**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:**

**Set Up a Spring Project:**

Create a Maven project named LibraryManagement.

Add Spring Core dependencies in the pom.xml file.

**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.

**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.

**Run the Application:**

Create a main class to load the Spring context and test the configuration.

**Answer:**

**LibraryManagement**

App.java

package com.library.LibraryManagement;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.repository.BookRepository;

import com.library.service.BookService;

/\*\*

\* Hello world!

\*

\*/

public class App

{

public static void main( String[] args )

{

System.out.println( "Hello World!" );

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

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

bs.bookServiceShow();

BookRepository br = (BookRepository) context.getBean("bookRepository");

br.bookRepositoryShow();

}

}

**Repository**

BookRepository.java

package com.library.repository;

public class BookRepository {

public void bookRepositoryShow()

{

System.out.println("Inside BookRepository");

}

}

**Service**

BookService.java

package com.library.service;

public class BookService {

public void bookServiceShow()

{

System.out.println("Inside BookService");

}

}

**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 Definitions -->

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

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

</beans>

**AppTest.java**

package com.library.LibraryManagement;

import junit.framework.Test;

import junit.framework.TestCase;

import junit.framework.TestSuite;

/\*\*

\* Unit test for simple App.

\*/

public class AppTest

extends TestCase

{

/\*\*

\* Create the test case

\*

\* @param testName name of the test case

\*/

public AppTest( String testName )

{

super( testName );

}

/\*\*

\* @return the suite of tests being tested

\*/

public static Test suite()

{

return new TestSuite( AppTest.class );

}

/\*\*

\* Rigourous Test :-)

\*/

public void testApp()

{

assertTrue( true );

}

}

**pom.properties**

#Generated by Maven Integration for Eclipse

#Tue Jul 30 11:25:57 IST 2024

artifactId=LibraryManagement

groupId=com.library

m2e.projectLocation=D\:\\DNP3\\Week 2\_Spring Core\_Maven\\LibraryManagement

m2e.projectName=LibraryManagement

version=0.0.1-SNAPSHOT

**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>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>LibraryManagement</name>

<url>http://maven.apache.org</url>

<properties>

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

</properties>

<dependencies>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>5.2.22.RELEASE</version>

</dependency>

<!--

https://mvnrepository.com/artifact/org.springframework/spring-context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.2.22.RELEASE</version>

