**Candidate Name**:- MEGHANA G K

**Superset ID**:- 6424295

**WEEK – 2 HANDS ON EXERCISE (JAVA FSE DEEPSKILLING)**

**(Spring Core and Maven)**

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

**Scenario:**

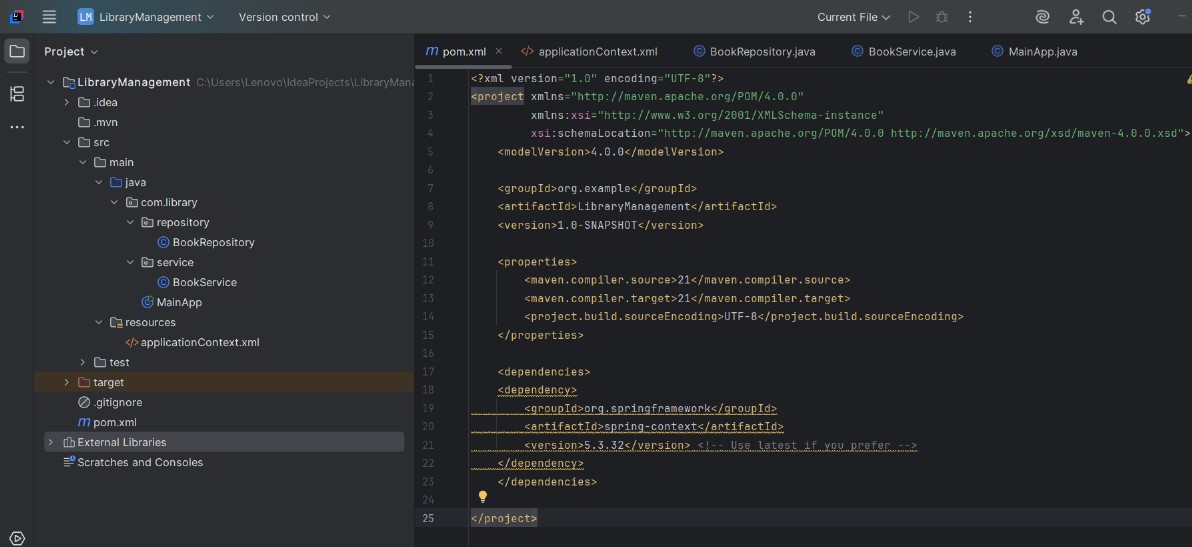
Your company is developing a web application for managing a library. You need to use the Spring Framework to handle the backend operations.

**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

**Dependency added in pom.xml :**



**Solution Code part :**

**1.applicationContext.xml**

<?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">  
  
 <!-- Define Repository Bean -->  
 <bean id="bookRepository" class="com.library.repository.BookRepository" />  
  
 <!-- Define Service Bean with DI (property injection) -->  
 <bean id="bookService" class="com.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository"/>  
 </bean>  
  
</beans>

**2. BookRepository.java :**

package com.library.repository;  
  
public class BookRepository {  
  
 public String fetchLatestBook() {  
 return " 'The Pragmatic Programmer' by Andrew Hunt & David Thomas";  
 }  
}

