# **WEEK 3 spring core AND maven HANDS ON**

**SUBMITTED BY :-**

NAME – AYUSH SAHOO

SUPERSET ID – 6372243

COURSE – DIGITAL NURTURE 4.0 JAVA FSE

EMAIL – [ayush.sahoo003@gmail.com](mailto:ayush.sahoo003@gmail.com)

COLLEGE – C.V. RAMAN GLOBAL UNIVERSITY

**-----------------NOTE: I HAVE ALSO DONE ADDITIONAL IMPORTANT HANDS ON -------------------**

**MANDATORY HANDS ON**

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

**Scenario:**

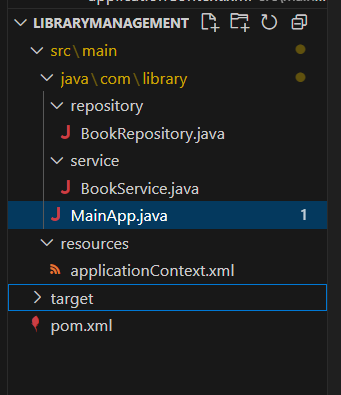
**Your company is developing a web application for managing a library. You need to use the Spring Framework to handle the backend operations.**

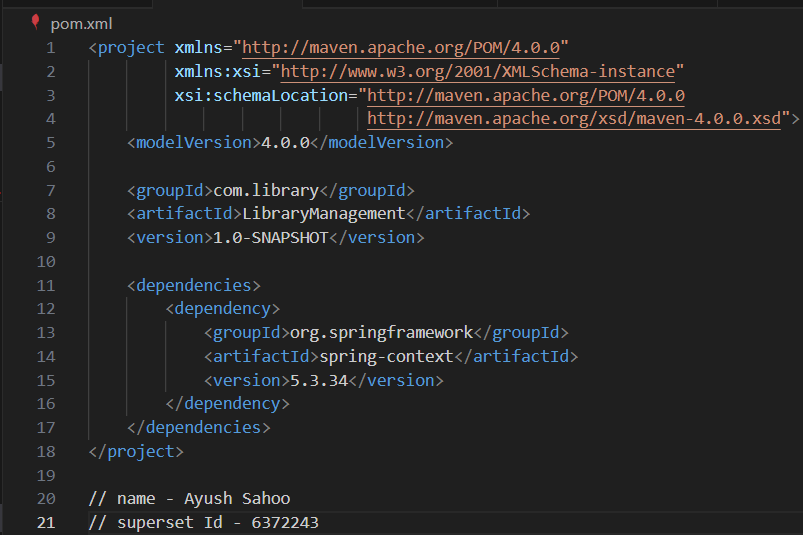
**Steps:**

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.**

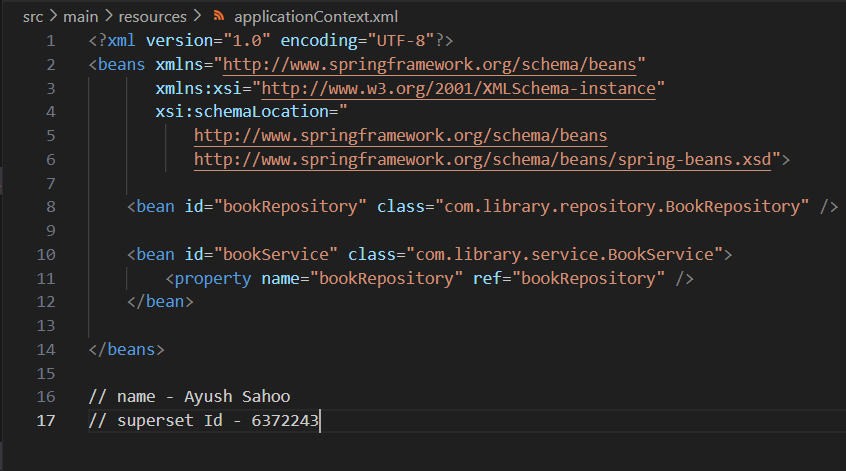
**SOLUTION:-**

Step 1:



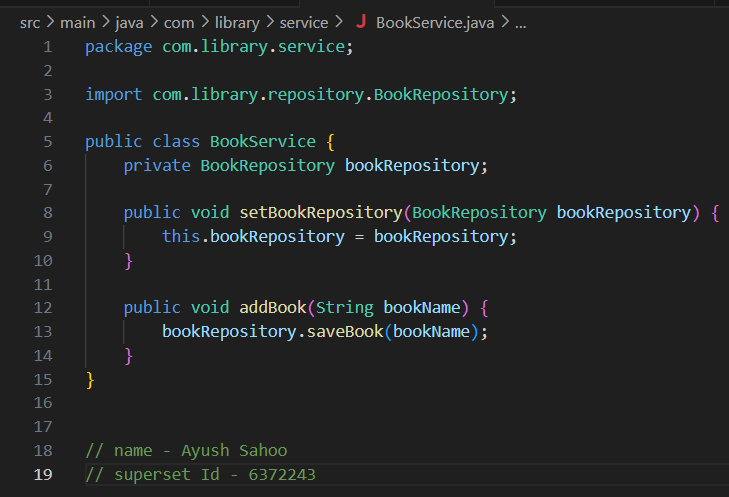


Step 2:

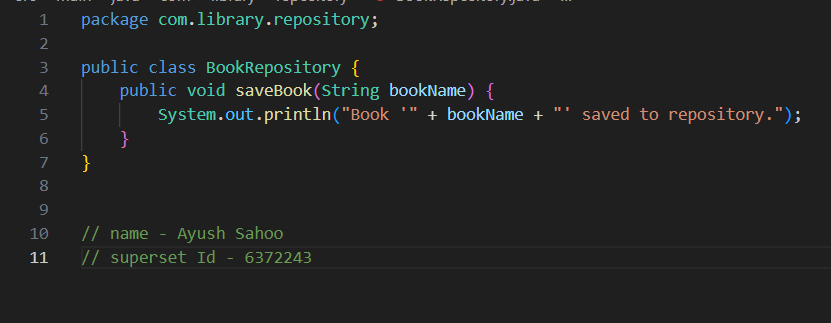


Step 3:

BookService:



BookRepository:

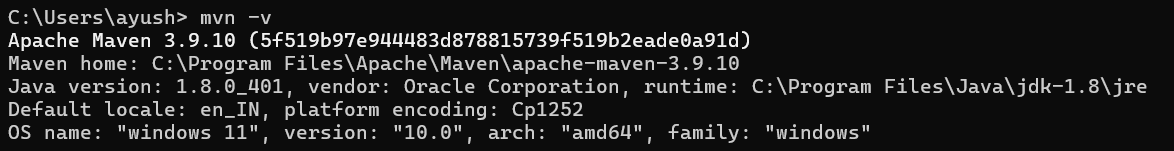


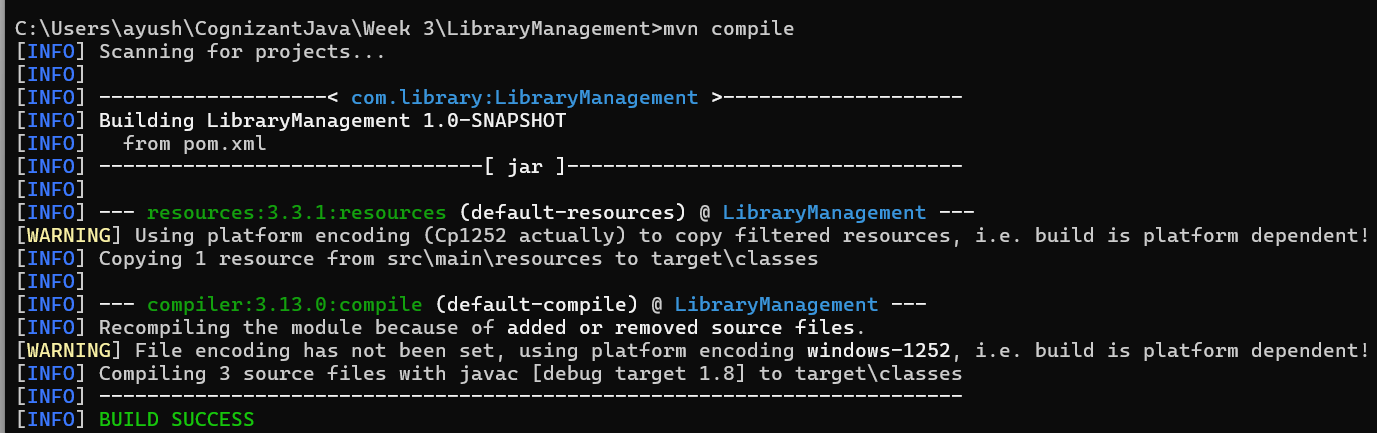
Step 5:



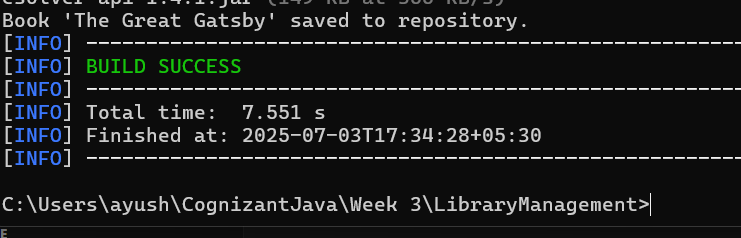
Let’s Set CLI for Maven

(The application was compiled and executed using Maven commands via Windows Command Prompt.)





OUTPUT:



**Exercise 2: Implementing Dependency Injection**

**Scenario:**

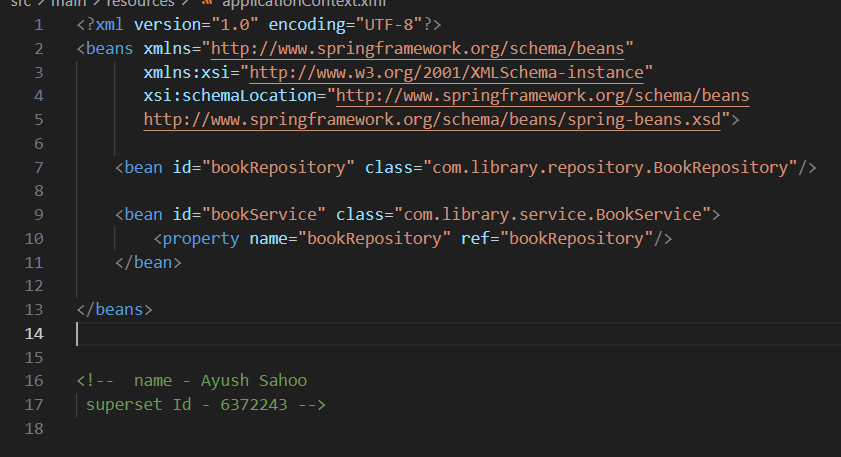
**In the library management application, you need to manage the dependencies between the BookService and BookRepository classes using Spring's IoC and DI.**

