**Spring Core and Maven**

***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.

**Step 1: Set Up Maven Project**

Add Spring dependencies in 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>5.3.36</version>

</dependency>

</dependencies>

</project>

**Step 2: Configure Beans**

Create applicationContext.xml in src/main/resources:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"https://www.springframework.org/schema/beans"*

xmlns:xsi=*"https://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans”*

*https://www.springframework.org/schema/beans/spring-beans.xsd"*>

<!-- Bean for BookRepository -->

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

<!-- Bean for BookService -->

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

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

</bean>

</beans>

**Step 3: Create Classes**

**BookRepository.java**

package com.library.repository;

public class BookRepository {

public void saveBook(String bookName) {

System.***out***.println("Book saved: " + 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);

}

}

**Step 4: Run Application**

**LibraryManagementApplication.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");

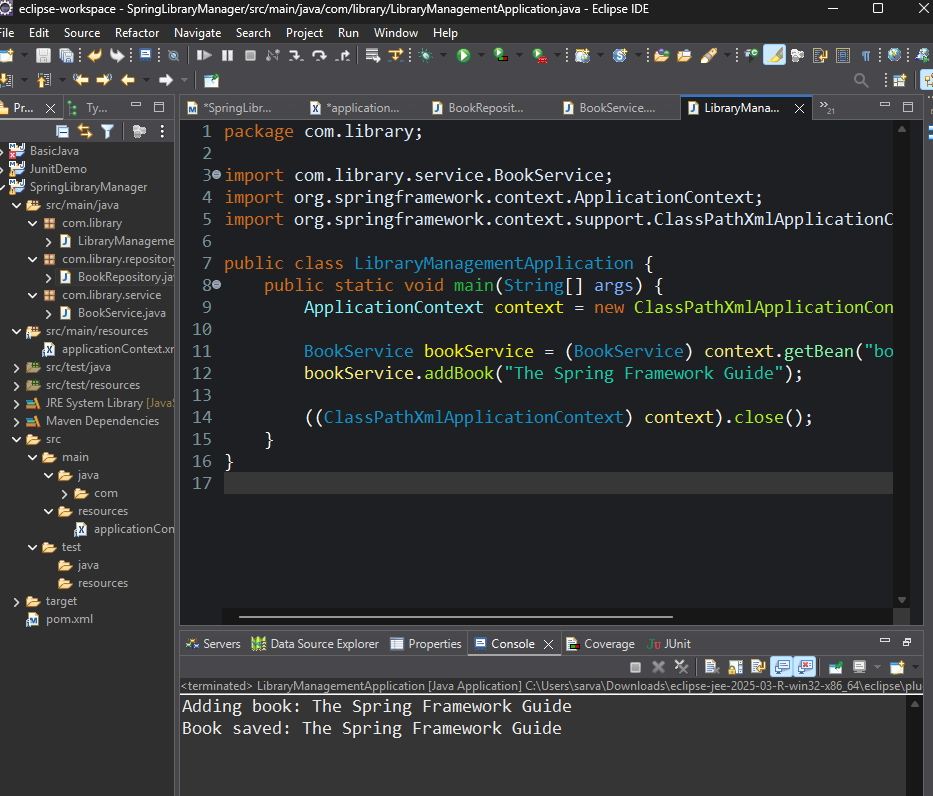
BookService bookService = (BookService) context.getBean("bookService");

bookService.addBook("The Spring Framework Guide");

((ClassPathXmlApplicationContext) context).close();

}

}**Output :**



***Exercise 2: Implementing Dependency Injection***

**Scenario**

In the Library Management application, implement dependency injection between BookService and BookRepository using Spring IoC (Inversion of Control).

**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.

**Step 1: Modify XML Configuration**

Update applicationContext.xml to wire BookRepository into BookService.

**applicationContext.xml**

<?**xml** version=*"1.0"* encoding=*"UTF-8"*?>

<**beans** xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"https://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**>

**Step 2: Update BookService Class**

Ensure BookService has a setter for BookRepository to allow Spring to inject the dependency.