</dependency>

<dependency>

<groupId>javax.annotation</groupId>

<artifactId>javax.annotation-api</artifactId>

<version>1.3.2</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

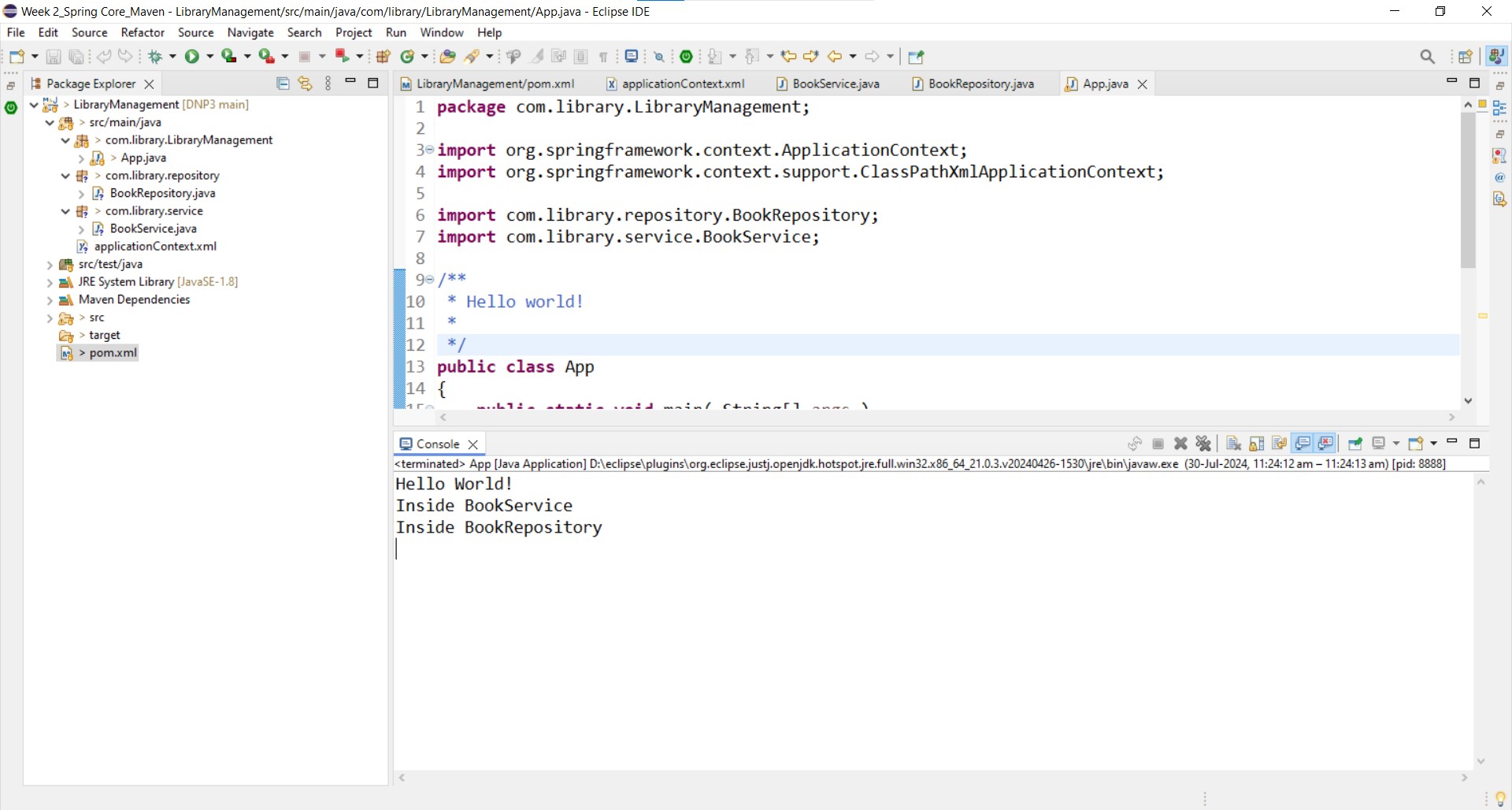
<scope>test</scope>

</dependency>

</dependencies>

</project>

**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:**

**Modify the XML Configuration:**

Update applicationContext.xml to wire BookRepository into BookService.

**Update the BookService Class:**

Ensure that BookService class has a setter method for BookRepository.

**Test the Configuration:**

Run the LibraryManagementApplication main class to verify the dependency injection.

**Answer:**

**App.java**

package com.library.LibraryManagement;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.repository.BookRepository;

import com.library.service.BookService;

/\*\*

\* Hello world!

\*

\*/

public class App

{

public static void main( String[] args )

{

System.out.println( "Hello World!" );

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

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

bs.bookServiceShow();

}

}

**BookRepository.java**

package com.library.repository;

public class BookRepository {

public void bookRepositoryShow()

{

System.out.println("Inside BookRepository");

}

}

**BookService.java**

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 bookServiceShow()

{

System.out.println("Inside BookService");

bookRepository.bookRepositoryShow();

}

}

**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 Definitions -->

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

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

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

</bean>

</beans>

**AppTest.java**

package com.library.LibraryManagement;

import junit.framework.Test;

import junit.framework.TestCase;

import junit.framework.TestSuite;

/\*\*

\* Unit test for simple App.

\*/

public class AppTest

extends TestCase

{

/\*\*

\* Create the test case

\*

\* @param testName name of the test case

\*/

public AppTest( String testName )

{

super( testName );

}

/\*\*

\* @return the suite of tests being tested

\*/

public static Test suite()

{

return new TestSuite( AppTest.class );

}

/\*\*

\* Rigourous Test :-)

\*/

public void testApp()

