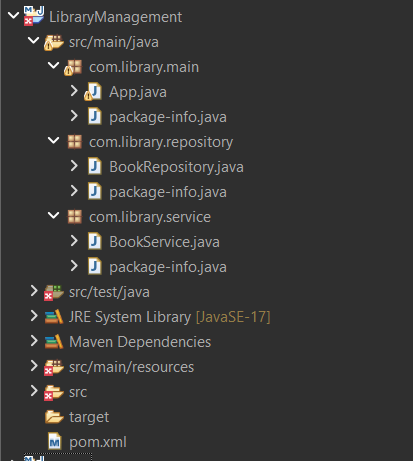
**Exercise 1: Configuring a Basic Spring Application**

**Scenario:**

Your company is developing a web application for managing a library. You need to use the Spring Framework to handle the backend operations.

**FOLDER STRUCTURE**

****

**pom.xml**

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

<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.maven</groupId>

<artifactId>spring</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring</name>

<url>http://www.example.com</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

</properties>

<dependencies>

<!-- Spring Core + Context for ApplicationContext & XML config -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.33</version>

</dependency>

<!-- Spring Beans (optional but useful) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>5.3.33</version>

</dependency>

<!-- JUnit for unit testing -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.11</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Maven Compiler Plugin -->

<plugin>

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

<version>3.8.1</version>

</plugin>

<!-- Exec plugin to run Java main class -->

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>exec-maven-plugin</artifactId>

<version>3.1.0</version>

<configuration>

<mainClass>com.maven.spring.App</mainClass>

</configuration>

</plugin>

</plugins>

</build>

</project>

**BookService.java(com.library.service-🡪package)**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository br;

public BookRepository getBr() {

return br;

}

public void setBr(BookRepository br) {

this.br = br;

}

public void display(){

System.***out***.println("books" +br);

}

}

**BookRepository.java(com.library.repository-🡪package)**

package com.library.repository;

public class BookRepository {

private String name;

private int bookid;

private String author;

private int price;

public int getBookid() {

return bookid;

}

public void setBookid(int bookid) {

this.bookid = bookid;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

public int getPrice() {

return price;

}

public void setPrice(int price) {

this.price = price;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

*@Override*

public String toString() {

return "BookRepository [name=" + name + ", bookid=" + bookid + ", author=" + author + ", price=" + price + "]";

}

}

**App.java (com.library.main-🡪package)**

package com.library.main;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class App

{

public static void main(String[] args)

{

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

BookService bs= (BookService)context.getBean("bookservice", BookService.class);

bs.display();

}

}

**applicationContext.xml(src/main/resources/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"

xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="

http://www.springframework.org/schema/beans

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

http://www.springframework.org/schema/context

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

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

<property name="br" ref="bookrepository"></property>

</bean>

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

<property name="name" value="uuuu"></property>

<property name="bookid" value="1089"></property>

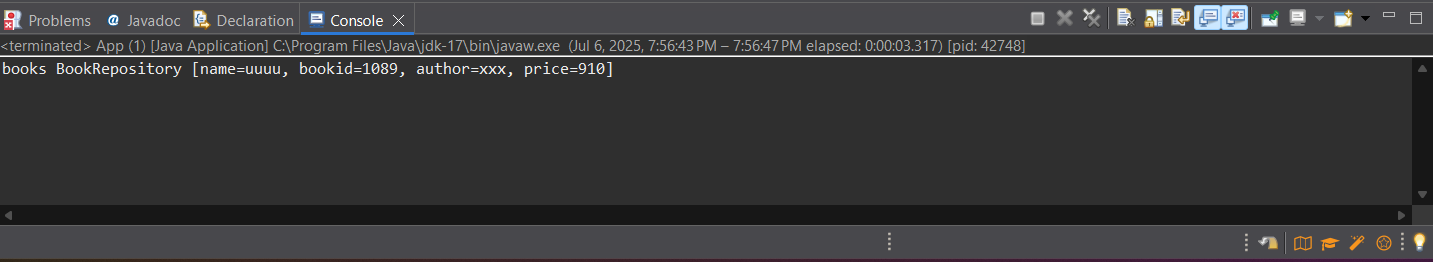
<property name="author" value="xxx"></property>

<property name="price" value="910"></property>

</bean>

</beans>

**OUTPUT**

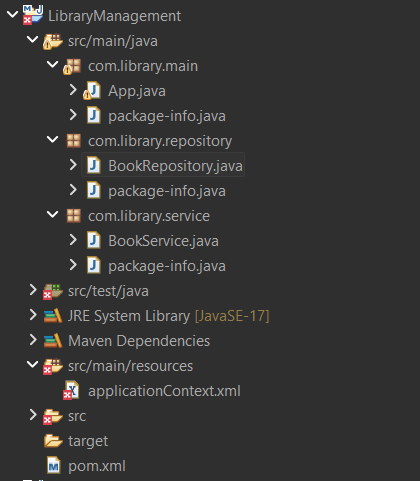
****

**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.

**FOLDER STRUCTURE**

****

**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"

xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="

http://www.springframework.org/schema/beans

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

http://www.springframework.org/schema/context

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

<context:component-scan base-package="com.library"/>

<context:annotation-config/>

</beans>

**BookRepository.java**

package com.library.repository;

import org.springframework.stereotype.Repository;

import org.springframework.beans.factory.annotation.Value;