**Steps:**

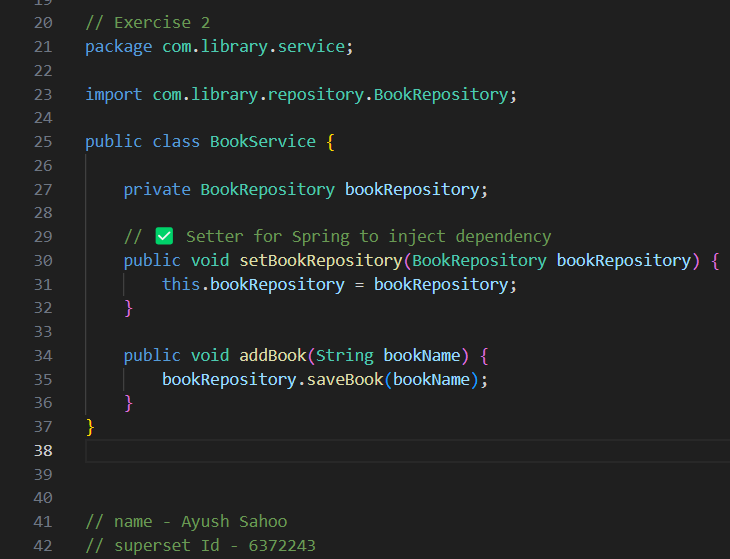
1. **Modify the XML Configuration:**
   * **Update applicationContext.xml to wire BookRepository into BookService.**
2. **Update the BookService Class:**
   * **Ensure that BookService class has a setter method for BookRepository.**
3. **Test the Configuration:**
   * **Run the LibraryManagementApplication main class to verify the dependency injection.**

**SOLUTION:**

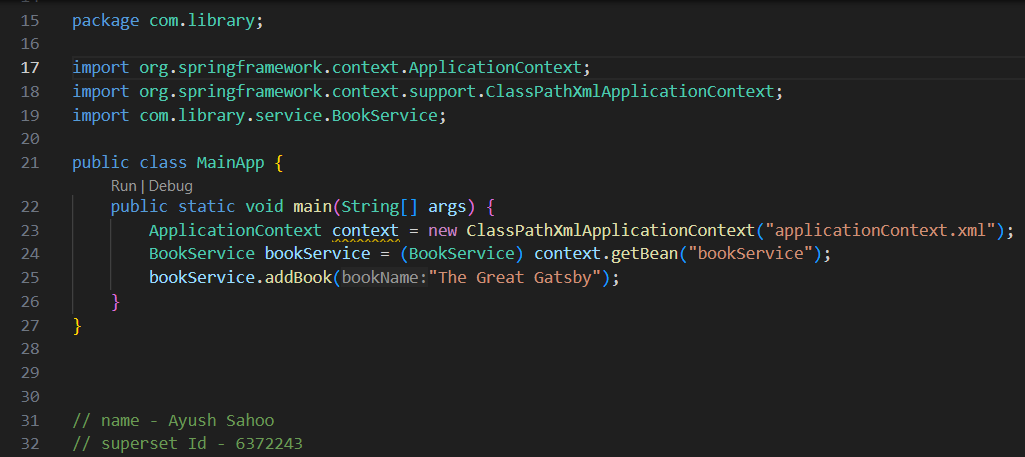
Step 1:



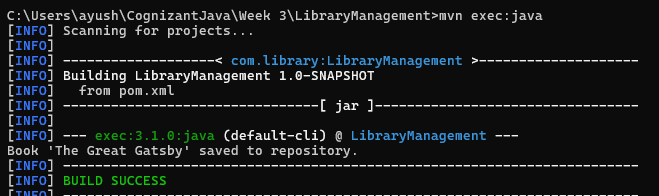
Step 2:



Step 3:



OUTPUT:



{Book ‘The Great Gatsby’ saved to repository.}

**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.**

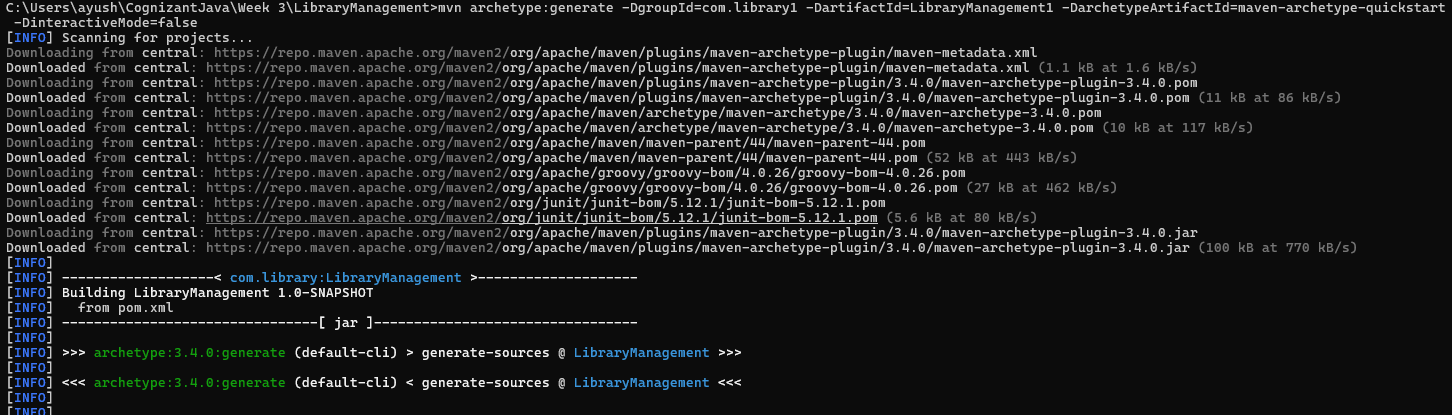
**Steps:**

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

**SOLUTION:**

**NOTE : HERE WE HAVE USED LIBRARYMANAGEMENT1 INSTEAD OF LIBRARYMANAGEMENT BECAUSE OF THE SAME NAME USED IN PREVIOUS QUESTION**

