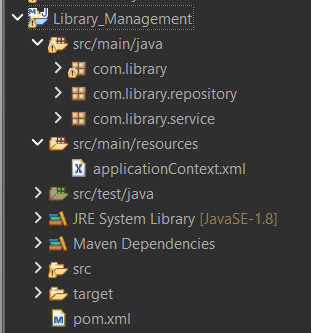
**Exercise 1: Configuring a Basic Spring Application**

**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:**

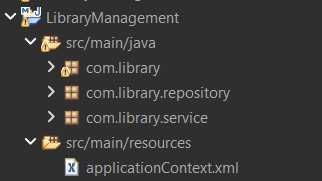
1. **Set Up a Spring Project:**
   * Create a Maven project named **LibraryManagement**.



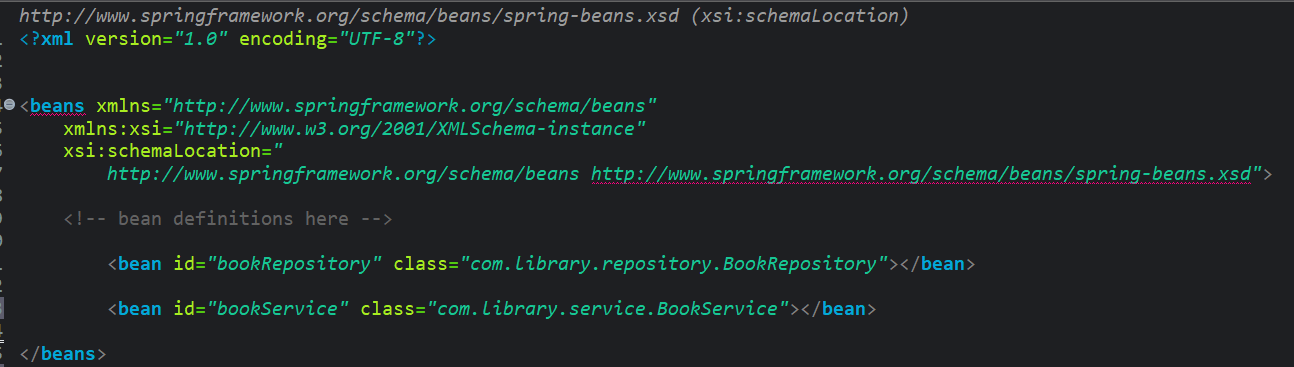
* + Add Spring Core dependencies in the **pom.xml** file.



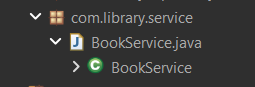
1. **Configure the Application Context:**
   * Create an XML configuration file named **applicationContext.xml** in the **src/main/resources** directory.



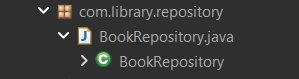
* + Define beans for **BookService** and **BookRepository** in the XML file.



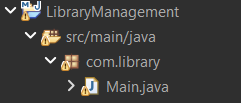
1. **Define Service and Repository Classes:**
   * Create a package **com.library.service** and add a class **BookService**.

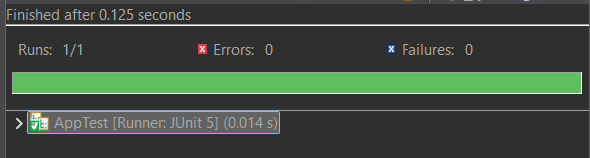


* + Create a package **com.library.repository** and add a class **BookRepository**.