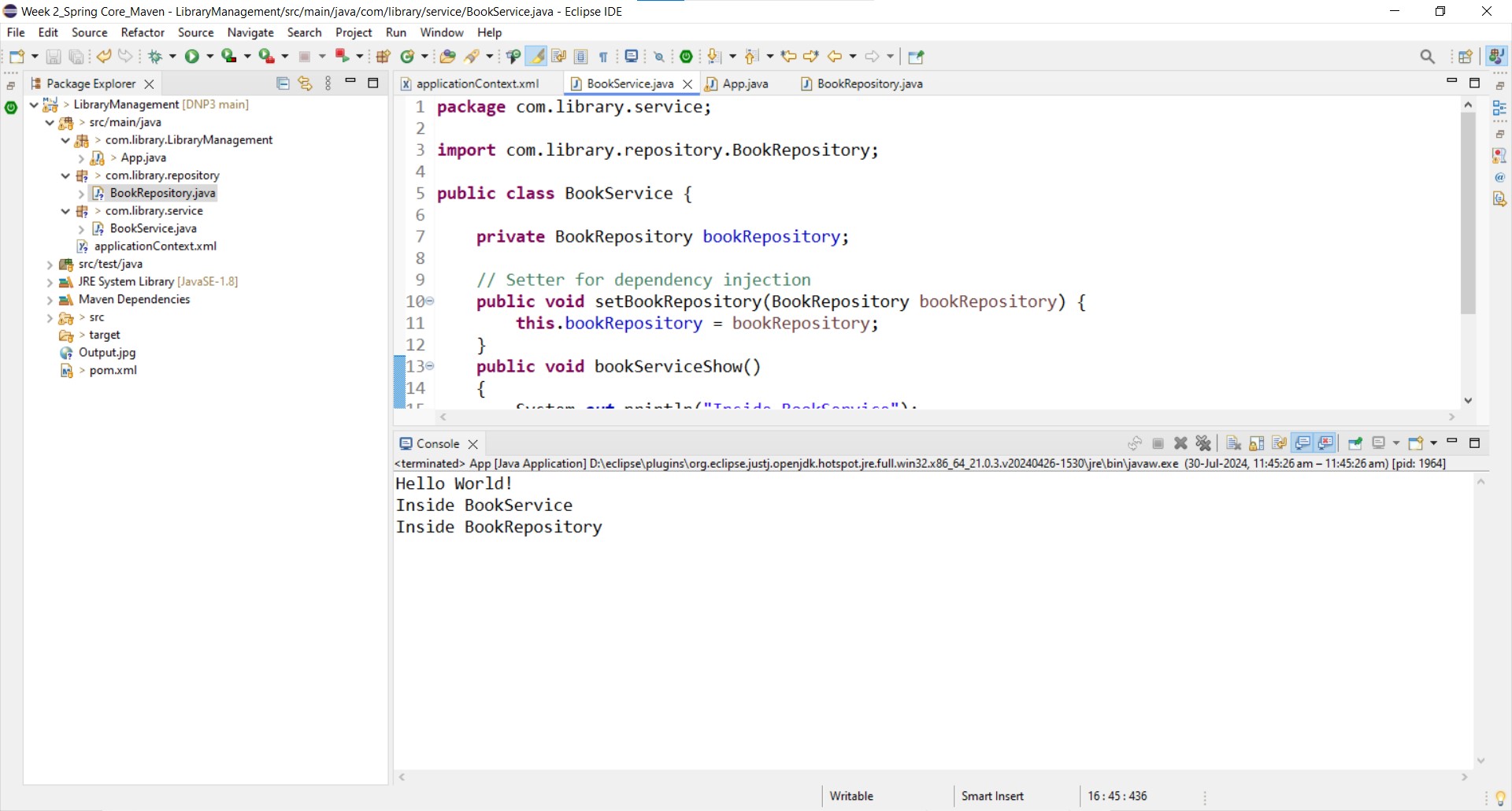
{

assertTrue( true );

}

}

**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:**

**Create a New Maven Project:**

Create a new Maven project named LibraryManagement.

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

Include dependencies for Spring Context, Spring AOP, and Spring WebMVC.

**Configure Maven Plugins:**

Configure the Maven Compiler Plugin for Java version 1.8 in the pom.xml file.

**App.java**

package com.spring\_maven.LibraryManagement;

/\*\*

\* Hello world!

\*

\*/

public class App

{

public static void main( String[] args )

{

System.out.println( "Hello World!" );

}

}

**AppTest.java**

package com.spring\_maven.LibraryManagement;

import junit.framework.Test;

import junit.framework.TestCase;

import junit.framework.TestSuite;

/\*\*

\* Unit test for simple App.

