**WEEK 3 HANDS ON EXERCISE – SPRING CORE AND MAVEN**

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

**PROBLEM STATEMENT:**

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

**CODE :**

**Main.java**

package com.library.main;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Main

{

public static void main(String[] args)

{

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

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

System.***out***.println("\n========= Library Management System =========\n");

bookService.addBook("The Great Gatsby");

bookService.addBook("The Alchemist");

bookService.addBook("1984");

bookService.updateBook("1984", "Nineteen Eighty-Four");

bookService.getBookDetails("The Alchemist");

bookService.removeBook("The Alchemist");

bookService.listAllBooks();

System.***out***.println("\n========= Operations Completed =========");

}

}

**BookRepository.java**

package com.library.repository;

import java.util.ArrayList;

import java.util.List;

public class BookRepository {

private final List<String> books = new ArrayList<>();

public void saveBook(String bookName) {

books.add(bookName);

System.***out***.println("Book saved: \"" + bookName + "\"");

}

public void deleteBook(String bookName) {

if (books.remove(bookName)) {

System.***out***.println("Book removed: \"" + bookName + "\"");

} else {

System.***out***.println("Book not found: \"" + bookName + "\"");

}

}

public void updateBook(String oldName, String newName) {

int index = books.indexOf(oldName);

if (index != -1) {

books.set(index, newName);

System.***out***.println("Book updated from \"" + oldName + "\" to \"" + newName + "\"");

} else {

System.***out***.println("Book not found: \"" + oldName + "\"");

}

}

public String findBook(String bookName) {

if (books.contains(bookName)) {

return "Book found: \"" + bookName + "\"";

} else {

return "Book not found: \"" + bookName + "\"";

}

}

public List<String> findAllBooks() {

return new ArrayList<>(books);

}

}

**BookService.java**

package com.library.service;

import com.library.repository.BookRepository;

import java.util.List;

public class BookService {

private BookRepository bookRepository;

// Setter for Spring dependency injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook(String bookName) {

System.***out***.println("Adding book: \"" + bookName + "\"");

bookRepository.saveBook(bookName);

}

public void removeBook(String bookName) {

System.***out***.println("Removing book: \"" + bookName + "\"");

bookRepository.deleteBook(bookName);

}

public void updateBook(String oldName, String newName) {

System.***out***.println("Updating book: \"" + oldName + "\" to \"" + newName + "\"");

bookRepository.updateBook(oldName, newName);

}

public void getBookDetails(String bookName) {

System.***out***.println("Searching for book: \"" + bookName + "\"");

System.***out***.println(bookRepository.findBook(bookName));

}

public void listAllBooks() {

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

System.***out***.println("List of Books in Library:");

if (books.isEmpty()) {

System.***out***.println(" [No books available]");

} else {

for (String book : books) {

System.***out***.println(" - " + book);

}

}

}

}

**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">

<!-- 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>

**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>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>LibraryManagement</name>

<url>http://maven.apache.org</url>

<properties>

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

</properties>

<dependencies>

<!-- Spring Core dependency -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.33</version>

</dependency>

<!-- Existing JUnit dependency -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