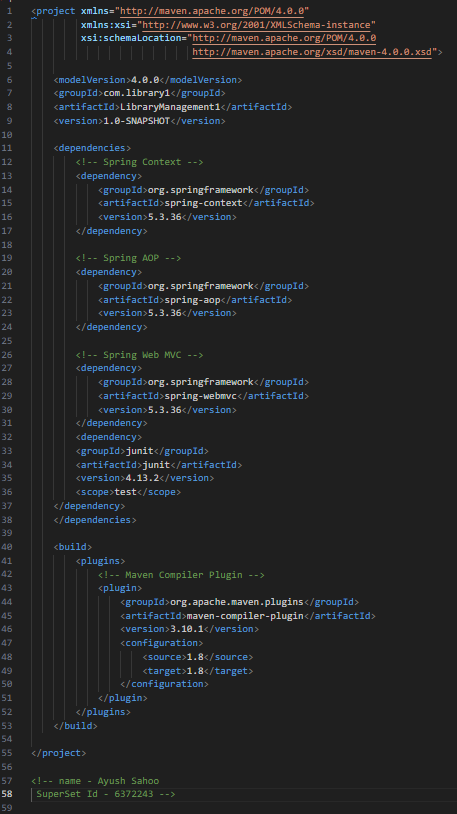
Step 1:



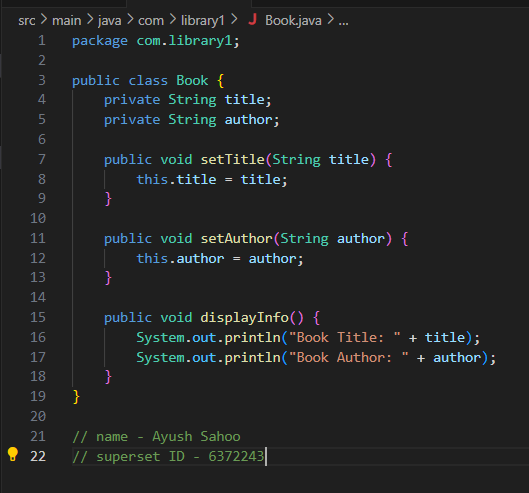
The Command that we have used here in CLI is

“mvn archetype:generate -DgroupId=com.library1 -DartifactId=LibraryManagement1 -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false”

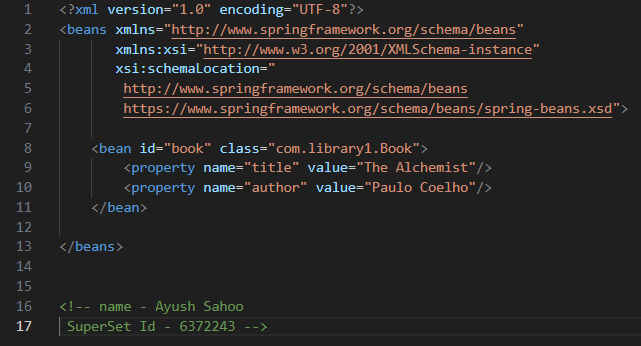
Step 2:



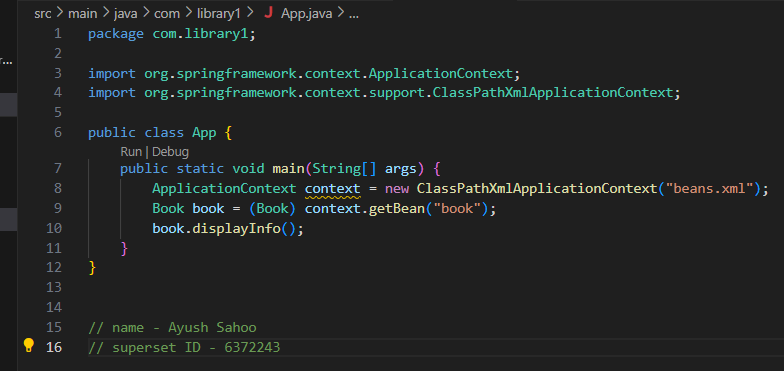
Step 3:



Step 4:



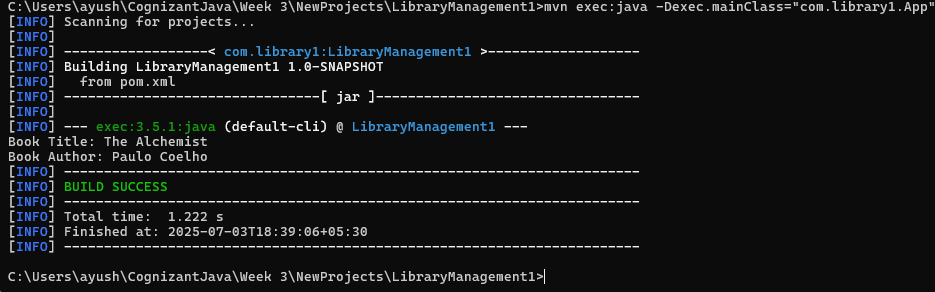
Step 5:



OUTPUT:

Book Title: The Alchemist

Book Author: Paulo Coelho



**Configure Maven Plugins:**

* + Configure the Maven Compiler Plugin for Java version 1.8 in the pom.xml file.

