**Implementing Dependency Injection**

**Spring Core\_Maven**

Vaishnavi

28/6/2025

In enterprise applications, managing dependencies manually becomes cumbersome and leads to tightly coupled code. Spring Framework addresses this issue with **Inversion of Control (IoC)** and **Dependency Injection (DI)**. In this exercise, we enhance the Library Management application by implementing DI between BookService and BookRepository using **XML-based Spring configuration**. This allows Spring to manage the lifecycle and wiring of components automatically, promoting clean, testable, and maintainable code.

**Objective:**

Understand and apply **Setter-based Dependency Injection** using Spring’s IoC container.

Modify the applicationContext.xml file to define bean dependencies.

Ensure that BookService receives an instance of BookRepository through Spring.

Test and validate that the injection is working as expected by running the main class.

**Implementation:**

### Setup a Maven Project

### ****Open IntelliJ IDEA****

### ****Create a New Project****:

Choose **Maven** > Click **Next**

Project Name: LibraryManagement

GroupId: com.library

ArtifactId: LibraryManagement

Click **Finish**

## Add Spring Core Dependencies in pom.xml

<dependencies>

<!-- Spring Core -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.36</version> <!-- use a compatible version -->

</dependency>

</dependencies>

**Create an 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">

<!-- Repository Bean -->

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

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

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

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

</bean>

</beans>

**Define BookService and BookRepository Classes**

**BookRepository**

package com.library.repository;

public class BookRepository {

public void saveBook(String bookName) {

System.out.println("Saving book: " + bookName);

}

}

**BookService**

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);

}

}

**MainApp**

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");

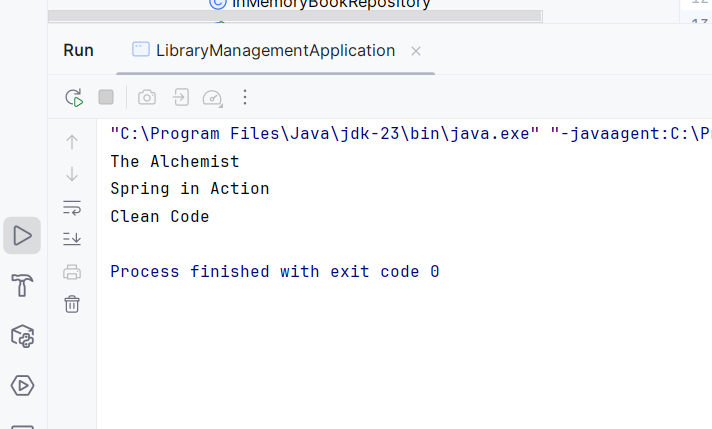
BookService bookService = context.getBean("bookService", BookService.class);

bookService.addBook("The Alchemist");

}

}

**Output:**

****