**Digital NUTURE 4.0 – DEEP SKILLING STAGE**

Spring Core\_Maven

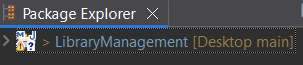
**Exercise 1: Configuring a Basic Spring Application**

**Scenario:**

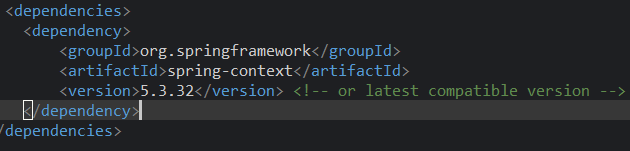
Your company is developing a web application for managing a library. You need to use the Spring Framework to handle the backend operations.

**Steps:**

1. **Set Up a Spring Project:**
   * Create a Maven project named **LibraryManagement**.



* + Add Spring Core dependencies in the **pom.xml** file.



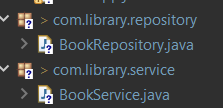
1. **Configure the Application Context:**
   * Create an XML configuration file named **applicationContext.xml** in the **src/main/resources** directory.



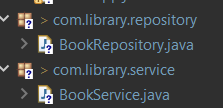
* + Define beans for **BookService** and **BookRepository** in the XML file.



1. **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**.

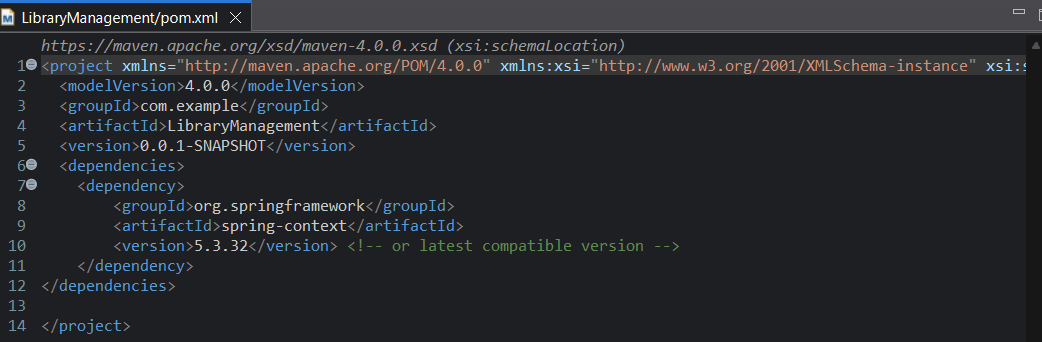


1. **Run the Application:**
   * Create a main class to load the Spring context and test the configuration.

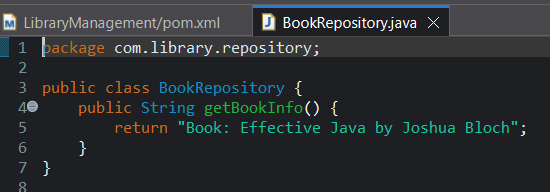


**Complete code**

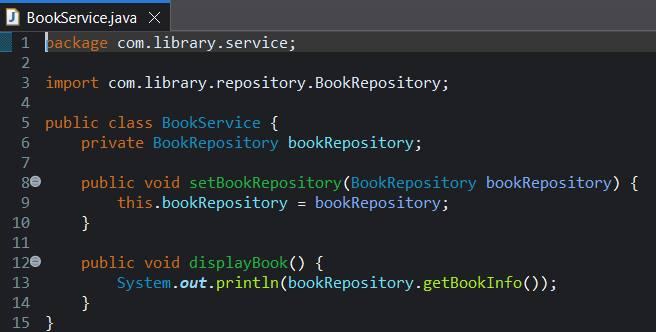
**Pox.xml**



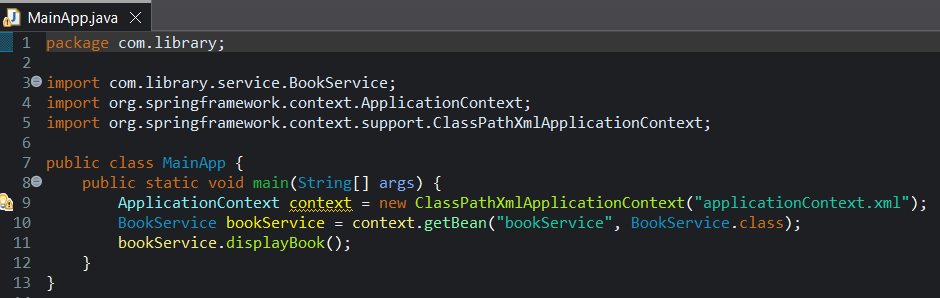
**BookRepository.java**



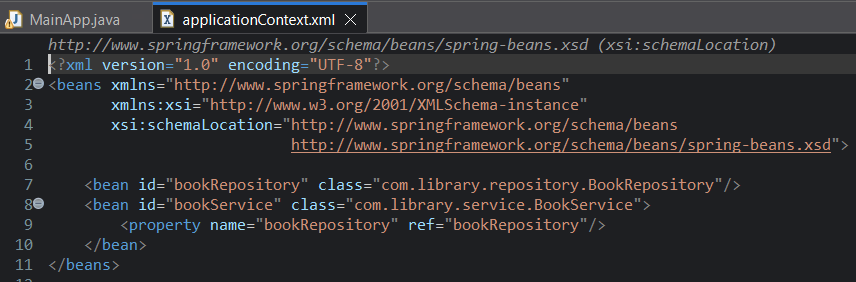
**BookService.java**



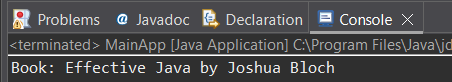
**MainApp.java**



**ApplicationContex.xml**



**Output**



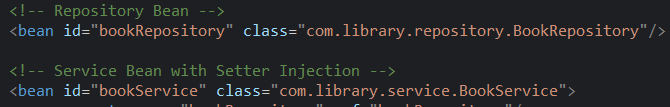
**Exercise 2: Implementing Dependency Injection**

**Scenario:**

In the library management application, you need to manage the dependencies between the BookService and BookRepository classes using Spring's IoC and DI.

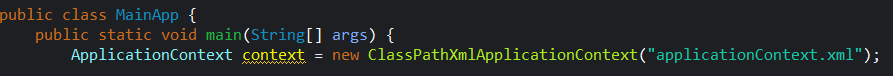
**Steps:**

1. **Modify the XML Configuration:**
   * Update **applicationContext.xml** to wire **BookRepository** into **BookService**.



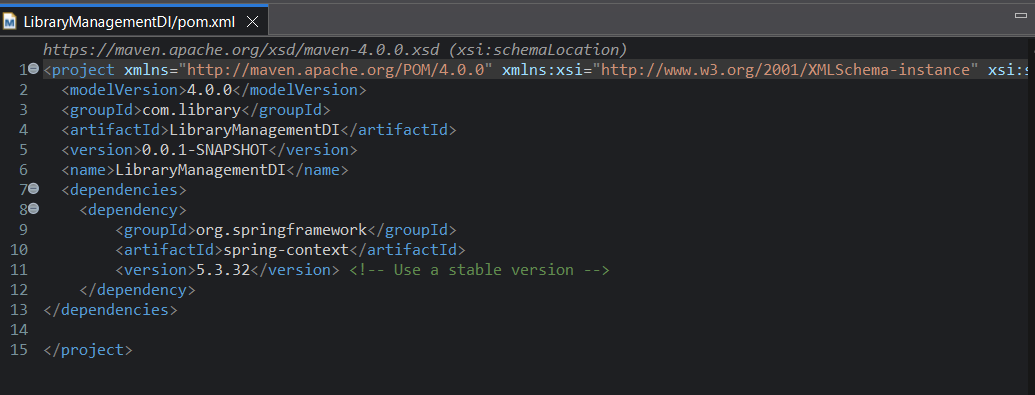
1. **Update the BookService Class:**
   * Ensure that **BookService** class has a setter method for **BookRepository**.



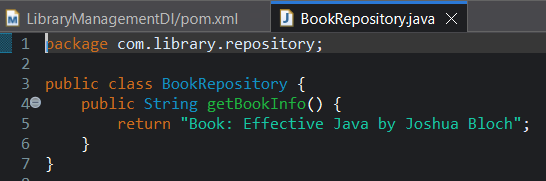
1. **Test the Configuration:**
   * Run the **LibraryManagementApplication** main class to verify the dependency injection.****

**Complete code**

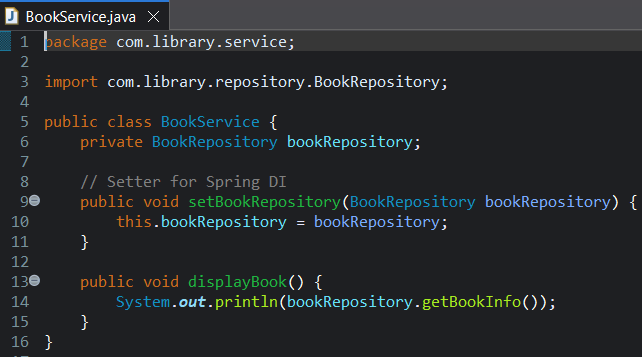
**Pox.xml**

****

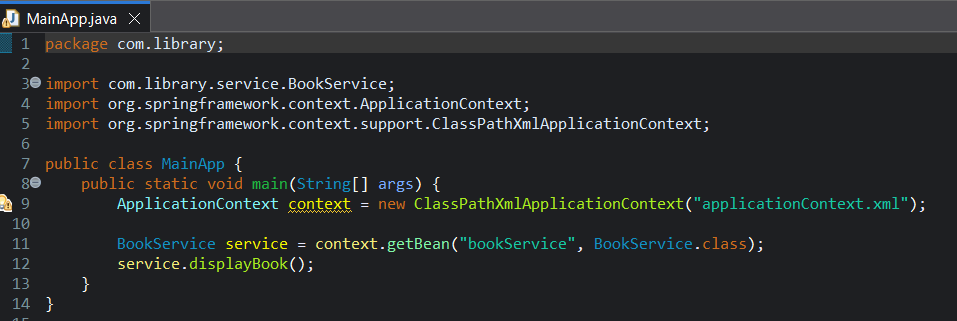
**BookRepository.java**

****

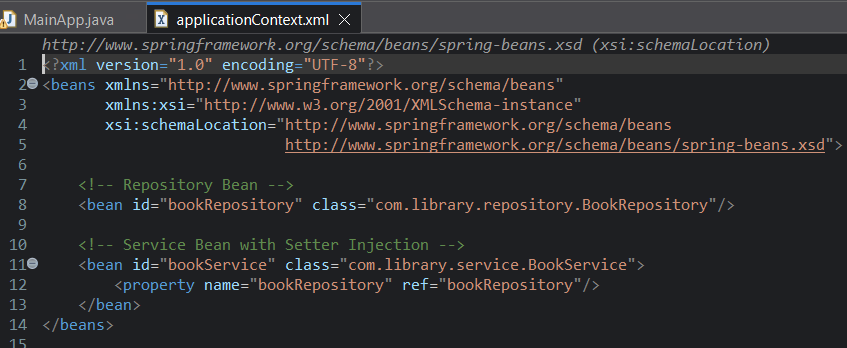
**BookService.java**



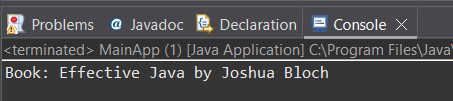
**MainApp.java**



**applicationContent.xml**



**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:**

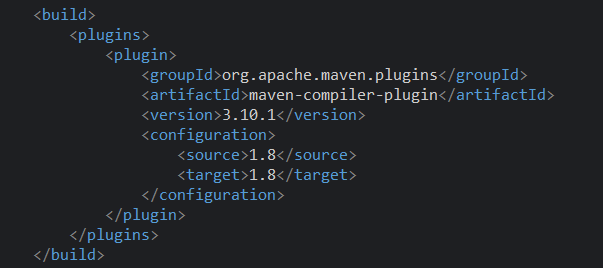
1. **Create a New Maven Project:**
   * Create a new Maven project named **LibraryManagement**.



1. **Add Spring Dependencies in pom.xml:**
   * Include dependencies for Spring Context, Spring AOP, and Spring WebMVC.

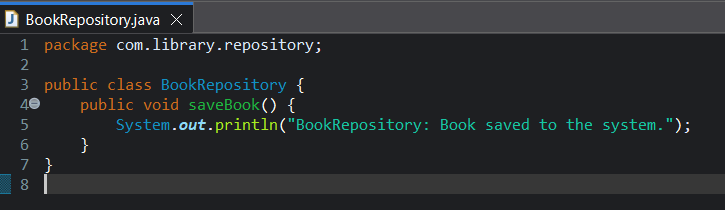


1. **Configure Maven Plugins:**
   * Configure the Maven Compiler Plugin for Java version 1.8 in the pom.xml file.