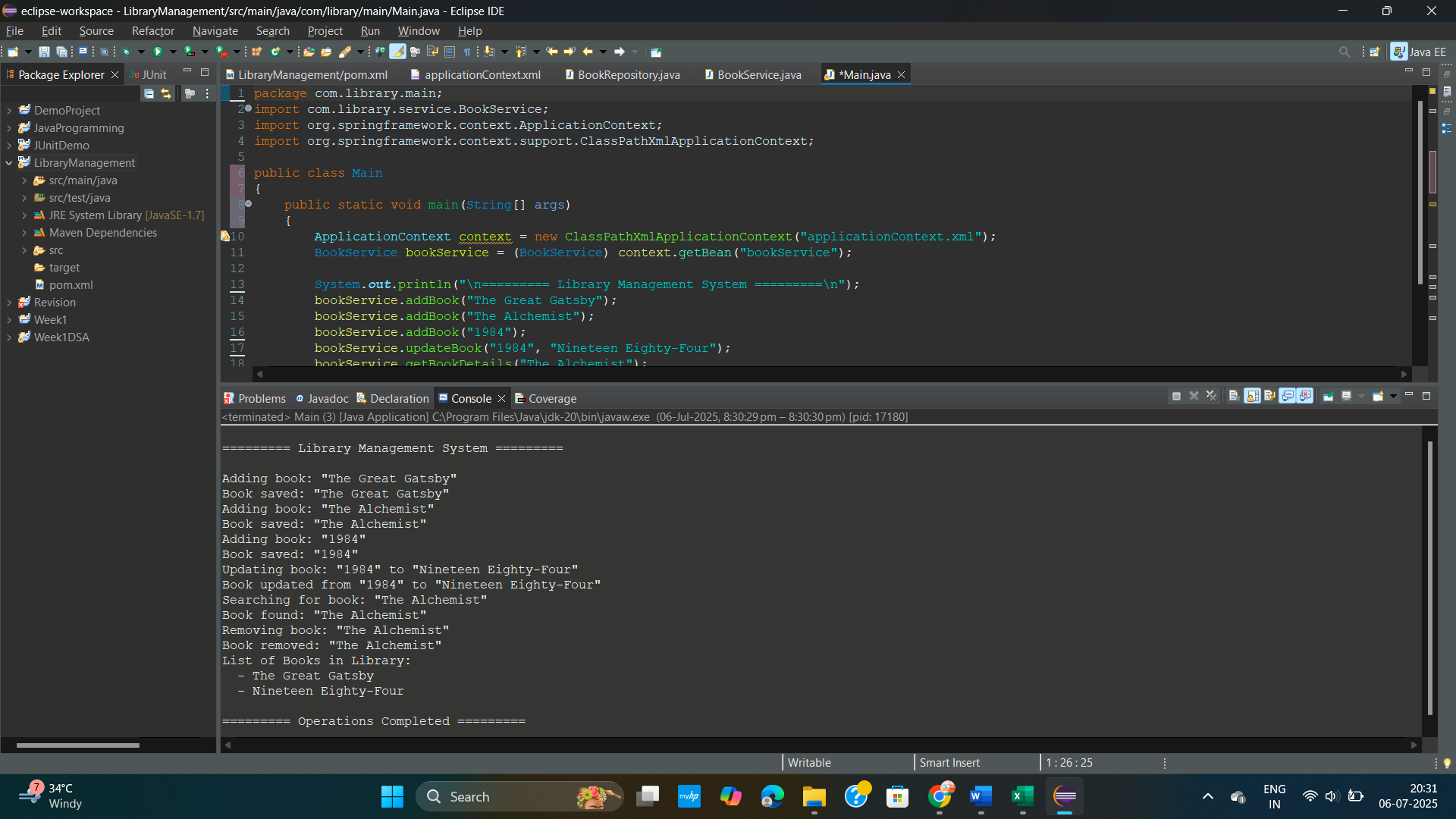
<scope>test</scope>

</dependency>

</dependencies>

</project>

**OUTPUT :**



**Exercise 2 : Implementing Dependency Injection**

**PROBLEM STATEMENT:**

In the library management application, you need to manage the dependencies between the BookService and BookRepository classes using Spring's IoC and DI.

**CODE :**

**Main.java**

package com.library.main;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Main

{

public static void main(String[] args)

{

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

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

System.***out***.println("\n--- Verifying Dependency Injection ---\n");

// If this works, DI is successful

bookService.addBook("Spring in Action");

// Confirm BookRepository was injected by listing books

bookService.listAllBooks();

System.***out***.println("\n--- Dependency Injection Verified ---");

}

}

**BookRepository.java**

package com.library.repository;

import java.util.ArrayList;

import java.util.List;

public class BookRepository {

private final List<String> books = new ArrayList<>();

public void saveBook(String bookName) {

books.add(bookName);

System.***out***.println("Book saved: \"" + bookName + "\"");

}

public void deleteBook(String bookName) {

if (books.remove(bookName)) {

System.***out***.println("Book removed: \"" + bookName + "\"");

} else {

System.***out***.println("Book not found: \"" + bookName + "\"");

}

}

public void updateBook(String oldName, String newName) {

int index = books.indexOf(oldName);

if (index != -1) {

books.set(index, newName);

System.***out***.println("Book updated from \"" + oldName + "\" to \"" + newName + "\"");

} else {

System.***out***.println("Book not found: \"" + oldName + "\"");

}

}

public String findBook(String bookName) {

if (books.contains(bookName)) {

return "Book found: \"" + bookName + "\"";

} else {

return "Book not found: \"" + bookName + "\"";

}

}

public List<String> findAllBooks() {

return new ArrayList<>(books);

}

}

**BookService.java**

package com.library.service;

import com.library.repository.BookRepository;

import java.util.List;

public class BookService {

private BookRepository bookRepository;

// Setter for Spring dependency injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook(String bookName) {

System.***out***.println("Adding book: \"" + bookName + "\"");

bookRepository.saveBook(bookName);

}

public void removeBook(String bookName) {

System.***out***.println("Removing book: \"" + bookName + "\"");

bookRepository.deleteBook(bookName);

}

public void updateBook(String oldName, String newName) {

System.***out***.println("Updating book: \"" + oldName + "\" to \"" + newName + "\"");

bookRepository.updateBook(oldName, newName);

}

public void getBookDetails(String bookName) {

System.***out***.println("Searching for book: \"" + bookName + "\"");

System.***out***.println(bookRepository.findBook(bookName));

}

public void listAllBooks() {

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

System.***out***.println("List of Books in Library:");

if (books.isEmpty()) {

System.***out***.println(" [No books available]");

} else {

for (String book : books) {

System.***out***.println(" - " + book);

}

}

}

}

**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">

<!-- 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>

**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>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>LibraryManagement</name>

<url>http://maven.apache.org</url>

<properties>

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

</properties>

<dependencies>

<!-- Spring Core dependency -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.33</version>

</dependency>

<!-- Existing JUnit dependency -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

