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

**Scenario:**

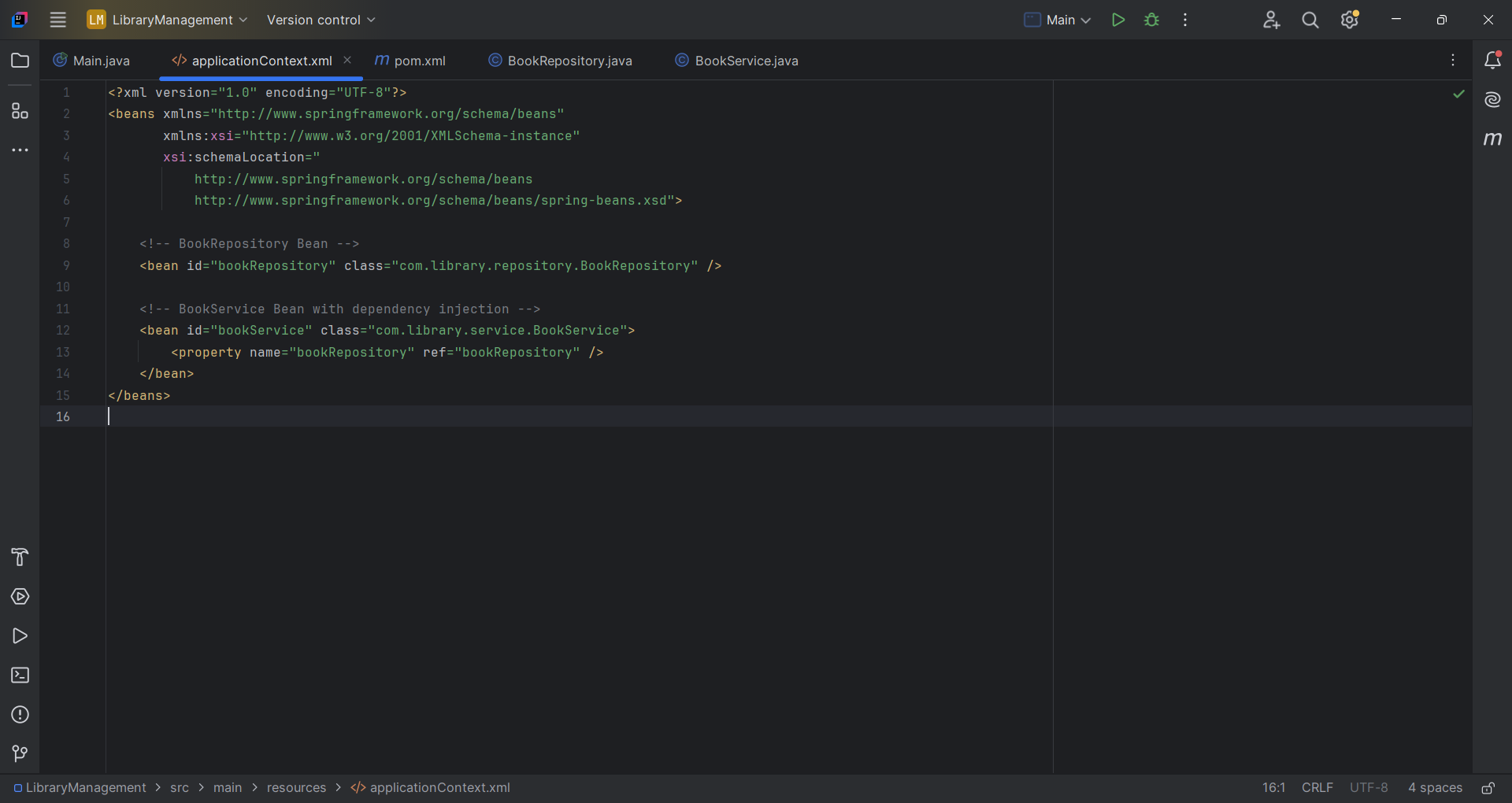
The library management application requires a central configuration for beans and dependencies.

