**Exercise 1: Configuring a Basic Spring Application**

**CODE:**  
File > New > Project > Maven Project

* **Group Id:** com.library
* **Artifact Id:** LibraryManagement
* **Version:** 0.0.1-SNAPSHOT (default is fine)
* **Packaging:** jar (default)

Add dependencies in pox.xml

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.30</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.30</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.30</version>

</dependency>

</dependencies>

Navigate to src/main/resources and create 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"*>

<!-- BookRepository Bean -->

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

<!-- BookService Bean with dependency injection -->

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

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

</bean>

</beans>

**BookRepository**

1. Right-click src/main/java > New > Package: com.library.repository
2. Right-click the package > New > Class: BookRepository
3. Add:

package com.library.repository;

public class BookRepository {

public void saveBook(String bookName) {

System.out.println("Book '" + bookName + "' saved to the repository.");

}

}

**BookService**

1. Create another package: com.library.service
2. Right-click the package > New > Class: BookService
3. Add:

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Setter for Dependency Injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook(String bookName) {

bookRepository.saveBook(bookName);

}

}

### Create Main Class to load Spring context and test

1. Create a package: com.library.main
2. Right-click > New > Class: LibraryApp
3. Add:

package com.library.main;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class LibraryApp {

public static void main(String[] args) {

// Load Spring context from XML

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

// Retrieve BookService bean

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

// Test the configuration

bookService.addBook("Clean Code by Robert C. Martin");

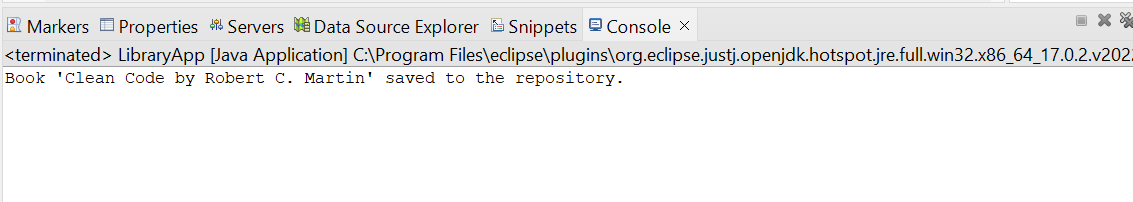
// Close context if needed

((ClassPathXmlApplicationContext) context).close();

}

}

**Output:**



**Exercise 2: Implementing Dependency Injection**

**CODE:**

**Open 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

https://www.springframework.org/schema/beans/spring-beans.xsd">

<!-- BookRepository Bean -->

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

<!-- BookService Bean with Dependency Injection -->

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

<!-- Inject bookRepository bean using setter injection -->

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

</bean>

</beans>

Ensure that BookService.java class has a setter method

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Required setter for Dependency Injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook(String bookName) {

bookRepository.saveBook(bookName);

}

}

Run your **LibraryApp** main class to verify:

java

CopyEdit

package com.library.main;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class LibraryApp {

public static void main(String[] args) {

// Load Spring context

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

// Get the bookService bean

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

// Test the dependency injection

bookService.addBook("The Pragmatic Programmer");

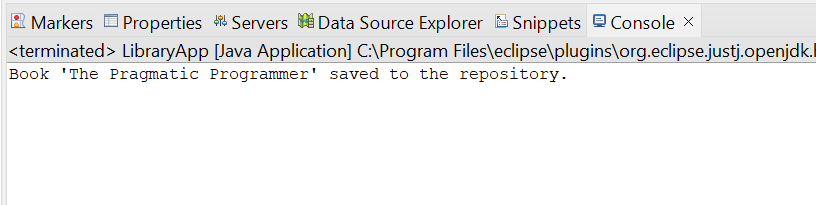
// Close context

((ClassPathXmlApplicationContext) context).close();

}

}

**Output:**



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

**CODE:**

Go to: File > New > Project > Maven Project

* **Group Id:** com.library
* **Artifact Id:** LibraryManagement
* **Version:** 0.0.1-SNAPSHOT (default)
* **Packaging:** jar
* **Name:** LibraryManagement

Add the following dependencies in pom.xml

<dependencies>

<!-- Spring Context for Core Container and Dependency Injection -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.38</version>

</dependency>

<!-- Spring AOP for Aspect-Oriented Programming -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.38</version>

</dependency>

<!-- Spring WebMVC for building web applications -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.38</version>

</dependency>

</dependencies>

Add the following plugins in pom.xml

<build>

<plugins>

<!-- Maven Compiler Plugin -->

<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>

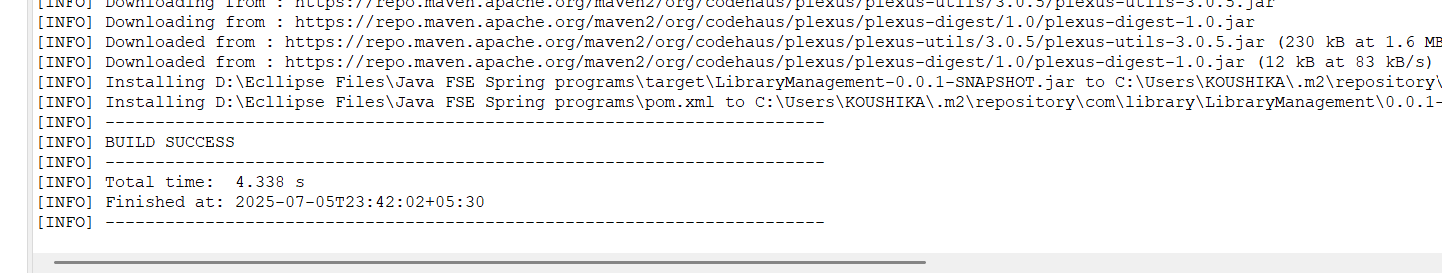
Right-click on the project > Run As > Maven build.  
Enter: clean install

If the pom.xml and setup are correct, Eclipse’s console will show the BUILD SUCCESS message and downloads all the required .jar files.

spring-context-5.3.38.jar

spring-aop-5.3.38.jar

spring-webmvc-5.3.38.jar

**Output:**  
****