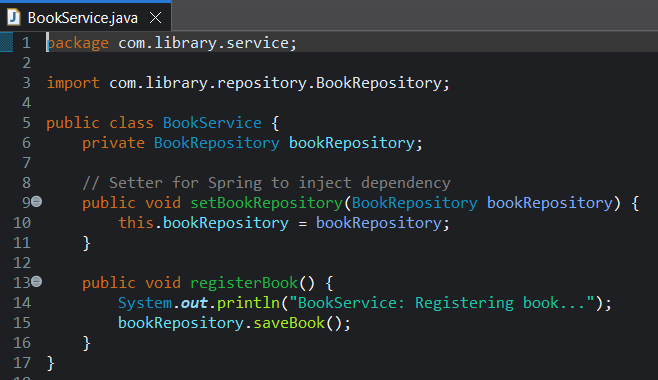


**Complete code**

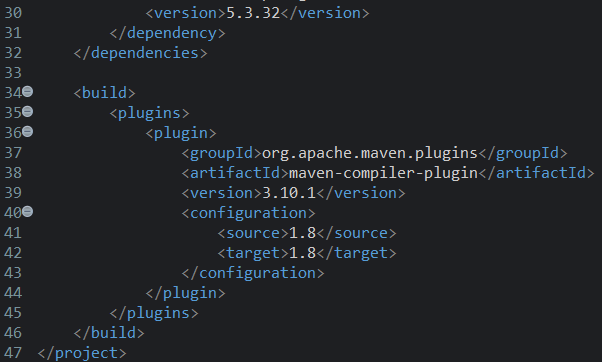
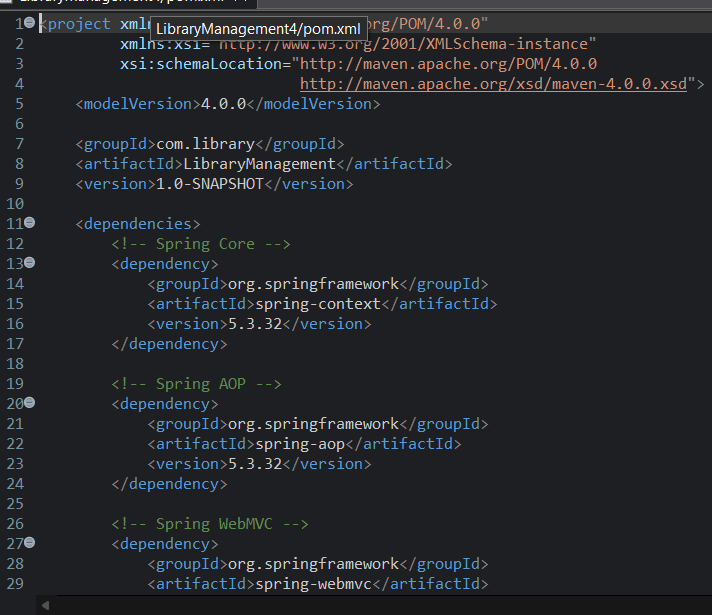
**BookRepository.java**



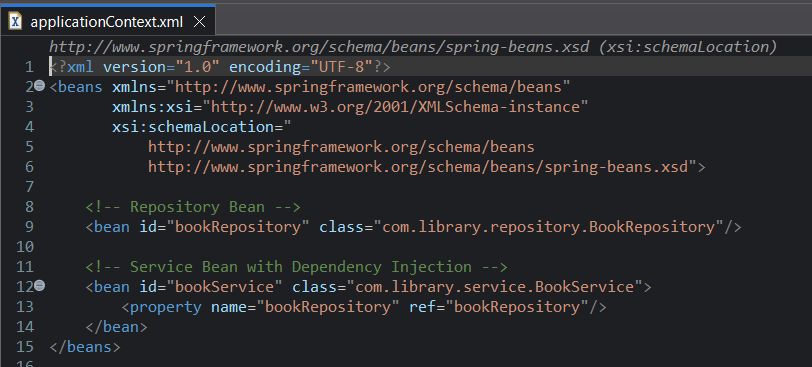
**BookService.java**



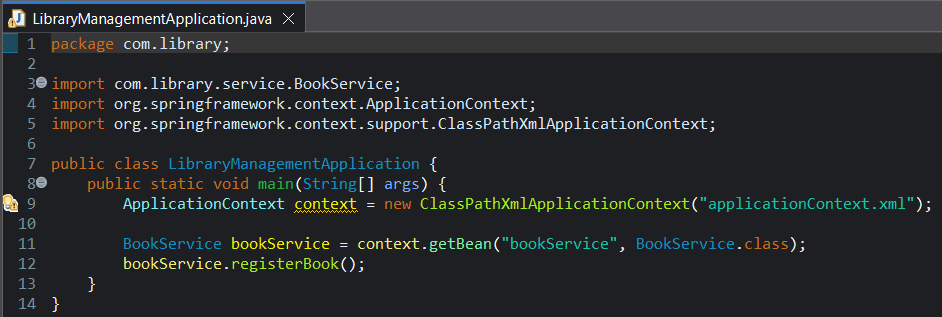
**Pom.xml**



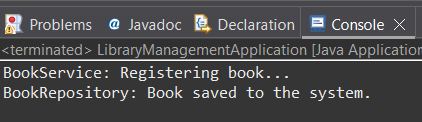
**ApplicationContext.xml**



**LibraryManagementAppliccation.java**



**Output**



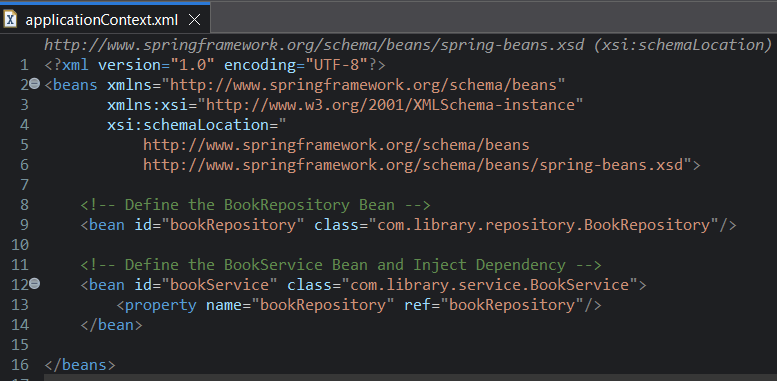
**Exercise 5: Configuring the Spring IoC Container**

**Scenario:**

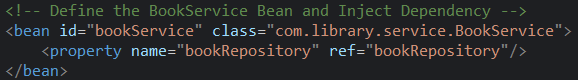
The library management application requires a central configuration for beans and dependencies.

**Steps:**

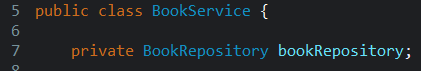
1. **Create Spring Configuration File:**
   * Create an XML configuration file named **applicationContext.xml** in the **src/main/resources** directory.



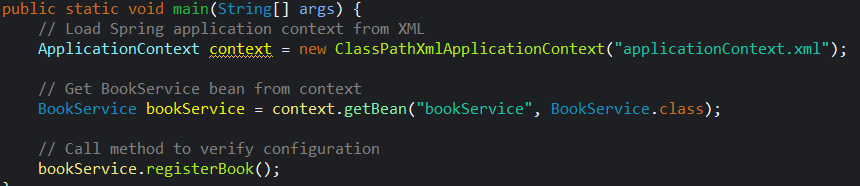
* + Define beans for **BookService** and **BookRepository** in the XML file.



1. **Update the BookService Class:**
   * Ensure that the **BookService** class has a setter method for **BookRepository**.

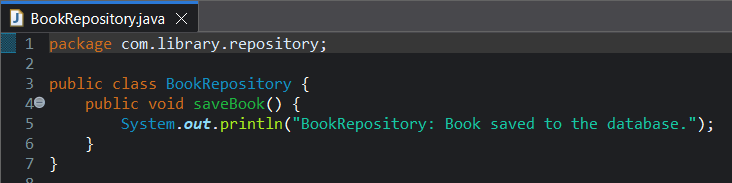


1. **Run the Application:**
   * Create a main class to load the Spring context and test the configuration.

