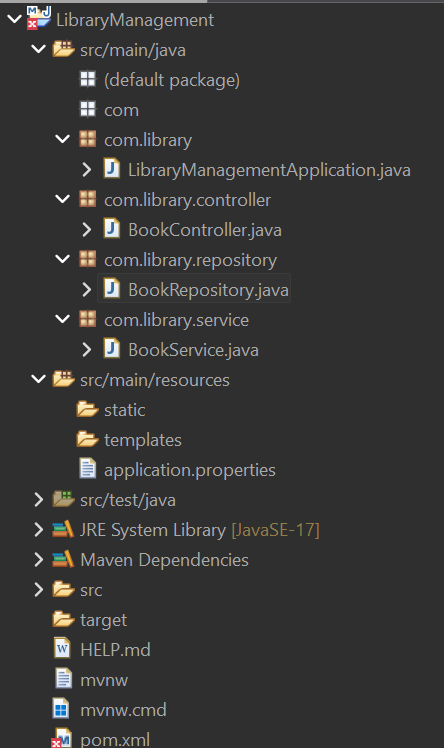
**Exercise 1: Configuring a Basic Spring Application**

**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**.

**Package com.library.service**

package com.library.service;

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

import org.springframework.stereotype.Service;

import com.library.repository.BookRepository;

*@Service*

public class BookService {

private final BookRepository bookRepository;

// Constructor-based dependency injection

*@Autowired*

public BookService(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void displayService() {

System.***out***.println("BookService: Service is working...");

bookRepository.displayRepo();

}

}

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

**package com.library.repository :**

package com.library.repository;

import org.springframework.stereotype.Repository;

*@Repository* // ✅ This is required so Spring can register the bean

public class BookRepository {

public void displayRepo() {

System.***out***.println("BookRepository: Fetching book data...");

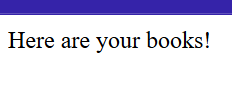
}

}

**Run the Application:**

* + Create a main class to load the Spring context and test the configuration.

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

****