**[Week 3] [Spring Core & Maven] HandsOn**

**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 backend operations.

**Steps:**

1. **Create a Maven project** named LibraryManagement.
2. **Add Spring Core dependencies** in pom.xml:

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.27</version>

</dependency>

</dependencies>

1. **Create 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="bookService" class="com.library.service.BookService"/>

<bean id="bookRepository" class="com.library.repository.BookRepository"/>

</beans>

1. **BookService.java:**

package com.library.service;

public class BookService {

public void display() {

System.out.println("BookService is working...");

}

}

1. **BookRepository.java:**

package com.library.repository;

public class BookRepository {

public void saveBook() {

System.out.println("Book saved!");

}

}

1. **Main class to test configuration:**

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

service.display();

}

}

**Exercise 2: Implementing Dependency Injection**

**Scenario:**  
In the library management application, you need to connect BookService and BookRepository using Spring's Dependency Injection.

**Steps:**

1. **Modify applicationContext.xml to wire dependencies:**

<bean id="bookRepository" class="com.library.repository.BookRepository"/>

<bean id="bookService" class="com.library.service.BookService">

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

</bean>

1. **Update 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 display() {

System.out.println("BookService is working...");

bookRepository.saveBook();

}

}

1. **Test using the same main class (LibraryManagementApplication).**

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

**Scenario:**  
You want to set up a new Maven project for the library management application.

**Steps:**

1. **Create a new Maven project** named LibraryManagement.
2. **Add dependencies in pom.xml:**

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.27</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.27</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.27</version>

</dependency>

</dependencies>

1. **Configure Maven Compiler Plugin:**

<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>