**Cognizant Digital Nurture 4.0**

***WEEK-2 Module 5 – Spring Core\_Maven***

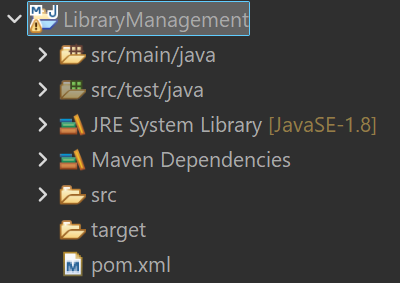
**Exercise 1: Configuring a Basic Spring Application (Mandatory)**

**Scenario:**

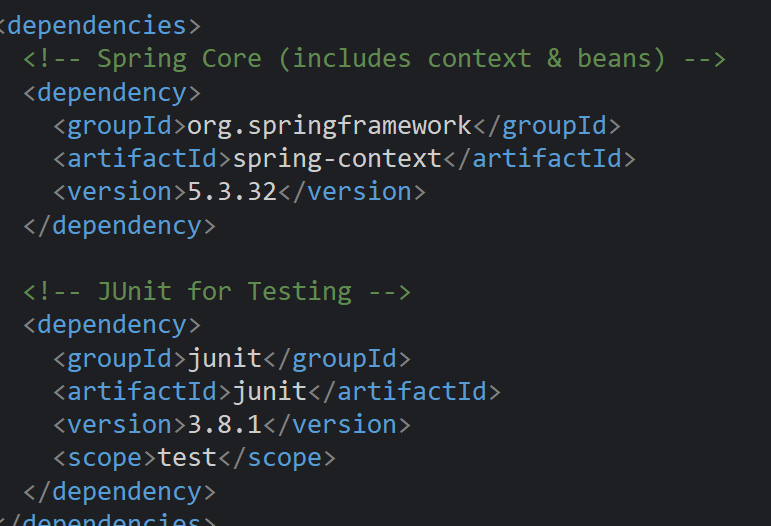
Your company is developing a web application for managing a library. You need to use the Spring Framework to handle the backend operations.

**Steps:**

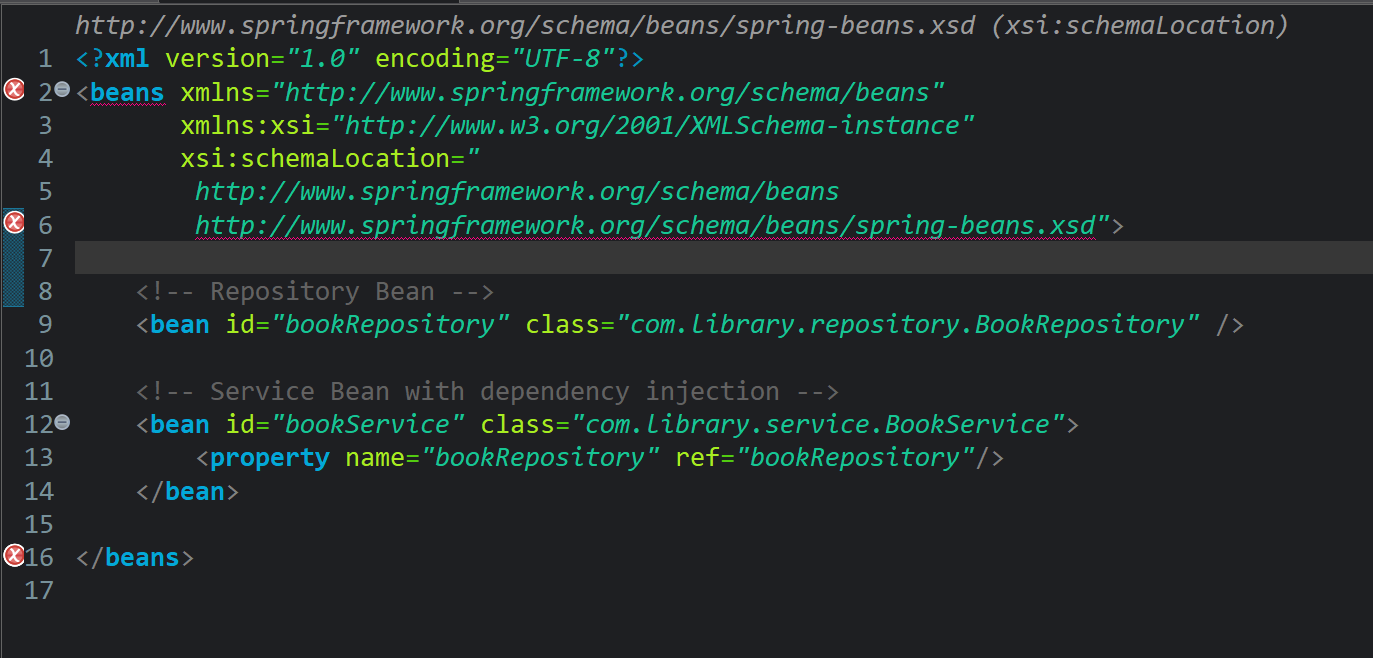
1. **Set Up a Spring Project:**
   * Create a Maven project named **LibraryManagement**.