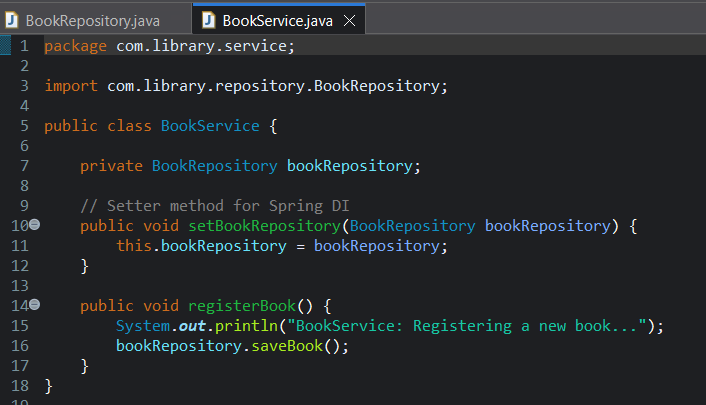


**Complete code**

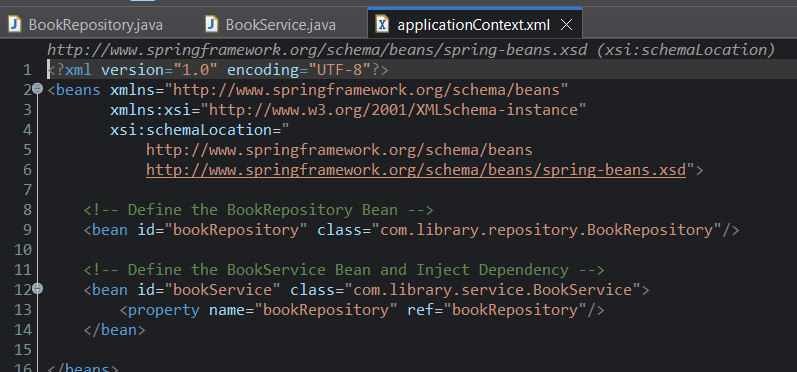
**BookRepository.java**



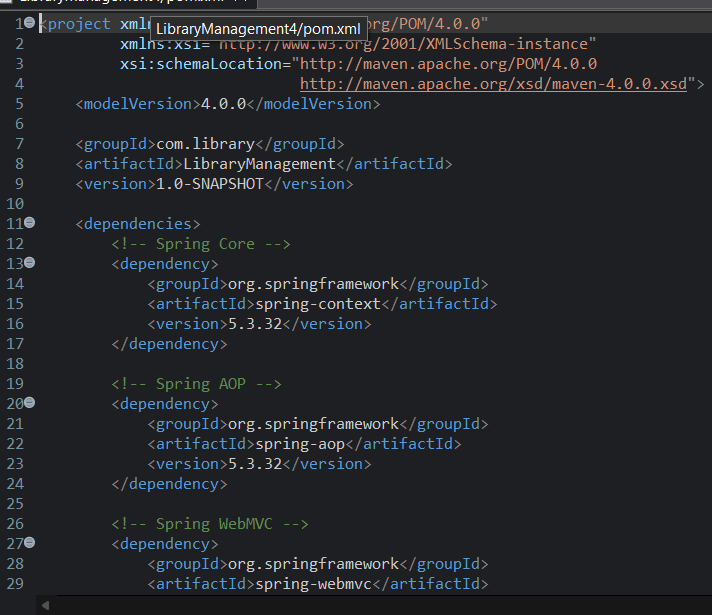
**BookService.java**

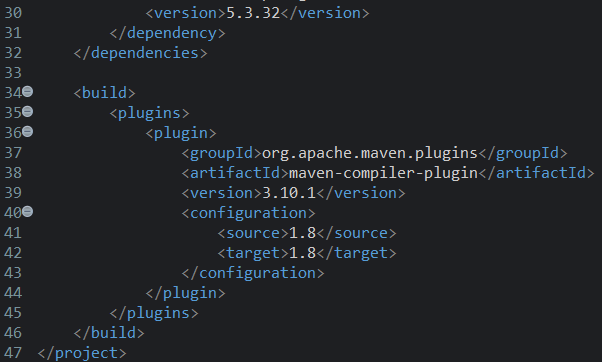


**applicationContext.xml**

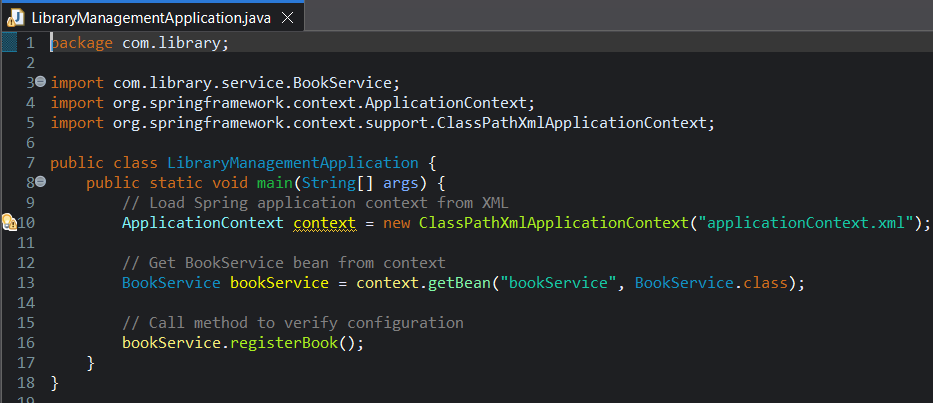


**Pom.xml**

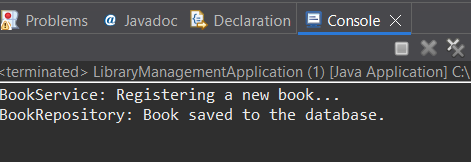




**LibraryManagementApplication.java**



**Output**



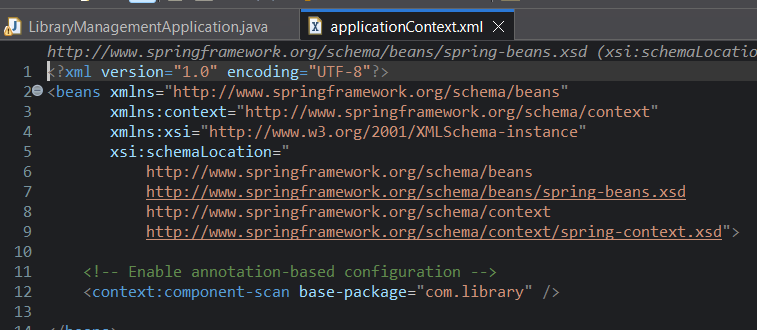
**Exercise 6: Configuring Beans with Annotations**

**Scenario:**

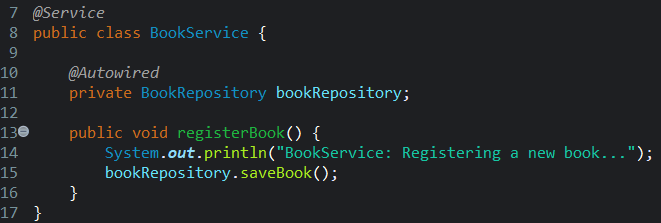
You need to simplify the configuration of beans in the library management application using annotations.

**Steps:**

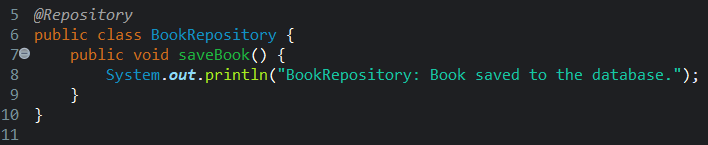
1. **Enable Component Scanning:**
   * Update **applicationContext.xml** to include component scanning for the **com.library** package.

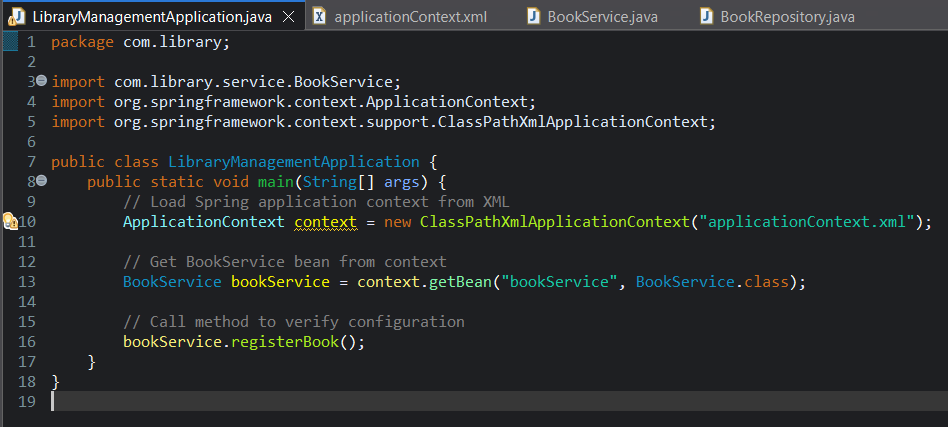


1. **Annotate Classes:**
   * Use **@Service** annotation for the **BookService** class.



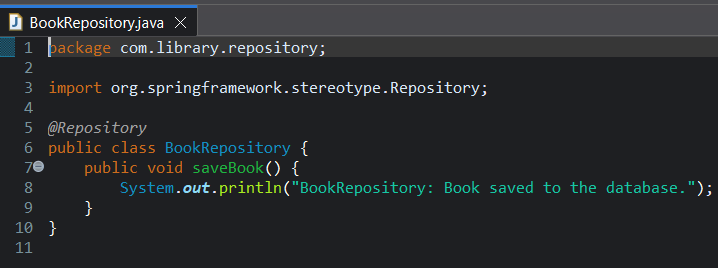
* + Use **@Repository** annotation for the **BookRepository** class.



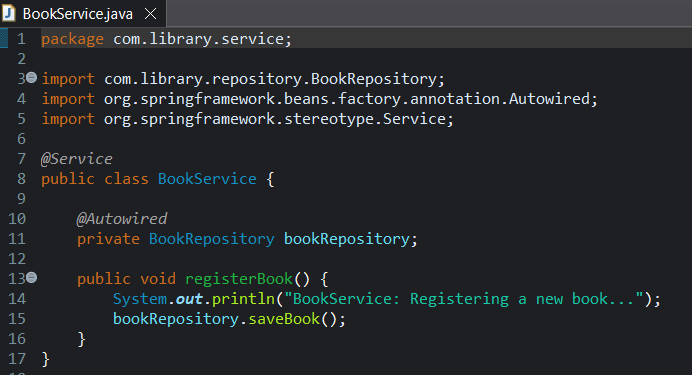
1. **Test the Configuration:**
   * Run the **LibraryManagementApplication** main class to verify the annotation-based configuration.