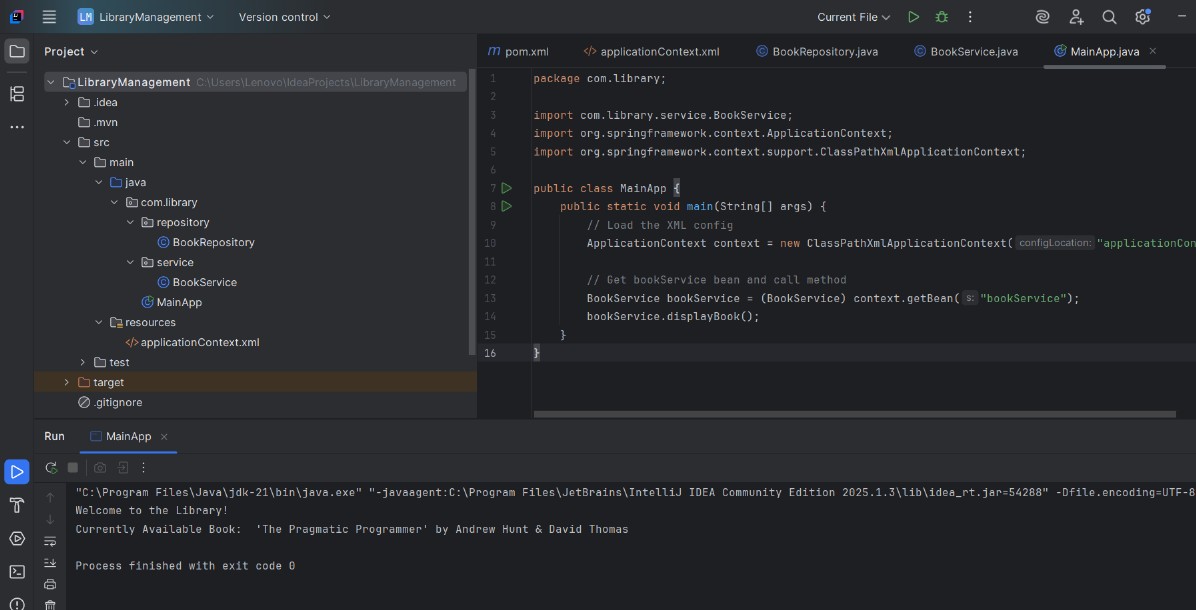
**3. BookService.java :**

package com.library.service;  
  
import com.library.repository.BookRepository;  
  
public class BookService {  
 private BookRepository bookRepository;  
  
 // Setter for Spring to inject BookRepository  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void displayBook() {  
 String book = bookRepository.fetchLatestBook();  
 System.*out*.println("Welcome to the Library!");  
  
 System.*out*.println("Currently Available Book: " + book);  
 }  
}

**4. MainApp.java :**

package com.library;  
  
import com.library.service.BookService;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class MainApp {  
 public static void main(String[] args) {  
 // Load the XML config  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 // Get bookService bean and call method  
 BookService bookService = (BookService) context.getBean("bookService");  
 bookService.displayBook();  
 }  
}

**Terminal Output :**



**Exercise 2 : Implementing Dependency Injection**

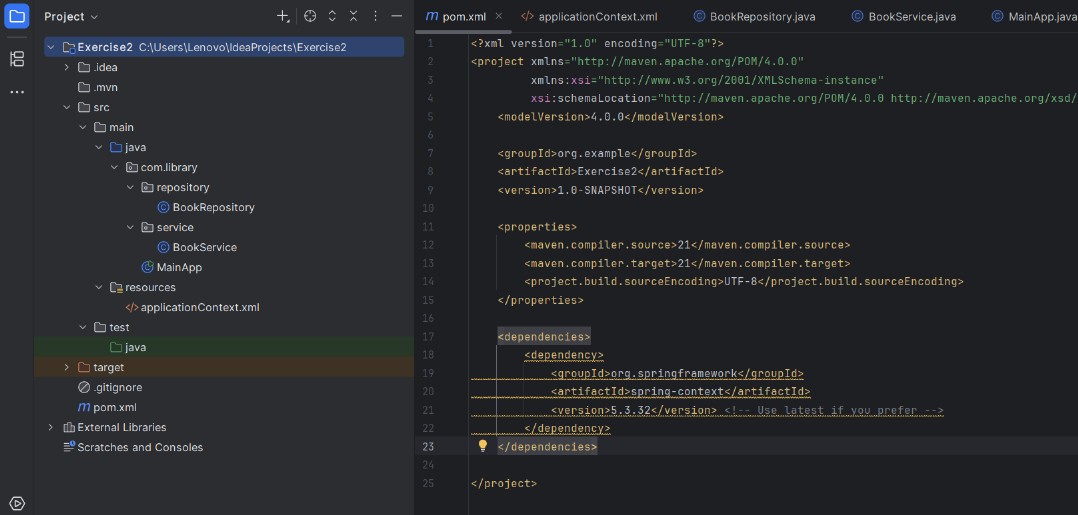
**Scenario:**

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.

