**Assignment Documentation: Spring Core + Maven (Library Management System)**

**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.

**Step 1: Maven Project Setup**

**Project Name:** LibraryManagement

**pom.xml:**

<!-- Spring Core Dependency -->

<dependency>

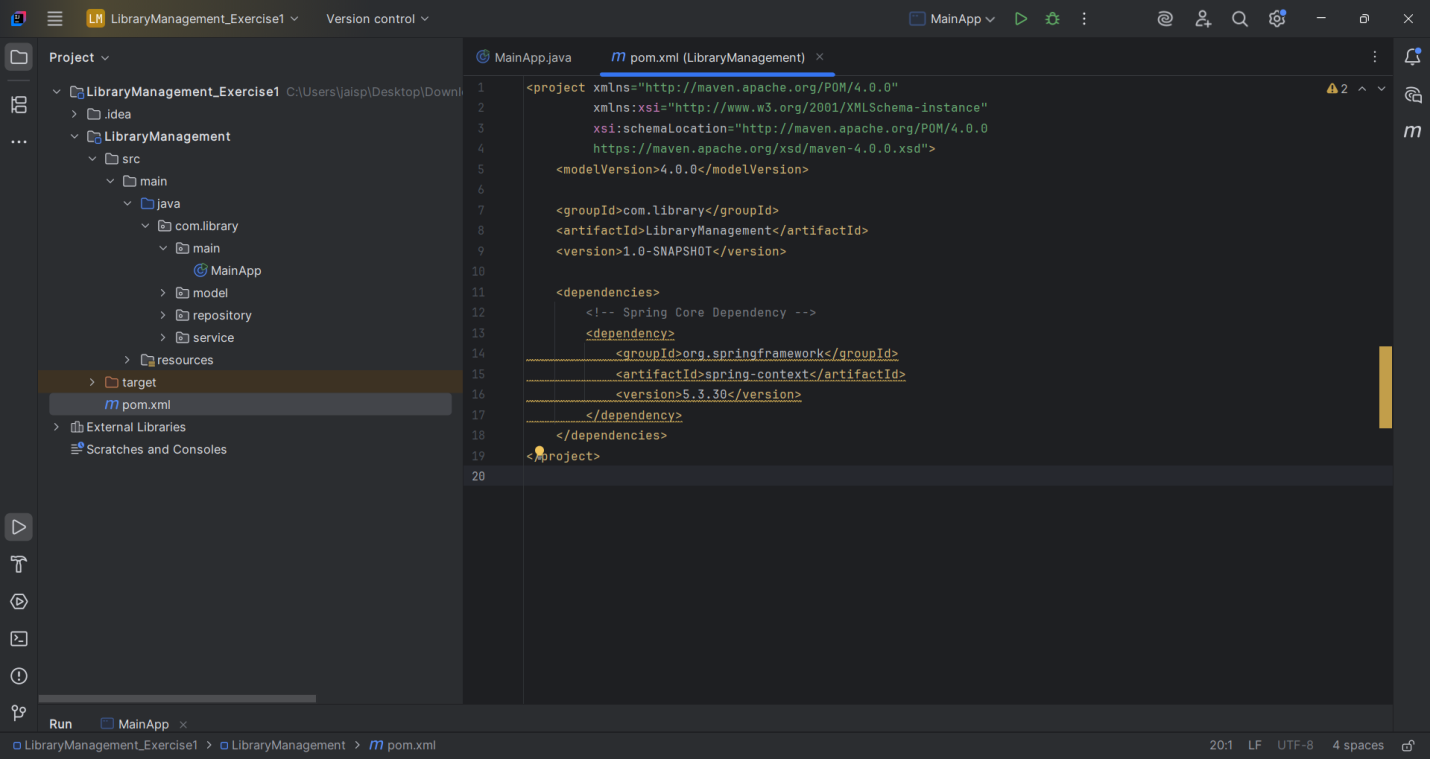
<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.30</version>