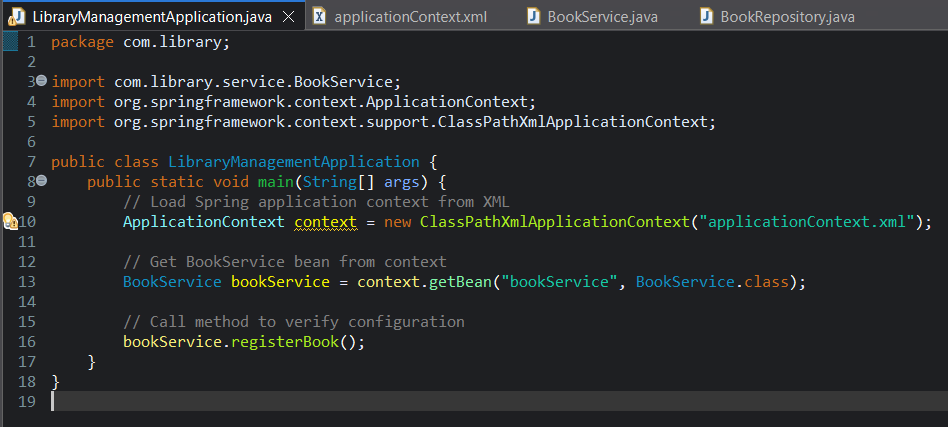
**Complete Code**

**BookRepository.java**

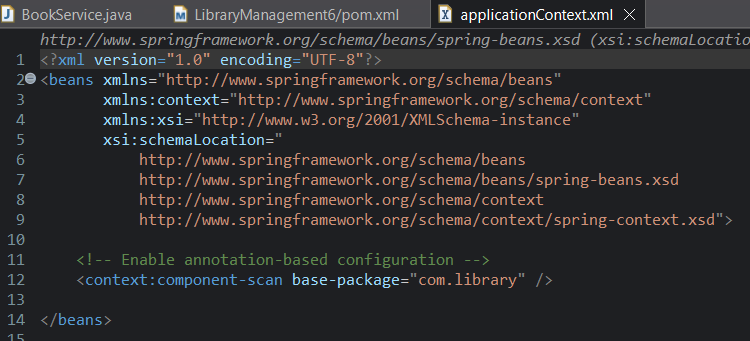
****

**BookService.java**

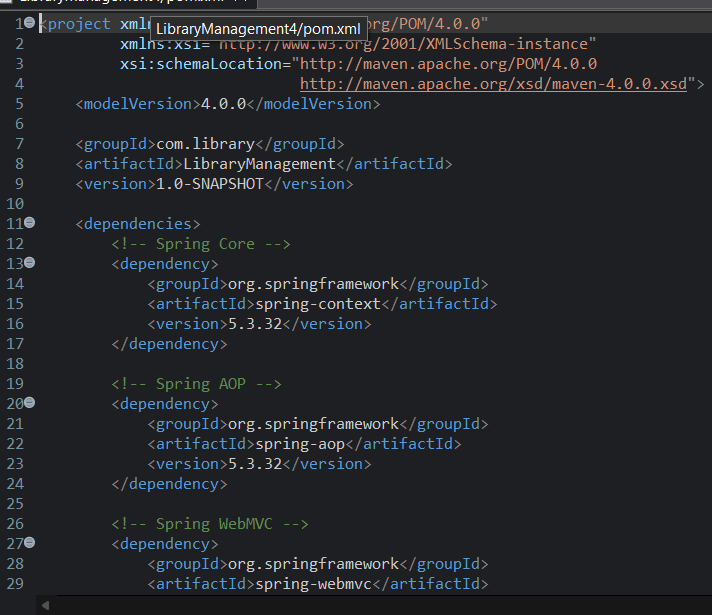
****

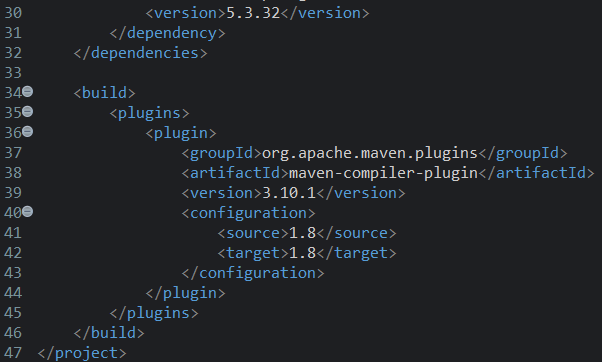
**LibraryManagementApplication.java**

**applicationContext.xml**

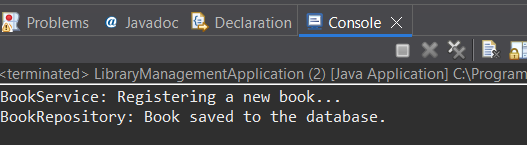
****

**Pom.xml**





**Output**

****

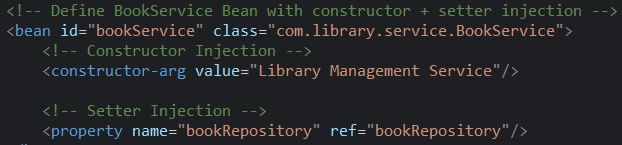
**Exercise 7: Implementing Constructor and Setter Injection**

**Scenario:**

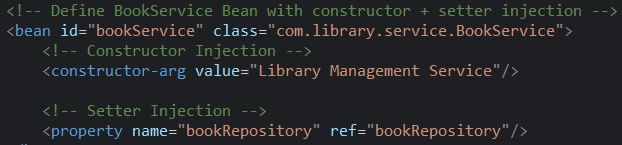
The library management application requires both constructor and setter injection for better control over bean initialization.

**Steps:**

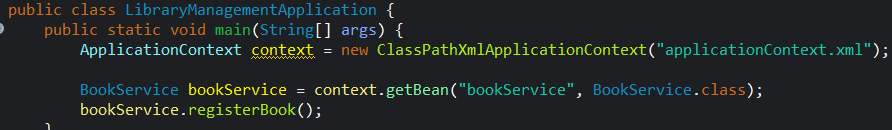
1. **Configure Constructor Injection:**
   * Update applicationContext.**xml** to configure constructor injection for **BookService**.



1. **Configure Setter Injection:**
   * Ensure that the **BookService** class has a setter method for **BookRepository** and configure it in **applicationContext.xml**.

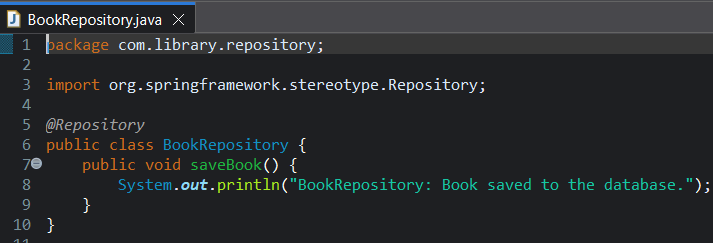


1. **Test the Injection:**
   * Run the **LibraryManagementApplication** main class to verify both constructor and setter injection.