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





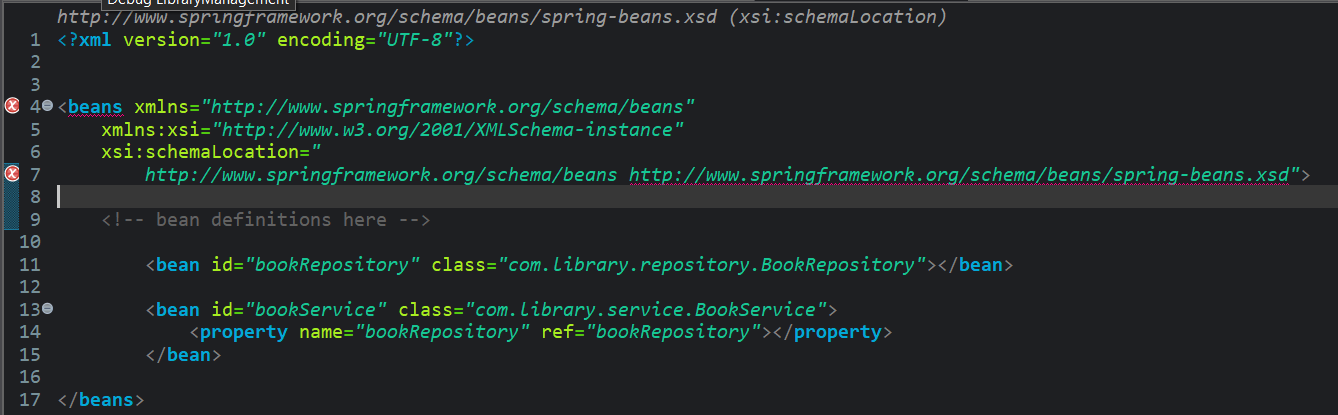
**Exercise 2: Implementing Dependency Injection**

**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**.



<?**xml** version=*"1.0"* encoding=*"UTF-8"*?>

<**beans** xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"*

*http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd"*>

<!-- bean definitions here -->

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

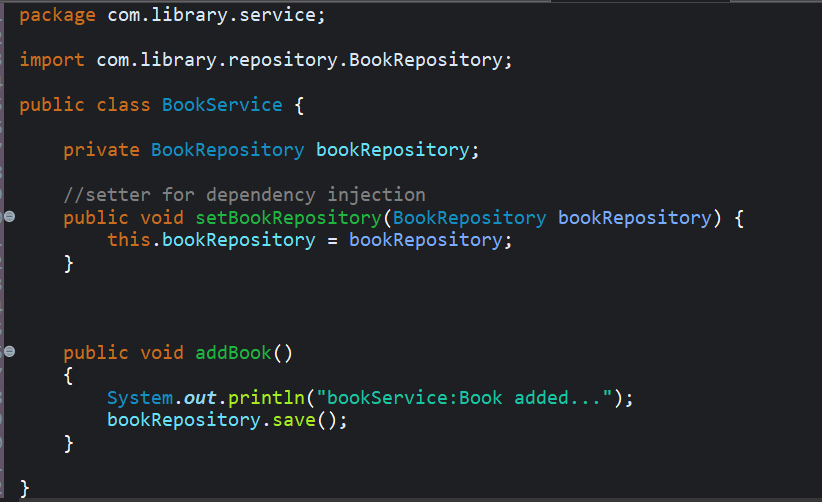
<**bean** id=*"bookService"* class=*"com.library.service.BookService"*>

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

</**bean**>

</**beans**>

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



package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

//setter for dependency injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook()

{

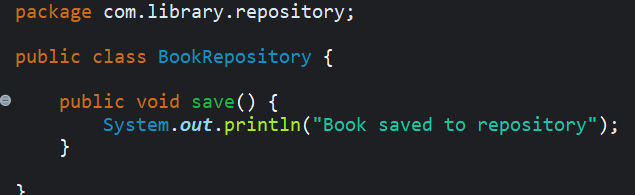
System.***out***.println("bookService:Book added...");

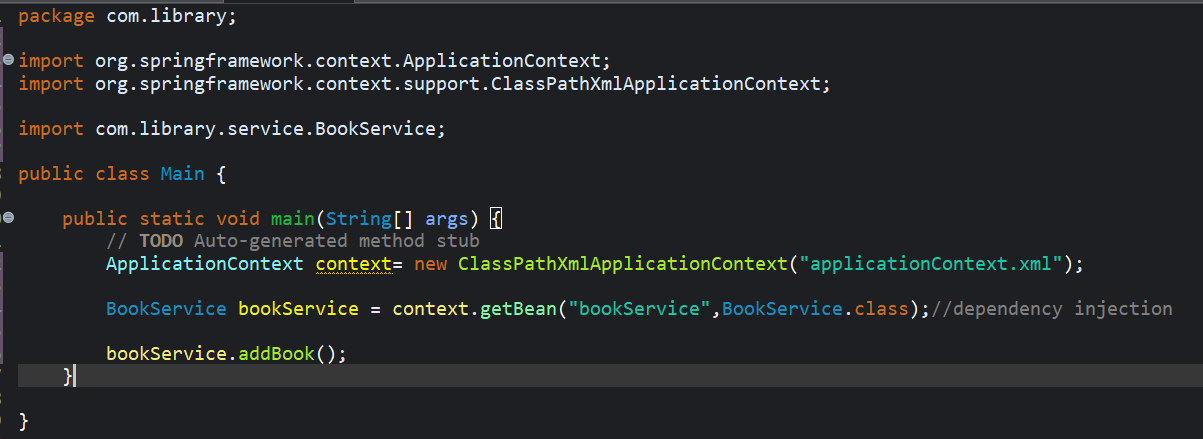
bookRepository.save();

