**WEEK 3**

**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.

**PROGRAM**

**package** com.library.repository;

**public** **class** BookRepository {

**public** **void** displayBooks() {

System.***out***.println("Fetching books from database...");

}

**public** String getBookDetails() {

**return** "Clean Code by Robert C. Martin";

}

}

**package** com.library.service;

**import** com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository = new BookRepository(); // ❌ Manual wiring

public void displayBook() {

System.out.println("Book info: " + bookRepository.getBookDetails());

}

}

**package** com.library;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** com.library.service.BookService;

**public** **class** MainApp {

**public** **static** **void** main(String[] args) {

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

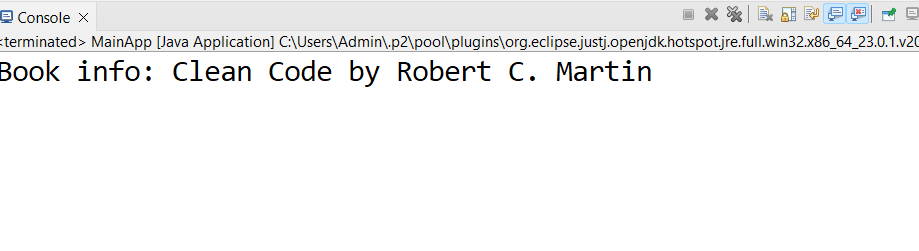
BookService bookService = context.getBean("bookService", BookService.**class**);

bookService.displayBook();

}

}

**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.

**PROGRAM**

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 displayBook() {

System.out.println("Book info: " + bookRepository.getBookDetails());

}

}

package com.library.repository;

public class BookRepository {

public String getBookDetails() {

return "Clean Code by Robert C. Martin";

}

}

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

<!-- BookRepository Bean -->

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

<!-- BookService Bean with DI using setter -->

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

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

</bean>

</beans>

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class LibraryManagementApplication {

public static void main(String[] args) {

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

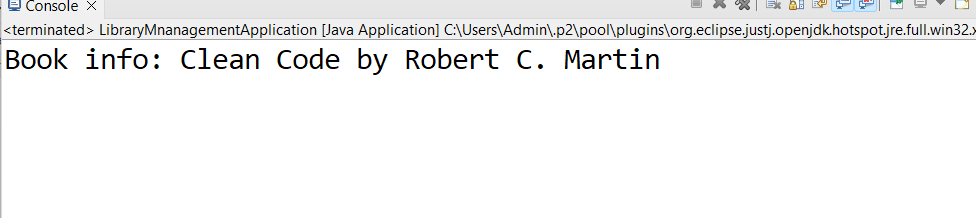
BookService service = context.getBean("bookService", BookService.class);

service.displayBook(); // DI worked

}

}

**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.

**PROGRAM**

**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 https://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>

<name>LibraryManagement</name>

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

<dependencies>

<!-- Spring Core Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.32</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.32</version>

</dependency>

<!-- Spring Web MVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.32</version>

</dependency>

</dependencies>

<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>

</project>