the BookService bean

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

// Call the method to test dependency injection

bookService.addBook("Pride and Prejudice");

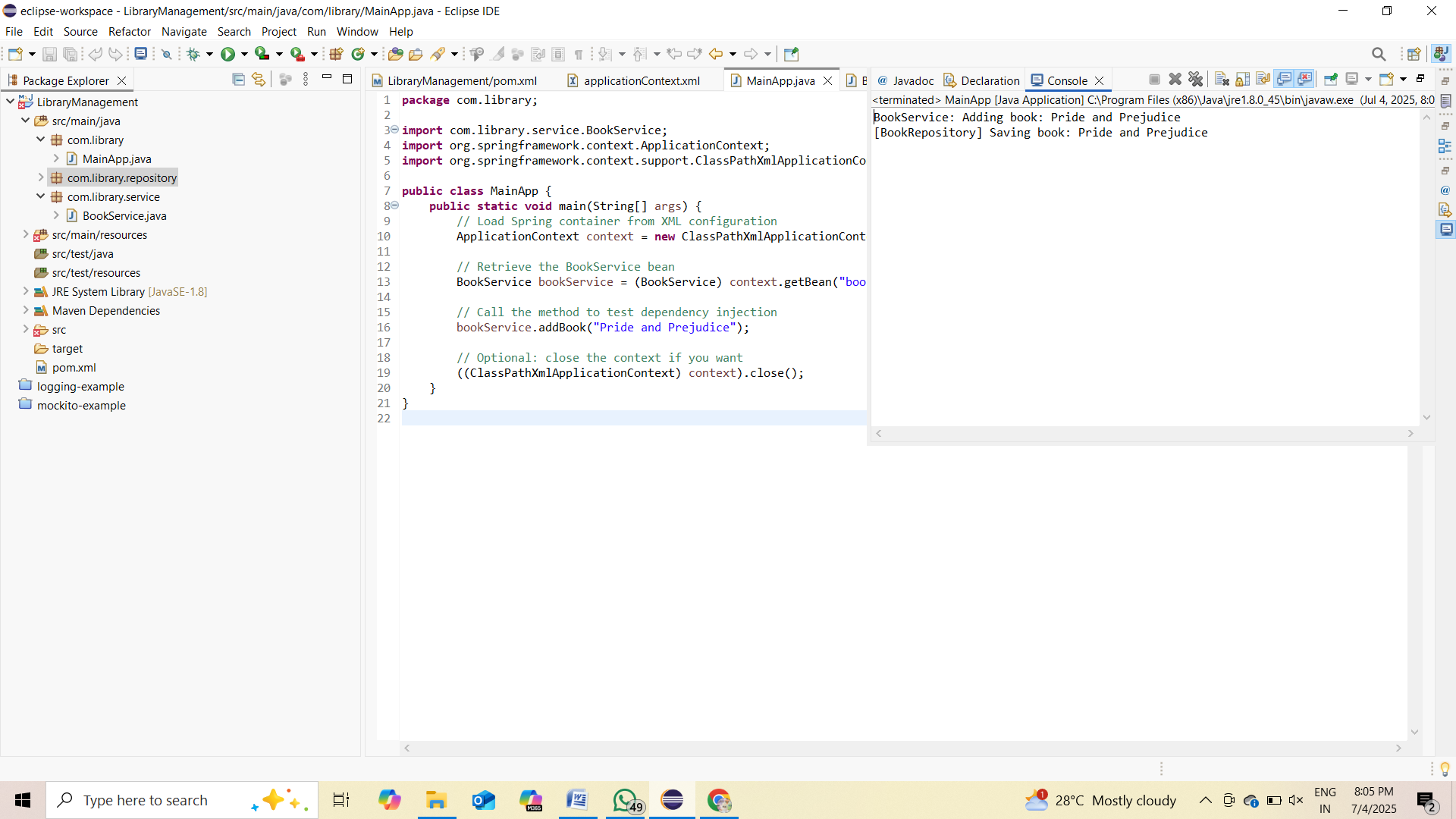
// Optional: close the context if you want

((ClassPathXmlApplicationContext) context).close();

}

}

**OUTPUT**

****

**Exercise 2: Implementing Dependency Injection**

**LibraryManagement /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-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

</dependencies>

</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">

<!-- Define BookRepository bean -->

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

<!-- Define BookService bean with BookRepository injected -->

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

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

</bean>

</beans>

**BookService**

**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(String title) {

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

bookRepository.saveBook(title);

}

}

**BookRepository**

**package** com.library.repository;

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

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

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

}

}

**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) {

// Load Spring context from XML

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

// Get the BookService bean (fully wired)

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

// Call method to test injection

bookService.addBook("Pride and Prejudice");

