# Skill – SpringCore 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.

## Implementation:

## Codes: pom.xml

**<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">**

**<modelVersion>4.0.0</modelVersion>**

**<groupId>com.library</groupId>**

**<artifactId>LibraryManagement</artifactId>**

**<version>0.0.1-SNAPSHOT</version>**

**<dependencies>**

**<!-- Spring Core Context -->**

**<dependency>**

**<groupId>org.springframework</groupId>**

**<artifactId>spring-context</artifactId>**

**<version>5.3.21</version>**

**</dependency>**

**</dependencies>**

**<build>**

**<plugins>**

**<plugin>**

**<artifactId>maven-compiler-plugin</artifactId>**

**<version>3.8.1</version>**

**<configuration>**

**<source>1.8</source>**

**<target>1.8</target>**

**</configuration>**

**</plugin>**

**</plugins>**

**</build>**

**</project>**

## BookRepository.java

**package** com.library.repository;

**public** **class** BookRepository {

**public** **void** saveBook() {

System.***out***.println("Book saved to the repository.");

}

}

## 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() {

bookRepository.saveBook();

}

}

## 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">**

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

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

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

**</bean>**

**</beans>**

## 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");

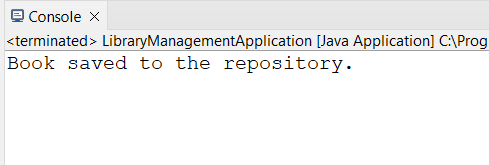
BookService bookService = context.getBean("bookService", BookService.**class**);

bookService.addBook();

}

}

## 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.

## Implementation: Codes which has to be updated in exercise 1 for Dependency Injection:

## BookService.java

## Update: Add setter method & use BookRepository:

**package com.library.service;**

**import com.library.repository.BookRepository;**

**public class BookService {**

**private BookRepository bookRepository;**

**// This setter method enables setter-based DI**

**public void setBookRepository(BookRepository bookRepository) {**

**this.bookRepository = bookRepository;**

**}**

**public void addBook() {**

**// Now using BookRepository to perform logic**

**bookRepository.saveBook();**

**}**

**}**

## applicationContext.xml

## Update: Wire the dependency using <property>:

**<?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 the repository bean -->**

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

**<!-- Define the service bean and inject the repository -->**

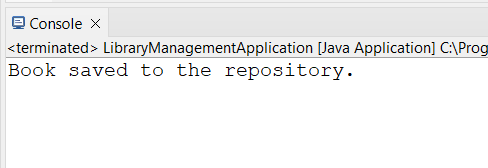
**<bean id="bookService" class="com.library.service.BookService">**

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

**</bean>**

**</beans>**

## 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.

## Implementation:

## Codes: pom.xml

## <project xmlns="http://maven.apache.org/POM/4.0.0"

## xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

## xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

## https://maven.apache.org/xsd/maven-4.0.0.xsd">

## <modelVersion>4.0.0</modelVersion>

## <groupId>com.library</groupId>

## <artifactId>LibraryManagement</artifactId>

## <version>0.0.1-SNAPSHOT</version>

## <dependencies>

## <!-- Spring Context -->

## <dependency>

## <groupId>org.springframework</groupId>

## <artifactId>spring-context</artifactId>

## <version>5.3.21</version>

## </dependency>

## <!-- Spring AOP -->

## <dependency>

## <groupId>org.springframework</groupId>

## <artifactId>spring-aop</artifactId>

## <version>5.3.21</version>

## </dependency>

## <!-- Spring WebMVC -->

## <dependency>

## <groupId>org.springframework</groupId>

## <artifactId>spring-webmvc</artifactId>

## <version>5.3.21</version>

## </dependency>

## </dependencies>

## <build>

## <plugins>

## <!-- Maven Compiler Plugin -->

## <plugin>

## <artifactId>maven-compiler-plugin</artifactId>

## <version>3.8.1</version>

## <configuration>

## <source>1.8</source>

## <target>1.8</target>

## </configuration>

## </plugin>

## </plugins>

## </build>

## </project>

## TestBean.java

**package com.library;**

**public class TestBean {**

**public void hello() {**

**System.out.println("Spring context loaded successfully!");**

**}**

**}**

## MainApp.java

**package com.library;**

**import org.springframework.context.ApplicationContext;**

**import org.springframework.context.support.ClassPathXmlApplicationContext;**

**public class MainApp {**

**public static void main(String[] args) {**

**ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");**

**TestBean testBean = (TestBean) context.getBean("testBean");**

**testBean.hello();**

**}**

**}**

## 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 a test bean -->**

**<bean id="testBean" class="com.library.TestBean"/>**

**</beans>**

## Output:

