Spring Core Maven Exercises

# 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. Maven Configuration (pom.xml)

<project xmlns="http://maven.apache.org/POM/4.0.0" ...>  
 <modelVersion>4.0.0</modelVersion>  
 <groupId>com.library</groupId>  
 <artifactId>LibraryManagement</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.33</version>  
 </dependency>  
 </dependencies>  
</project>

## 2. XML Configuration (applicationContext.xml)

<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. Java Classes

BookRepository.java

package com.library.repository;  
  
public class BookRepository {  
 public void saveBook(String title) {  
 System.out.println("Book '" + title + "' saved.");  
 }  
}

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 addBook(String title) {  
 bookRepository.saveBook(title);  
 }  
}

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 service = context.getBean("bookService", BookService.class);  
 service.addBook("The Alchemist");  
 }  
}

Expected Output:

# Exercise 2: Implementing Dependency Injection

Already accomplished in Exercise 1 using setter-based Dependency Injection. Bean was wired via XML configuration.

# Exercise 4: Creating and Configuring a Maven Project

Project configuration and setup steps were covered in Exercise 1. Below is a complete Maven configuration with compiler plugin.

## Updated pom.xml with Maven Plugin

<build>  
 <plugins>  
 <plugin>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.10.1</version>  
 <configuration>  
 <source>1.8</source>  
 <target>1.8</target>  
 </configuration>  
 </plugin>  
 </plugins>  
</build>  