*@Repository*

public class BookRepository {

*@Value* ("vvv")

private String name;

*@Value*("9089")

private int bookid;

*@Value*("jjjjjj")

private String author;

*@Value*("90")

private int price;

public int getBookid() {

return bookid;

}

public void setBookid(int bookid) {

this.bookid = bookid;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

public int getPrice() {

return price;

}

public void setPrice(int price) {

this.price = price;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

*@Override*

public String toString() {

return "BookRepository [name=" + name + ", bookid=" + bookid + ", author=" + author + ", price=" + price + "]";

}

}

**BookService.java**

package com.library.service;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.library.repository.BookRepository;

*@Service*

public class BookService {

*@Autowired*

private BookRepository br;

public BookRepository getBr() {

return br;

}

public void setBr(BookRepository br) {

this.br = br;

}

public void display()

{

System.***out***.println("books " +br);

}

}

**App.java**

package com.library.main;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class App

{

public static void main(String[] args)

{

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

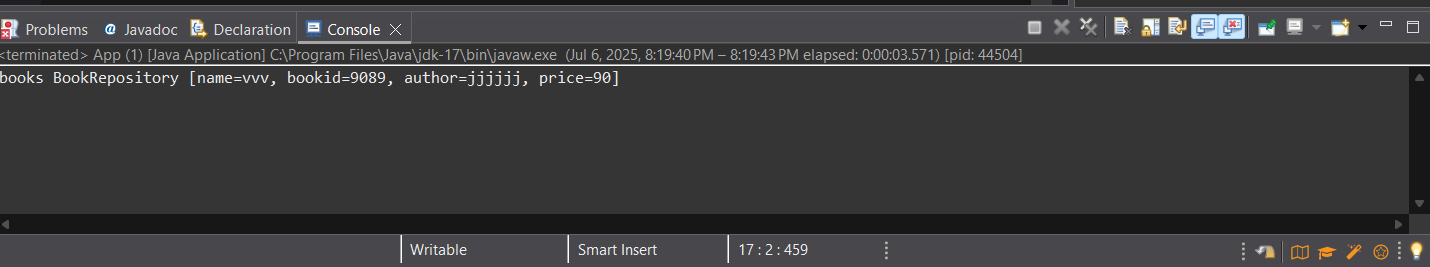
BookService bs= (BookService)context.getBean( BookService.class);

bs.display();

}

}

**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**

**Pom.xml**

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

<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.maven</groupId>

<artifactId>spring</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring</name>

<url>http://www.example.com</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

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

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

</properties>

<dependencies>

<!-- Spring Core + Context for ApplicationContext & XML config -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.33</version>

</dependency>

<!-- Spring Beans -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>5.3.33</version>

</dependency>

<!-- JUnit for unit testing -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.11</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.33</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.33</version>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Maven Compiler Plugin -->

<plugin>

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

<version>3.8.1</version>

</plugin>

<!-- Exec plugin to run Java main class -->

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>exec-maven-plugin</artifactId>

<version>3.1.0</version>

<configuration>

<mainClass>com.maven.spring.App</mainClass>

</configuration>

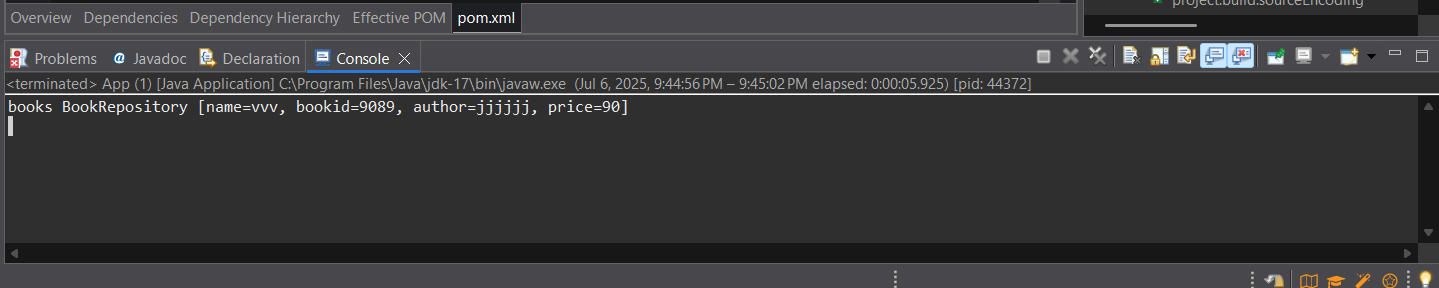
</plugin>

</plugins>

</build>

</project>

**OUTPUT**

****