Exercise 2: Implementing Dependency Injection

Subject: Spring Framework

Title: LibraryManagement - Dependency Injection using Spring

Submitted by: Likhitha Sri Dasari

# Objective

To demonstrate dependency injection using Spring's Inversion of Control (IoC) container in the LibraryManagement application by wiring BookRepository into BookService.

# Step 1: Modify the XML Configuration

The following is the updated `applicationContext.xml` to configure dependency injection:

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

# Step 2: Update the BookService Class

Ensure that the BookService class includes a setter method for BookRepository:

package com.library.service;  
  
import com.library.repository.BookRepository;  
  
public class BookService {  
 private BookRepository bookRepository;  
  
 // Setter-based Dependency Injection  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void showBookInfo() {  
 System.out.println("BookService: Calling repository...");  
 bookRepository.displayBookInfo();  
 }  
}

# Step 3: Test the Configuration

The following Java class tests the DI setup by loading the Spring context and calling a method on BookService:

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 = (BookService) context.getBean("bookService");  
 bookService.showBookInfo();  
 }  
}

## Output

BookService: Calling repository...  
BookRepository: Fetching book data...

# Conclusion

This exercise illustrates how Spring's IoC container enables loose coupling between components through dependency injection. BookService is no longer responsible for instantiating BookRepository, making the system more modular and testable.