Explanation: We did use java version 1.8 in the pom.xml file

ADDITIONAL HANDS ON

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

**Scenario:**

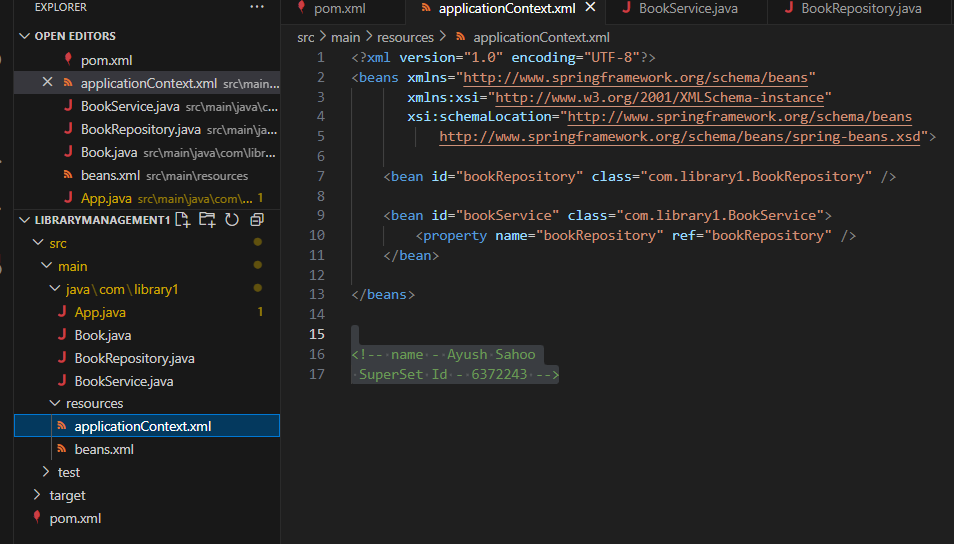
**The library management application requires a central configuration for beans and dependencies.**

**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.**
2. **Update the BookService Class:**
   * **Ensure that the BookService class has a setter method for BookRepository.**
3. **Run the Application:**
   * **Create a main class to load the Spring context and test the configuration.**

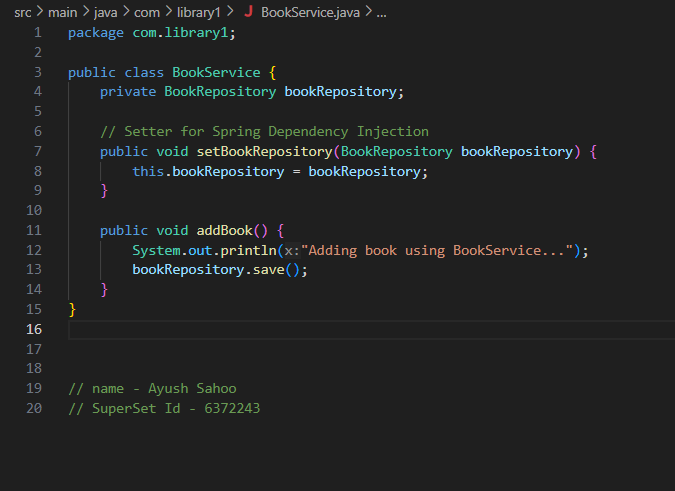
SOLUTION:

Step 1:

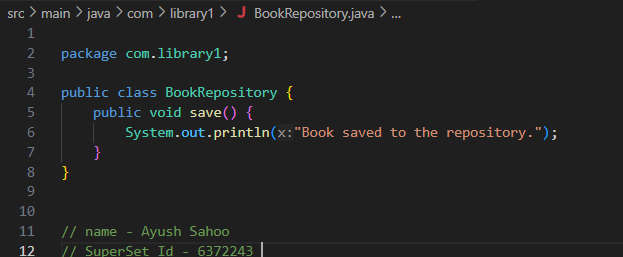


Step 2:

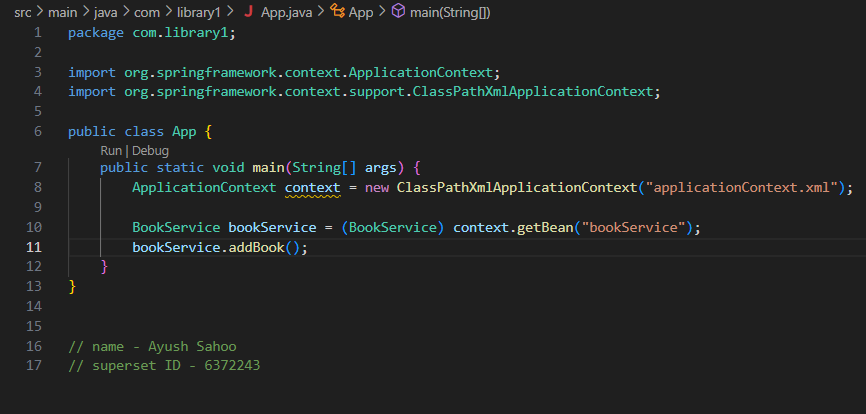
BookService :



BookRepository:



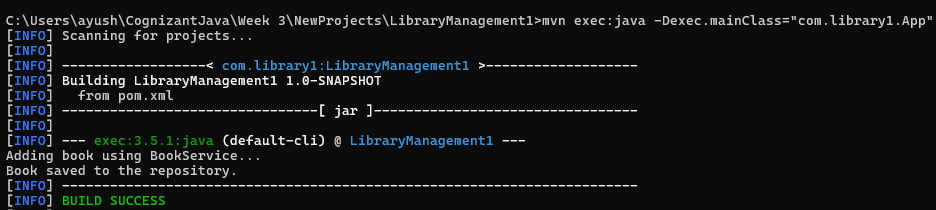
Step 3:



OUTPUT:

mvn clean install

mvn exec:java -Dexec.mainClass="com.library1.App"



Output will be :

Displaying book details...

Saving book to repository...

