**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.

**Solution:**

**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>

**BookRepository Class:**

**package** com.library.repository;

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

**public** **void** getAllBooks() {

System.***out***.println("📚 Getting all books from the repository...");

}

}

**Mainclass.java:**

**package** com.library;

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

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

**import** com.library.service.BookService;

**public** **class** MainApp {

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

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

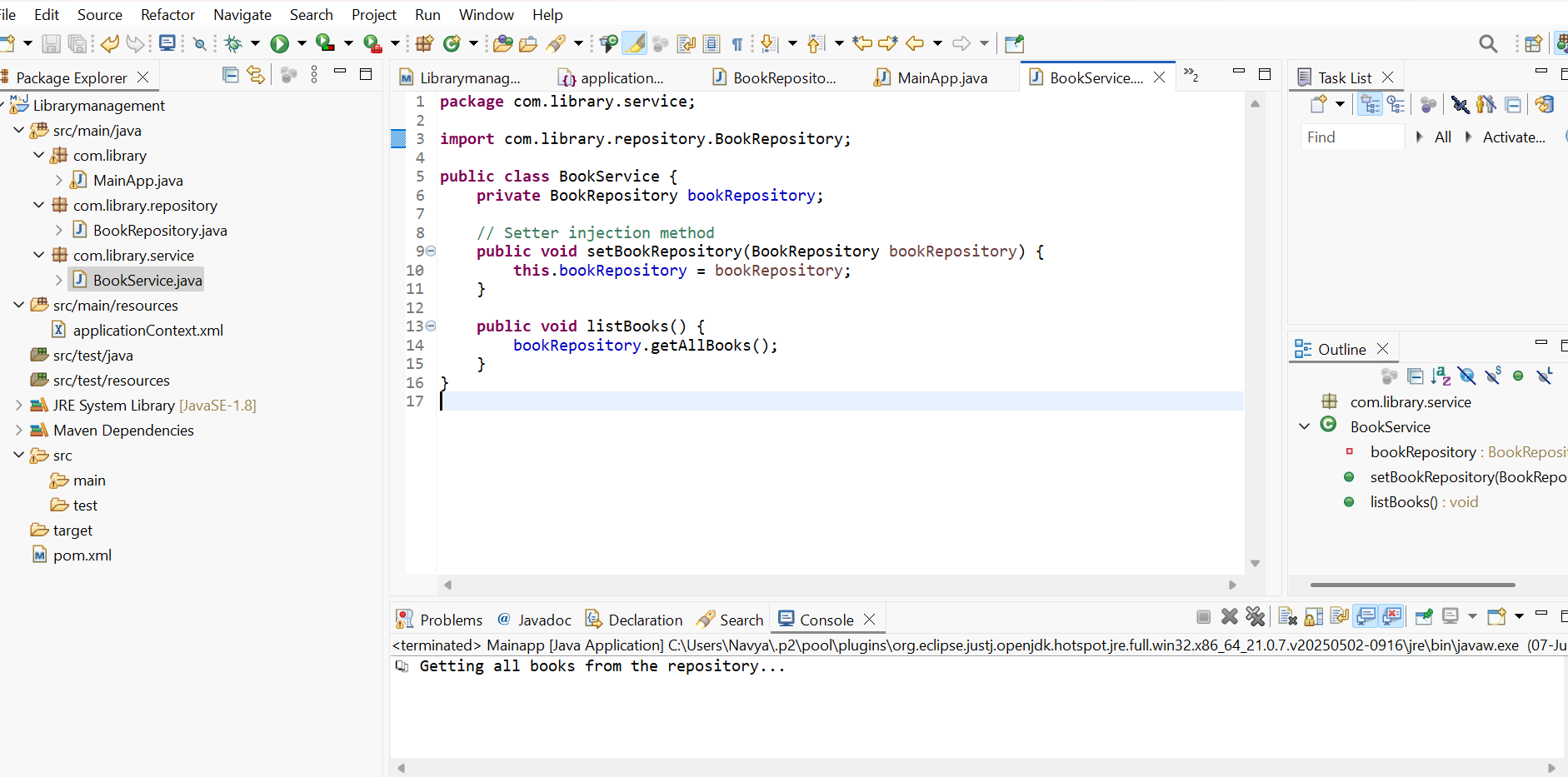
BookService service = (BookService) context.getBean("bookService");

service.listBooks();

}

}

OUTPUT:



**Exercise 2: Implementing Dependency Injection**

**Scenario:**

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

**SOLUTION:**

**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>

**BookService.java:**

**package** com.library.service;

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

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

**private** BookRepository bookRepository;

// Setter injection method

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

**this**.bookRepository = bookRepository;

}

**public** **void** listBooks() {

bookRepository.getAllBooks();

}

}

**Bookrepository.java:**

**package** com.library.repository;

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

**public** **void** getAllBooks() {

System.***out***.println("📚 Getting all books from the repository...");