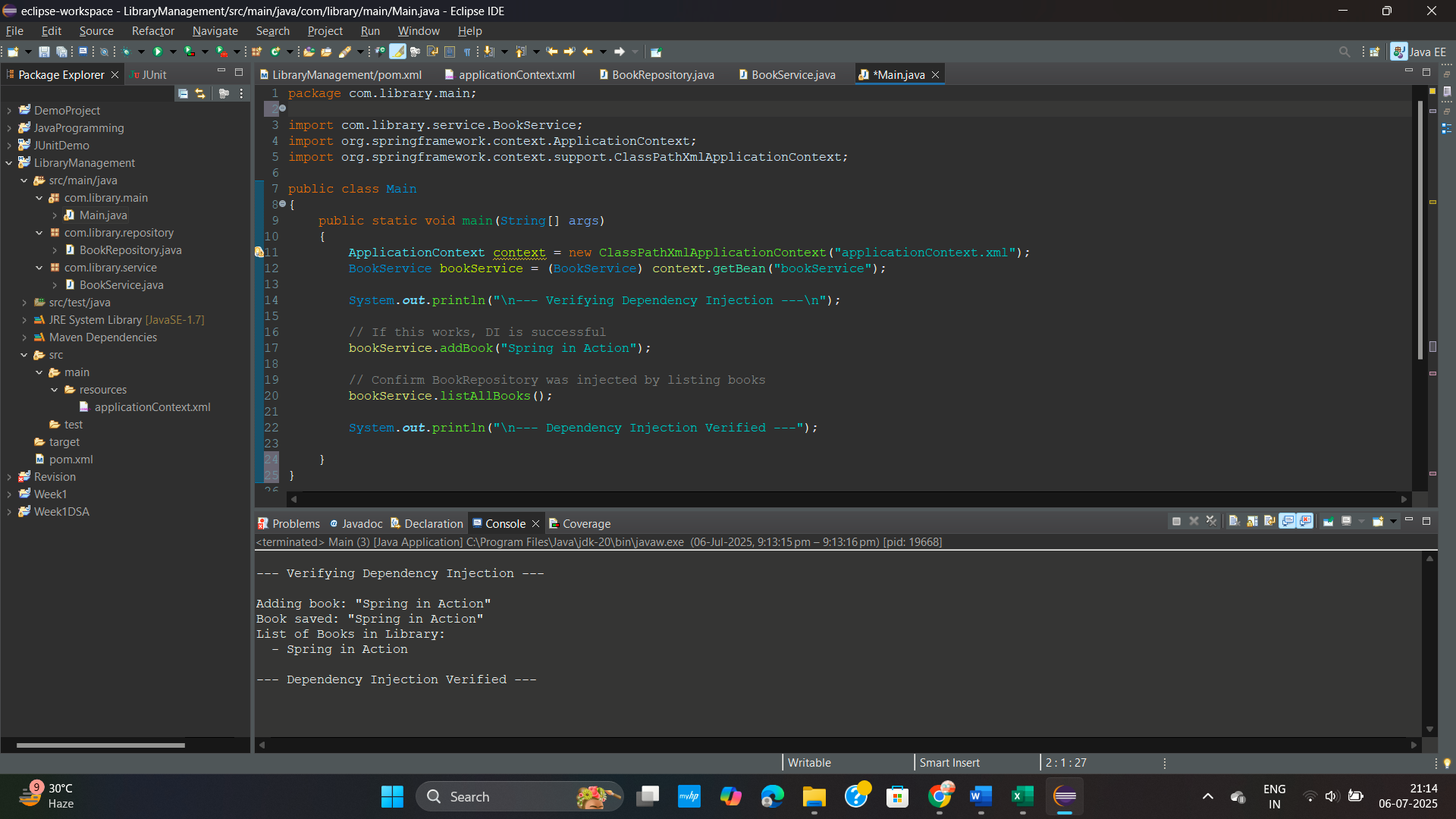
<scope>test</scope>

</dependency>

</dependencies>

</project>

**OUTPUT :**



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

**PROBLEM STATEMENT :**

You need to set up a new Maven project for the library management application and add Spring dependencies.

**CODE :**

1. A new Maven project named LibraryManagement was created using Eclipse IDE.

* Group ID: com.library
* Artifact ID: LibraryManagement
* Packaging: Jar
* Version: 0.0.1-SNAPSHOT

1. Adding Spring Dependencies in pom.xml

* Spring Context: For core IoC container and application context
* Spring AOP: For aspect-oriented programming features
* Spring WebMVC: For future web layer and controller support
* JUnit: For unit testing

1. Configuring Maven Compiler Plugin

* The project is configured to use **Java 1.8** via the maven-compiler-plugin. This ensures compatibility with standard Java 8 syntax and features.

**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>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>LibraryManagement</name>

<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 Core + Context (for DI and ApplicationContext) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.33</version>

</dependency>

<!-- Spring AOP (for aspect-oriented programming features) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.33</version>

</dependency>

<!-- Spring Web MVC (in case you later build a web controller) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.33</version>

</dependency>

<!-- JUnit for testing -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Maven Compiler Plugin to use Java 1.8 -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.10.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

**OUTPUT :**

