**SPRING CORE AND MAVEN**

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

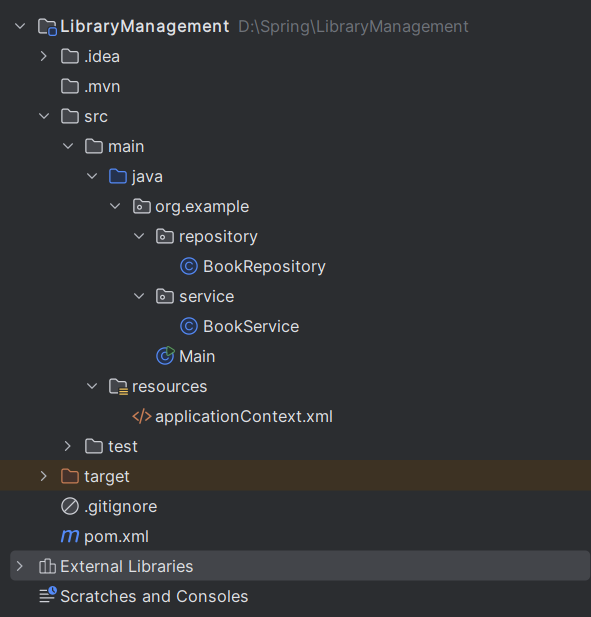
**Scenario**

Your company is developing a web application for managing a library. You are tasked with setting up a basic Spring Framework application to handle the backend operations such as managing books, authors, and users.