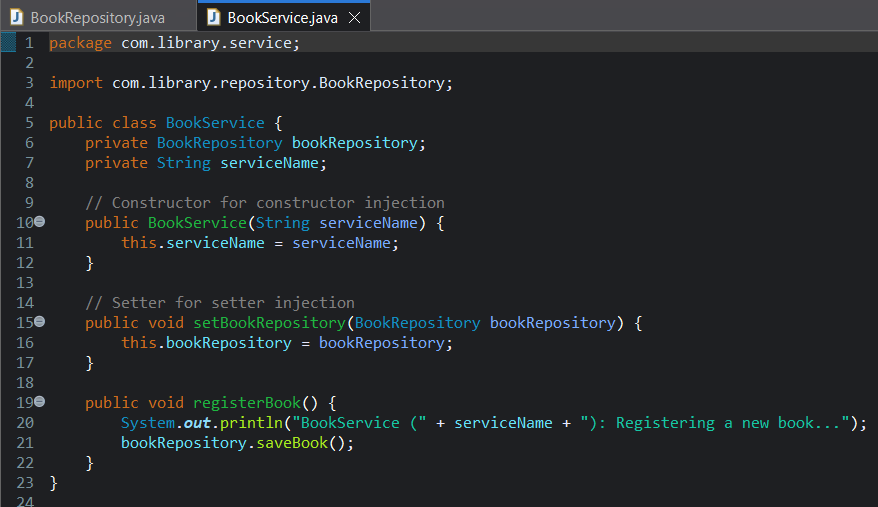


**Complete Code**

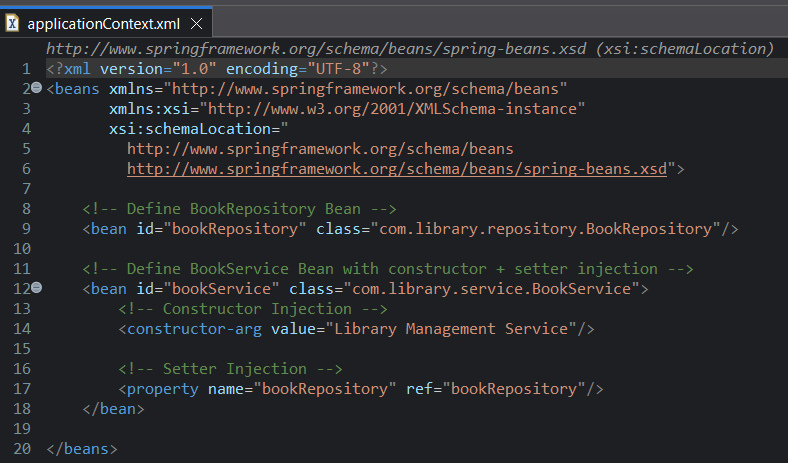
**BookRepository.java**

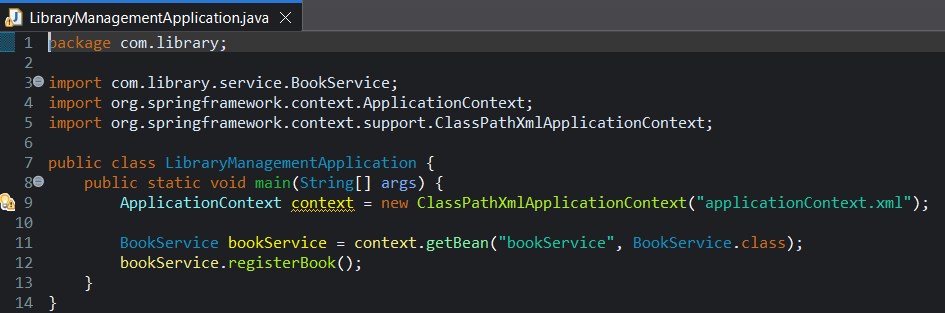


**BookServices.java**

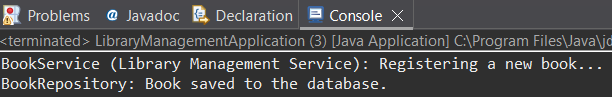


**applicationContext.java**



**LibraryManagementApplication.java**

**Output**



**Exercise 9: Creating a Spring Boot Application**

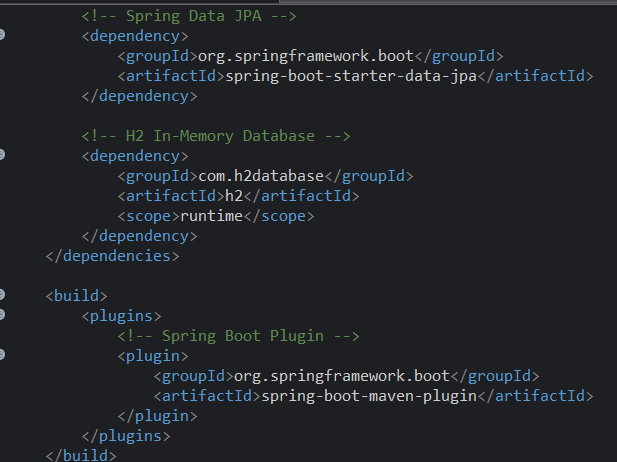
**Scenario:**

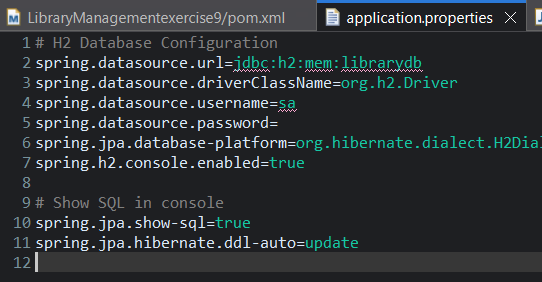
You need to create a Spring Boot application for the library management system to simplify configuration and deployment.

**Steps:**

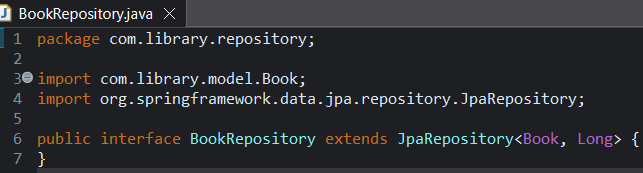
1. **Create a Spring Boot Project:**
   * Use **Spring Initializr** to create a new Spring Boot project named **LibraryManagement**.



1. **Add Dependencies:**
   * Include dependencies for **Spring Web, Spring Data JPA, and H2 Database**. 
2. **Create Application Properties:**
   * Configure database connection properties in **application.properties**.



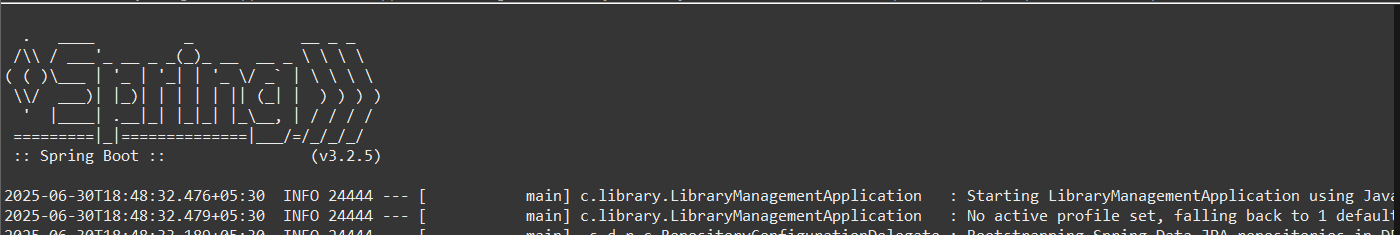
1. **Define Entities and Repositories:**
   * Create **Book** entity and **BookRepository** interface.