</dependency>

**Screenshot: Maven Project Structure in IDE**



**Step 2: XML Configuration - applicationContext.xml**

**File Location:** src/main/resources/applicationContext.xml

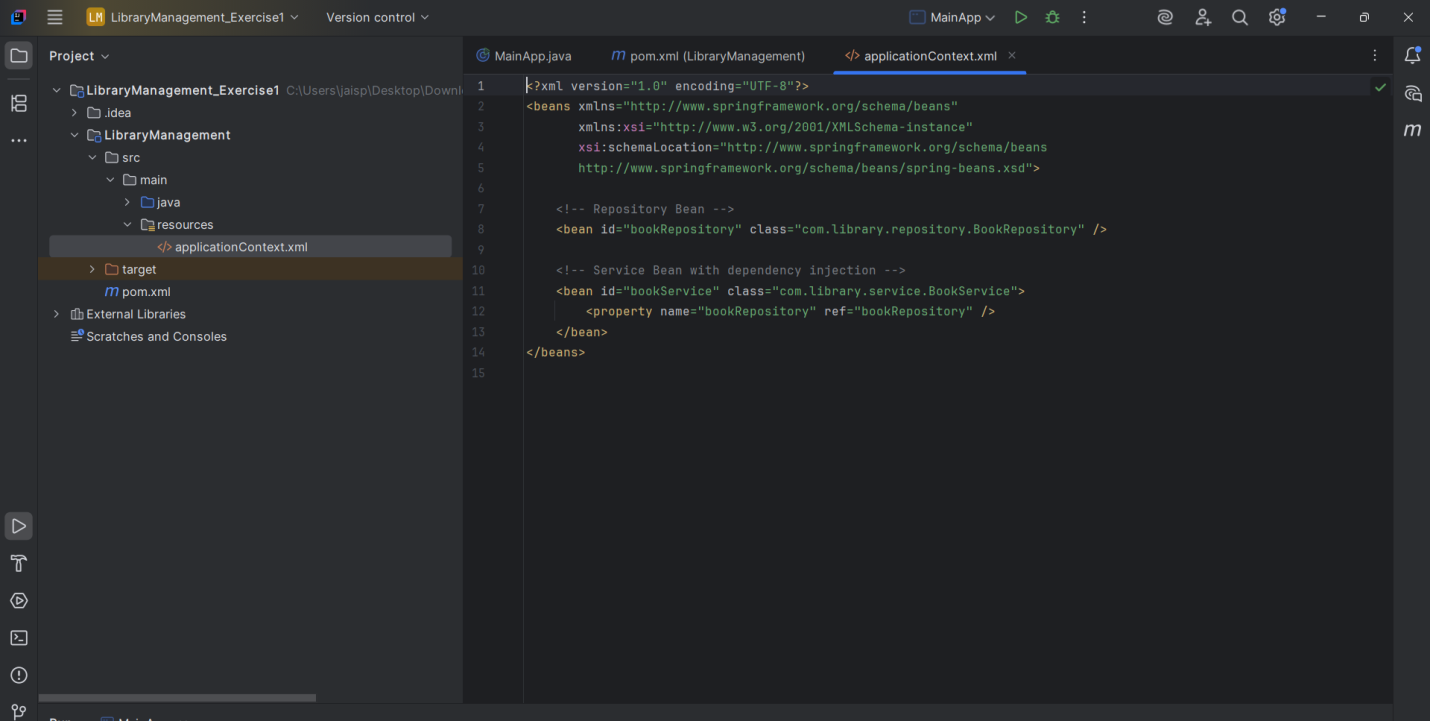
<bean id="bookRepository" class="com.library.repository.BookRepository" />

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

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

</bean>

**Screenshot: XML Configuration**



**Step 3: Java Classes**

**Book.java (Model)**

package com.library.model;

public class Book {

private int id;

private String title;

private String author;

public Book(int id, String title, String author) {

this.id = id;

this.title = title;

this.author = author;

}

public int getId() { return id; }

public String getTitle() { return title; }

public String getAuthor() { return author; }

@Override

public String toString() {

return "Book ID: " + id + ", Title: \"" + title + "\"" + ", Author: " + author;