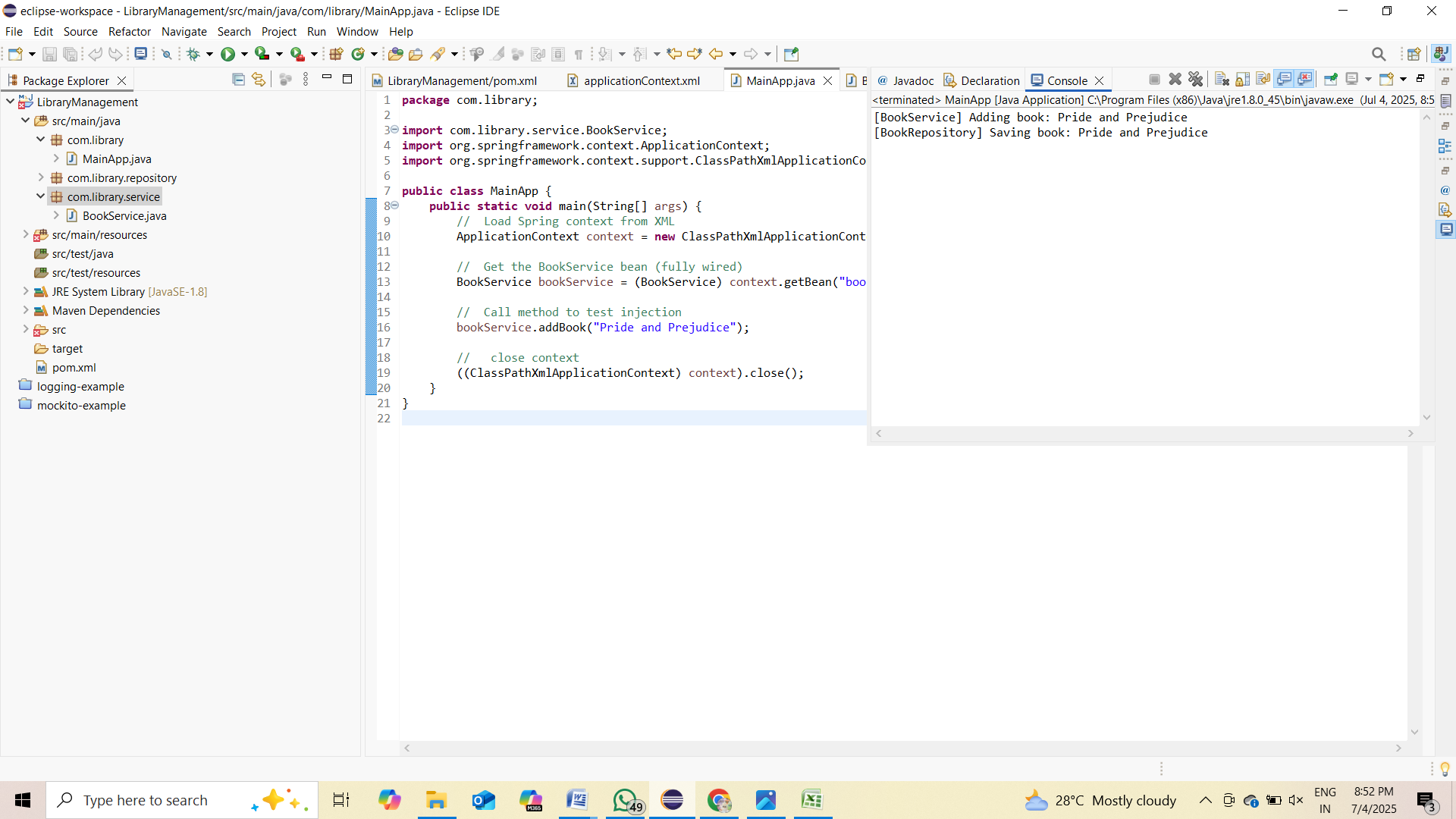
// close context

((ClassPathXmlApplicationContext) context).close();

}

}

**OUTPUT**



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

**LibraryManagement /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-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.34</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.34</version>

</dependency>

</dependencies>

<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>

</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">

<!-- Define BookRepository Bean -->

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

<!-- Define BookService Bean and inject BookRepository -->

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

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

</bean>

</beans>

**BookService**

**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(String bookName) {

System.***out***.println("[BookService] Adding book: " + bookName);

bookRepository.saveBook(bookName);

}

}

**BookRepository**

**package** com.library.repository;

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

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

System.***out***.println("[BookRepository] Saving book: " + 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 = (BookService) context.getBean("bookService");