**Dependency added in pom.xml :**



**Solution Code Part :**

**1.applicationContext.xml**

<?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  
 https://www.springframework.org/schema/beans/spring-beans.xsd">  
  
 <!-- Bean for BookRepository -->  
 <bean id="bookRepository" class="com.library.repository.BookRepository" />  
  
 <!-- Bean for BookService with Dependency Injection via setter -->  
 <bean id="bookService" class="com.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository" />  
 </bean>  
  
</beans>

**2. BookRepository.java :**

package com.library.repository;  
  
public class BookRepository {  
  
 public String getBookTitle() {  
 return "The Hobbit by J.R.R. Tolkien";  
 }  
}

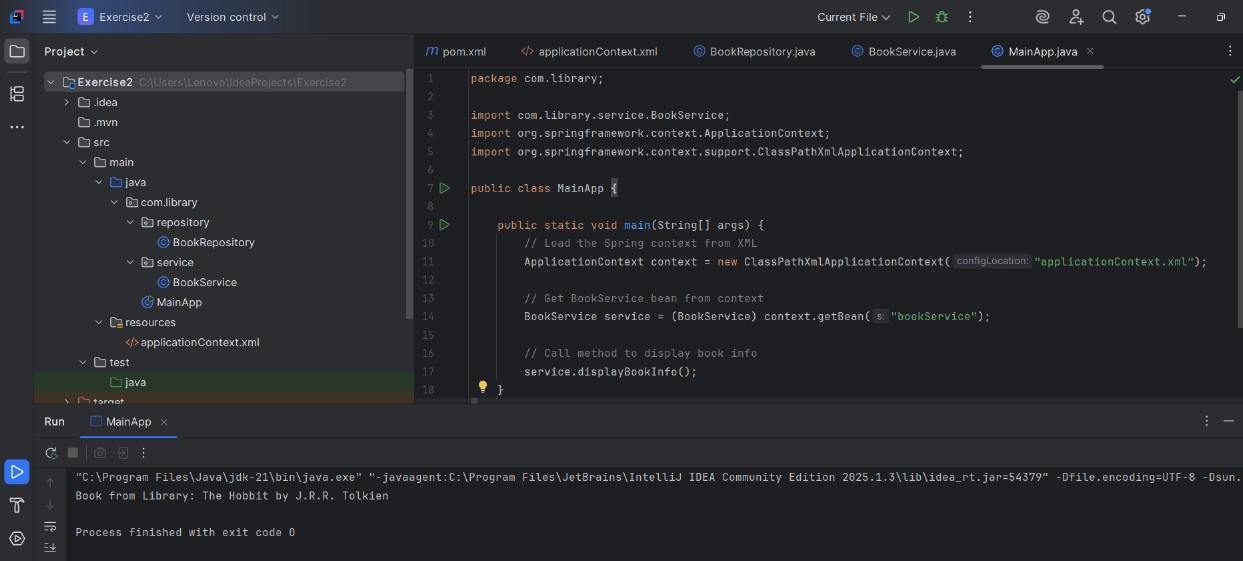
**3. BookService.java :**

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 displayBookInfo() {  
 String book = bookRepository.getBookTitle();  
 System.*out*.println("Book from Library: " + book);  
 }  
}

**4. MainApp.java :**

package com.library;  
  
import com.library.service.BookService;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class MainApp {  
  
 public static void main(String[] args) {  
 // Load the Spring context from XML  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 // Get BookService bean from context  
 BookService service = (BookService) context.getBean("bookService");  
  
 // Call method to display book info  
 service.displayBookInfo();  
 }  
}

**Terminal Output :**



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

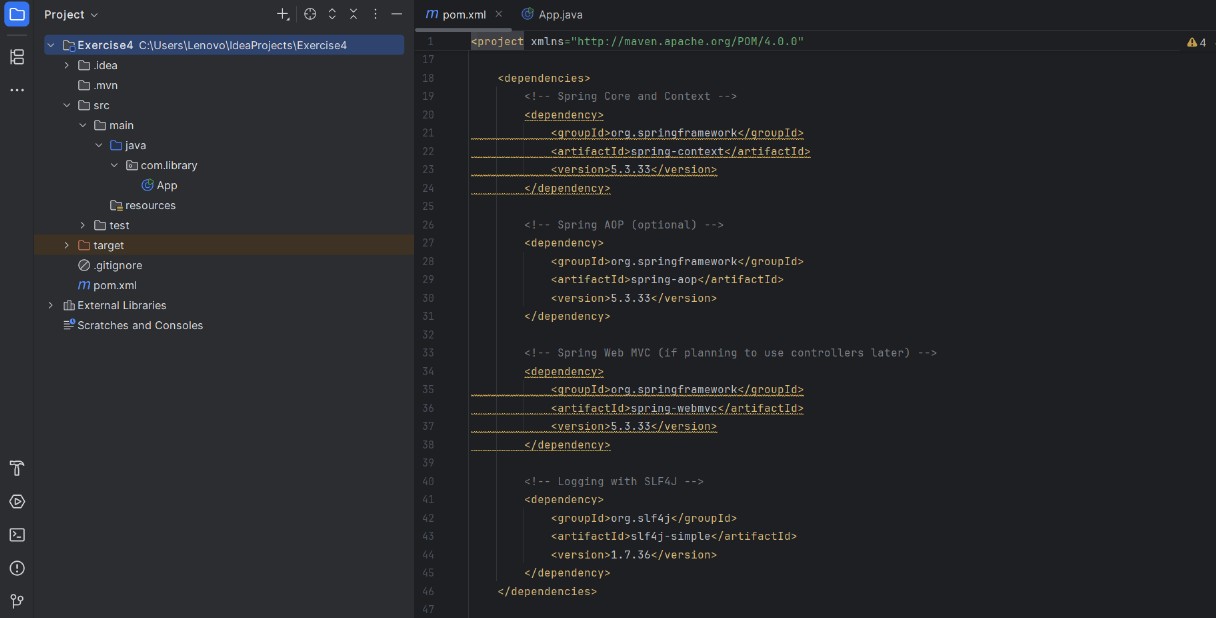
**Scenario:**

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

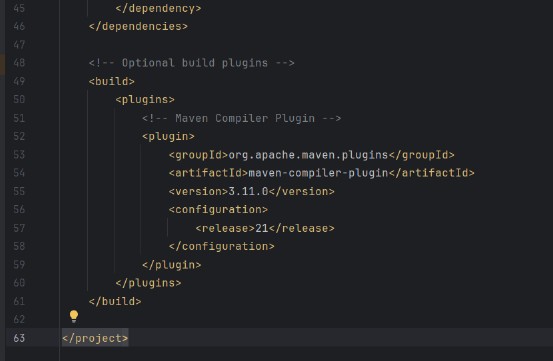
**Steps:**

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

**Dependency added in pom.xml :**



**Plugins added in pom.xml :**



**Solution Code Part :**

**1.App.java:**

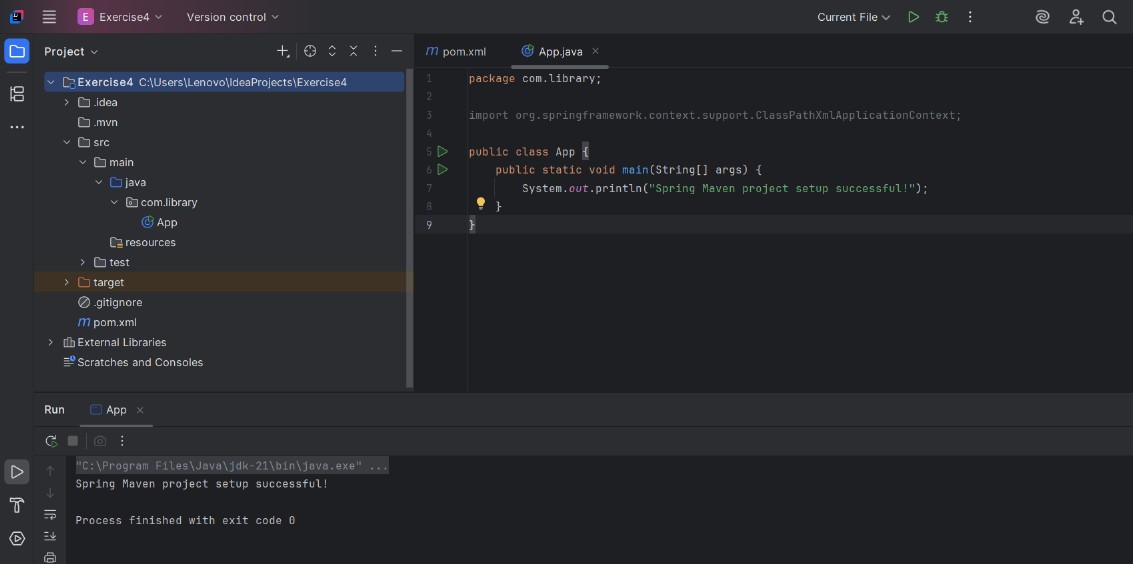