}

}

**BookRepository.java**

package com.library.repository;

import com.library.model.Book;

public class BookRepository {

public Book findBookById(int id) {

if (id == 101) return new Book(101, "Effective Java", "Joshua Bloch");

else if (id == 102) return new Book(102, "Clean Code", "Robert C. Martin");

else return new Book(id, "Unknown Title", "Unknown Author");

}

}

**BookService.java**

package com.library.service;

import com.library.model.Book;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void displayBookDetails(int bookId) {

Book book = bookRepository.findBookById(bookId);

System.out.println("Book Details:");

System.out.println(book);

}

}

**MainApp.java**

package com.library.main;

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");

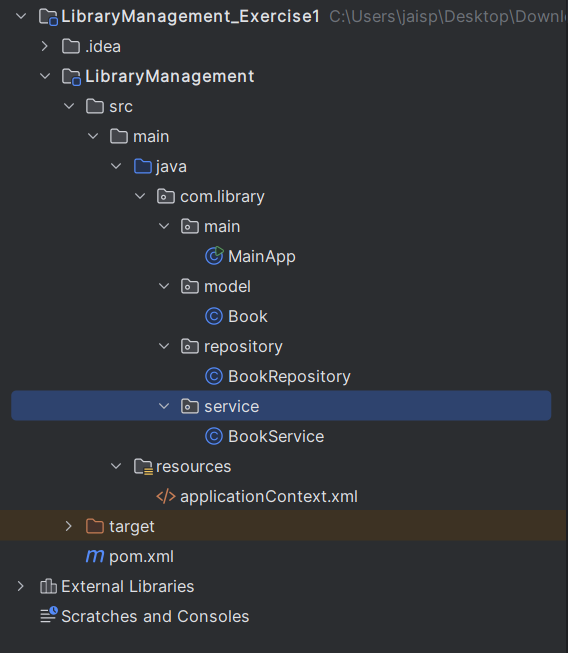
BookService bookService = context.getBean("bookService", BookService.class);

bookService.displayBookDetails(101);

}

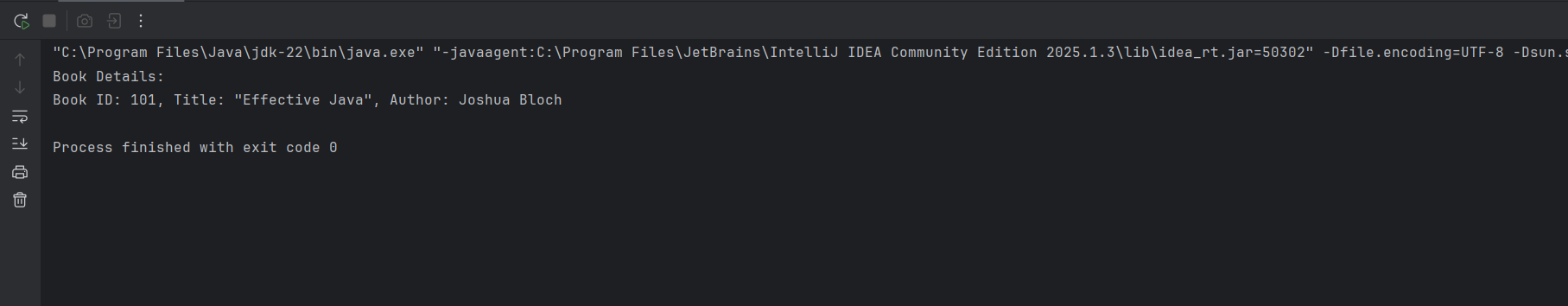
}

**Screenshot: Java Class Structure**



**Step 4: Output**

**Screenshot: Console Output**



**Final Project Structure**

LibraryManagement/

├── pom.xml

└── src/

└── main/

├── java/

│ ├── com/library/model/Book.java

│ ├── com/library/repository/BookRepository.java

│ ├── com/library/service/BookService.java

│ └── com/library/main/MainApp.java

└── resources/

└── applicationContext.xml

**Conclusion**

All assignment requirements have been satisfied:

* Maven and Spring Core setup
* XML-based Bean Configuration
* Functional Service and Repository classes

The project is clean, functional, and extensible.