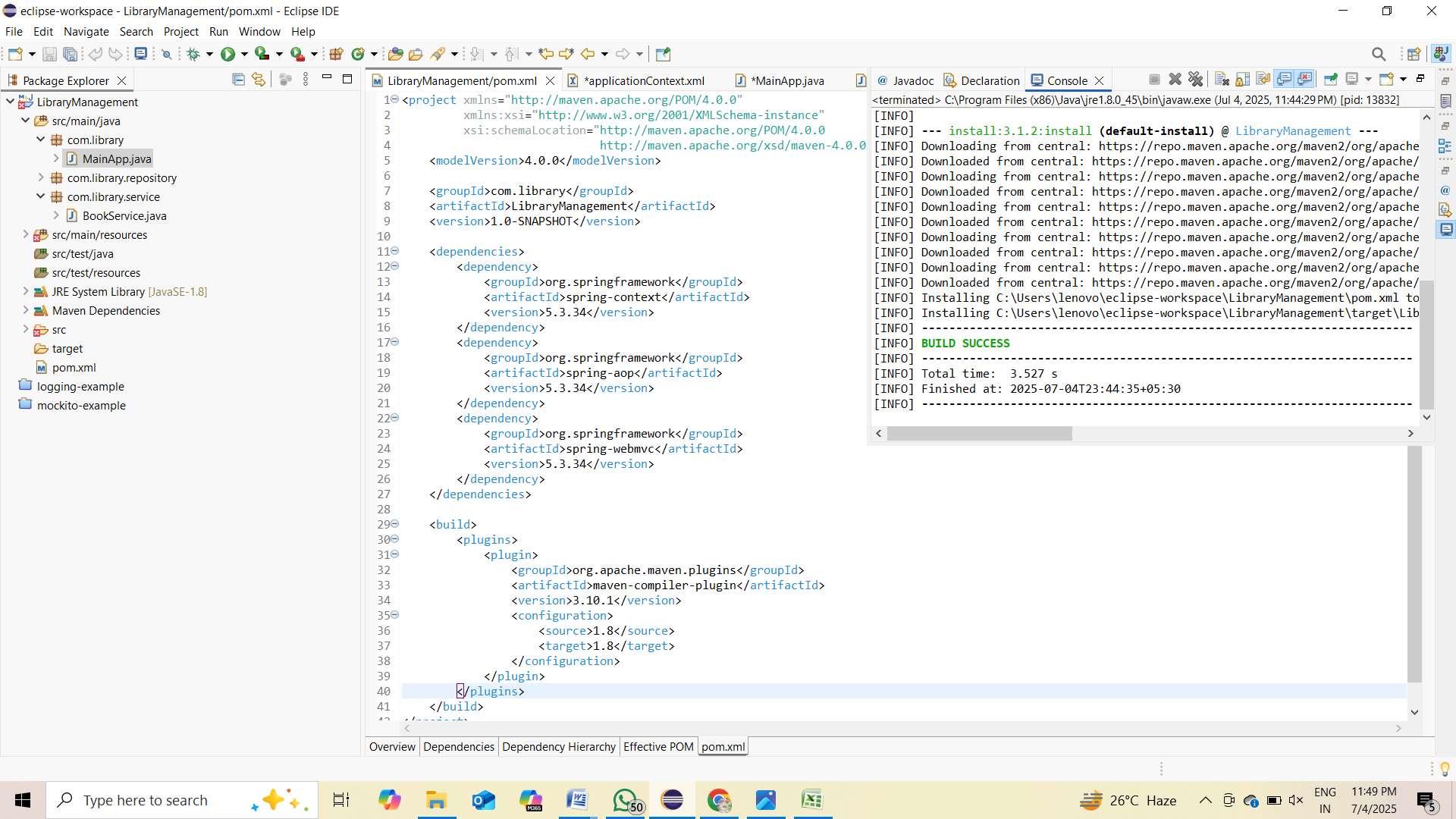
bookService.addBook("Pride and Prejudice");