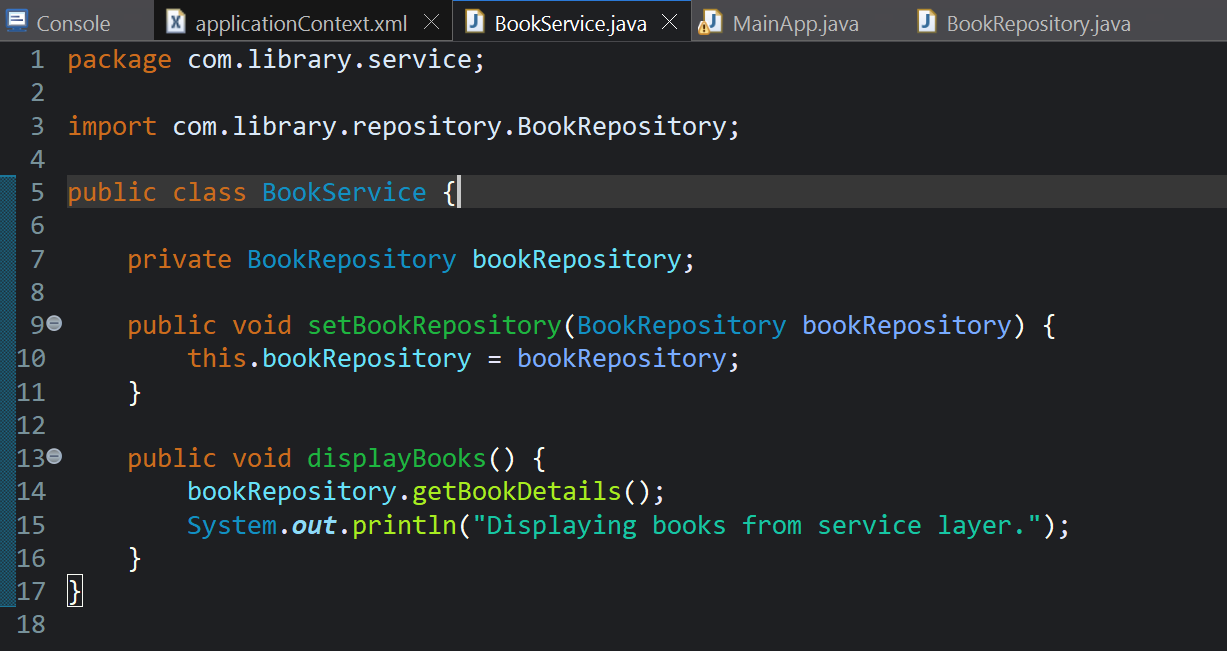
* + Add Spring Core dependencies in the **pom.xml** file.



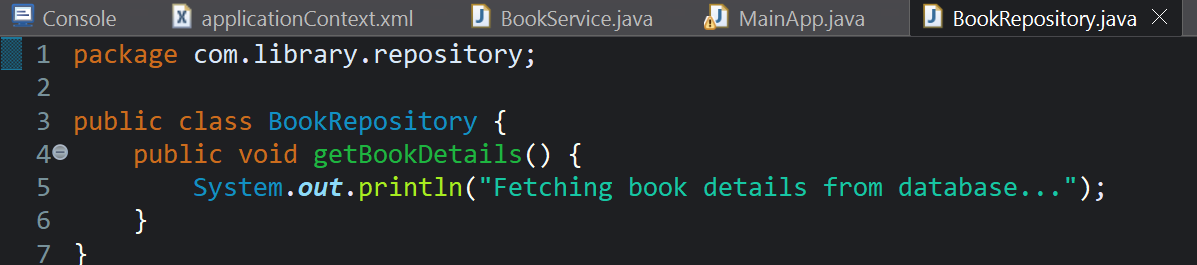
1. **Configure the Application Context:**
   * Create an XML configuration file named **applicationContext.xml** in the **src/main/resources** directory.
   * Define beans for **BookService** and **BookRepository** in the XML file.



