**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.

**Code:**  
  
1. pom.xml – Adding Spring Dependency

<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.library

</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- Spring Core -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.33</version>

</dependency>

</dependencies>

</project>

2. BookRepository.java – Repository Layer

package com.library.repository;

public class BookRepository {

public void saveBook(String bookName) {

System.***out***.println("Book '" + bookName + "' saved to repository.");

}

}

3. BookService.java – Service Layer

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Setter injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook(String bookName) {

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

bookRepository.saveBook(bookName);

}

}

4. applicationContext.xml – Spring Configuration

<?**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 dependency injection -->

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

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

</**bean**>

</**beans**>

5. MainApp.java – Entry Point (Client Code)

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 bean

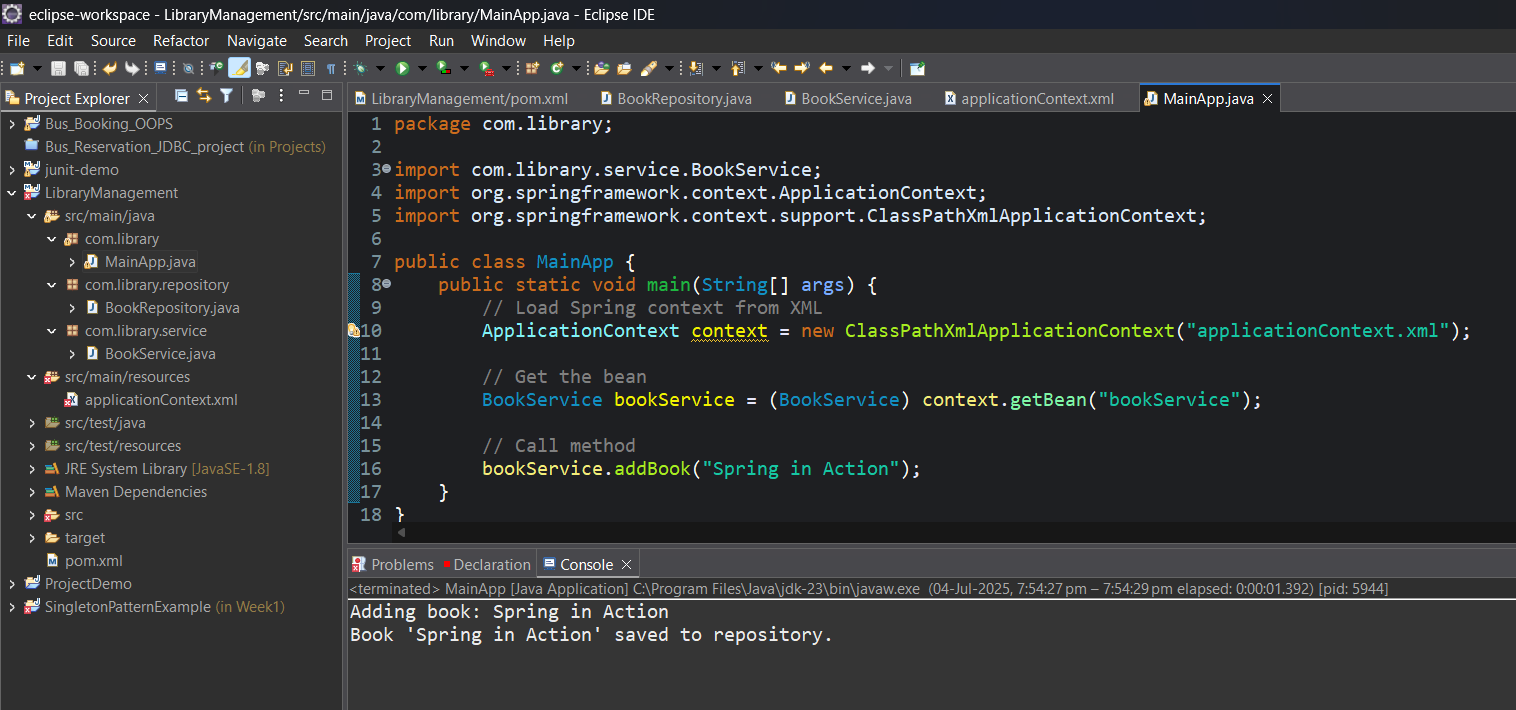
BookService bookService = (BookService) context.getBean("bookService");