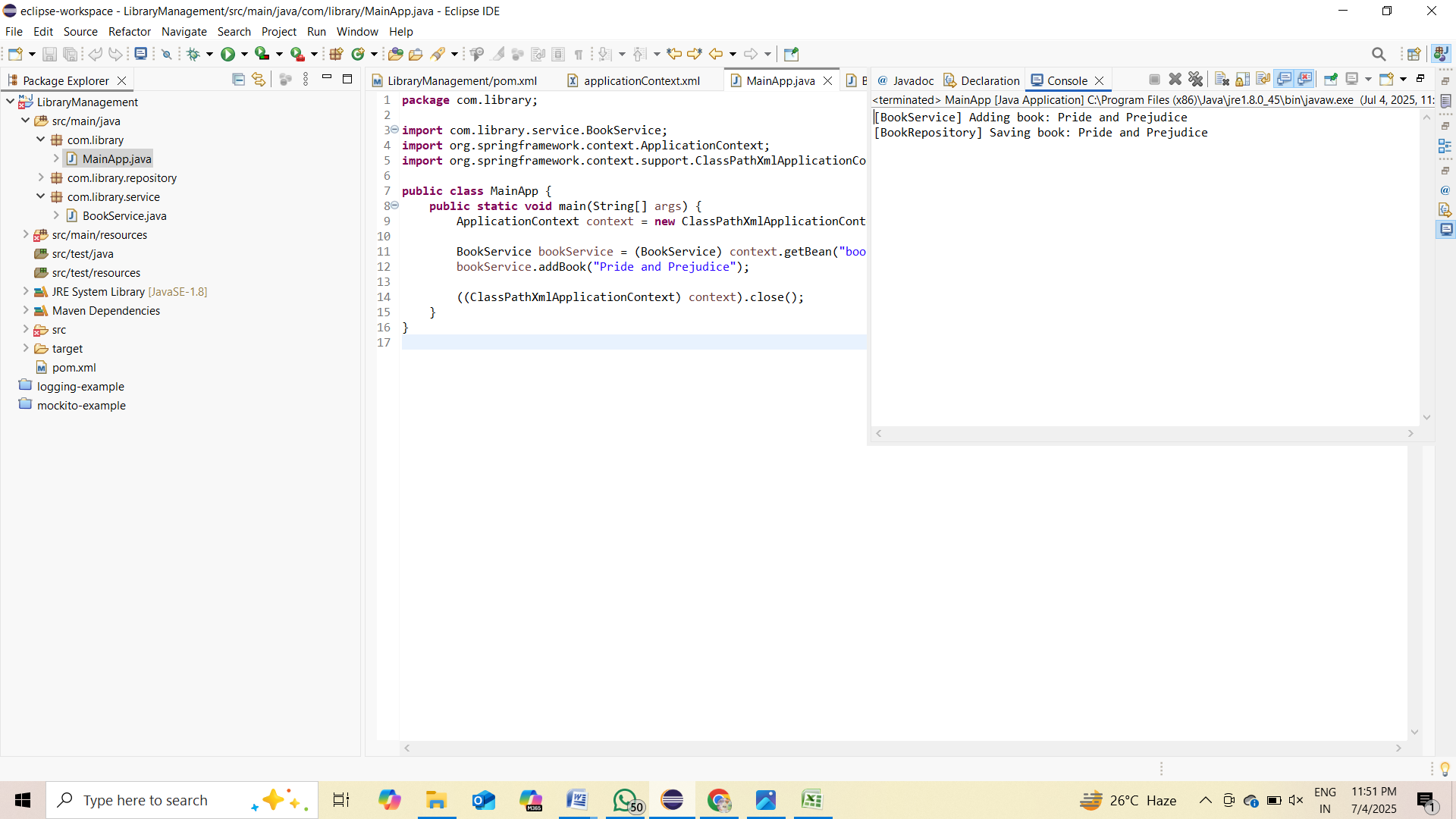
((ClassPathXmlApplicationContext) context).close();

}

}

**OUTPUT**

****

****

**ADDITIONAL HANDS-ON EXERCISES**

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

**LibraryManagement /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-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.34</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.34</version>

</dependency>

</dependencies>

<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>

</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">

<!-- Define BookRepository Bean -->

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

<!-- Define BookService Bean with DI -->

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

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

</bean>

</beans>

**BookService**

**package** com.library.service;

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

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

**private** BookRepository bookRepository;

// Setter for DI

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

**this**.bookRepository = bookRepository;

}

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

System.***out***.println("[BookService] Adding book: " + bookName);

bookRepository.saveBook(bookName);

}

}

**BookRepository**

**package** com.library.repository;

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

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

System.***out***.println("[BookRepository] Saving book: " + 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) {

