# **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.

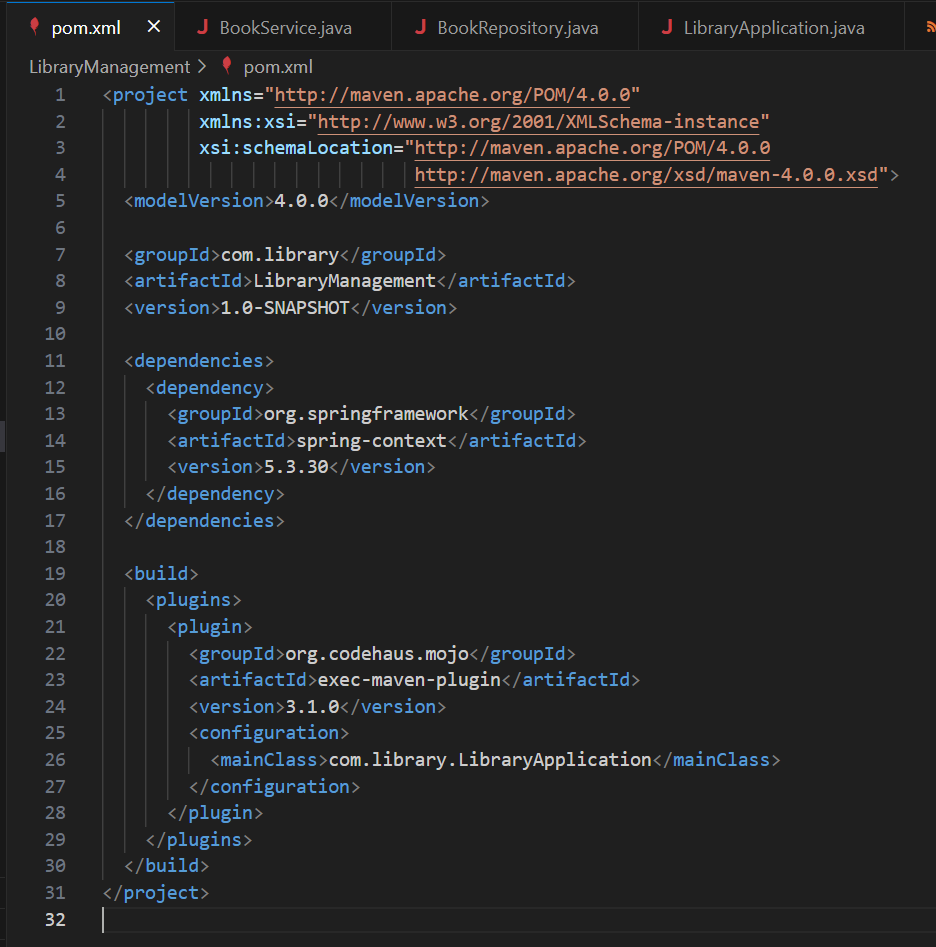
## **Step 1: Set Up a Spring Project**

* Created a Maven project named LibraryManagement.
* Added Spring Core dependency in the pom.xml:

<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.30</version>  
</dependency>

Included the exec-maven-plugin to run the main() class easily:

<build>  
 <plugins>  
 <plugin>  
 <groupId>org.codehaus.mojo</groupId>  
 <artifactId>exec-maven-plugin</artifactId>  
 <version>3.1.0</version>  
 <configuration>  
 <mainClass>com.library.LibraryApplication</mainClass>  
 </configuration>  
 </plugin>  
 </plugins>  
</build>

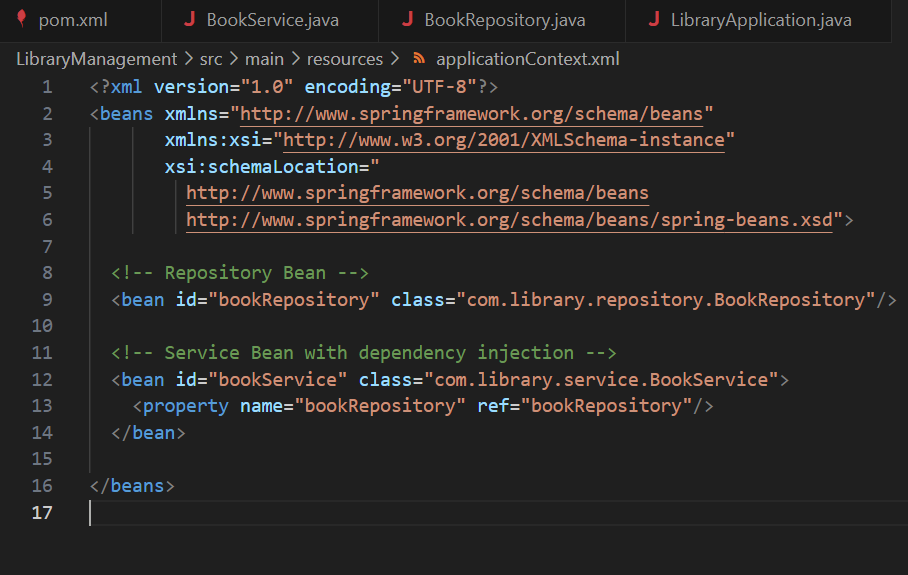


## **Step 2: Configure the Application Context**

**Created 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"  
 xsi:schemaLocation="  
 http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd">  
  
 <!-- Repository Bean -->  
 <bean id="bookRepository" class="com.library.repository.BookRepository"/>  
  
 <!-- Service Bean -->  
 <bean id="bookService" class="com.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository"/>  
 </bean>  
  
</beans>

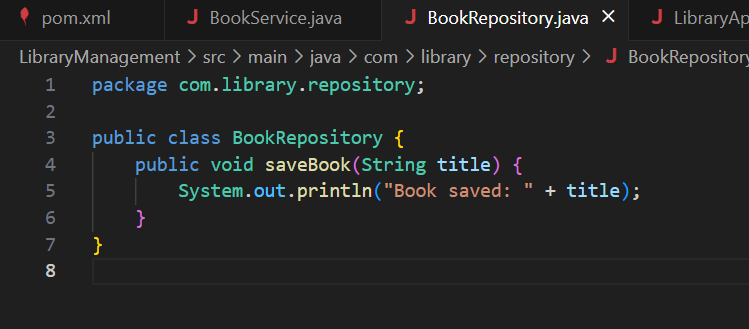
This configuration defines two beans and injects BookRepository into BookService.

