# Exercise 1 – Configuring a Basic Spring Application

## 1. Objective

To create a basic Spring application for managing a library, using XML configuration for beans and a simple service-repository structure.

## 2. Problem Statement / Scenario

Your company is developing a web application for managing a library. You need to use the Spring Framework to handle the backend operations.

## 3. Approach / Steps

3.1 Set Up a Spring Project

- Create a Maven project named `LibraryManagement`.

- Add Spring Core dependency (`spring-context`) in the `pom.xml`.

3.2 Configure the Application Context

- Create an XML configuration file named `applicationContext.xml` in the `src/main/resources` directory.

- Define beans for `BookService` and `BookRepository`.

3.3 Define Service and Repository Classes

- Create a package `com.library.service` and add a class `BookService`.

- Create a package `com.library.repository` and add a class `BookRepository`.

- Implement a method in `BookService` that prints books retrieved from `BookRepository`.

3.4 Run the Application

- Create a main class `LibraryApp` to load the Spring context and invoke the service method.

## 4. Code

\*\*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>1.0.0-SNAPSHOT</version>

<properties>

<spring.version>6.2.8</spring.version>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

</dependencies>

</project>

\*\*applicationContext.xml\*\*

<beans xmlns="https://www.springframework.org/schema/beans" ...>

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

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

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

</bean>

</beans>

\*\*BookRepository.java\*\*

package com.library.repository;

import java.util.Arrays;

import java.util.List;

public class BookRepository {

public List<String> findAll() {

return Arrays.asList("Effective Java", "Clean Code", "Spring in Action");

}

}

\*\*BookService.java\*\*

package com.library.service;

import com.library.repository.BookRepository;

import java.util.List;

public class BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void listBooks() {

List<String> books = bookRepository.findAll();

books.forEach(book -> System.out.println("• " + book));

}

}

\*\*LibraryApp.java\*\*

package com.library;

import com.library.service.BookService;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryApp {

public static void main(String[] args) {

try (ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml")) {

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

System.out.println("Books in the library:");

service.listBooks();

}

}

}

## 5. Output Verification

- The application compiles successfully.

- On running the `LibraryApp`, the console prints:

Books in the library:

• Effective Java

• Clean Code

• Spring in Action

## 6. Conclusion

This exercise demonstrated how to set up a basic Spring application using XML configuration. We created service and repository beans and integrated them using the Spring IoC container. The application outputs a list of books, confirming successful configuration and dependency injection.