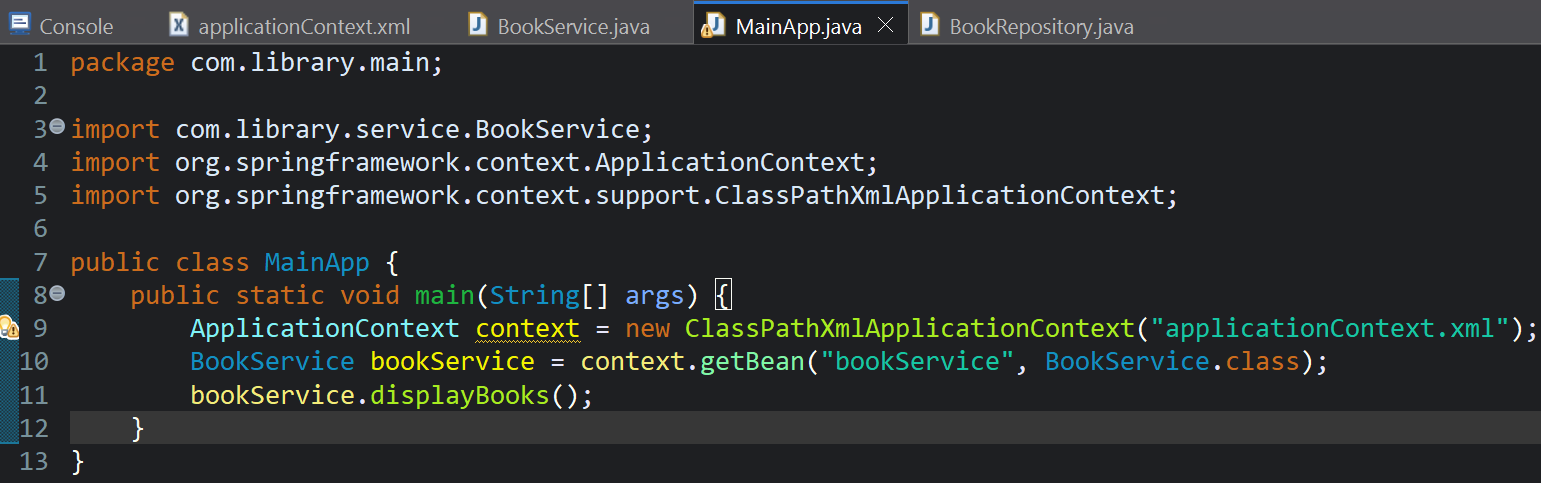
1. **Define Service and Repository Classes:**
   * Create a package **com.library.service** and add a class **BookService**.



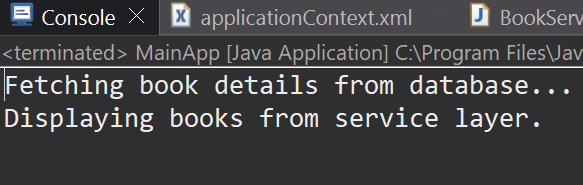
* + Create a package **com.library.repository** and add a class **BookRepository**.



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



Output :