****

## **Step 3: Define Service and Repository Classes**

### **File : com.library.repository.BookRepository**

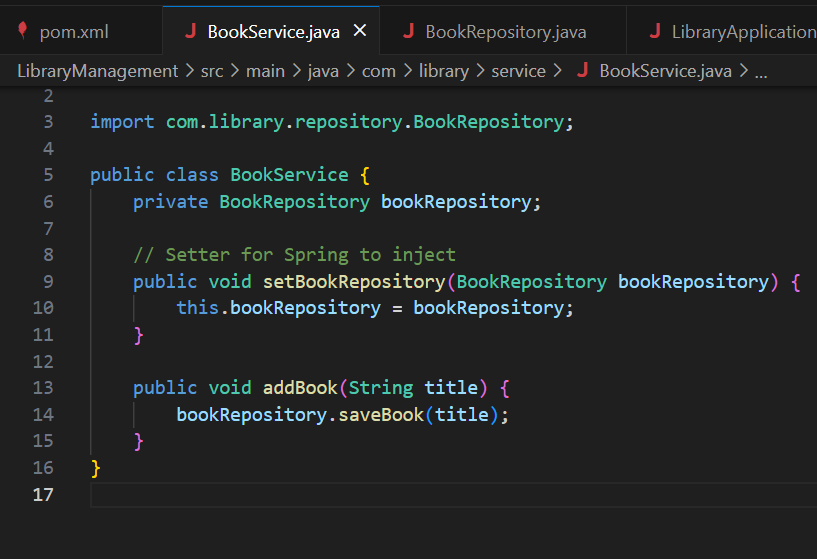
package com.library.repository;  
  
public class BookRepository {  
 public void saveBook(String title) {  
 System.out.println("Book saved: " + title);  
 }  
}



**file : com.library.service.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 title) {  
 bookRepository.saveBook(title);  
 }  
}

These classes demonstrate dependency injection in Spring.

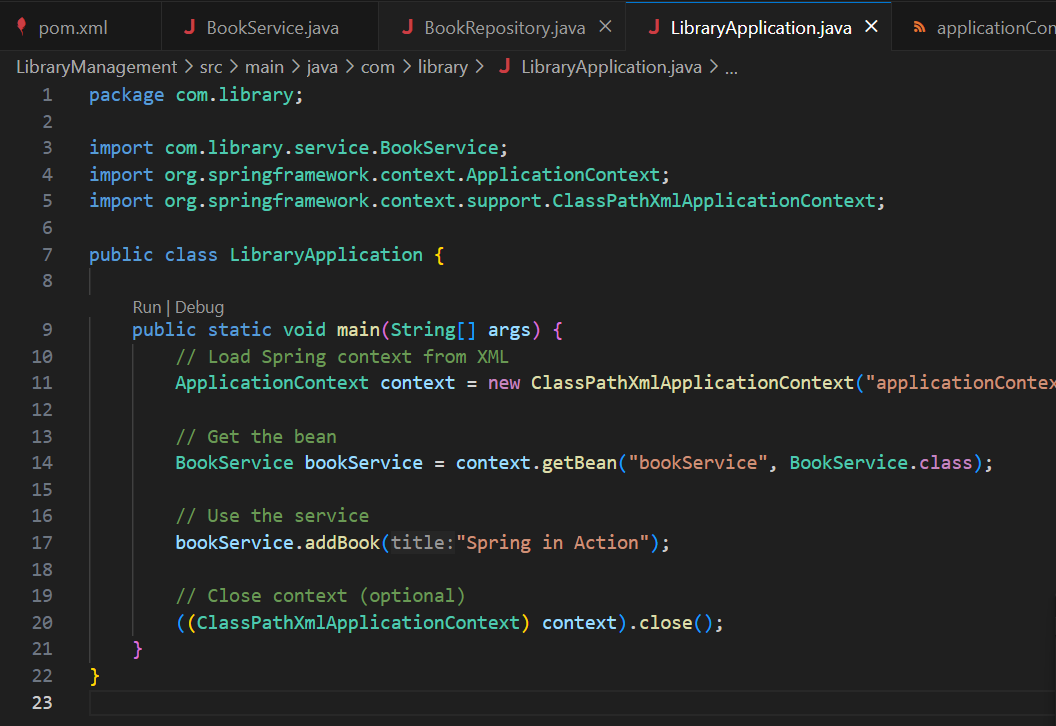


## **Step 4: Run the Application**

### **File : com.library.LibraryApplication**

package com.library;  
  
import com.library.service.BookService;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class LibraryApplication {  
 public static void main(String[] args) {  
  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
  
   
 BookService bookService = context.getBean("bookService", BookService.class);  
  
  
 bookService.addBook("Spring Framework Basics");  
  
 ((ClassPathXmlApplicationContext) context).close();  
 }  
}

This main class loads the XML config, gets the bean, and calls the service method.



## **How to Run**

In VS Code terminal:

mvn compile  
mvn exec:java

**output:**

Book saved: Spring Framework Basics