// Load Spring IoC container

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

// Get BookService bean

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

// Test

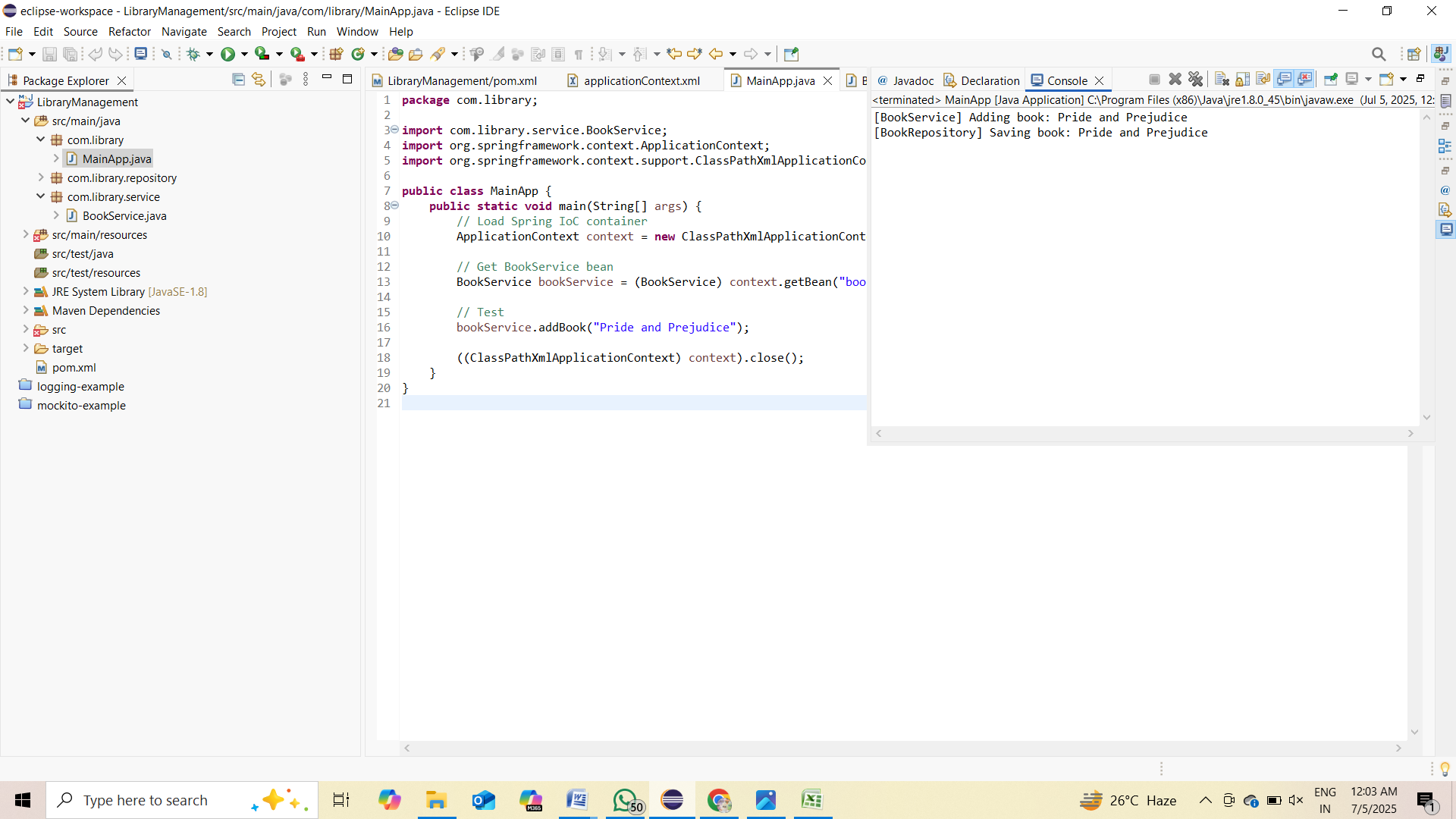
bookService.addBook("Pride and Prejudice");

((ClassPathXmlApplicationContext) context).close();

}

}

**OUTPUT**

****

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

**LibraryManagement /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-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.34</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.34</version>

</dependency>

</dependencies>

<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>

</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">

<!-- Define BookRepository Bean -->

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

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

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

<!-- Constructor Injection -->

<constructor-arg value="LibraryManagementService"/>

<!-- Setter Injection -->

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

</bean>

</beans>

**BookService**

**package** com.library.service;

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

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

**private** String serviceName;

**private** BookRepository bookRepository;

// Constructor for Constructor Injection

**public** BookService(String serviceName) {

**this**.serviceName = serviceName;

}

// Setter for Setter Injection

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

**this**.bookRepository = bookRepository;

}

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

System.***out***.println("[" + serviceName + "] Adding book: " + bookName);

bookRepository.saveBook(bookName);

}

}

**BookRepository**

**package** com.library.repository;

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

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

System.***out***.println("[BookRepository] Saving book: " + 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) {