1. **Create a REST Controller:**
   * Create a **BookController** class to handle CRUD operations.

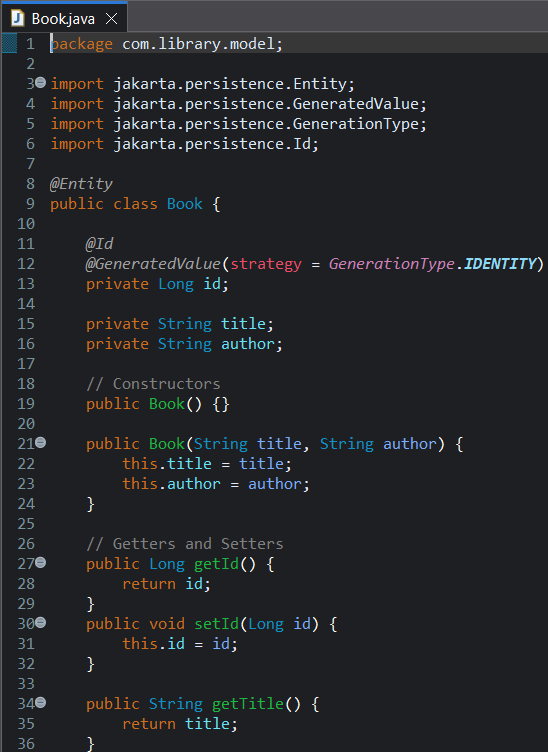
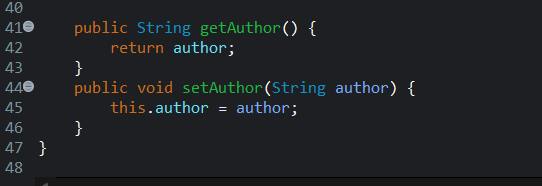


1. **Run the Application:**
   * Run the Spring Boot application and test the REST endpoints.

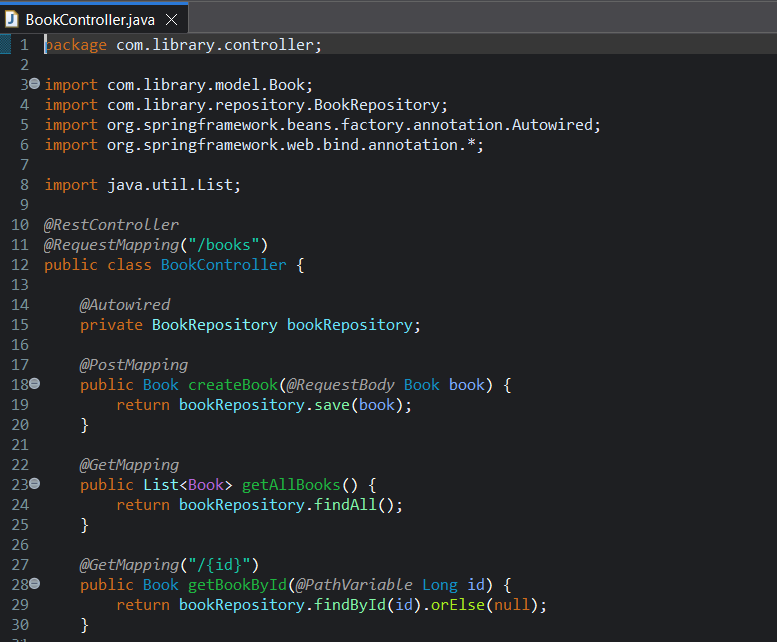


**Complete code**

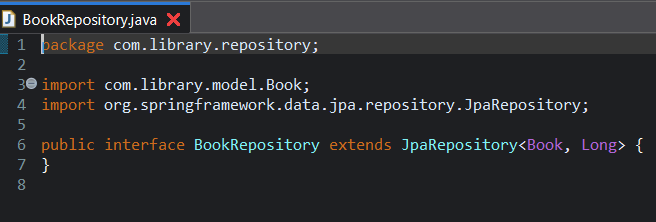
**Book.java**

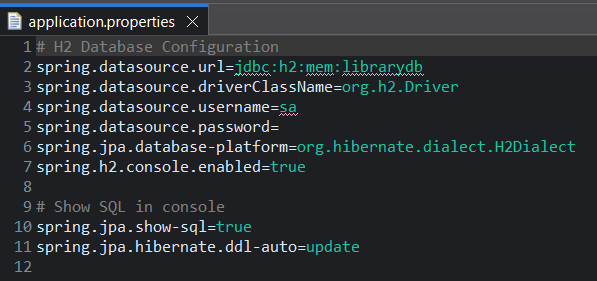
**BookController.java**

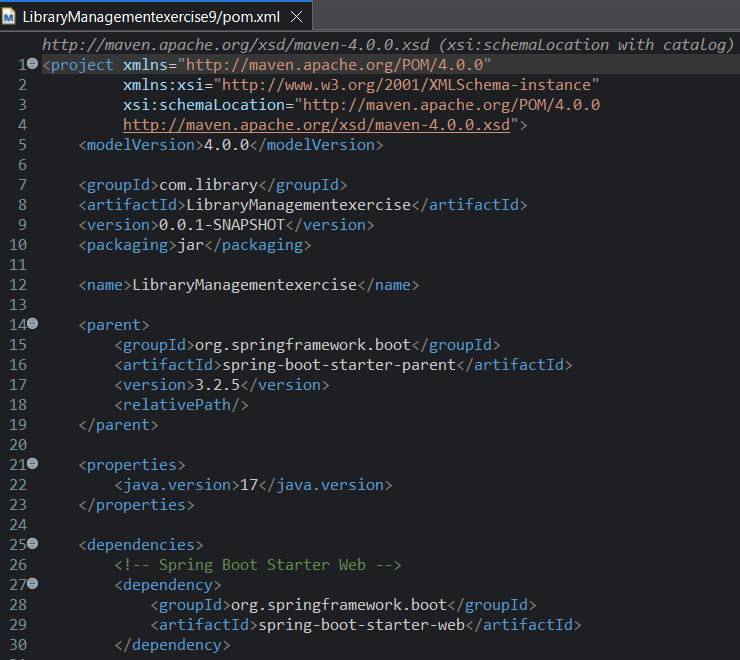
**BookRepository.java**



**applicationProperties**



**Pom.xml**





**Output**