package com.library;  
  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class App

{  
 public static void main(String[] args)

{  
 System.*out*.println("Spring Maven project setup successful!");  
 }

}

**Terminal Output :**



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

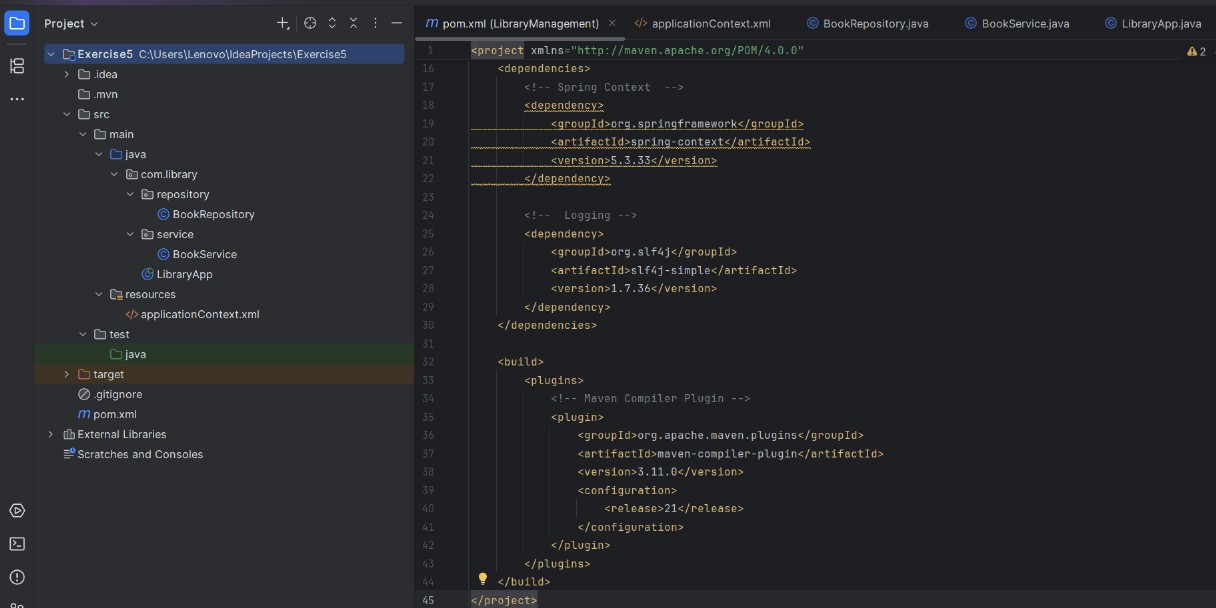
**Scenario:**

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.