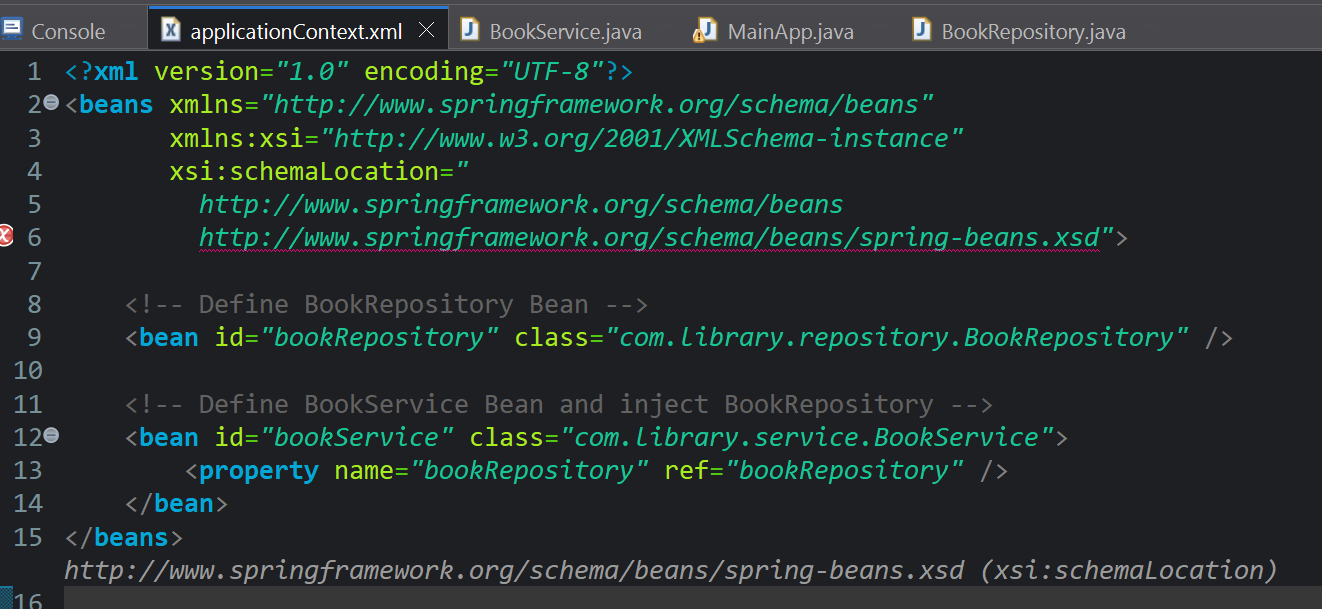
**Exercise 2: Implementing Dependency Injection**

**Scenario:**

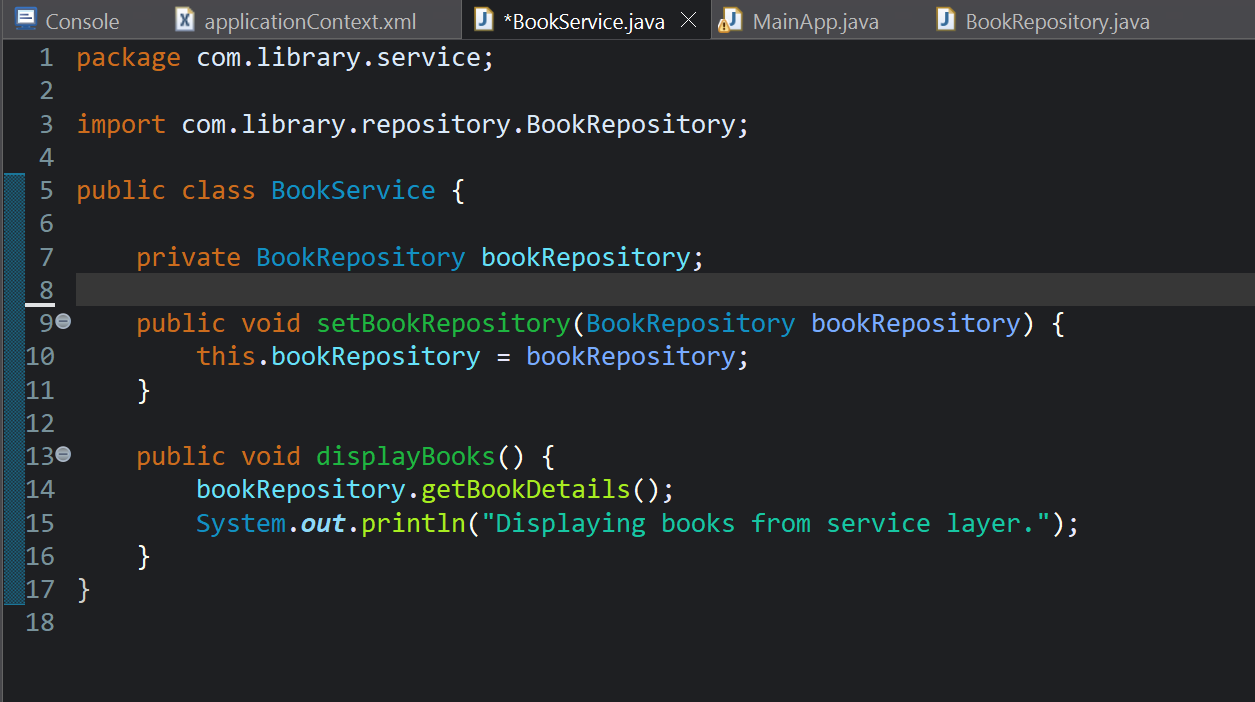
In the library management application, you need to manage the dependencies between the BookService and BookRepository classes using Spring's IoC and DI.

