**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.

**Book Repository.java**

package com.library.repository;public class BookRepository { public void saveBook(String bookName) { System.out.println("Book saved: " + bookName); }}

**Book Service.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 to library: " + bookName); bookRepository.saveBook(bookName); }}

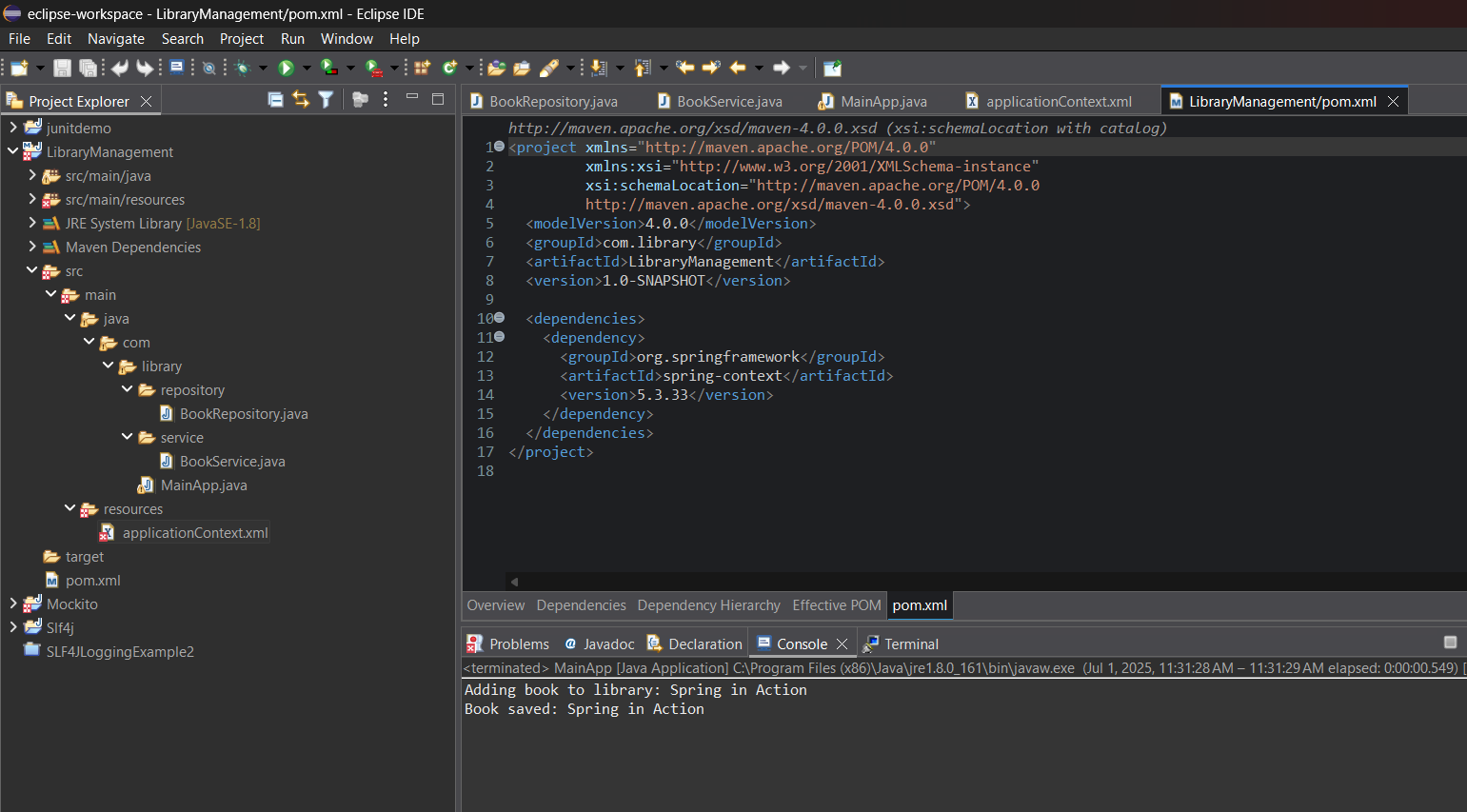
**Main.app**

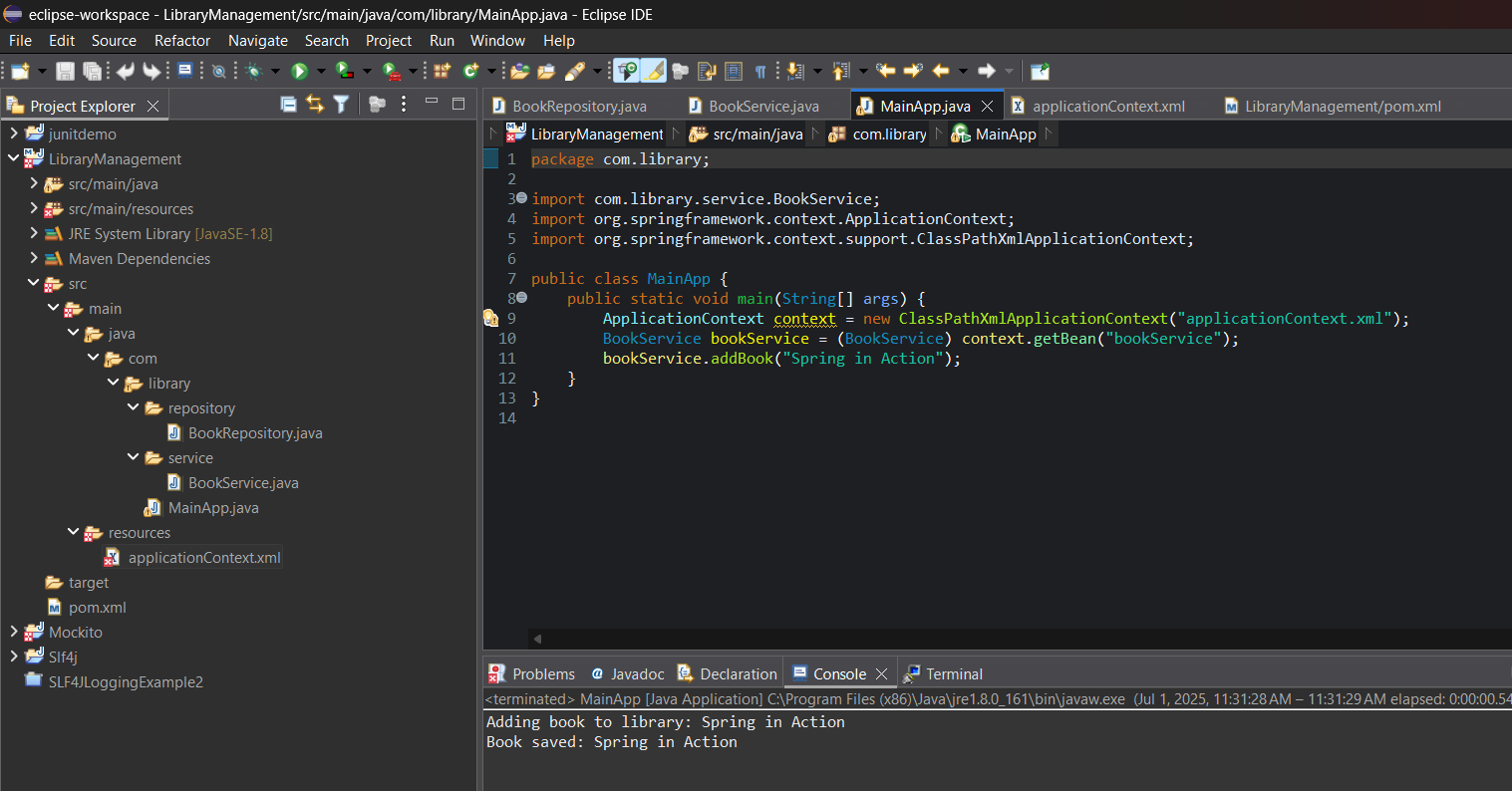
package com.library;import com.library.service.BookService;import org.springframework.context.ApplicationContext;import org.springframework.context.support.ClassPathXmlApplicationContext;public class MainApp { public static void main(String[] args) { ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml"); BookService bookService = (BookService) context.getBean("bookService"); bookService.addBook("Spring in Action"); }}

**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>5.3.33</version> </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:**

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.

**Book Service.java**

package com.library.service;import com.library.repository.BookRepository;public class BookService { private BookRepository bookRepository; // Setter method for Dependency Injection public void setBookRepository(BookRepository bookRepository) { this.bookRepository = bookRepository; System.out.println("BookRepository injected via setter."); } public void addBook(String bookName) { System.out.println("Adding book to library: " + bookName); bookRepository.saveBook(bookName); }}

**Book Repository.java**

package com.library.repository;public class BookRepository { public void saveBook(String bookName) { System.out.println("Book saved: " + bookName); }}