**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:**

1. **Create Spring Configuration File:**

****

2. **Update the BookService Class:**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Setter method for DI

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void listBooks() {

System.out.println("BookService: Fetching books...");

bookRepository.displayBooks();

}

}

// BookRepository.java

package com.library.repository;

public class BookRepository {

public void displayBooks() {

System.out.println("BookRepository: Displaying books from the repository.");

}

}

//Main.java

package com.library;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Main {

public static void main(String[] args) {

// Load Spring context from XML

ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

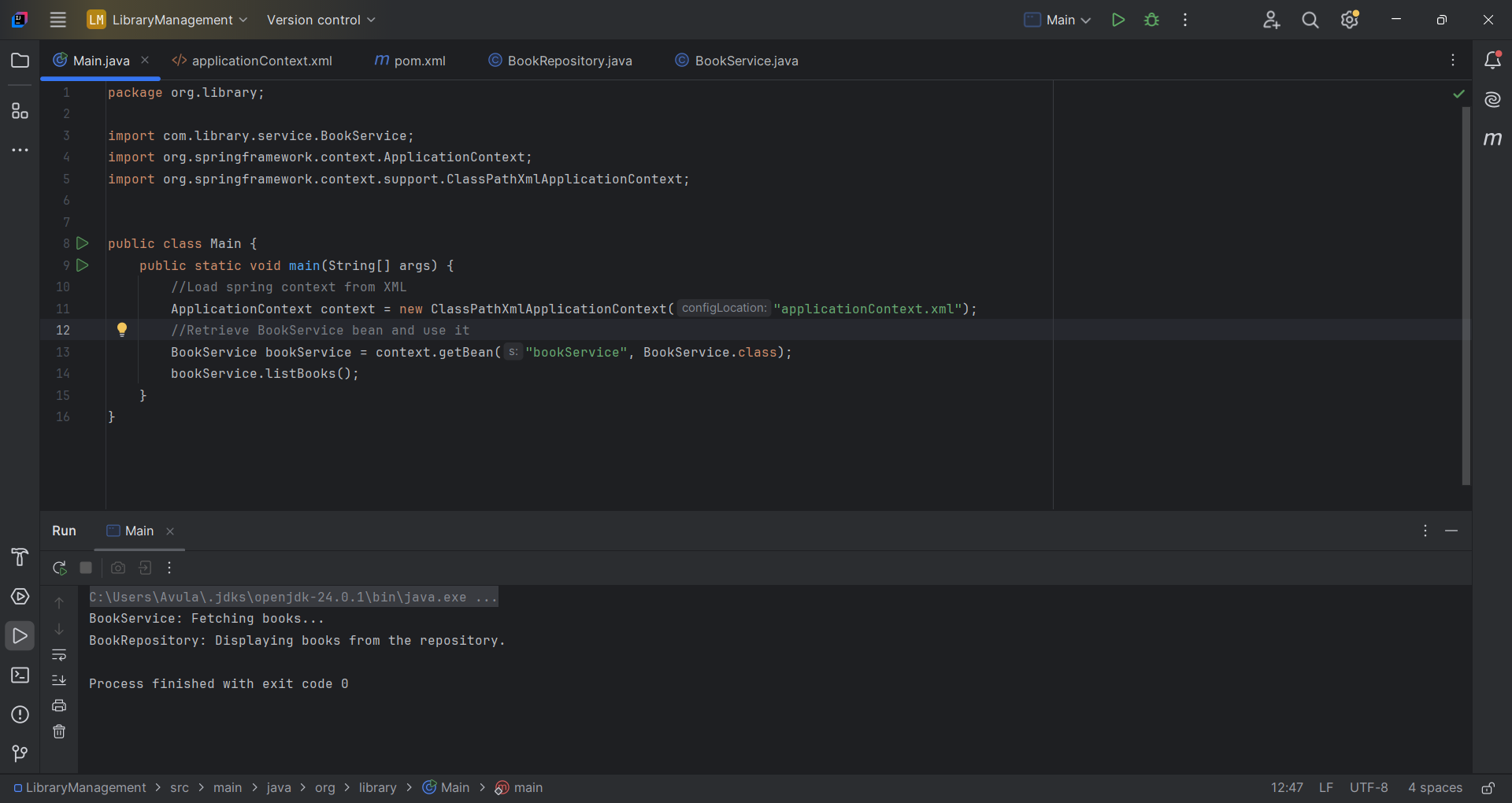
// Retrieve BookService bean and use it

BookService bookService = context.getBean("bookService", BookService.class);

bookService.listBooks();

}

}



**What Is Spring's IoC Container?**

* **IoC (Inversion of Control)** means the control of creating and injecting dependencies is given to a container (Spring).
* In Spring, the **IoC Container** is the **ApplicationContext** interface (or sometimes BeanFactory).

ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml"); in main.java file does:

* **Creates the Spring IoC container** (context) using:

ClassPathXmlApplicationContext (which implements ApplicationContext)

* **Reads the XML config file** applicationContext.xml from the src/main/resources folder
* **Creates and wires the beans** (e.g., BookService, BookRepository) as defined in XML
* **Manages the lifecycle** of those beans (singleton by default)
* **Injects dependencies** (DI) like:

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

This injects bookRepository into BookService using its **setter method**.