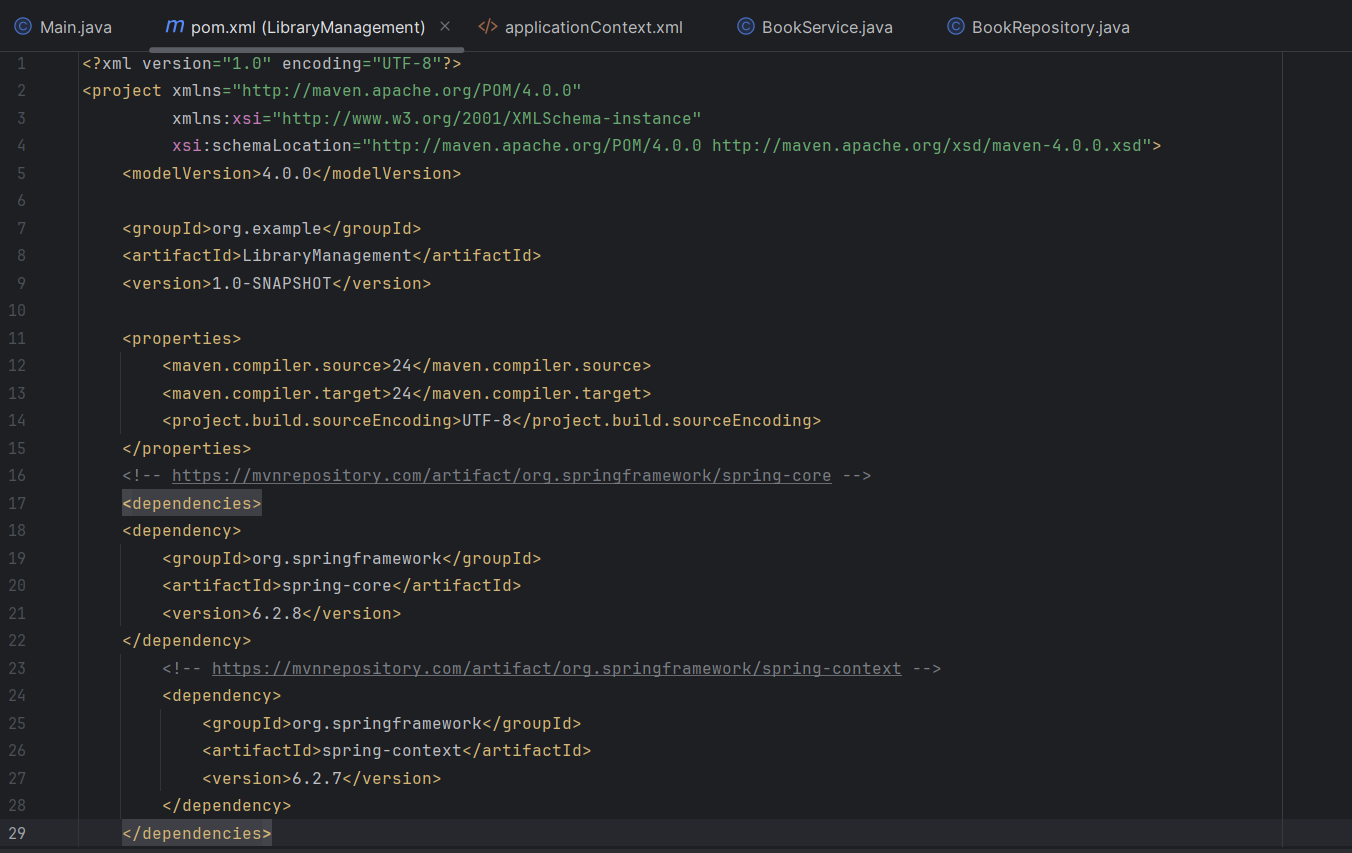
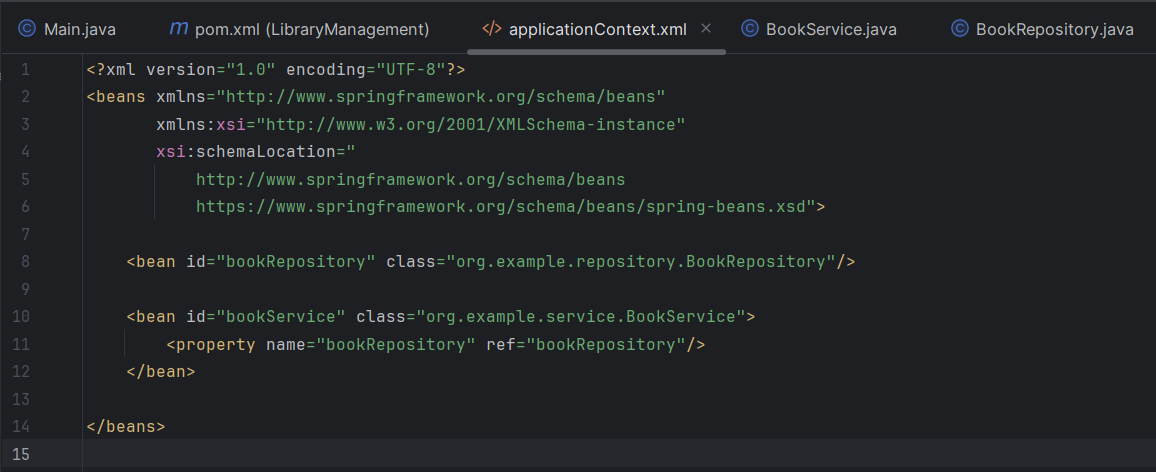
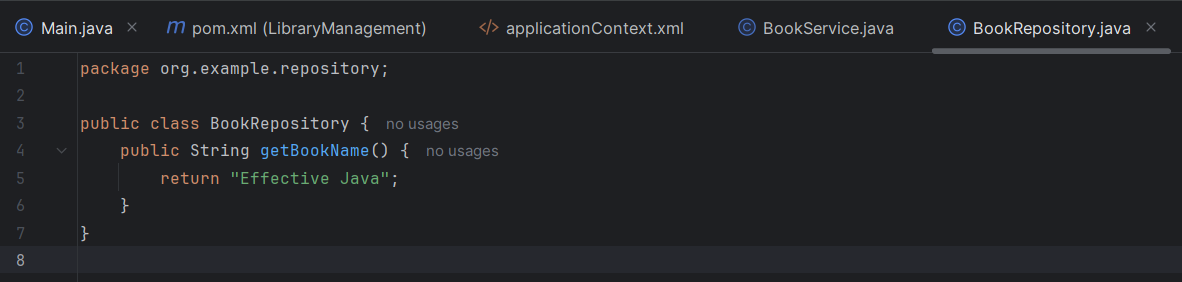
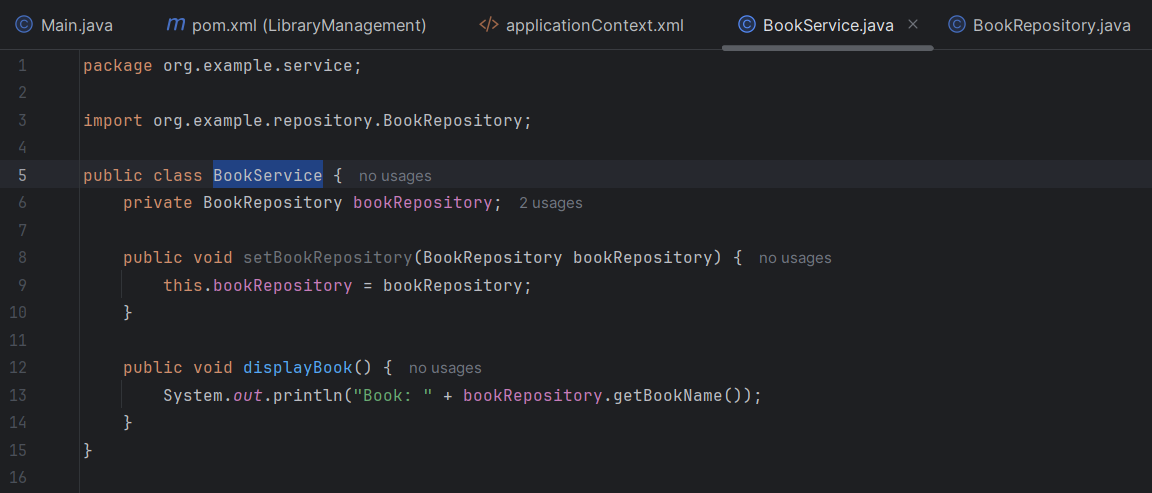
**Objective**

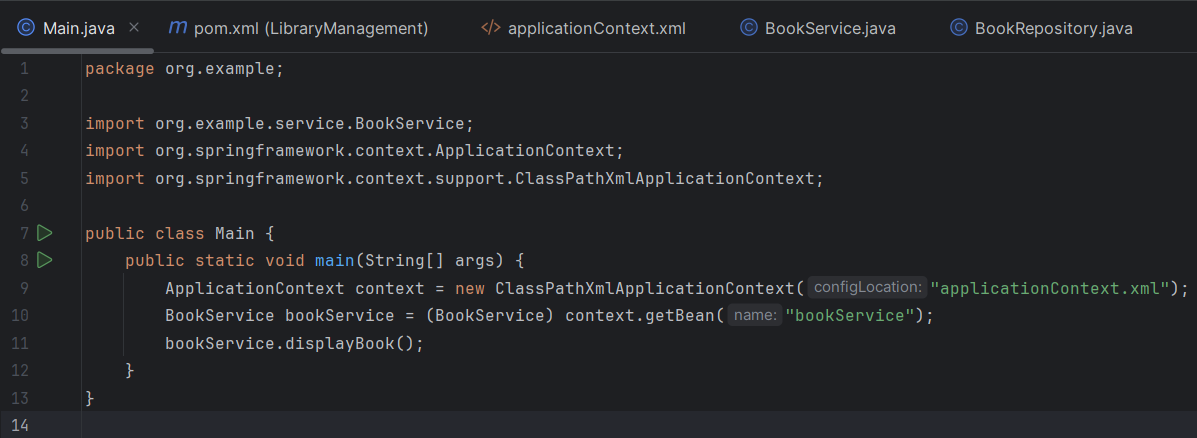
Configure a simple Spring Boot application that sets up the basic backend structure for the Library Management System.

**Project Structure**

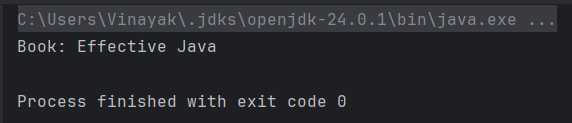
****

***Code Snippets***

1. ***pom.xml***
2. ***applicationContext.xml***
3. ***BookRepository.java***
4. ***BookService.java***
5. ***Main.java***

****

***Output Screenshot***

****

**Exercise 2: Implementing Dependency Injection**

**Scenario**

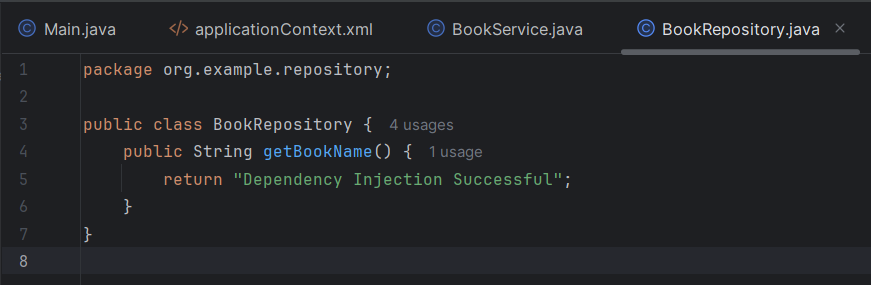
In the Library Management application, you need to manage dependencies between the BookService and BookRepository classes using Spring’s Inversion of Control (IoC) and Dependency Injection (DI) mechanisms.