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);

}

}

**Step 3: Test the Configuration**

Run the main class to verify dependency injection.

**LibraryManagementApplication.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");

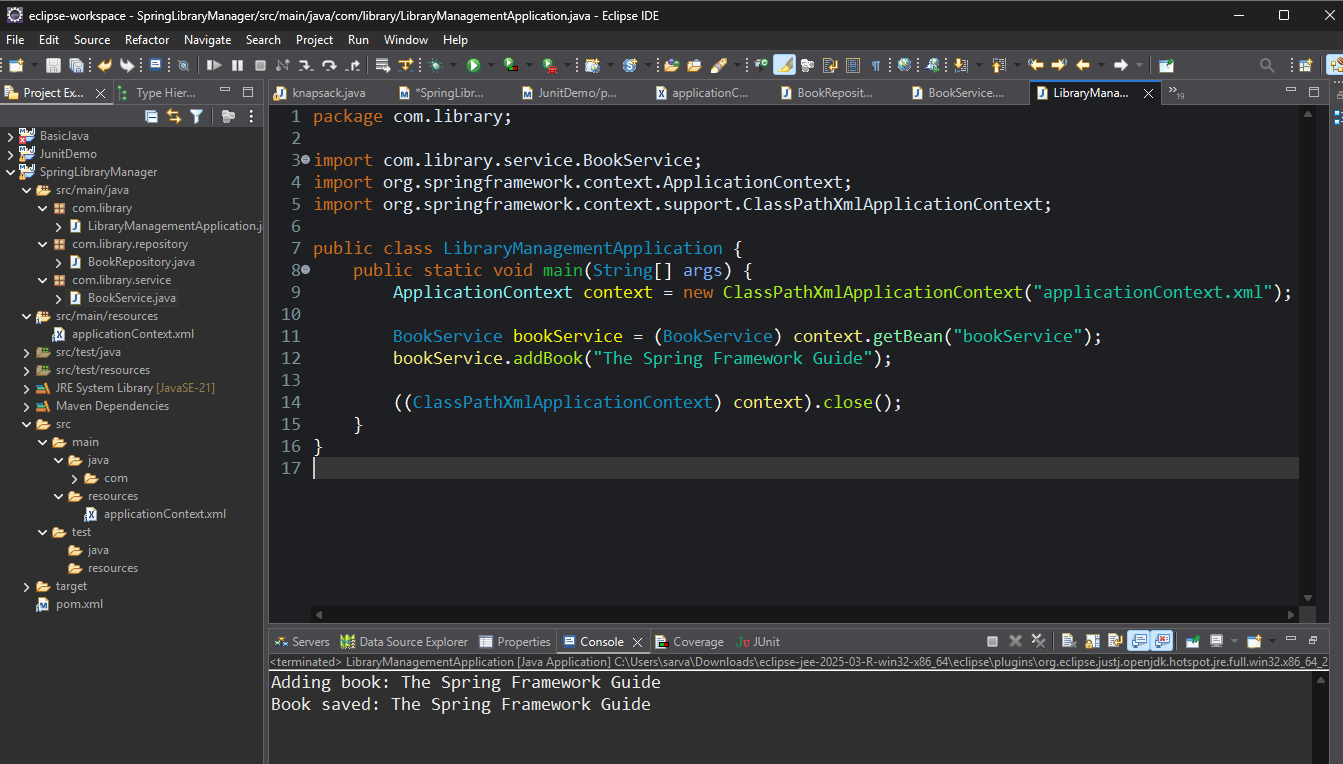
bookService.addBook("The Spring Framework Guide");

((ClassPathXmlApplicationContext) context).close();

}

}

**Output :**

****

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

**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.

**Step 1: Create a New Maven Project**

* Create a Maven project named LibraryManagement.
* File ->New->Maven Project -> Name: LibraryManagement

**Step 2: Add Spring Dependencies in 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>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.36</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.36</version>

</dependency>

<!-- Spring Web MVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.36</version>

</dependency>

</dependencies>

**Step 3: Configure Maven Compiler Plugin**

<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>

**Result**

* Maven project with Spring dependencies and plugins is now ready.