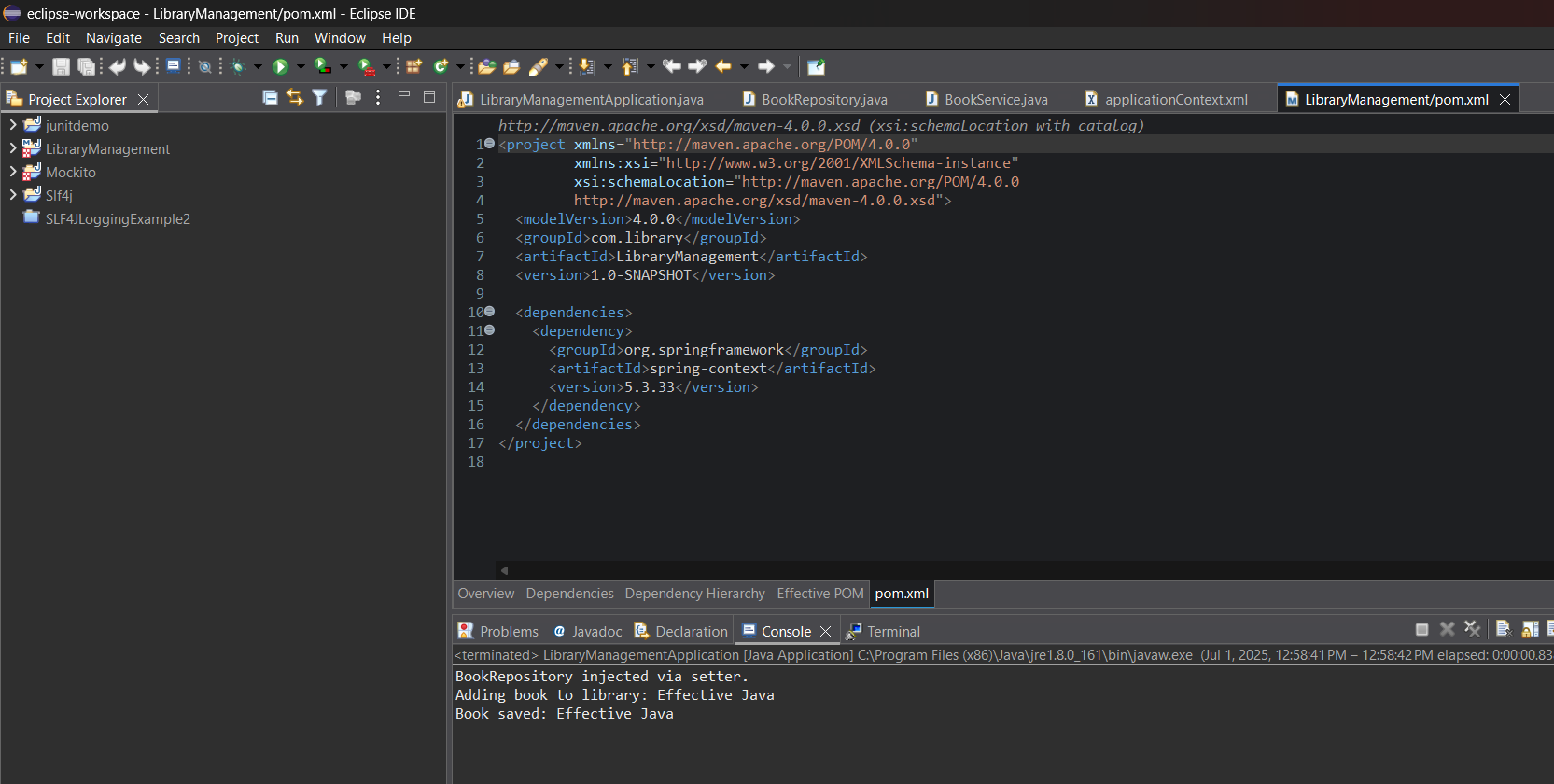
**Application Context.xml**

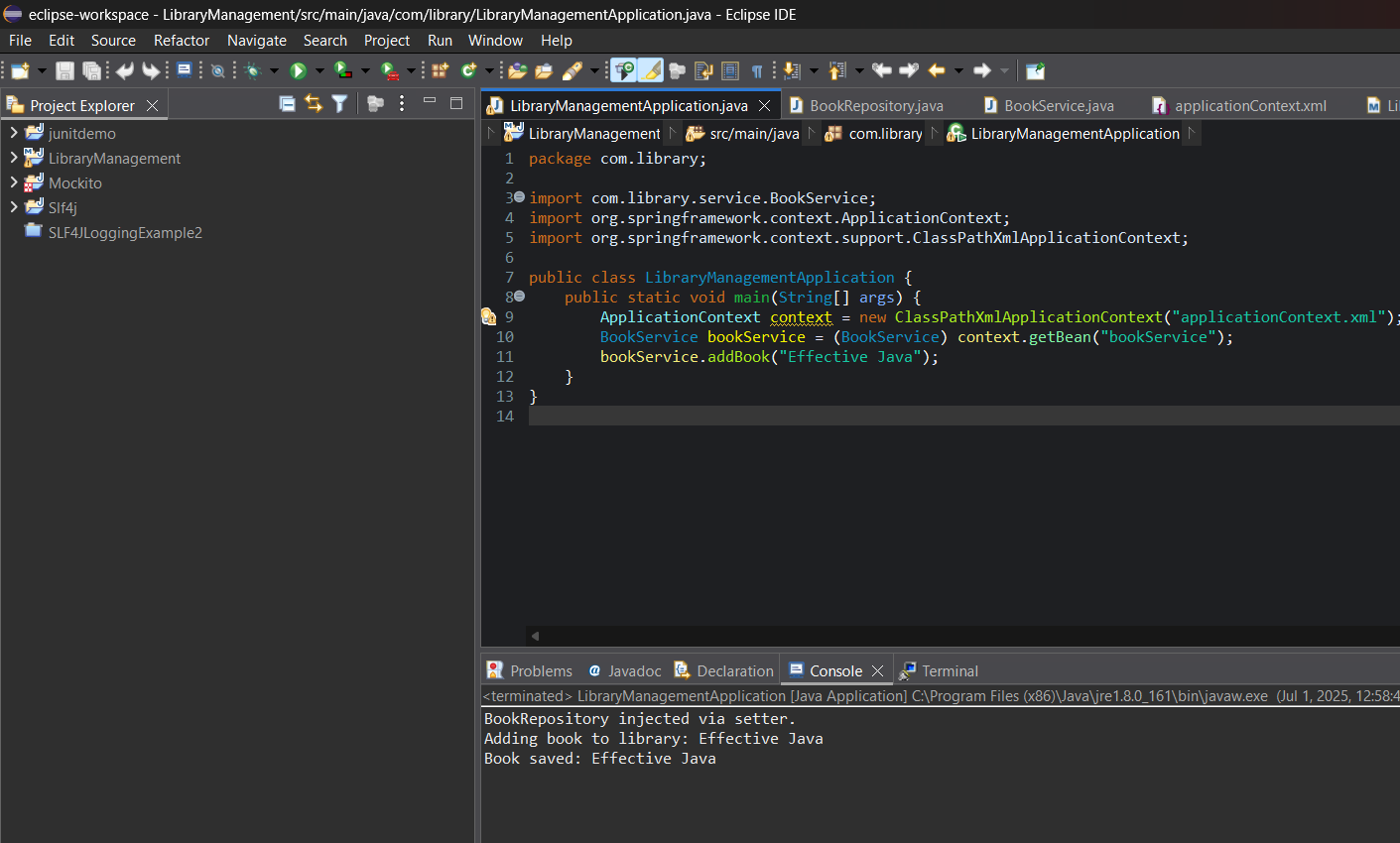
<?xml version="1.0" encoding="UTF-8"?><beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd"> <bean id="bookRepository" class="com.library.repository.BookRepository"/> <bean id="bookService" class="com.library.service.BookService"> <property name="bookRepository" ref="bookRepository"/> </bean></beans>

**Library Management Application.java**

package com.library;import com.library.service.BookService;import org.springframework.context.ApplicationContext;import org.springframework.context.support.ClassPathXmlApplicationContext;public class LibraryManagementApplication { public static void main(String[] args) { ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml"); BookService bookService = (BookService) context.getBean("bookService"); bookService.addBook("Effective Java"); }}

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.

**Library management 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> <packaging>jar</packaging> <name>LibraryManagement</name> <properties> <java.version>1.8</java.version> <spring.version>5.3.30</spring.version> </properties> <dependencies> <dependency> <groupId>org.springframework</groupId> <artifactId>spring-context</artifactId> <version>${spring.version}</version> </dependency>

<dependency> <groupId>org.springframework</groupId> <artifactId>spring-aop</artifactId> <version>${spring.version}</version> </dependency>

<dependency> <groupId>org.springframework</groupId> <artifactId>spring-webmvc</artifactId> <version>${spring.version}</version> </dependency> <dependency> <groupId>commons-logging</groupId> <artifactId>commons-logging</artifactId> <version>1.2</version> </dependency> </dependencies> <build> <plugins> <plugin> <groupId>org.apache.maven.plugins</groupId> <artifactId>maven-compiler-plugin</artifactId> <version>3.8.1</version> <configuration> <source>1.8</source> <target>1.8</target> </configuration> </plugin> </plugins> </build></project>

**MainApp.java**

package com.library1;import org.springframework.context.ApplicationContext;import org.springframework.context.support.ClassPathXmlApplicationContext;public class MainApp { public static void main(String[] args) { System.out.println("Library Management App is running..."); }}

Output:

