**4)**

**Aim:**

**1a) Basic Spring Boot Application with Auto-Wiring. Description**

This is a **basic Spring Boot application** that demonstrates **Auto-Wiring** and **Bean Scopes**.

* The Student class depends on the Marks class, and Spring automatically injects Marks into Student using @Autowired.
* @Component is used to register both classes (Student and Marks) as Spring Beans.
* In the main class (Prac4Application), we retrieve the Student bean from the

**ApplicationContext** and call its methods along with the Marks methods.

**Program:**

# AutoWiringApplication.java

**package** com.example;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.context.ApplicationContext;

@SpringBootApplication

**public** **class** AutowiringApplication {

**public** **static** **void** main(String[] args) {

ApplicationContext context=SpringApplication.*run*(AutowiringApplication.**class**, args);

Student s = context.getBean(Student.**class**);

s.display();

s.m.display();

s.m.read();

}

}

**Marks.java**

**package** com.example;

**import** org.springframework.stereotype.Component;

@Component

**public** **class** Marks {

**public** **void** display() {

System.***out***.println("Marks class display method");

}

**public** **void** read() {

System.***out***.println("Marks class read method");

}

}

**Student.java**

**package** com.example;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Component;

@Component

**public** **class** Student {

@Autowired

**public** Marks m;

**public** **void** display() {

System.***out***.println("Student class display method");

}

}

**application.properties**

spring.application.name=Autowiring

server.port=8546

**pom.xml**

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

<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> https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent><groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.5.5</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.example</groupId>

<artifactId>SampleApplication</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>prac-4</name>

<description>Demo project for Spring Boot</description>

<url/>

<licenses><license/></licenses>

<developers><developer/></developers>

<scm><connection/>

<developerConnection/><tag/>

<url/></scm>

<properties><java.version>21</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build><plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

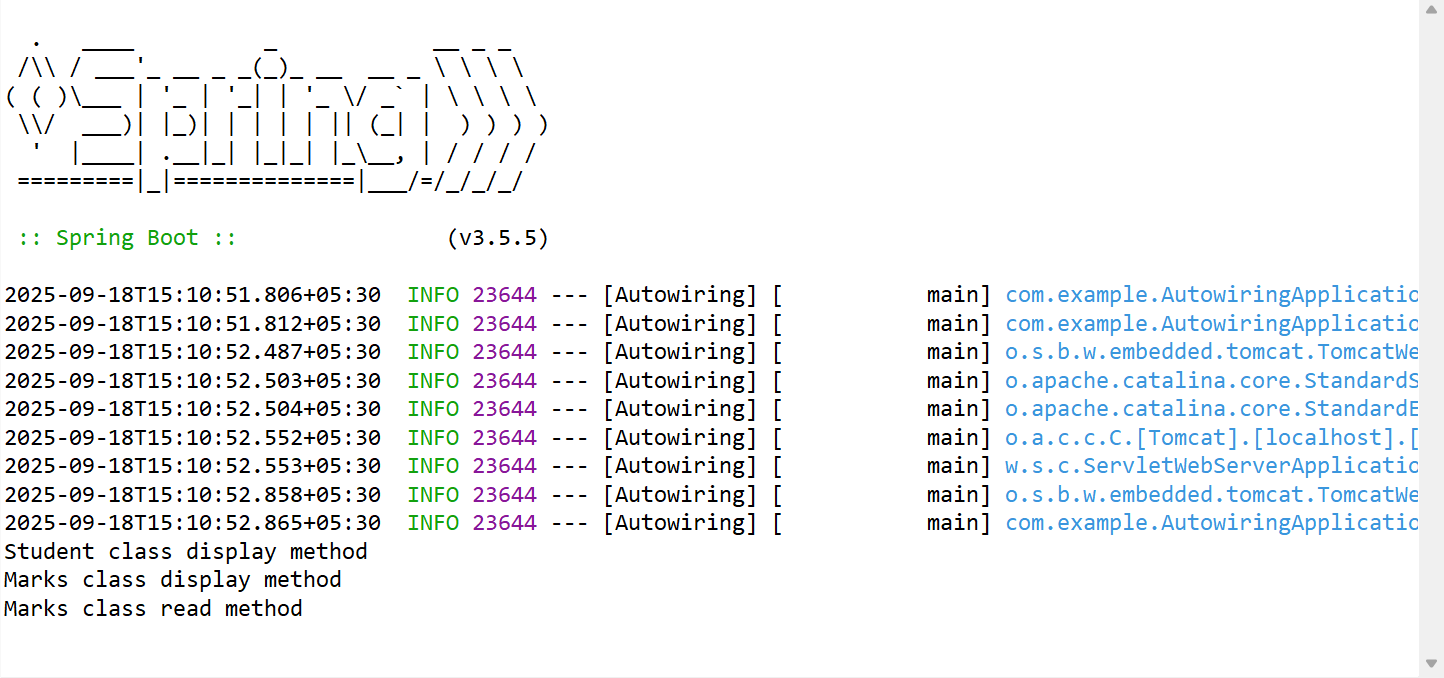
<artifactId>spring-boot-maven-plugin</artifactId>