**Steps:**

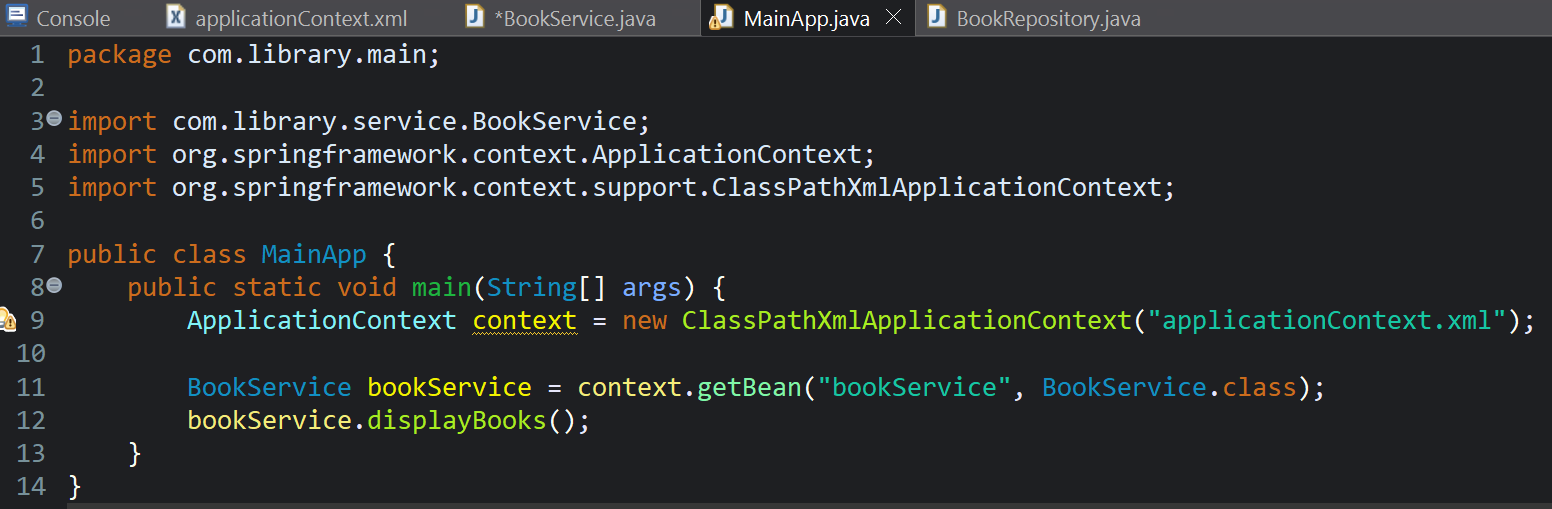
1. **Modify the XML Configuration:**
   * Update **applicationContext.xml** to wire **BookRepository** into **BookService**.



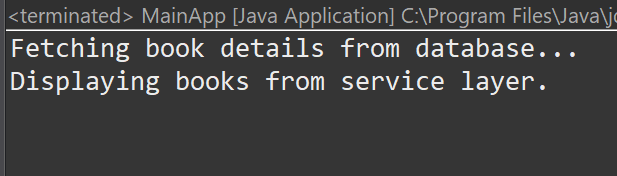
1. **Update the BookService Class:**
   * Ensure that **BookService** class has a setter method for **BookRepository**.