**Dependency added in pom.xml :**



**Solution Code Part :**

**1.applicationContext.xml**

<?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  
 https://www.springframework.org/schema/beans/spring-beans.xsd">  
  
 <!-- Define BookRepository Bean -->  
 <bean id="bookRepository" class="com.library.repository.BookRepository" />  
  
 <!-- Define BookService Bean and inject BookRepository -->  
 <bean id="bookService" class="com.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository" />  
 </bean>  
  
</beans>

**2. BookRepository.java :**

package com.library.repository;  
  
public class BookRepository {  
 public String getBookTitle() {  
 return "Pride and Prejudice by Jane Austen";  
 }  
}

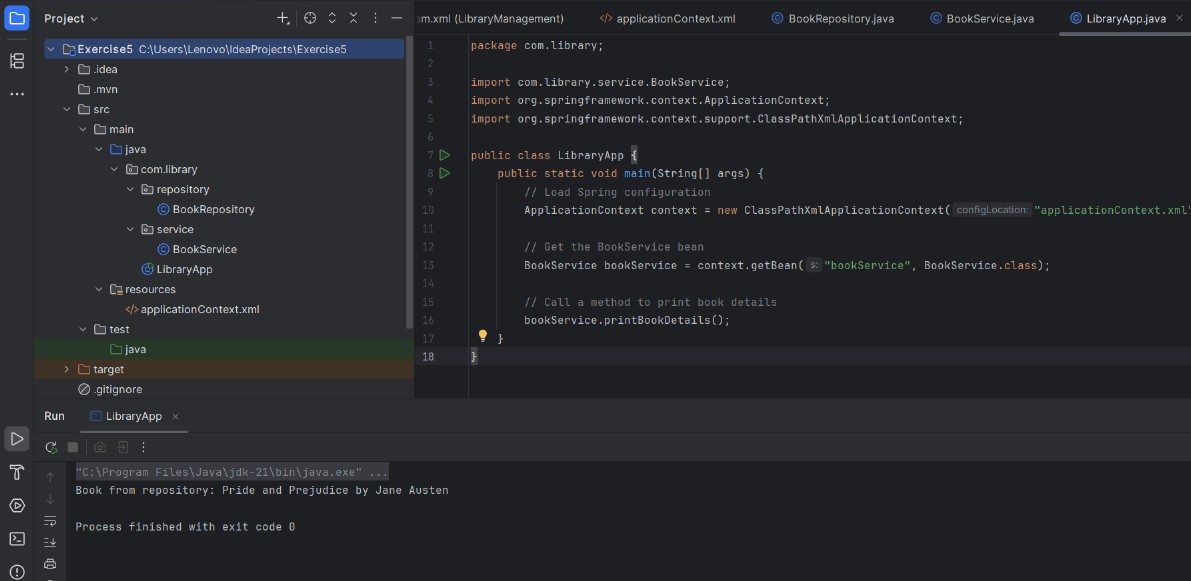
**3. BookService.java :**

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 printBookDetails() {  
 System.*out*.println("Book from repository: " + bookRepository.getBookTitle());  
 }  
}

**4. LibraryApp.java :**

package com.library;  
  
import com.library.service.BookService;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class LibraryApp {  
 public static void main(String[] args) {  
 // Load Spring configuration  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 // Get the BookService bean  
 BookService bookService = context.getBean("bookService", BookService.class);  
  
 // Call a method to print book details  
 bookService.printBookDetails();  
 }  
}