</plugin></plugins>

</build>

</project>

**Output:**



**4)**

**Aim:**

**1b) Basic Spring Boot Application with Bean Scopes.**

**Description:**

This program explains the difference between **Singleton** and **Prototype** bean scopes in Spring Boot:

* **Singleton Scope** (@Scope("singleton")): Only one instance of the bean is created for the entire Spring container. Any request for that bean will return the same instance.
* **Prototype Scope** (@Scope("prototype")): A new instance of the bean is created every time it is requested from the container.
* SingletonBean is shared across requests. Changes in one reference affect the other.
* PrototypeBean creates new objects each time, so changes in one object do not affect another.

**Program:**

# SingletonBean.java

**package** com.example;

**import** org.springframework.beans.factory.annotation.Value;

**import** org.springframework.context.annotation.Scope;

**import** org.springframework.stereotype.Component;

@Component

@Scope("prototype")

**public** **class** PrototypeBean {

@Value("93")

**public** **int** count;

**public** **int** getCount() {

**return** count;

}

**public** **void** setCount(**int** count) {

**this**.count = count;

}

**ScopeDemoApplication.java**

**package** com.example;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.context.ConfigurableApplicationContext;

@SpringBootApplication

**public** **class** ScopeDemoApplication {

**public** **static** **void** main(String[] args) {

ConfigurableApplicationContext context=SpringApplication.*run*(ScopeDemoApplication.**class**, args);

SingletonBean sb1=context.getBean(SingletonBean.**class**);

SingletonBean sb2=context.getBean(SingletonBean.**class**);

sb1.setCount(100);

sb2.setCount(200);

System.***out***.println(sb1.getCount());

System.***out***.println(sb2.getCount());

PrototypeBean pb1=context.getBean(PrototypeBean.**class**);

PrototypeBean pb2=context.getBean(PrototypeBean.**class**);

pb1.setCount(700);

pb2.setCount(900);

System.***out***.println(pb1.getCount());

System.***out***.println(pb2.getCount());

}

}

**PrototypeBean.java**

**package** com.example;

**import** org.springframework.beans.factory.annotation.Value;

**import** org.springframework.context.annotation.Scope;

**import** org.springframework.stereotype.Component;

@Component

@Scope("prototype")

**public** **class** PrototypeBean {

@Value("93")

**public** **int** count;

**public** **int** getCount() {

**return** count;

}

**public** **void** setCount(**int** count) {

**this**.count = count;

}

}

**application.properties**

spring.application.name=ScopeDemo

server.port=1234

**pom.xml**

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

<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>

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

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.5.5</version>

<relativePath/> <!-- lookup parent from repository -->

</parent><groupId>com.example</groupId>

<artifactId>SampleApplication</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>prac-5</name>

<description>Demo project for Spring Boot</description>

<url/><licenses><license/></licenses>

<developers><developer/></developers>

<scm><connection/>

<developerConnection/><tag/><url/></scm>

<properties><java.version>21</java.version>

</properties>

<dependencies>

<!-- Spring Boot Web Starter -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency><!-- Spring Boot Test Starter -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build><plugins><!-- Spring Boot Maven Plugin -->

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin></plugins></build>

</project>

**Output:**



**5)**

**Aim:**

**Basic Spring Boot Application with Bean Scopes. Description:**

This program demonstrates how to implement **logging** in a Spring Boot application using

**SLF4J (Simple Logging Facade for Java)** and LoggerFactory.

* We created a LogController that logs messages at different log levels: **TRACE, DEBUG, INFO, WARN, and ERROR**.
* The logging behavior is configured in application.properties.
* Another controller MyController is added to test a simple message response.

**Program:**

**LogController.java**

**package** com.example;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.RestController;

@RestController

**public** **class** MyController {

@GetMapping("/get")

**public** String msg()

{

**return** "Spring Boot Application successfully executed";

}

}

**MyController.java**

**package** com.example;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.RestController;

@RestController

**public** **class** MyController {

@GetMapping("/get")

**public** String msg() {

**return** "Spring Boot Application successfully executed";

}

}

**Application.java**

**package** com.example;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** Application {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(Application.**class**, args);

}

}

**application.properties**

spring.application.name=SpringBootApplication

logging.level.root=INFO

logging.level.com.example=DEBUG

server.port=0000

**pom.xml**

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

<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>

https://maven.apache.org/xsd/maven- 4.0.0.xsd"><modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.5.5</version>

<relativePath/> <!-- lookup parent from repository -->

</parent><groupId>com.example</groupId>

<artifactId>SampleApplication</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>prac-5.1</name>

<description>Demo project for Spring Boot</description>

<url/><licenses><license/></licenses>

<developers><developer/></developers>

<scm><connection/>

<developerConnection/><tag/><url/></scm>

<properties><java.version>21</java.version>

</properties>

<dependencies><!-- Spring Boot Web Starter -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- Spring Boot Test Starter -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope></dependency>

</dependencies><build>

<plugins><!-- Spring Boot Maven Plugin -->

<plugin>

<groupId>org.springframework.boot</groupId>

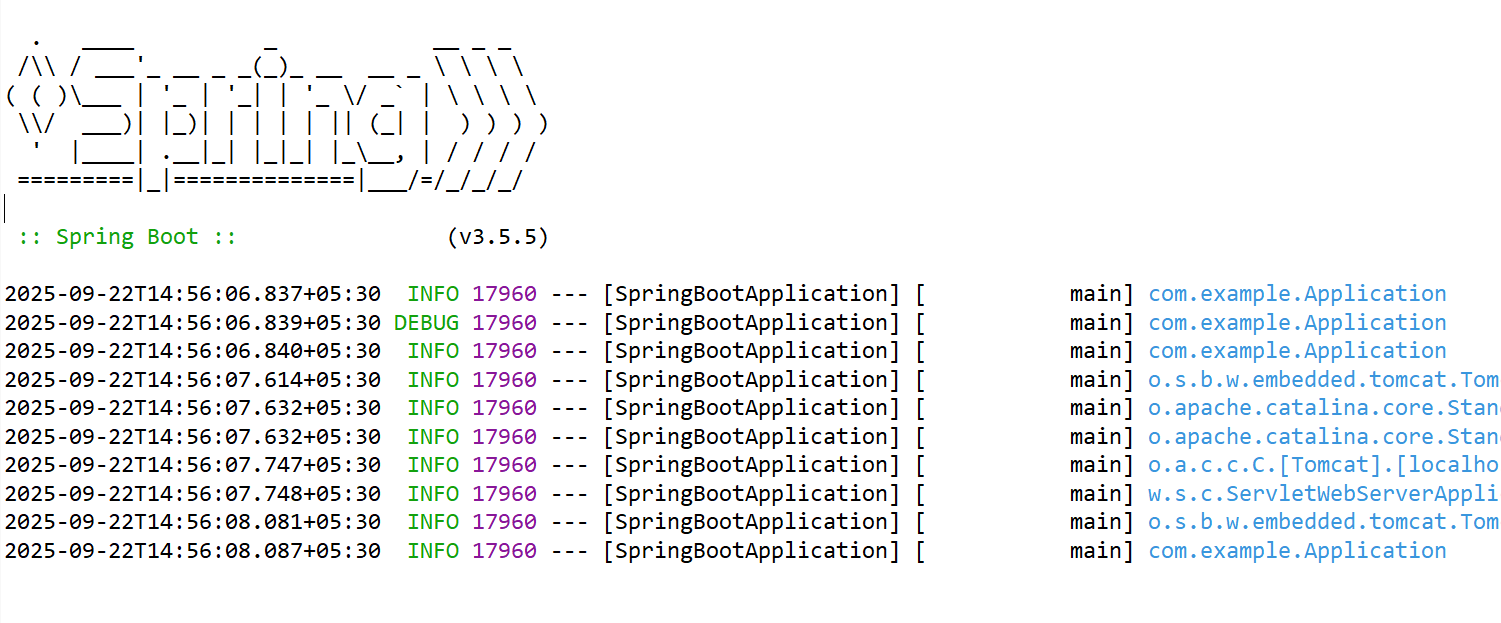
<artifactId>spring-boot-maven-plugin</artifactId>

</plugin></plugins>

</build>

</project>

**Output:**



**6)**

**Aim:**

**Implementing and Using AOP in Spring Boot Description:**

Aspect-Oriented Programming (AOP) in Spring Boot is used to separate cross-cutting concerns (like logging, transactions, security) from the main business logic.

In this example:

* We create a **Service (**MyService**)** that prints a message.
* We define an **Aspect (**MyAspect**)** with @Before and @After advice that runs before and after the sayHello() method.
* The **Main Application (**Prac6Application**)** runs the service using CommandLineRunner.
* Dependencies in pom.xml include spring-boot-starter-aop to enable AOP support.

**Program: MyAspect.java**

package com.demo;

import org.aspectj.lang.annotation.After; import org.aspectj.lang.annotation.Aspect; import org.aspectj.lang.annotation.Before;

import org.springframework.stereotype.Component;

@Aspect @Component

public class MyAspect {

@Before("execution(\* com.demo.MyService.sayHello(..))") public void beforeMethod() {

System.out.println("BEFORE : Method run");

}

@After("execution(\* com.demo.MyService.sayHello(..))") public void afterMethod() {

System.out.println("AFTER : Method run");

}

}

**MyService.java**

package com.demo;

import org.springframework.stereotype.Service; @Service

public class MyService {

public void sayHello()

{ System.out.println("Welcome to AOP");

}

}

**Prac6Application.java**

package com.demo;

import org.springframework.boot.CommandLineRunner; import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication; @SpringBootApplication

public class Prac6Application implements CommandLineRunner

{ public MyService myservice;

public Prac6Application(MyService myservice) { super();

this.myservice = myservice;

}

public static void main(String[] args)

{ SpringApplication.run(Prac6Application.class, args);

}

@Override

public void run(String... args)

{ myservice.sayHello();

}}

**application.properties :**

spring.application.name = AOP spring.aop.auto = true

**pom.xml**

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

<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> https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent><groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.5.5</version>

<relativePath/> <!-- lookup parent from repository --></parent>

<groupId>com.example</groupId>

<artifactId>SampleApplication</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>prac-6</name>

<description>Demo project for Spring Boot AOP</description>

<url/><licenses><license/></licenses>

<developers><developer/></developers>

<scm><connection/>

<developerConnection/><tag/><url/></scm>

<properties>

<java.version>21</java.version>

</properties>

<dependencies><dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope></dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-aop</artifactId></dependency>

</dependencies>

<build><plugins><plugin>

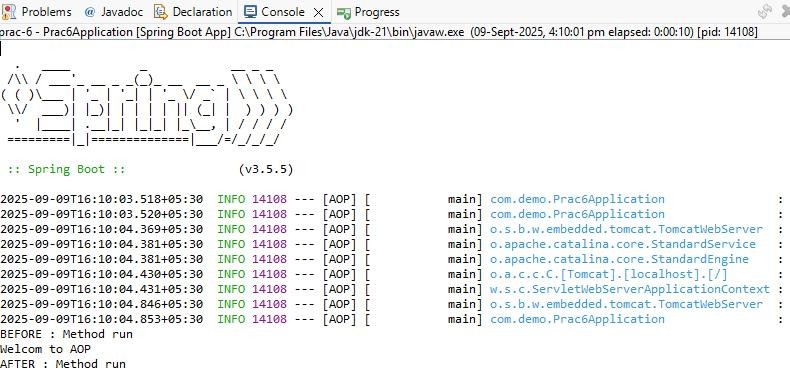
<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin></plugins></build>

</project>

**Output:**

****