}

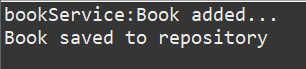
}

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





**Output**

****

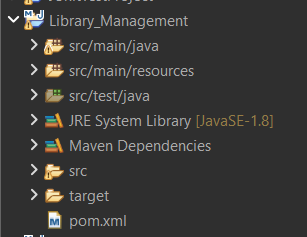
**Exercise 4: Creating and Configuring a Maven Project**

**Scenario:**

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

**Steps:**

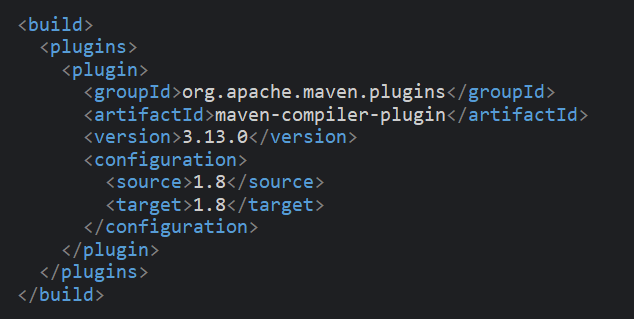
1. **Create a New Maven Project:**
   * Create a new Maven project named **LibraryManagement**.



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.



**pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.library</groupId>

<artifactId>Library\_Management</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>Library\_Management</name>

<url>http://www.example.com</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.junit</groupId>

<artifactId>junit-bom</artifactId>

<version>5.11.0</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<dependencies>

<!-- JUnit 5 -->

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-api</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-params</artifactId>

<scope>test</scope>

</dependency>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.36</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.36</version>

</dependency>

<!-- Spring WebMVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.36</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.13.0</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

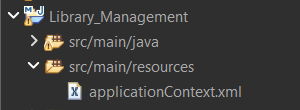
**Exercise 5: Configuring the Spring IoC Container**

**Scenario:**

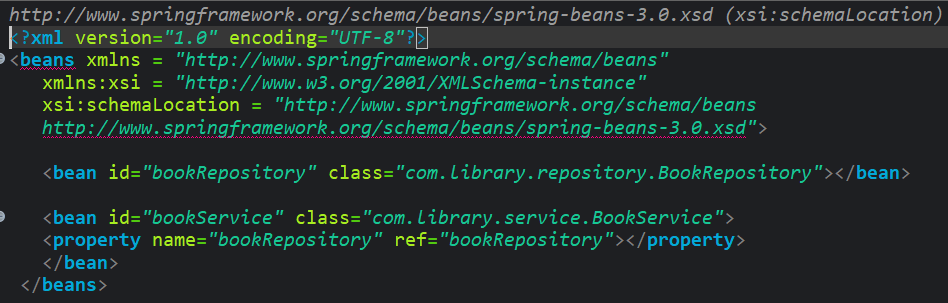
The library management application requires a central configuration for beans and dependencies.

**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.



<?**xml** version=*"1.0"* encoding=*"UTF-8"*?>

<**beans** xmlns = *"http://www.springframework.org/schema/beans"*

xmlns:xsi = *"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation = *"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"*>

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

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

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

</**bean**>

</**beans**>

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



package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook()

{

System.***out***.println("Book is added");

bookRepository.saveBook();