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

**Scenario:**

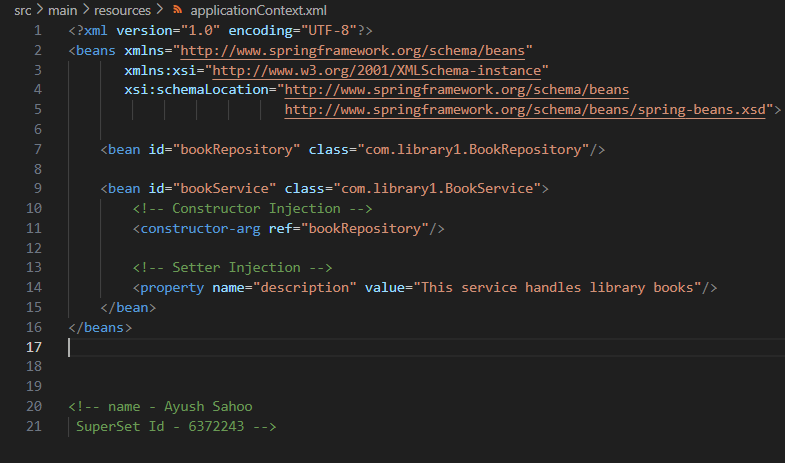
**The library management application requires both constructor and setter injection for better control over bean initialization.**

**Steps:**

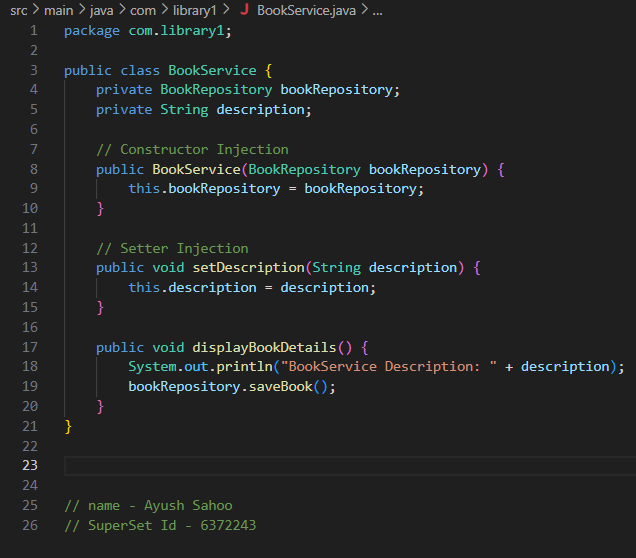
1. **Configure Constructor Injection:**
   * **Update applicationContext.xml to configure constructor injection for BookService.**
2. **Configure Setter Injection:**
   * **Ensure that the BookService class has a setter method for BookRepository and configure it in applicationContext.xml.**
3. **Test the Injection:**
   * **Run the LibraryManagementApplication main class to verify both constructor and setter injection.**

SOLUTION:

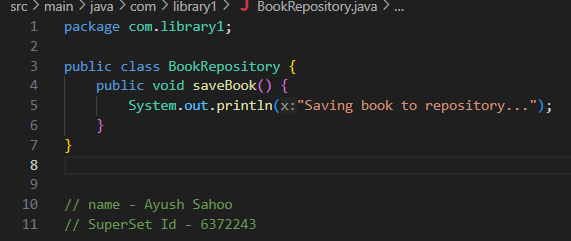
Step 1:



Step 2:



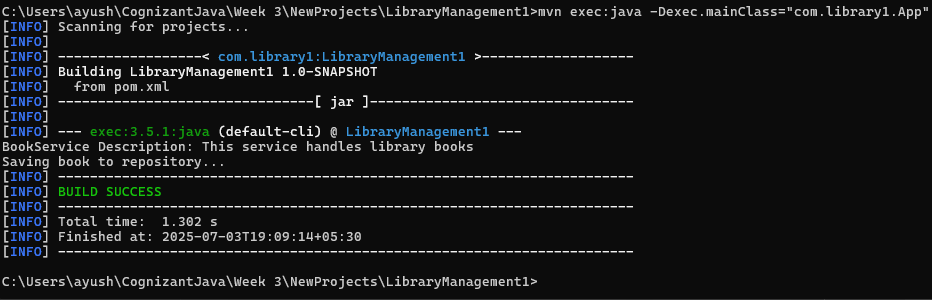
Step 3:



Step 4:



**OUTPUT:**



Command used : mvn exec:java -Dexec.mainClass="com.library1.App"

Output: BookService Description: This service handles library books

Saving book to repository...

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

**Scenario:**

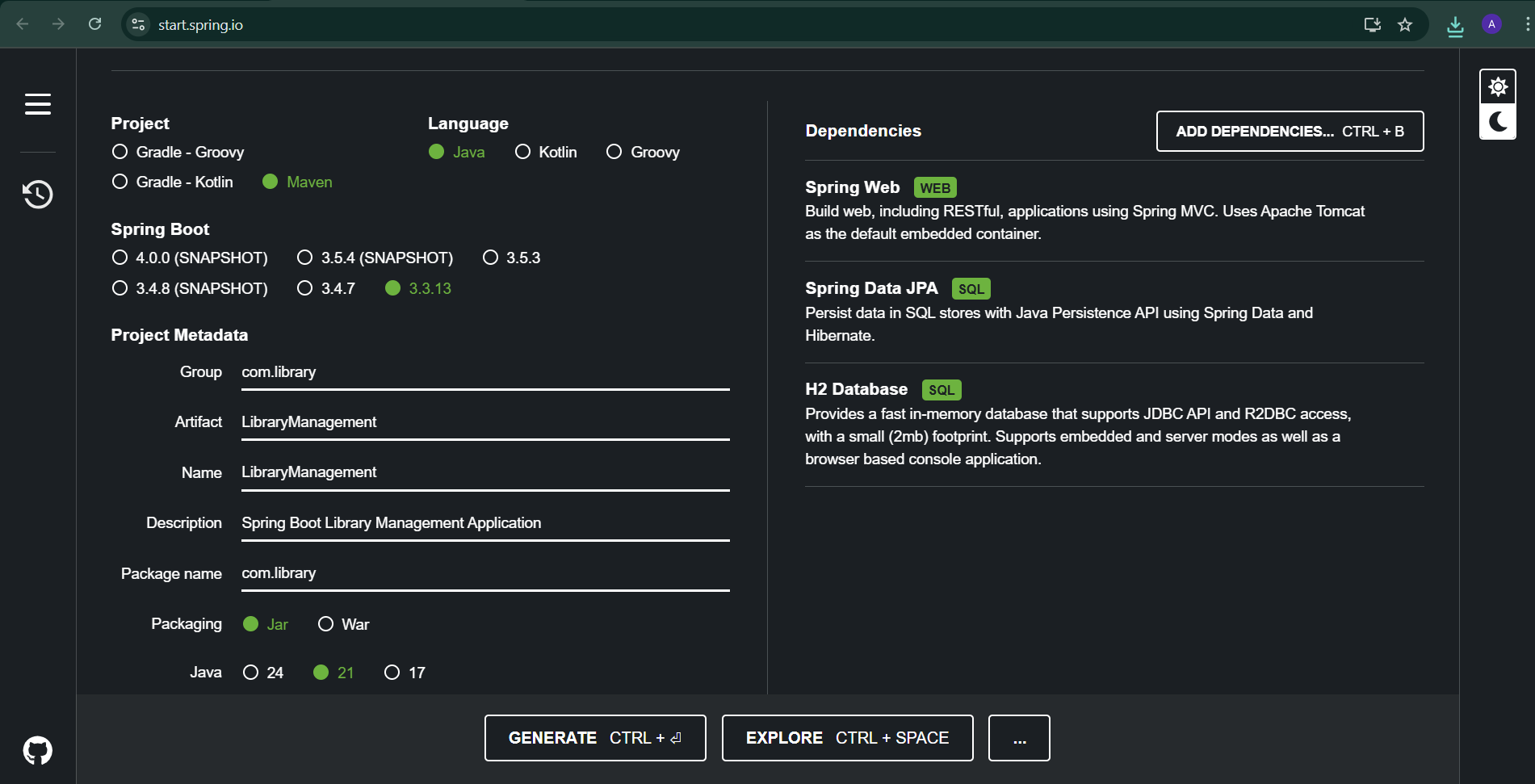
**You need to create a Spring Boot application for the library management system to simplify configuration and deployment.**