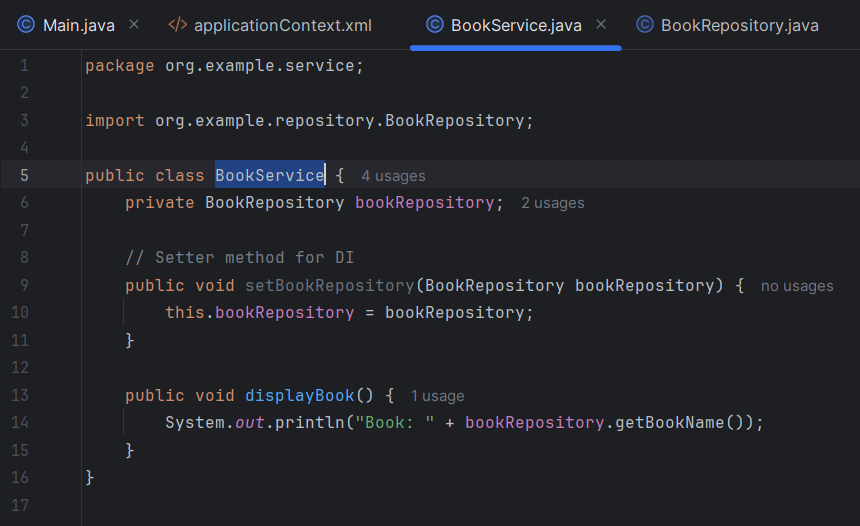
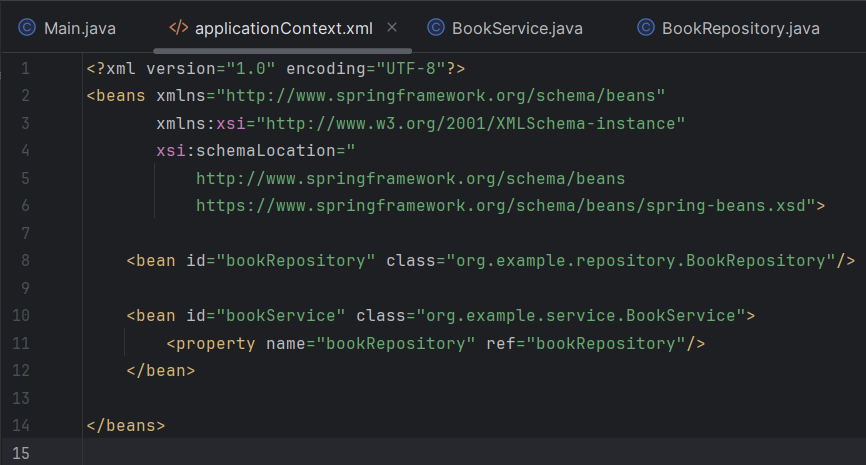
**Objective**

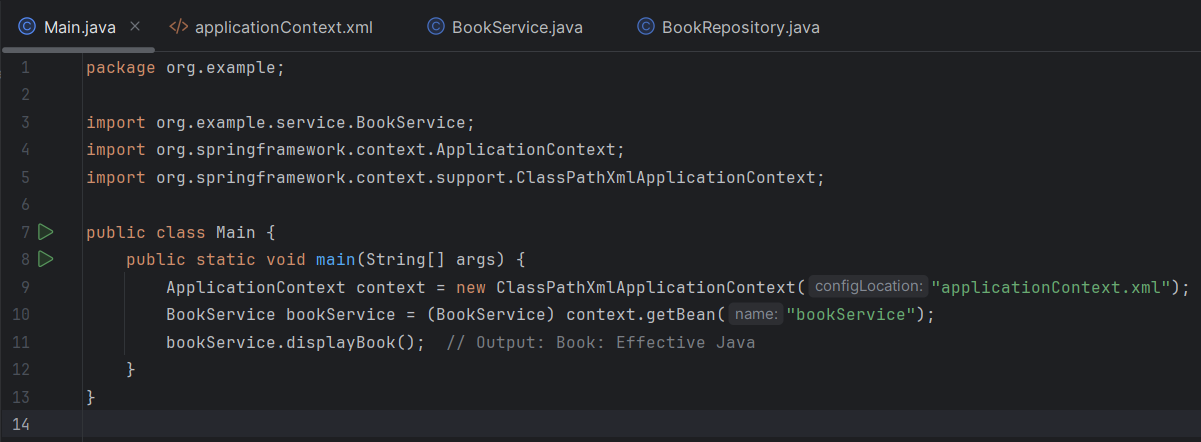
Use Spring's XML-based configuration to inject BookRepository into BookService using setter-based dependency injection.

