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

**1. Set Up a Spring Maven Project**

**➤ pom.xml**

Create a Maven project named LibraryManagement, and use the following minimal pom.xml:

xml

CopyEdit

<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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- Spring Core Dependency -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.33</version>

</dependency>

</dependencies>

</project>

**2. Configure the Application Context**

**➤ Directory structure:**

Create the file:  
src/main/resources/applicationContext.xml

**➤ applicationContext.xml**

xml

CopyEdit

<?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 for BookRepository -->

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

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

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

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

</bean>

</beans>

**3. Define Service and Repository Classes**

**➤ BookRepository.java**

Location: src/main/java/com/library/repository/BookRepository.java

java

CopyEdit

package com.library.repository;

public class BookRepository {

public String getBook() {

return "Effective Java by Joshua Bloch";

}

}

**➤ BookService.java**

Location: src/main/java/com/library/service/BookService.java

java

CopyEdit

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Setter for DI

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void displayBook() {

System.out.println("Book: " + bookRepository.getBook());

}

}

**4. Run the Application**

**➤ Main class to load Spring context**

Location: src/main/java/com/library/MainApp.java

java

CopyEdit

package com.library;

import com.library.service.BookService;

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");

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

bookService.displayBook();

}

}

**Exercise 2: Implementing Dependency Injection**

**1. Modify the XML Configuration**

Ensure your applicationContext.xml file (in src/main/resources) correctly **injects BookRepository into BookService** using **setter injection**.

**🔹 applicationContext.xml**

xml

CopyEdit

<?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 for BookRepository -->

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

<!-- Bean for BookService with DI via setter -->

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

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

</bean>

</beans>

**2. Update the BookService Class**

Make sure BookService.java has a **setter method** to accept a BookRepository object.

**🔹 BookService.java**

java

CopyEdit

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Setter for DI

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void displayBook() {

System.out.println("Book: " + bookRepository.getBook());

}

}

**3. Test the Configuration**

Use the main application class to **load the Spring container** and test the dependency injection.

**🔹 MainApp.java**

java

CopyEdit

package com.library;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

public static void main(String[] args) {

// Load Spring ApplicationContext from XML

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

// Get the BookService bean

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

// Call method to check if DI worked

bookService.displayBook();

}

}

**Expected Output**

When you run MainApp.java, you should see:

vbnet

CopyEdit

Book: Effective Java by Joshua Bloch

This confirms that:

* Spring successfully created the beans
* BookRepository was injected into BookService using setter injection

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

**1. Create a New Maven Project**

In **Eclipse** or any IDE:

* File → New → Maven Project
* Choose maven-archetype-quickstart
* Set:
  + **GroupId**: com.library
  + **ArtifactId**: LibraryManagement
* Finish

This creates the basic Maven structure.

**2. Add Spring Dependencies to pom.xml**

Open pom.xml and add the following dependencies for:

* **Spring Context**: Core functionality and DI
* **Spring AOP**: Aspect-oriented programming
* **Spring WebMVC**: For creating web applications

xml

CopyEdit

<dependencies>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.33</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.33</version>

</dependency>

<!-- Spring Web MVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.33</version>

</dependency>

<!-- Servlet API (for Web MVC) -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>4.0.1</version>

<scope>provided</scope>

</dependency>

</dependencies>

**3. Configure Maven Compiler Plugin**

Still in pom.xml, configure the **Maven Compiler Plugin** for **Java 1.8**:

xml

CopyEdit

<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>

**Final pom.xml Example**

xml

CopyEdit

<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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<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>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.33</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.33</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>4.0.1</version>

<scope>provided</scope>

</dependency>

</dependencies>

<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>

</project>

**After Setup**

* Right-click project → Maven → Update Project
* Build and run to ensure no errors
* You are now ready to start using **Spring MVC**, **IoC**, and **AOP in your app**