**Terminal Output :**



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

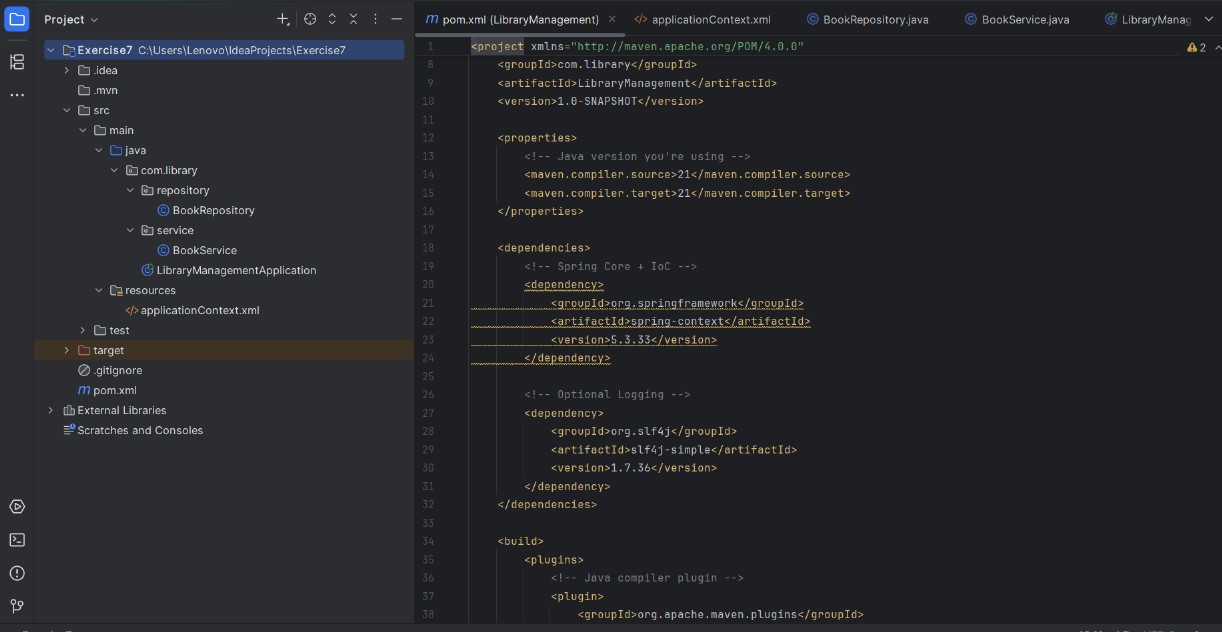
**Scenario:**

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.
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.

**Dependency added in pom.xml :**



**Solution Code Part :**

**1.applicationContext.xml**

<?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  
 https://www.springframework.org/schema/beans/spring-beans.xsd">  
  
 <!-- BookRepository bean -->  
 <bean id="bookRepository" class="com.library.repository.BookRepository" />  
  
 <!-- BookService bean with both constructor and setter injection -->  
 <bean id="bookService" class="com.library.service.BookService">  
 <!-- Constructor Injection: service name -->  
 <constructor-arg value="Main Library Service" />  
  
 <!-- Setter Injection: bookRepository -->  
 <property name="bookRepository" ref="bookRepository" />  
 </bean>  
  
</beans>

**2. BookRepository.java :**

package com.library.repository;  
  
public class BookRepository

{  
 public String getBookTitle()

{  
 return "The Great Gatsby by F. Scott Fitzgerald";  
 }  
}

**3. BookService.java :**

package com.library.service;  
  
import com.library.repository.BookRepository;  
  
public class BookService

{  
 private String serviceName;  
 private BookRepository bookRepository;  
  
 // Constructor Injection  
 public BookService(String serviceName)

{  
 this.serviceName = serviceName;  
 System.*out*.println("Constructor Injection: " + serviceName);  
 }  
  
 // Setter Injection  
 public void setBookRepository(BookRepository bookRepository)

{  
 this.bookRepository = bookRepository;  
 System.*out*.println("Setter Injection: BookRepository injected");  
 }  
  
 public void displayServiceInfo()

{  
 System.*out*.println("Service Name: " + serviceName);  
 System.*out*.println("Book: " + bookRepository.getBookTitle());  
 }  
 }

**4. LibraryManagementApplication.java :**

package com.library;  
  
import com.library.service.BookService;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class LibraryManagementApplication

{  
 public static void main(String[] args)

{  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 BookService bookService = context.getBean("bookService", BookService.class);  
  
 bookService.displayServiceInfo();  
 }  
}

**Terminal Output :**

