**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.

**ANSWER**

**Answer: no. 1: Set Up a Spring Project:**

a) Create Maven Project using the option in create java project option in vscode and select maven as the build tool.

Project name: LibraryManagement

b) Add Dependencies to pom.xml

**Code:**

<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>5.3.32</version>

</dependency>

</dependencies>

</project>

Answer: no. 2:-- **Configure the Application Context:**

Create an XML configuration file named **applicationContext.xml** in the **src/main/resources** directory and define beans for **BookService** and **BookRepository** in the XML file.

Path: src/main/resources/applicationContext.xml

Code:

<?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>

Answer 3 and 4:

Path: **com.library.repository.BookRepository.java**

**Code:**

package com.library.repository;

public class BookRepository

{

    public void saveBook(String bookName)

    {

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

    }

}

Path: **com.library.service.BookService.java**

**Code:**

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);

    }

}

Main.java:

package com.library;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Main

{

    public static void main(String[] args)

    {

        ApplicationContext context =

            new ClassPathXmlApplicationContext("applicationContext.xml");

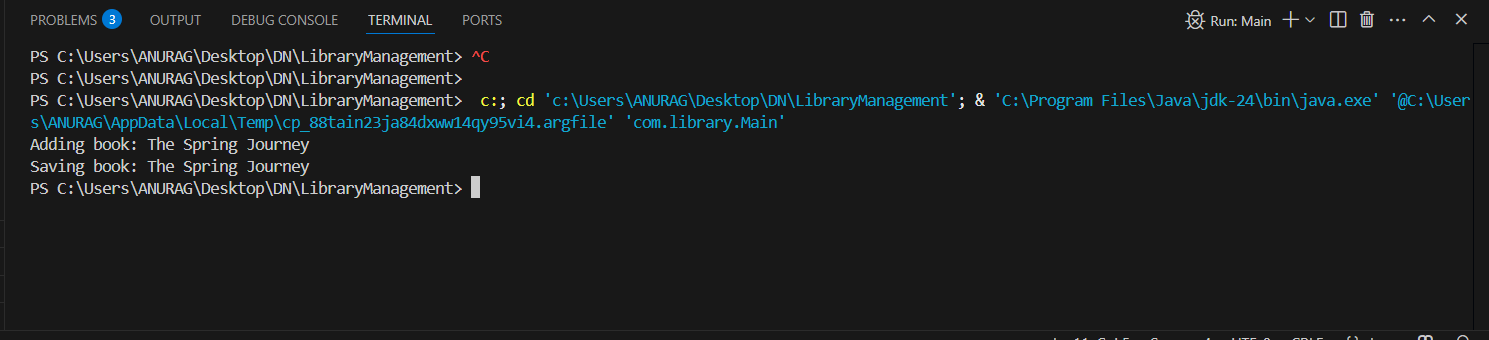
        BookService bookService = context.getBean("bookService", BookService.class);

        bookService.addBook("The Spring Journey");

    }

}

**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.

**Modify the XML Configuration: answer:**

New code:

<?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>

**Update the BookService Class; answer:**

New code:

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);

}

}

**Test the Configuration: answer:**

Main.java 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)

{

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

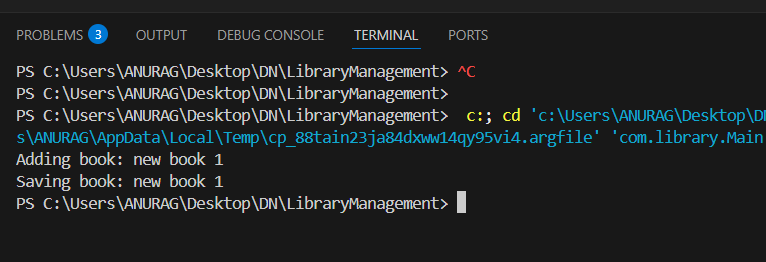
BookService bookService = context.getBean("bookService", BookService.class);

bookService.addBook("new book 1");

}

}

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:**

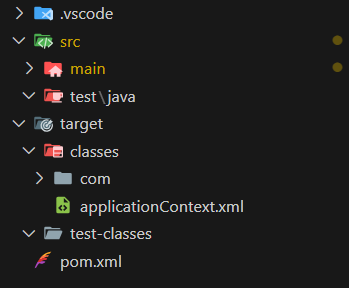
1. **Create a New Maven Project:**
   * Create a new Maven project named **LibraryManagement**.
2. **Add Spring Dependencies in pom.xml:**
   * Include dependencies for Spring Context, Spring AOP, and Spring WebMVC.
3. **Configure Maven Plugins:**
   * Configure the Maven Compiler Plugin for Java version 1.8 in the pom.xml file.

**Answer no. 1: create a new maven project:--**

**a)** Command line: mvn archetype:generate

**b)**  Vscode IDE: create java project -> select maven -> select archtype -> select destination folder.

**Output (maven project structure) :-**



**Answer no.2:- Add spring dependencies to pom.xml:-**

Write the following code 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>5.3.32</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.32</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.32</version>

</dependency>

</dependencies>

</project>

**Answer no. 3:- configure maven plugins:--**

Add the following code in the pom.xml file inside <build> tag :-

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.11.0</version> <!-- or latest stable version -->

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>