Spring Framework Exercises - Library Management System

# 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.

## 1. Set Up a Spring Project

Create a Maven project named LibraryManagement and add the following Spring Core dependencies in pom.xml:

<dependencies>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.20</version>  
 </dependency>  
</dependencies>

## 2. Configure the Application Context

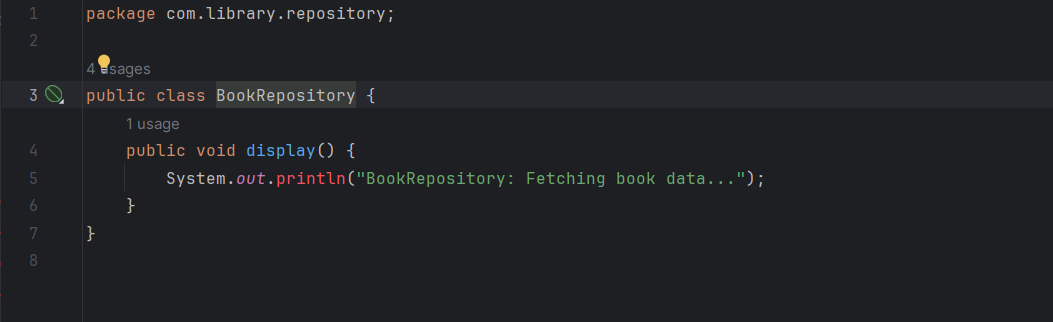
Create an XML configuration file named applicationContext.xml under src/main/resources with the following:

<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 id="bookRepository" class="com.library.repository.BookRepository"/>  
 <bean id="bookService" class="com.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository"/>  
 </bean>  
</beans>

## 3. Define Service and Repository Classes

BookRepository.java

package com.library.repository;  
  
public class BookRepository {  
 public void display() {  
 System.out.println("BookRepository: Fetching book data...");  
 }  
}



BookService.java

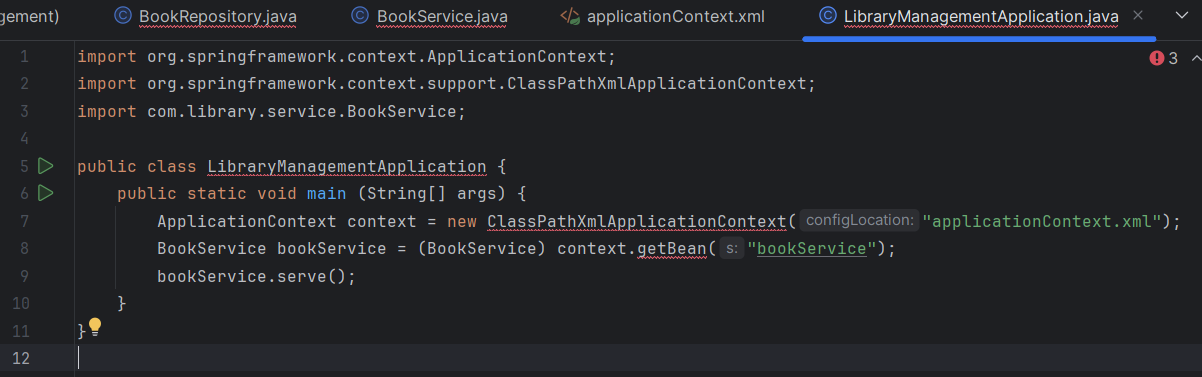
package com.library.service;  
  
import com.library.repository.BookRepository;  
  
public class BookService {  
 private BookRepository bookRepository;  
  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void serve() {  
 System.out.println("BookService: Service called");  
 bookRepository.display();  
 }  
}

## 

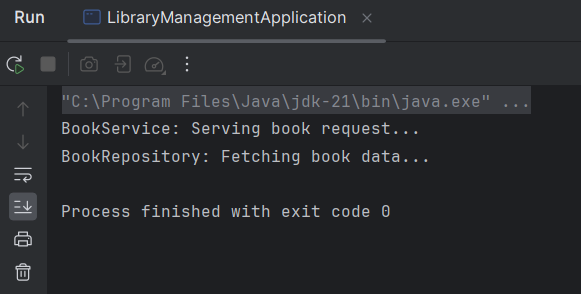
## 4. Run the Application

LibraryManagementApplication.java

import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
import com.library.service.BookService;  
  
public class LibraryManagementApplication {  
 public static void main(String[] args) {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
 BookService service = (BookService) context.getBean("bookService");  
 service.serve();  
 }  
}



## 4. 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 (Dependency Injection).

## 1. Modify the XML Configuration

Update applicationContext.xml to inject BookRepository into BookService using setter injection:

<bean id="bookRepository" class="com.library.repository.BookRepository"/>  
<bean id="bookService" class="com.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository"/>  
</bean>

## 2. Update the BookService Class

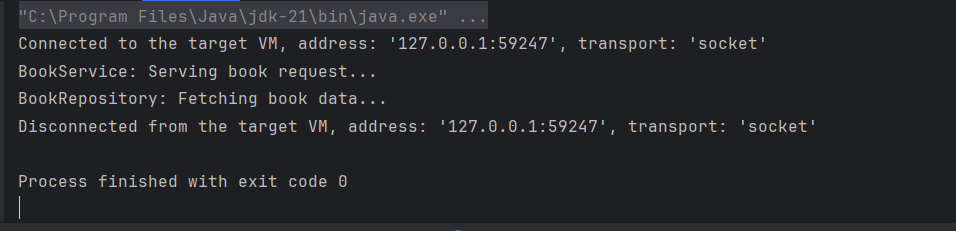
Ensure the BookService class has a setter method for BookRepository to allow Spring to inject it:

package com.library.service;  
  
import com.library.repository.BookRepository;  
  
public class BookService {  
 private BookRepository bookRepository;  
  
 // Setter for DI  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void serve() {  
 System.out.println("BookService: Serving book request...");  
 bookRepository.display();  
 }  
}

## 3. Test the Configuration

Create or update the main class LibraryManagementApplication to test the DI setup:

import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
import com.library.service.BookService;  
  
public class LibraryManagementApplication {  
 public static void main(String[] args) {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
 BookService bookService = (BookService) context.getBean("bookService");  
 bookService.serve();  
 }  
}



Exercise 4: Creating and Configuring a Maven Project

## 1. Create a Maven Project

Create a Maven project named LibraryManagement.

## 2. Add Spring Dependencies in pom.xml

<dependencies>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.20</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>5.3.20</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aop</artifactId>  
 <version>5.3.20</version>  
 </dependency>  
</dependencies>



## 3. Configure Maven Plugins

<build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.8.1</version>  
 <configuration>  
 <source>1.8</source>  
 <target>1.8</target>  
 </configuration>  
 </plugin>  
 </plugins>  
</build>