// Load Spring IoC container

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

// Get BookService bean

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

// Test

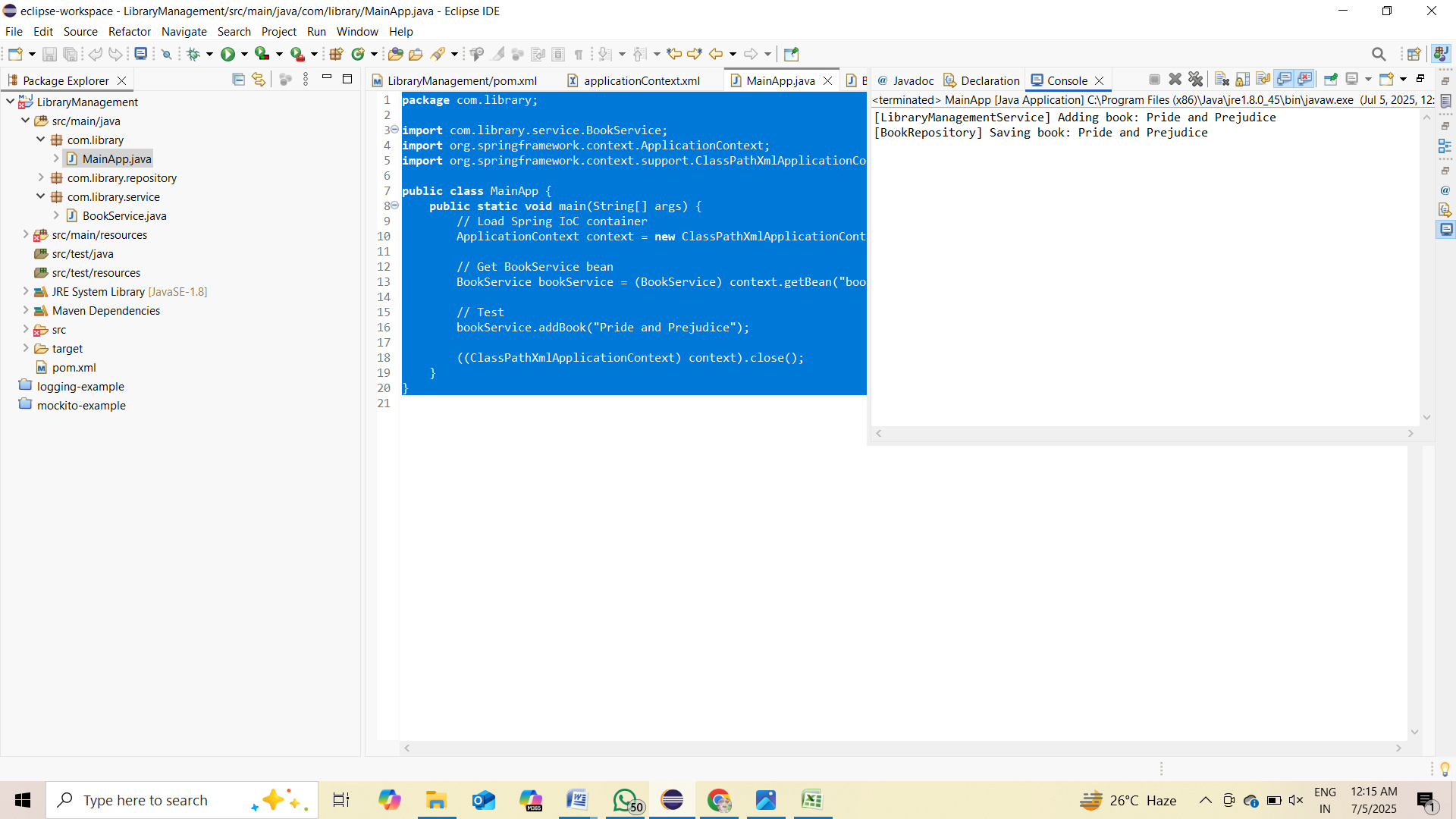
bookService.addBook("Pride and Prejudice");

((ClassPathXmlApplicationContext) context).close();

}

}

**OUTPUT**

****

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

**LibraryManagement/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>

<groupId>com.example</groupId>

<artifactId>LibraryManagement</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>LibraryManagement</name>

<description>Library Management Spring Boot Application</description>

<parent>

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

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

<version>3.2.5</version>

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

</parent>

<properties>

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

</properties>

<dependencies>

<!-- Spring Web -->

<dependency>

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

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

</dependency>

<!-- Spring Data JPA -->

<dependency>

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

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<!-- H2 Database -->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**src/main/resources/application.properties**

# H2 Database config

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.h2.console.enabled=true

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=update

**src/main/java/com/example/librarymanagement/model/Book.java**

**package** com.example.librarymanagement.model;

**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;

**private** String isbn;

// Default constructor

**public** Book() {}

// Parameterized constructor

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

**this**.title = title;

**this**.author = author;

**this**.isbn = isbn;

}

// Getters and 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;

}

**public** String getIsbn() {

**return** isbn;

}

**public** **void** setIsbn(String isbn) {

**this**.isbn = isbn;

}

}

**src/main/java/com/example/librarymanagement/repository/BookRepository.java**

**package** com.example.librarymanagement.repository;

**import** com.example.librarymanagement.model.Book;

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

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

}

**src/main/java/com/example/librarymanagement/controller/BookController.java**

**package** com.example.librarymanagement.controller;

**import** com.example.librarymanagement.model.Book;

**import** com.example.librarymanagement.repository.BookRepository;

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

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

**import** java.util.List;

**import** java.util.Optional;

@RestController

@RequestMapping("/api/books")

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

@Autowired

**private** BookRepository bookRepository;

@GetMapping

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

**return** bookRepository.findAll();

}

@GetMapping("/{id}")

**public** Optional<Book> getBookById(@PathVariable Long id) {

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

}

@PostMapping

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

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

}

@PutMapping("/{id}")

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

**return** bookRepository.findById(id)

.map(book -> {

book.setTitle(updatedBook.getTitle());

book.setAuthor(updatedBook.getAuthor());

book.setIsbn(updatedBook.getIsbn());

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

})

.orElseGet(() -> {

updatedBook.setId(id);

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

});

}

@DeleteMapping("/{id}")

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

bookRepository.deleteById(id);

}

}

**LibraryManagementApplication.java**

**package** com.example.librarymanagement;

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

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

@SpringBootApplication

**public** **class** LibraryManagementApplication {

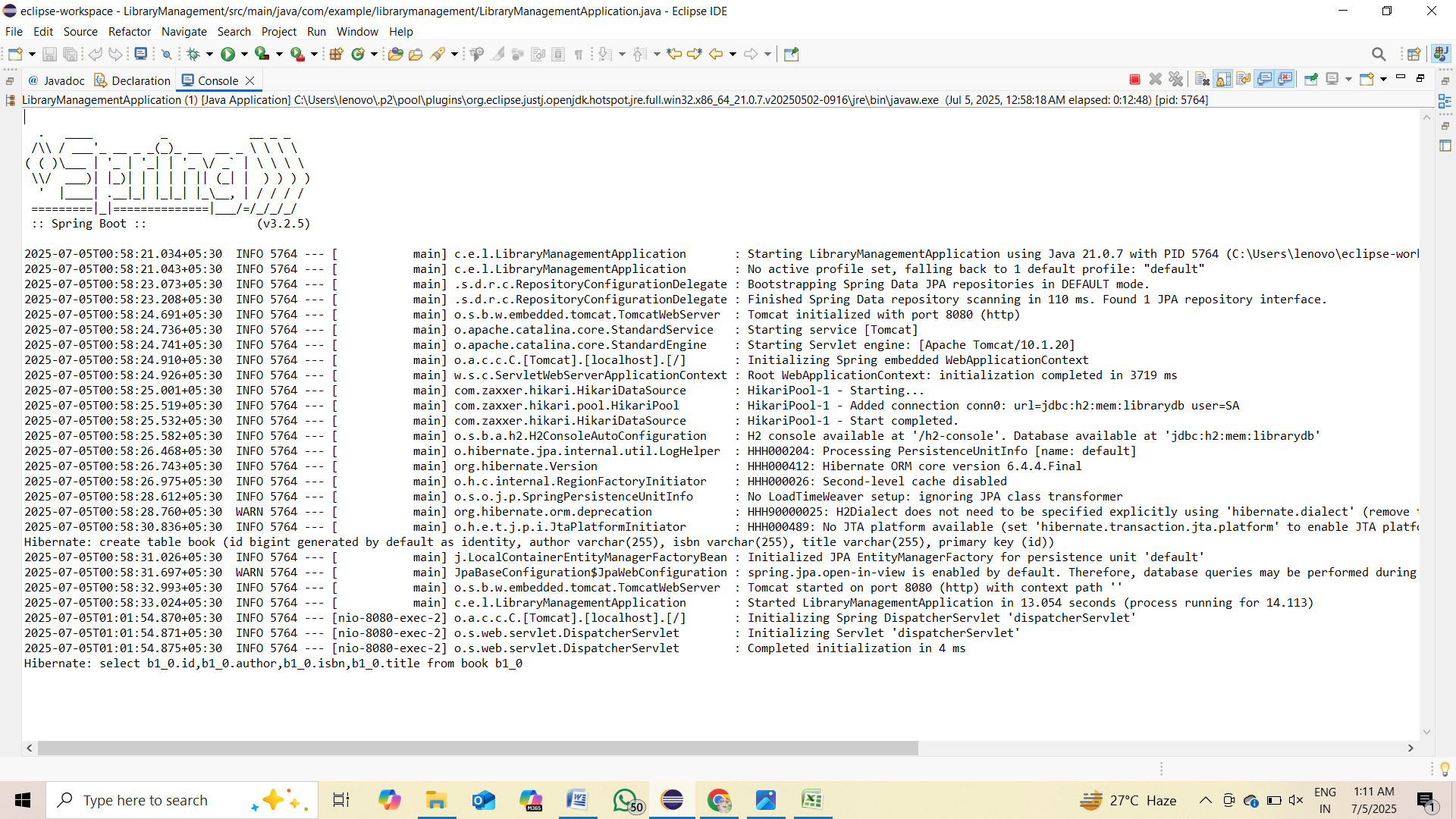
**public** **static** **void** main(String[] args) {

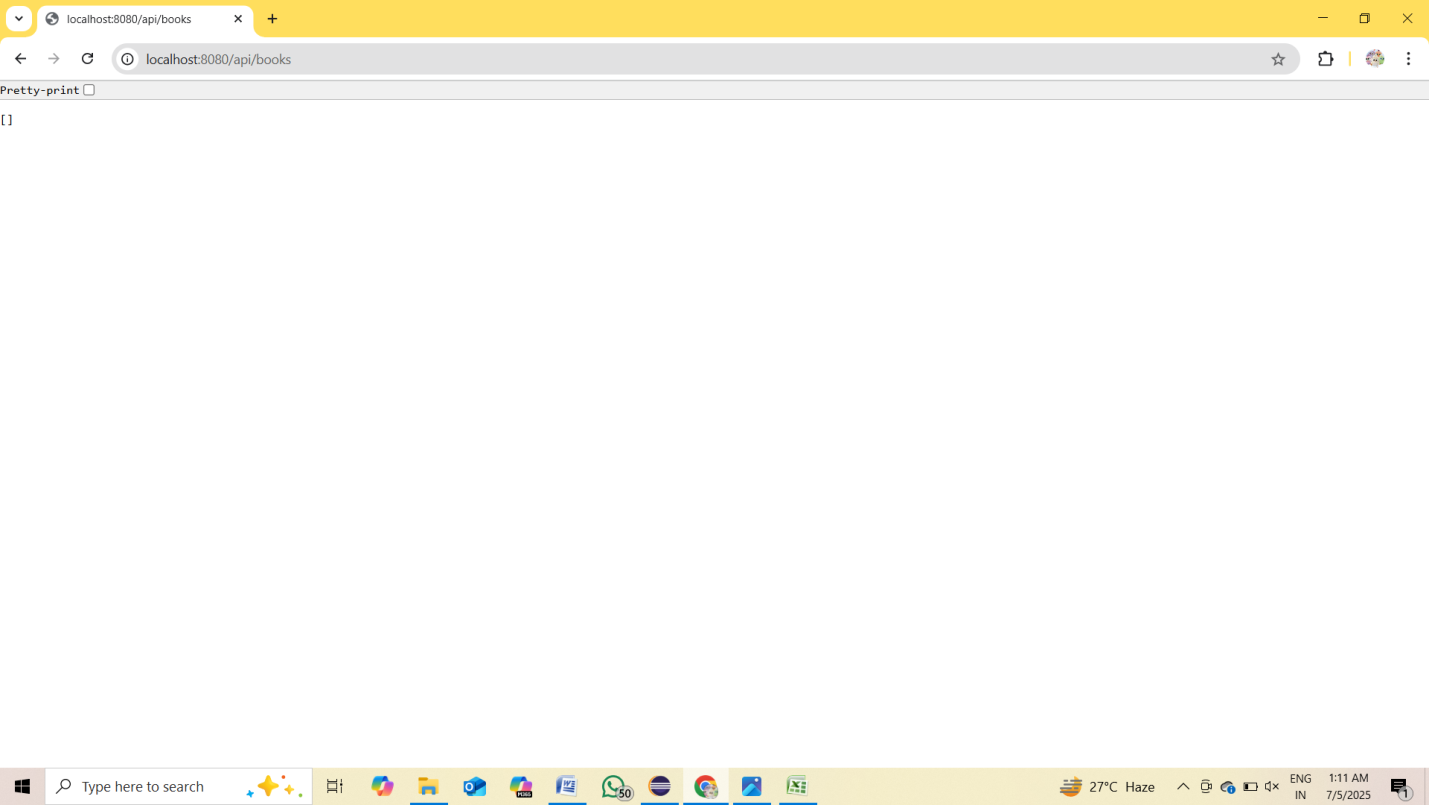
SpringApplication.*run*(LibraryManagementApplication.**class**, args);

}

}

**OUTPUT**

****

****