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

****

**Output:**



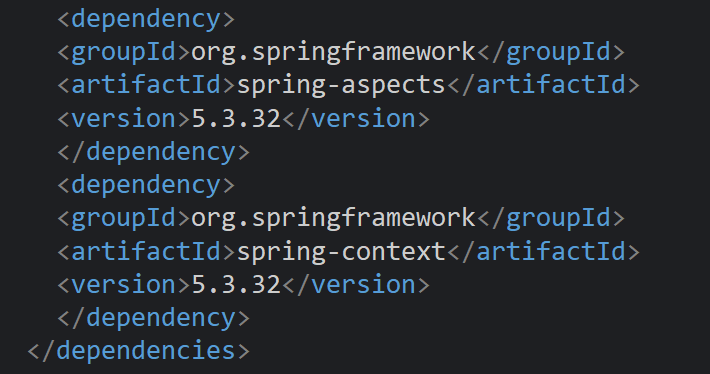
**Exercise 3: Implementing Logging with Spring AOP**

**Scenario:**

The library management application requires logging capabilities to track method execution times.

**Steps:**

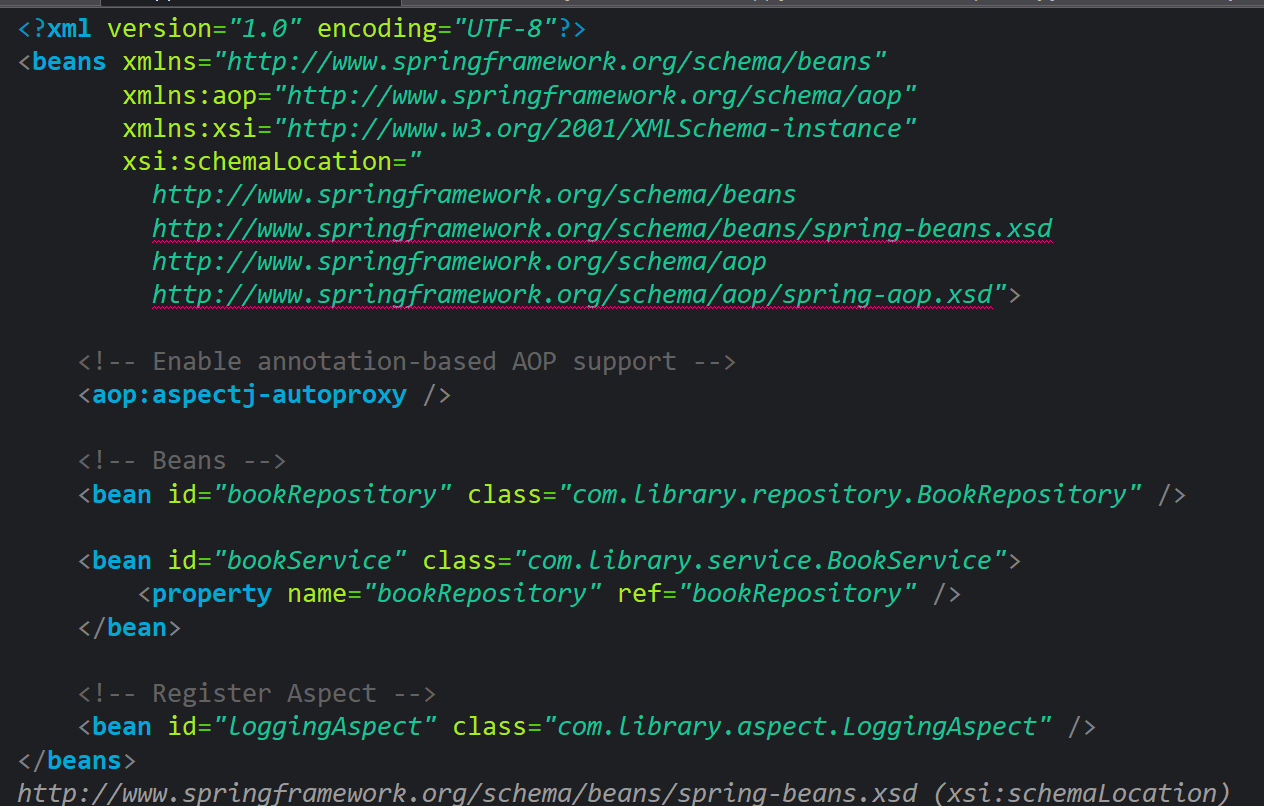
1. **Add Spring AOP Dependency:**
   * Update **pom.xml** to include Spring AOP dependency.