\*/

public class AppTest

extends TestCase

{

/\*\*

\* Create the test case

\*

\* @param testName name of the test case

\*/

public AppTest( String testName )

{

super( testName );

}

/\*\*

\* @return the suite of tests being tested

\*/

public static Test suite()

{

return new TestSuite( AppTest.class );

}

/\*\*

\* Rigourous Test :-)

\*/

public void testApp()

{

assertTrue( true );

}

}

**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.spring-maven</groupId>

<artifactId>LibraryManagement</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>LibraryManagement</name>

<url>http://maven.apache.org</url>

<properties>

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

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<!--

https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

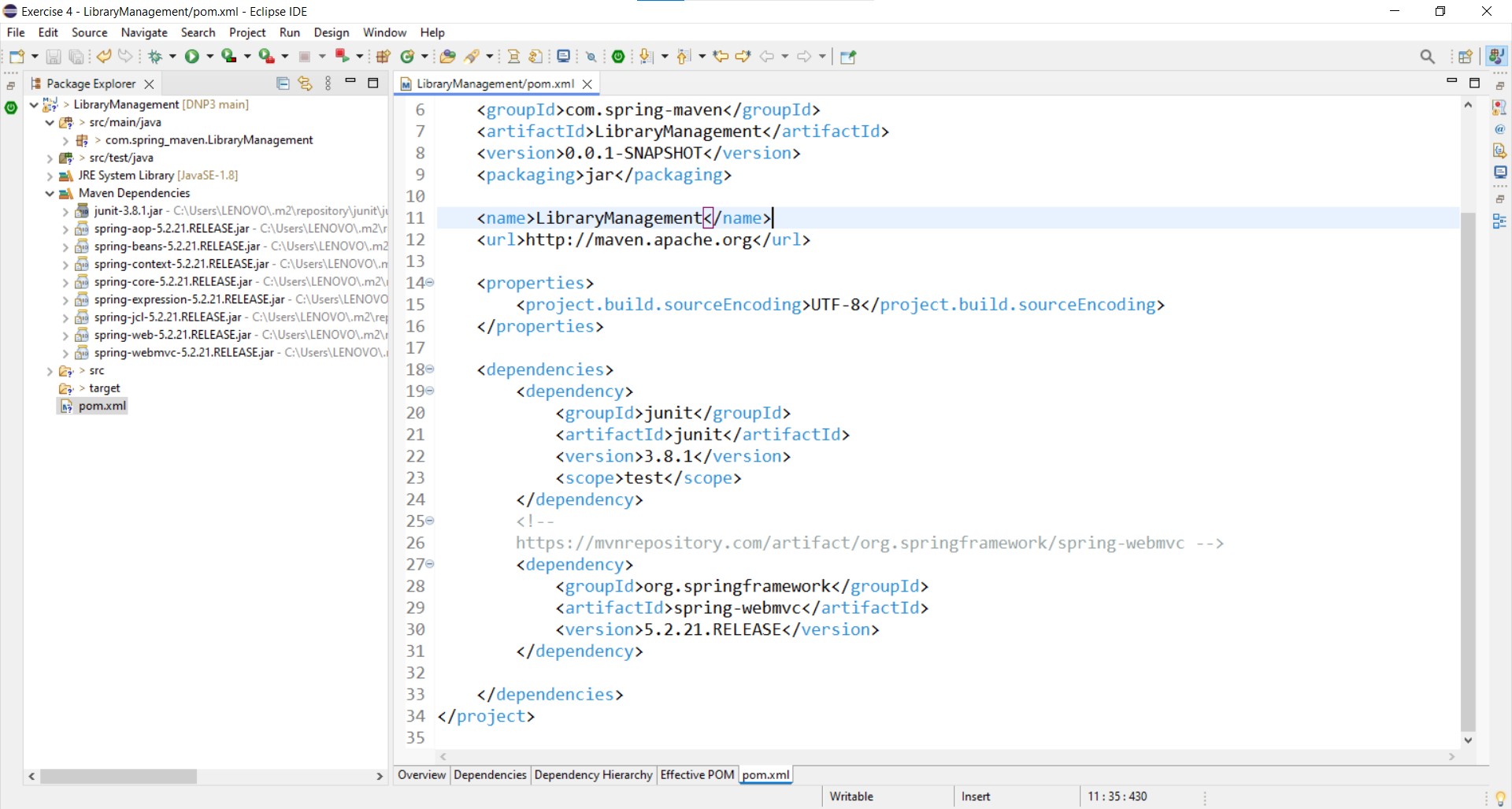
<version>5.2.21.RELEASE</version>

</dependency>

</dependencies>

</project>

**Output:**

****