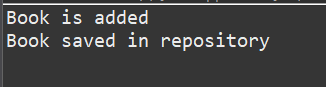
}

}

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



**Output**

****

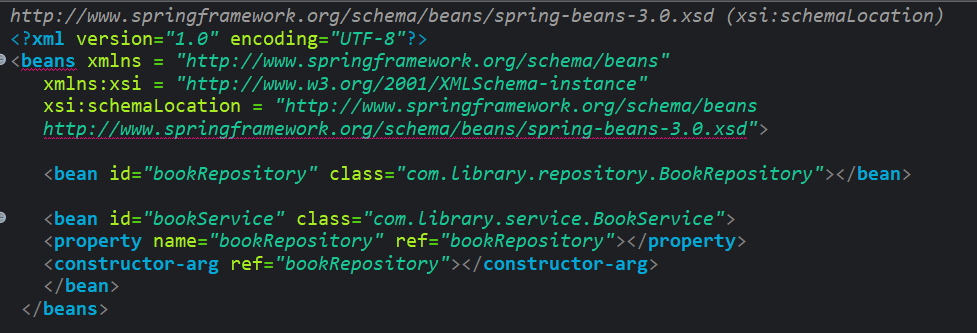
**Exercise 7: Implementing Constructor and Setter Injection**

**Scenario:**

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

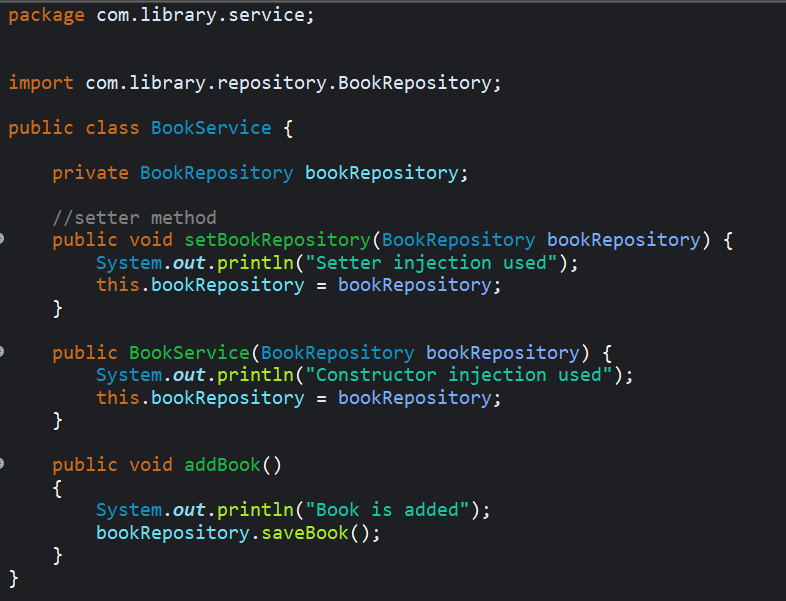
**Steps:**

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**.

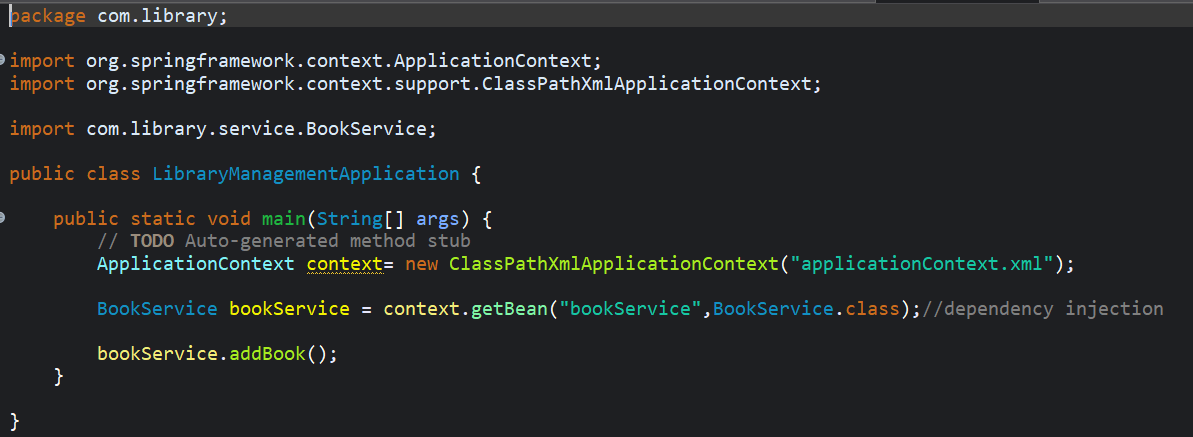
**Class BookService**



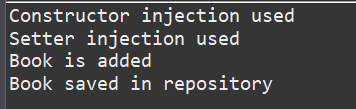
**applicationContext.xml**.



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



**Output**



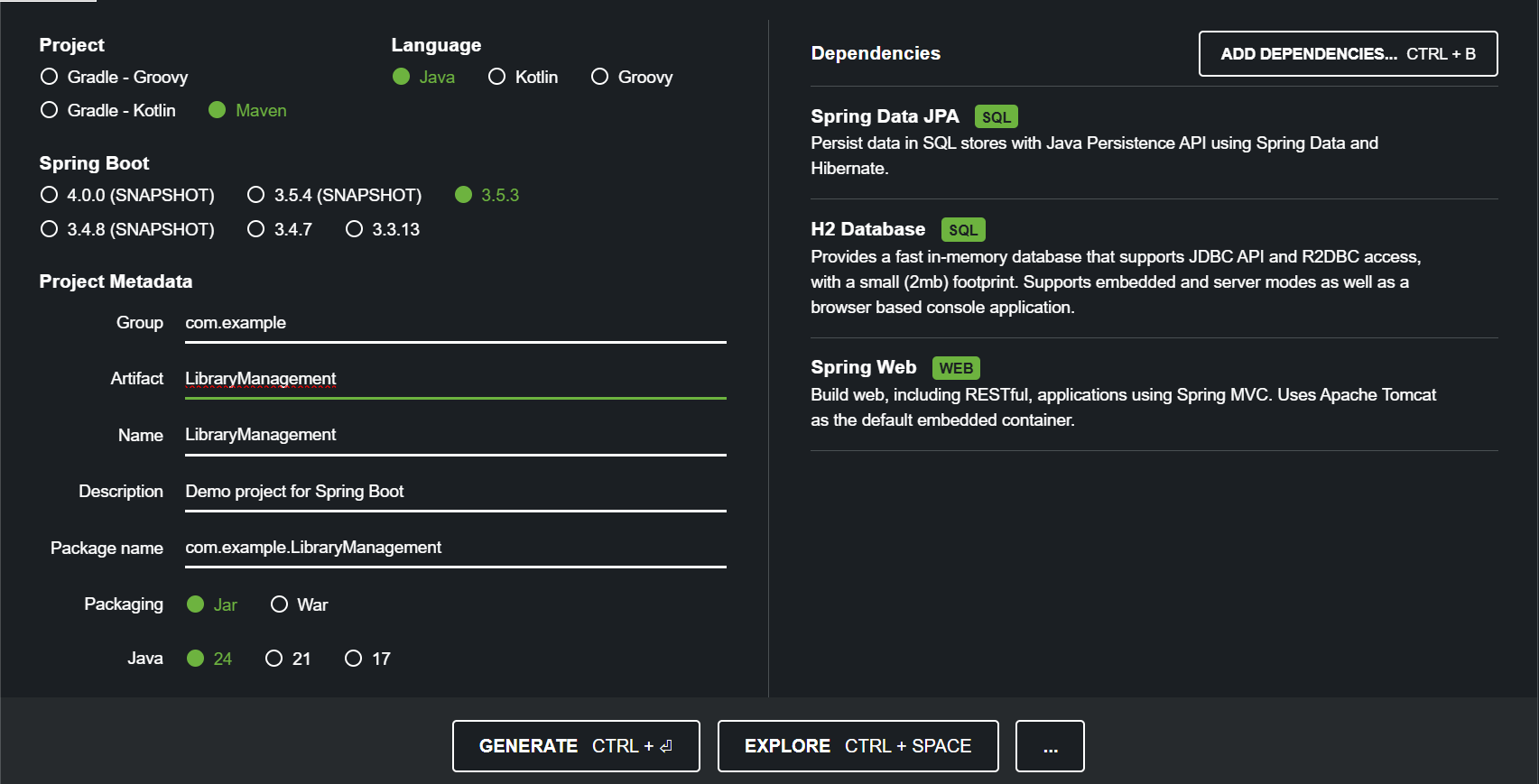
**Exercise 9: Creating a Spring Boot Application**