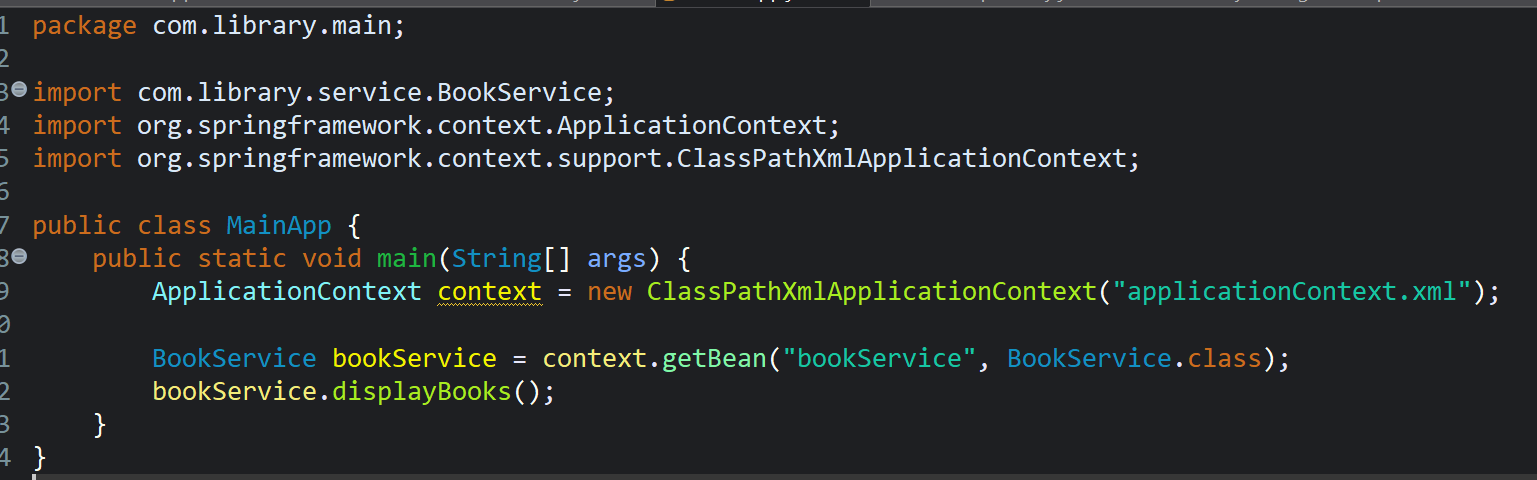
1. **Create an Aspect for Logging:**
   * Create a package **com.library.aspect** and add a class **LoggingAspect** with a method to log execution times.



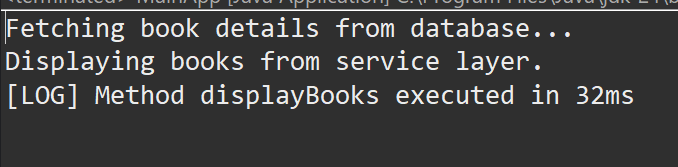
1. **Enable AspectJ Support:**
   * Update **applicationContext.xml** to enable **AspectJ** support and register the aspect.



1. **Test the Aspect:**
   * Run the **LibraryManagementApplication** main class and observe the console for log messages indicating method execution times.



Output:



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

**Scenario:**

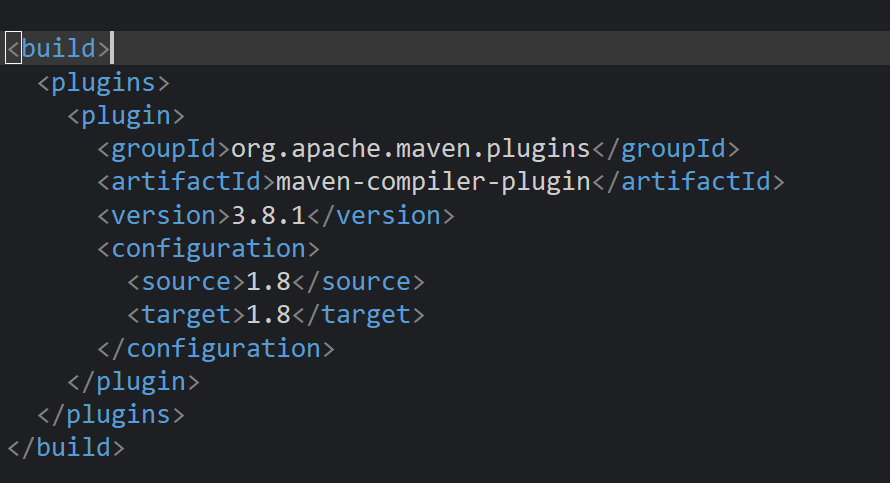
You need to set up a new Maven project for the library management application and add Spring dependencies.

