**Creating and Configuring a Maven Project**

**Spring Core\_Maven**

Vaishnavi

28/6/2025

In Java-based enterprise development, **Maven** is a powerful build automation and dependency management tool. It helps developers manage libraries, plugins, project structure, and lifecycle with ease. In this exercise, you will set up a Maven project for the Library Management system and configure it to use key Spring modules like **Spring Context**, **Spring AOP**, and **Spring WebMVC**. This provides a strong foundation for building scalable and modular web applications.

**Objective:**

Understand and apply **Setter-based Dependency Injection** using Spring’s · Set up a **Maven-based Java project** named LibraryManagement.

Add necessary **Spring dependencies** for core features, AOP (Aspect-Oriented Programming), and MVC-based web components.

Configure the **Maven Compiler Plugin** to compile the code using Java 1.8, ensuring compatibility with Spring.

**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

<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>

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.36</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.36</version>

</dependency>

<!-- Spring WebMVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.36</version>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Compiler Plugin for Java 1.8 -->

<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>

**MainApp.java:**

package org.example;

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.displayBooks();

}

}

**Book.java:**

package org.example;

public class Book {

private String title;

private String author;

public Book(String title, String author) {

this.title = title;

this.author = author;

}

public String getTitle() {

return title;

}

public String getAuthor() {

return author;

}

}

**BookRepository.java:**

package org.example;

import java.util.ArrayList;

import java.util.List;

public class BookRepository {

public List<Book> getAllBooks() {

List<Book> books = new ArrayList<>();

books.add(new Book("Effective Java", "Joshua Bloch"));

books.add(new Book("Spring in Action", "Craig Walls"));

books.add(new Book("Clean Code", "Robert C. Martin"));

books.add(new Book("Head First Design Patterns", "Eric Freeman"));

books.add(new Book("Java Concurrency in Practice", "Brian Goetz"));

books.add(new Book("Pro Spring 5", "Iuliana Cosmina"));

books.add(new Book("Beginning Spring Boot 3", "Jude Joseph"));

books.add(new Book("Introduction to Algorithms", "Thomas H. Cormen"));

books.add(new Book("Refactoring", "Martin Fowler"));

books.add(new Book("The Pragmatic Programmer", "Andrew Hunt"));

return books;

}

}

**BookService.java:**

package org.example;

import java.util.List;

public class BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void displayBooks() {

printBanner();

List<Book> books = bookRepository.getAllBooks();

System.out.printf("%-30s | %-20s%n", "Title", "Author");

for (Book book : books) {

System.out.printf("%-30s | %-20s%n", book.getTitle(), book.getAuthor());

}

System.out.println("Total Books: " + books.size());

}

private void printBanner() {

System.out.println(" WELCOME TO SPRING LIBRARY MANAGEMENT SYSTEM ");

}

}

**LoggingAspect.java:**

package org.example;

import org.aspectj.lang.JoinPoint;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Before;

@Aspect

public class LoggingAspect {

@Before("execution(\* org.example.BookService.displayBooks(..))")

public void logBeforeDisplay(JoinPoint joinPoint) {

System.out.println("LOG: " + joinPoint.getSignature().getName() + "() method is about to be called.");

}

}

**Application.properties:**

<?xml version="1.0" encoding="UTF-8"?>

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

xmlns:aop="http://www.springframework.org/schema/aop"

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

http://www.springframework.org/schema/aop

http://www.springframework.org/schema/aop/spring-aop.xsd">

<aop:aspectj-autoproxy />

<bean id="bookRepository" class="org.example.BookRepository" />

<bean id="bookService" class="org.example.BookService">

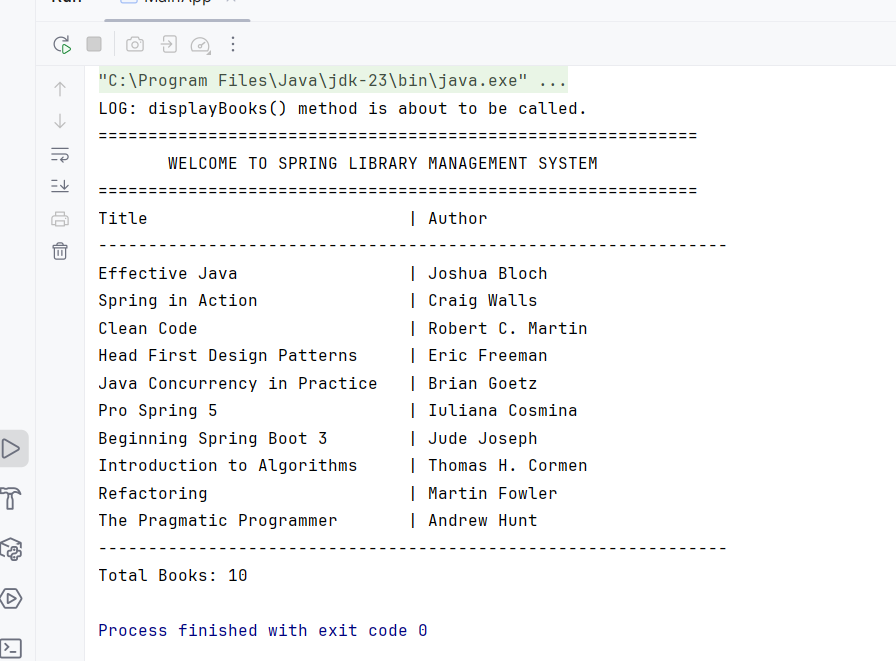
<property name="bookRepository" ref="bookRepository" />

</bean>

<bean id="loggingAspect" class="org.example.LoggingAspect" />

</beans>

**Output:**

****