**Scenario:**

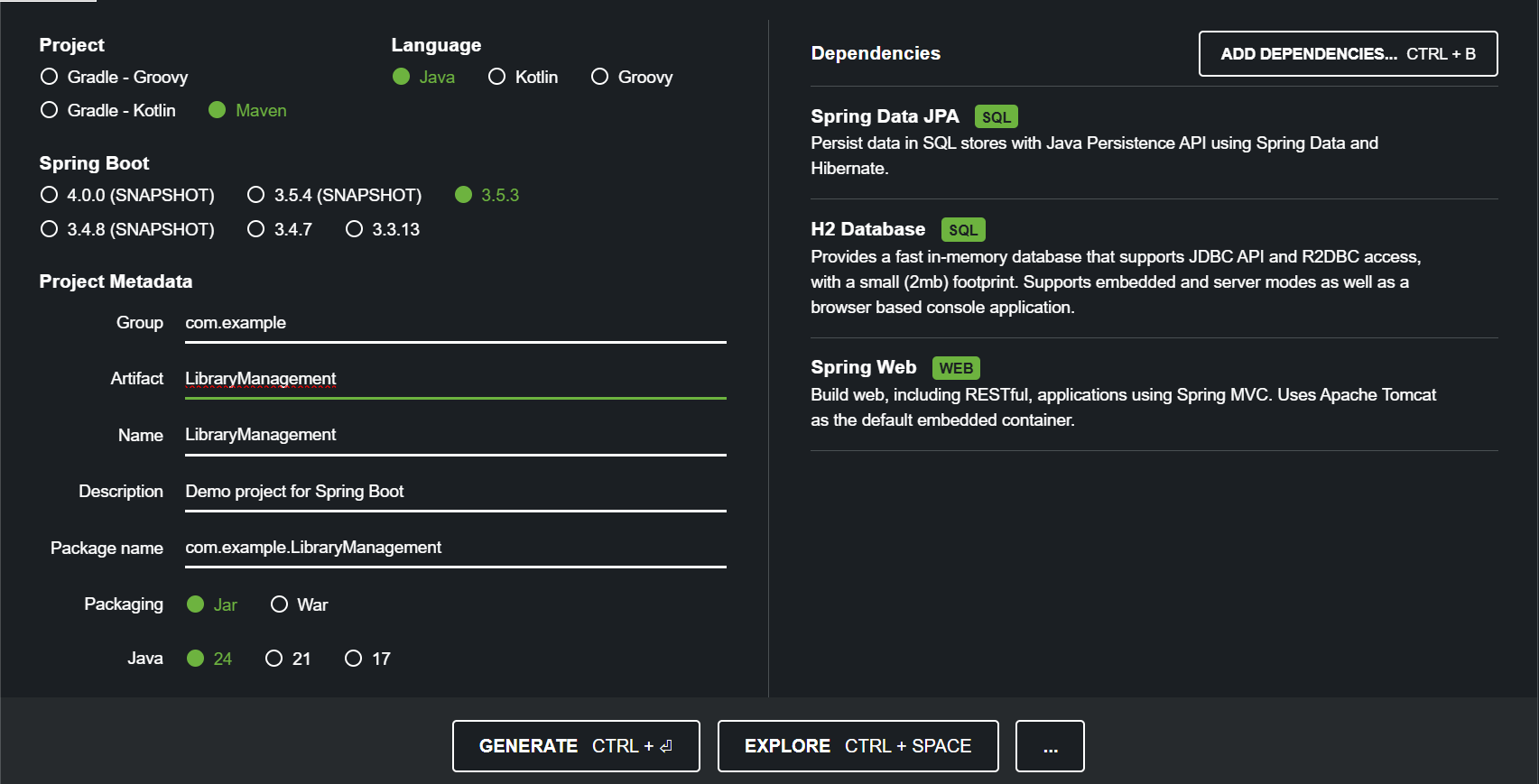
You need to create a Spring Boot application for the library management system to simplify configuration and deployment.

**Steps:**

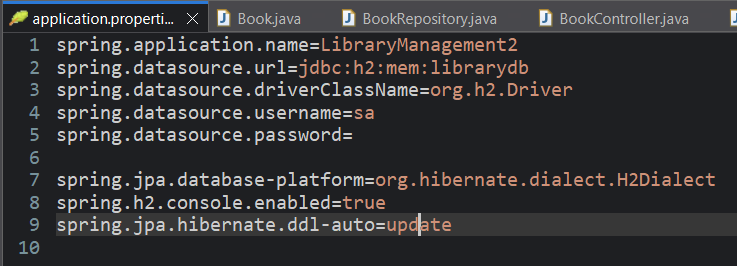
1. **Create a Spring Boot Project:**
   * Use **Spring Initializr** to create a new Spring Boot project named **LibraryManagement**.



1. **Add Dependencies:**
   * Include dependencies for **Spring Web, Spring Data JPA, and H2 Database**.