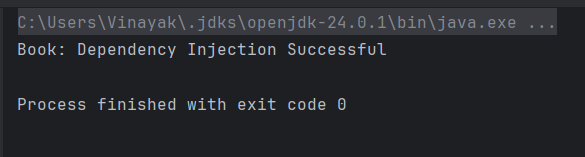
**Code Snippets**

1. ***BookRepository***

****

1. ***BookService***
2. ***applicationContext.xml***
3. ***Main.java***

****

***Output Screenshot***

**Exercise 4: Creating and Configuring a Maven Project**

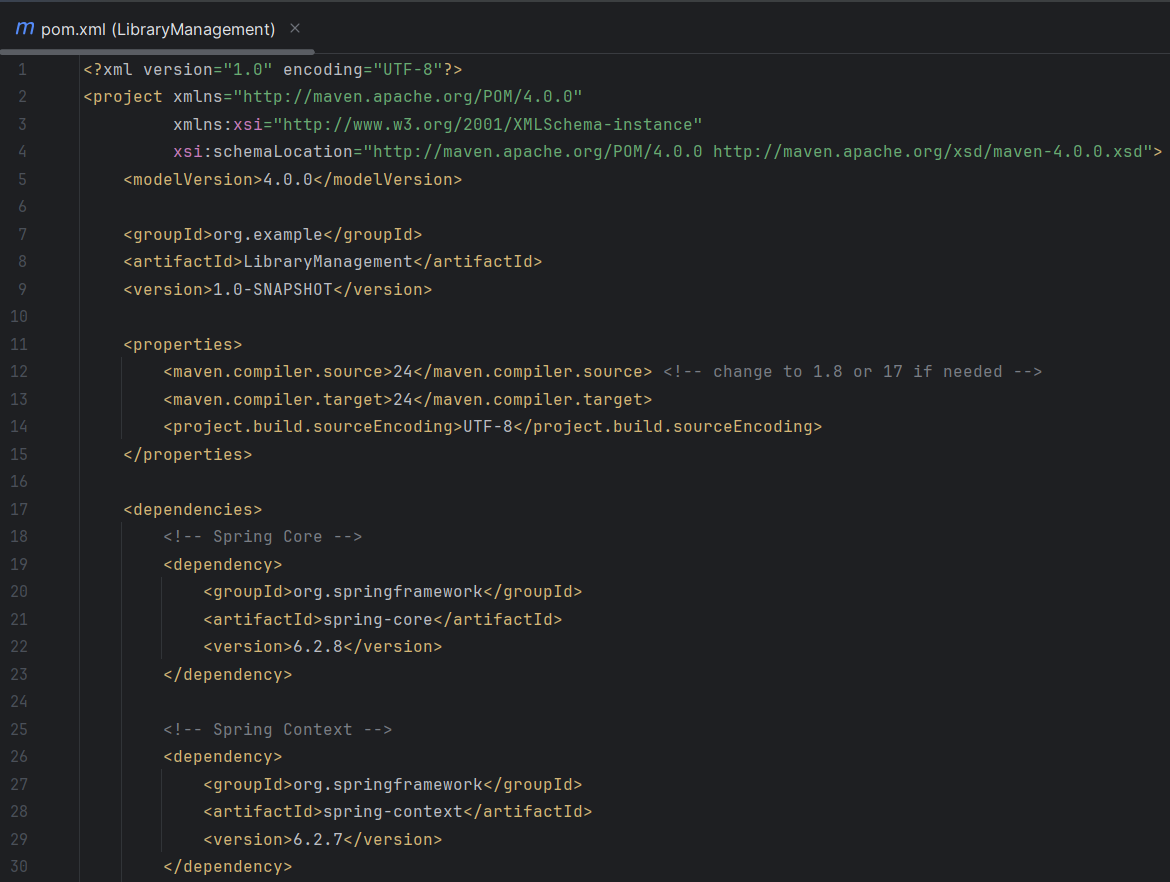
**Scenario**

You are required to set up a new Maven project for the Library Management application and configure it with the necessary Spring dependencies and Maven plugins.

**Objective**

Create a Maven-based Java project and configure it with Spring dependencies and build plugins using pom.xml.

**Code Snippets**

***pom.xml file with dependencies and plugins***

****