**Steps:**

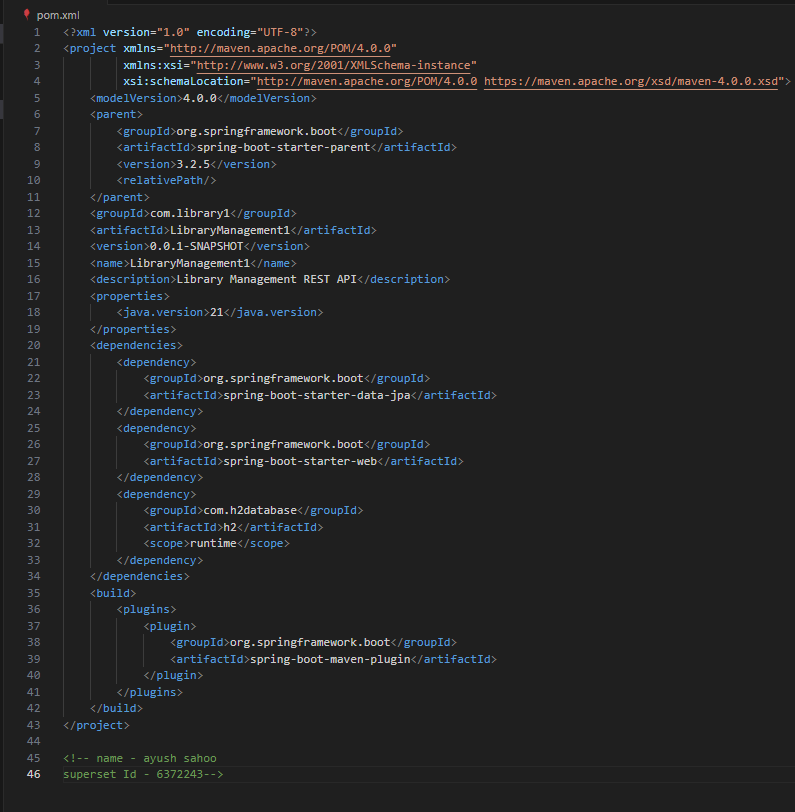
1. **Create a Spring Boot Project:**
   * **Use Spring Initializr to create a new Spring Boot project named LibraryManagement.**
2. **Add Dependencies:**
   * **Include dependencies for Spring Web, Spring Data JPA, and H2 Database.**
3. **Create Application Properties:**
   * **Configure database connection properties in application.properties.**
4. **Define Entities and Repositories:**
   * **Create Book entity and BookRepository interface.**
5. **Create a REST Controller:**
   * **Create a BookController class to handle CRUD operations.**
6. **Run the Application:**
   * **Run the Spring Boot application and test the REST endpoints.**

**SOLUTION:**

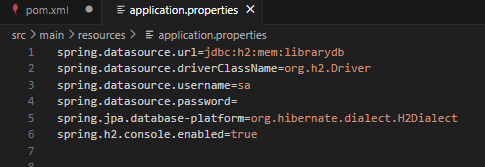
Step 1:



Step 2:

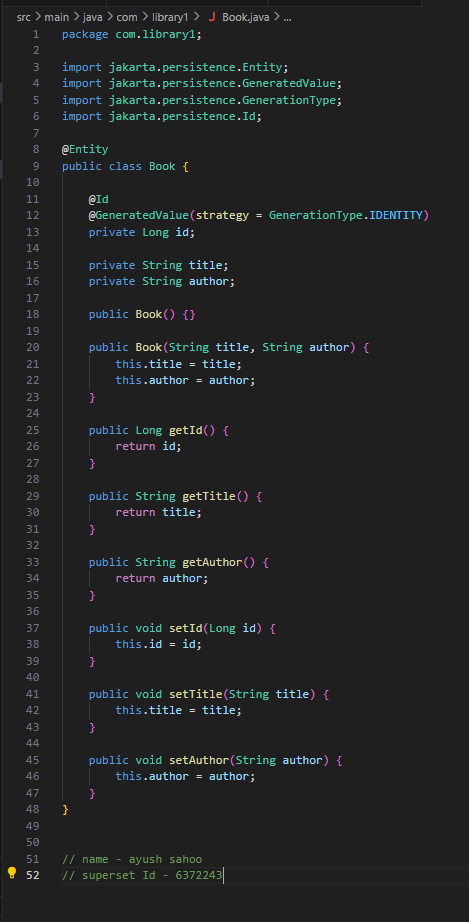


Step 3:

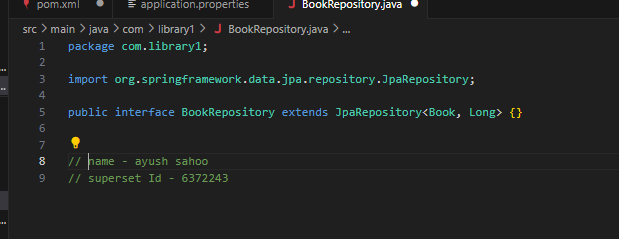


Step 4:

Book.java

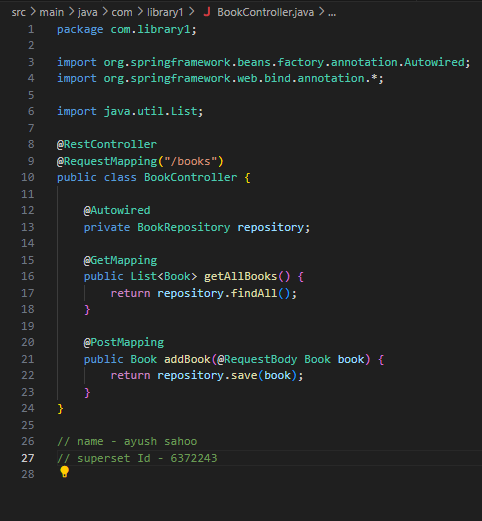


BookRepository.java

****

**JPA handles CRUD automatically**

Step 5:

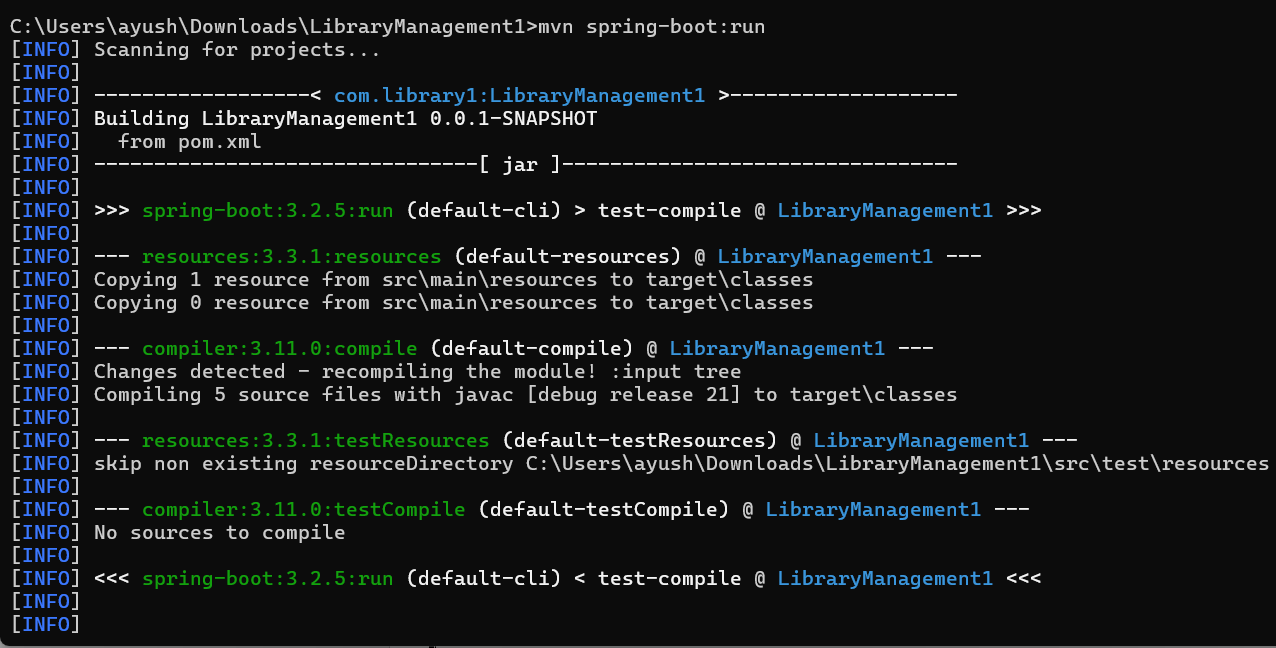


GET/ books – List all books

POST/ books – Add a new book

Step 6:

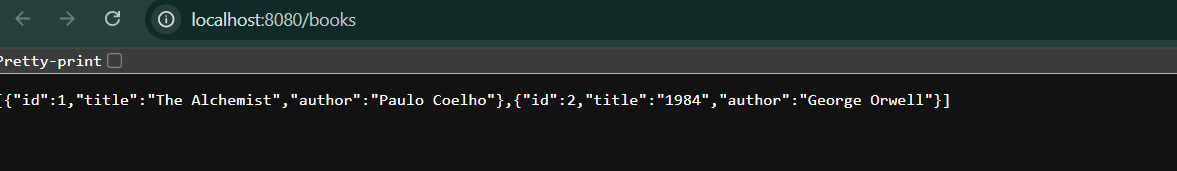
Use Command : mvn spring-boot:run



Then after opening

[**http://localhost:8080/books**](http://localhost:8080/books)

**OUTPUT:**

****

**x----x----x**