1. **Create Application Properties:**
   * Configure database connection properties in **application.properties**.



spring.application.name=LibraryManagement2

spring.datasource.url=jdbc:h2:mem:librarydb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

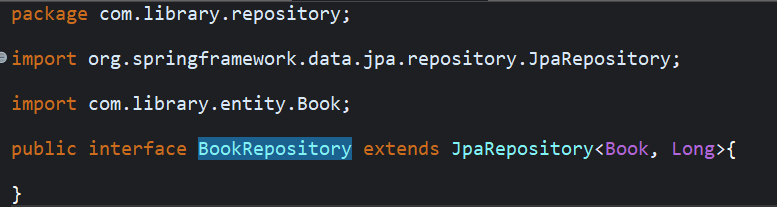
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true

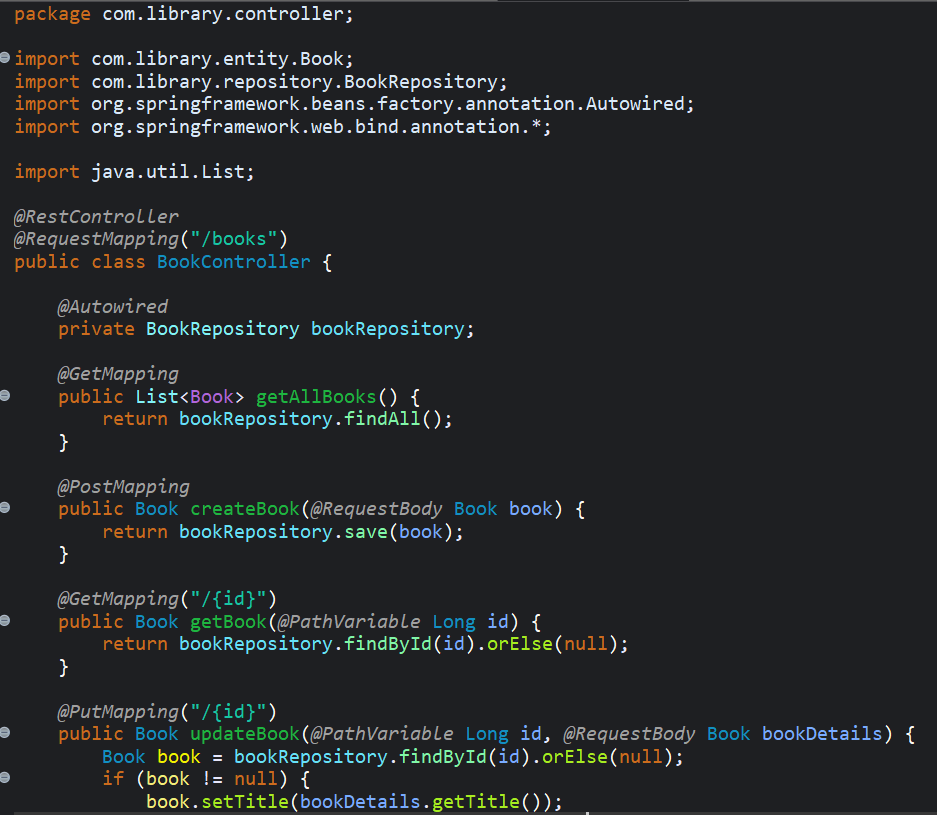
spring.jpa.hibernate.ddl-auto=update

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





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





package com.library.controller;

import com.library.entity.Book;

import com.library.repository.BookRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

*@RestController*

*@RequestMapping*("/books")

public class BookController {

*@Autowired*

private BookRepository bookRepository;

*@GetMapping*

public List<Book> getAllBooks() {

return bookRepository.findAll();

}

*@PostMapping*

public Book createBook(*@RequestBody* Book book) {

return bookRepository.save(book);

}

*@GetMapping*("/{id}")

public Book getBook(*@PathVariable* Long id) {

return bookRepository.findById(id).orElse(null);

}

*@PutMapping*("/{id}")

public Book updateBook(*@PathVariable* Long id, *@RequestBody* Book bookDetails) {

Book book = bookRepository.findById(id).orElse(null);

if (book != null) {

book.setTitle(bookDetails.getTitle());

book.setAuthor(bookDetails.getAuthor());

return bookRepository.save(book);

}

return null;

}

*@DeleteMapping*("/{id}")

public void deleteBook(*@PathVariable* Long id) {

bookRepository.deleteById(id);

}

}

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