**Steps:**

1. **Create a New Maven Project:**
   * Create a new Maven project named **LibraryManagement**.
2. **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.



**Additional important hands-on**

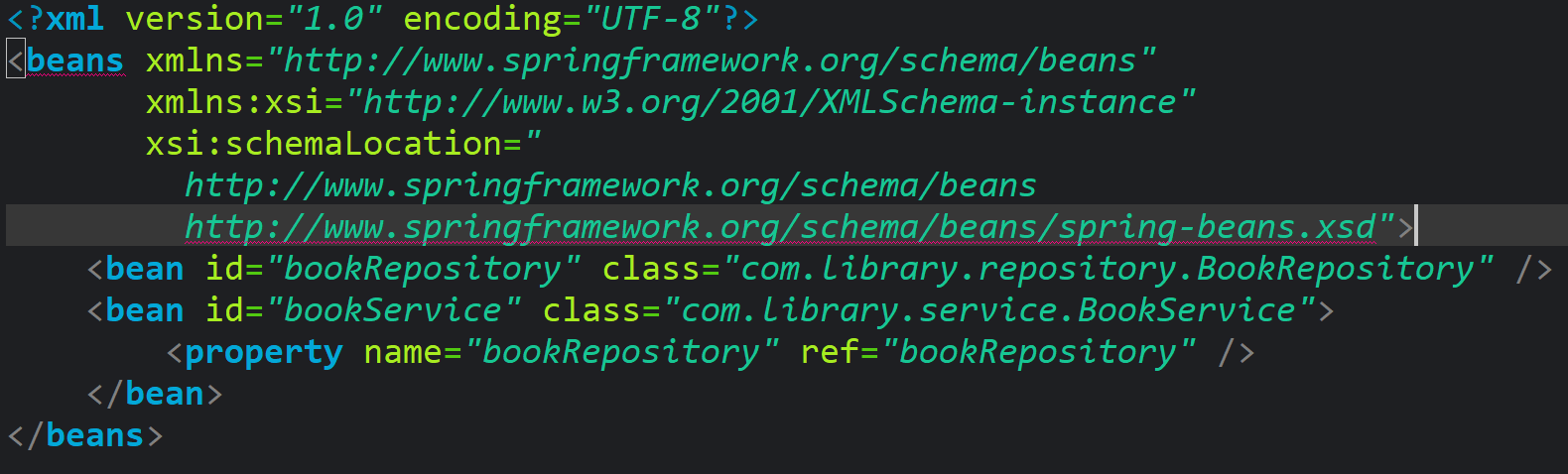
**Exercise 5: Configuring the Spring IoC Container**

**Scenario:**

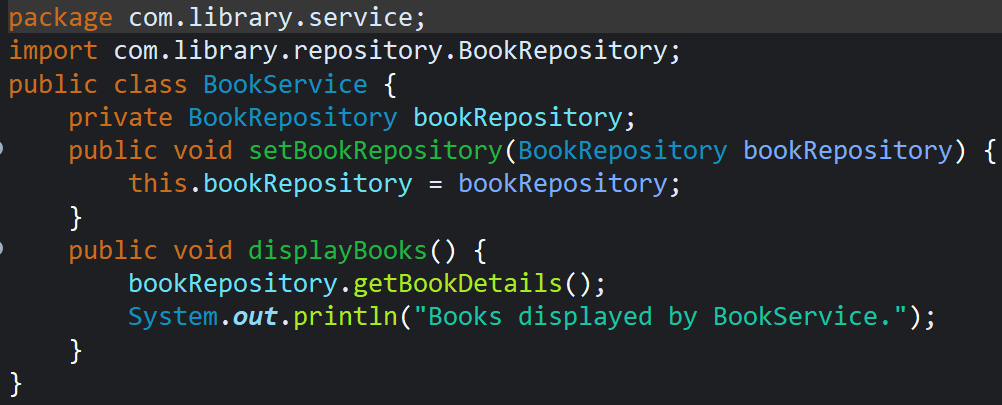
The library management application requires a central configuration for beans and dependencies.

**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.

