**WEEK 3**

**Spring Core\_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.**

<projectxmlns="https://maven.apache.org/POM/4.0.0" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="https://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.LibraryManagement</groupId>

<artifactId>LibraryManagement</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.30</version>

</dependency>

</dependencies>

</project>

1. **Configure the Application Context:**
   * **Create an XML configuration file named applicationContext.xml in the src/main/resources directory.**

<?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">

<!-- BookRepository Bean -->

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

<!-- BookService Bean with dependency on BookRepository -->

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

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

</bean>

</beans>

* + Define beans for BookService and BookRepository in the XML file.

1. **Define Service and Repository Classes:**
   * **Create a package com.library.service and add a class BookService. 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) {

bookRepository.saveBook(bookName);

}

}

* + **Create a package com.library.repository and add a class BookRepository**.

package com.library.repository;

public class BookRepository {

public void saveBook(String bookName) {

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

}

}

1. **Run the Application:**
   * Create a main class to load the Spring context and test the configuration

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");

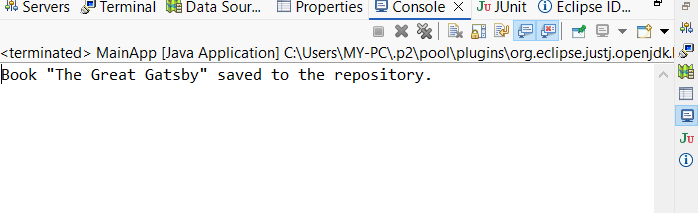
BookService bookService = context.getBean("bookService", BookService.class);

bookService.addBook("The Great Gatsby");

}

}

**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.**

<?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"*>

<!-- BookRepository Bean -->

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

<!-- BookService Bean with DI -->

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

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

</bean>

</beans>

1. **Update the BookService Class:**
   * **Ensure that BookService class has a setter method for BookRepository.**

public class BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook(String bookName) {

bookRepository.saveBook(bookName);

}

}

1. **Test the Configuration:**
   * **Run the LibraryManagementApplication main class to verify the dependency injection.**

public class BookRepository {

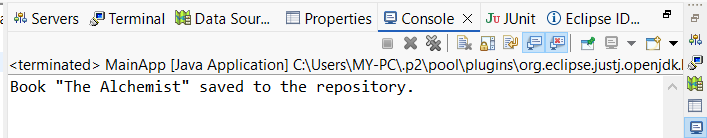
public void saveBook(String bookName) {

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

}

}

**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

**.OUTPUT:** 