// Call method

bookService.addBook("Spring in Action");

}

}

**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.

**Code:**

1. Modify 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 definition for BookRepository -->

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

<!-- Bean definition for BookService -->

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

<!-- Inject BookRepository into BookService via setter -->

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

</**bean**>

</**beans**>

2. Ensure BookService Has a Setter

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Setter method for DI

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook(String bookName) {

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

bookRepository.saveBook(bookName);

}

}

3. Test the Configuration

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

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

// Get the service bean

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

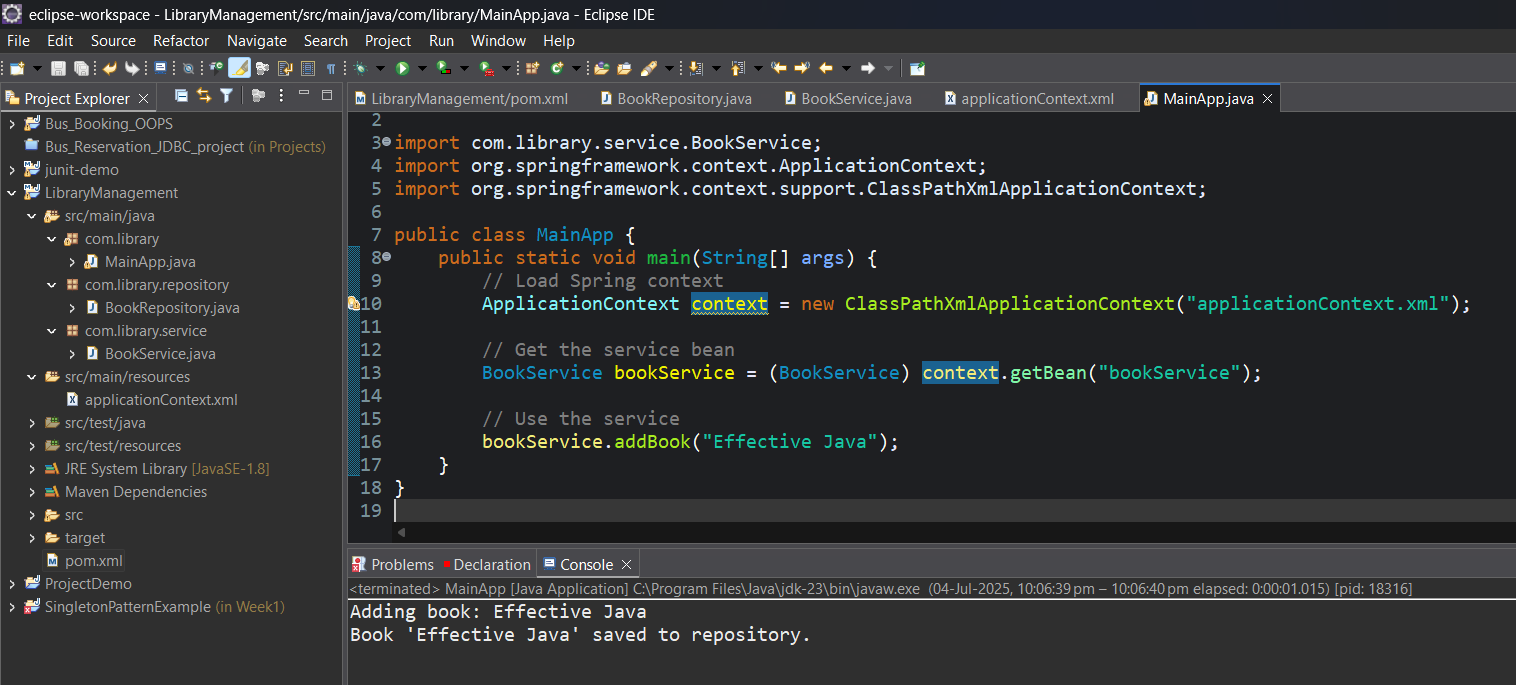
// Use the service

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.

**Code:**

1. Adding Spring Dependencies in pom.xml:

<dependencies>

<!-- Spring Context for Core Container and DI -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.33</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.33</version>

</dependency>

<!-- Spring Web MVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.33</version>

</dependency>

</dependencies>

1. Configure Maven Compiler Plugin:

<build>

<plugins>

<!-- Maven Compiler Plugin -->

<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>

**Simple Java class test output:**