}

}

**MainApp.java:**

**package** com.library;

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

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

**import** com.library.service.BookService;

**public** **class** MainApp {

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

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

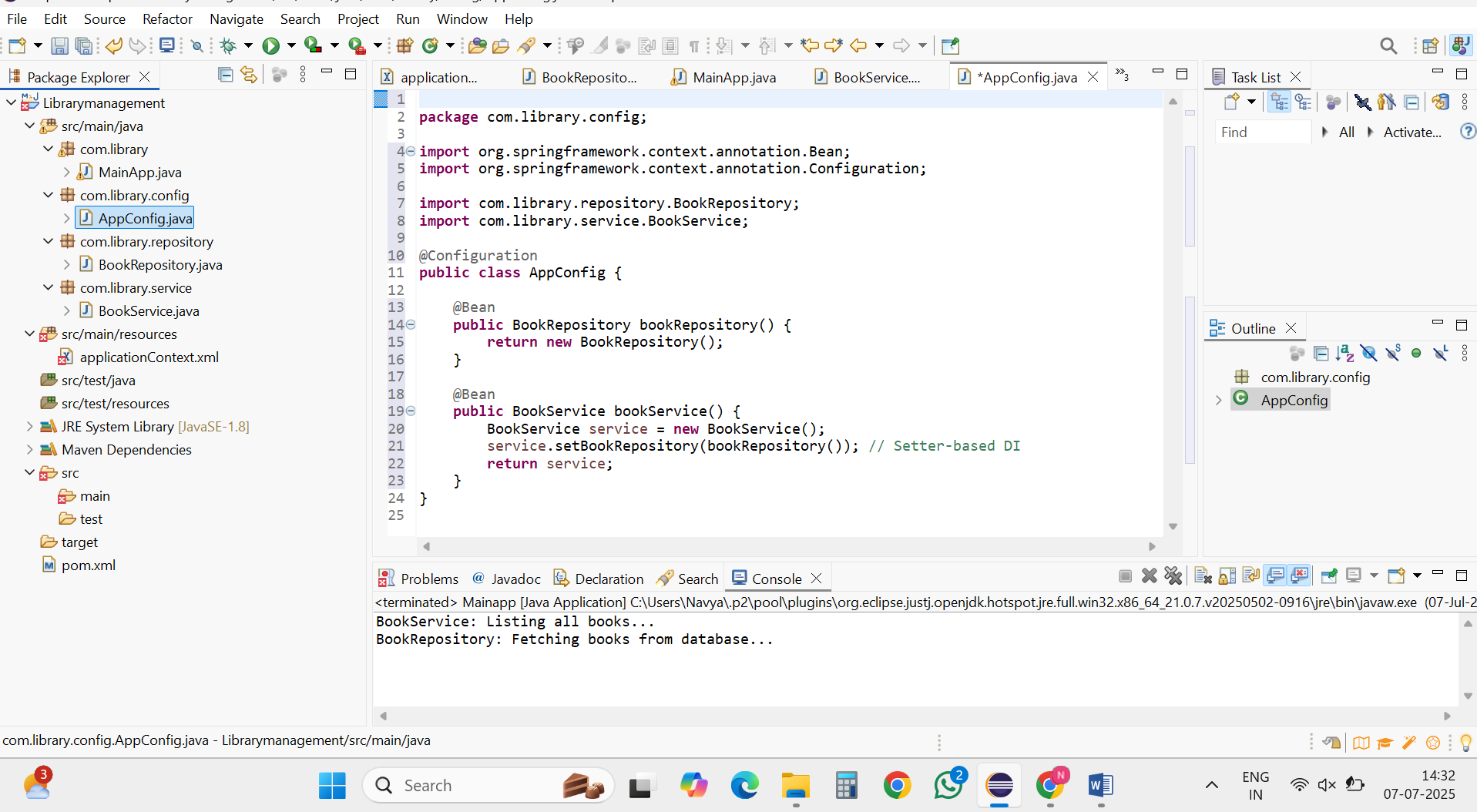
BookService service = (BookService) context.getBean("bookService");

service.listBooks();

}

}

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.

**SOLUTION:**

**Appconfig.java:**

package com.library.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import com.library.repository.BookRepository;

import com.library.service.BookService;

@Configuration

public class AppConfig {

@Bean

public BookRepository bookRepository() {

return new BookRepository();

}

@Bean

public BookService bookService() {

BookService service = new BookService();

service.setBookRepository(bookRepository()); // Setter-based DI

return service;

}

}

BookRepository.java:

package com.library.repository;

public class BookRepository {

public void saveBook(String bookName) {

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

}  
}  
**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) {

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

bookRepository.saveBook(bookName);

}

}  
  
**MainApp.java:**

package com.library;

import com.library.service.BookService;

import com.library.config.AppConfig;

import org.springframework.context.ApplicationContext;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class MainApp {

public static void main(String[] args) {

ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);

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

bookService.addBook("The Alchemist");

}

}

OUTPUT:

