**Practical No: 8**

**Aim: Assignment based Aspect Oriented Programming**

**1. Write a program to demonstrate Spring AOP – before advice.**

**Application Context.xml**

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

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

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

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

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-4.3.xsd

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

http://www.springframework.org/schema/context/spring-context-4.3.xsd

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

http://www.springframework.org/schema/aop/spring-aop-4.3.xsd">

<aop:aspectj-autoproxy />

<bean id="employeeService" class="com.sush.service.EmployeeService"></bean>

<!-- Aspect -->

<bean id="logAspect" class="com.ram.Aspect.LoggingAspect" />

</beans>

**LoggingAspect**

package com.sush.Aspect;

import org.aspectj.lang.JoinPoint;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Before;

@Aspect

public class LoggingAspect

{

@Before("execution(\* com.ram.service.EmployeeService.addEmployee())")

public void logBefore(JoinPoint joinPoint)

{

System.out.print("logBefore() is running!");

System.out.println(", before " + joinPoint.getSignature().getName() + " method");

System.out.println("\*\*\*\*\*\*");

}

}

**APP.java**

package com.sush.core;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.ram.service.EmployeeService;

public class App

{

public static void main(String[] args)

{

ApplicationContext context = new ClassPathXmlApplicationContext(

"applicationContext.xml");

System.out.println("---------------------------------------");

EmployeeService employeeService = context

.getBean("employeeService", EmployeeService.class);

employeeService.addEmployee();

employeeService.modifyEmployee();

employeeService.deleteEmployee();

}

}

**EmployeeService.java**

package com.ram.service;

public class EmployeeService

{

public void addEmployee()

{

System.out.println("Add Employee ");

}

public void modifyEmployee()

{

System.out.println("Modify Employee");

}

public void deleteEmployee()

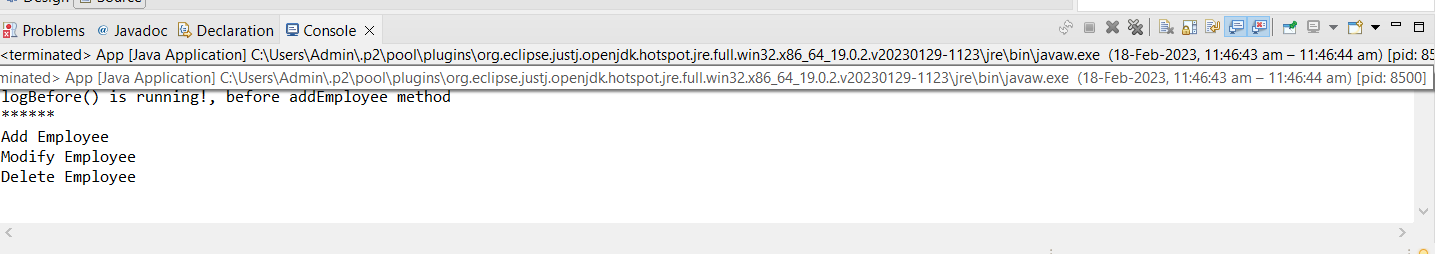
{

System.out.println("Delete Employee");

}

}

**Output:**



**2. Write a program to demonstrate Spring AOP – after advice.**

**Logging Aspect.Java**

package com.sush.Aspect;

import org.aspectj.lang.JoinPoint;

import org.aspectj.lang.annotation.After;

import org.aspectj.lang.annotation.Aspect;

@Aspect

public class LoggingAspect

{

@After("execution(\* com.sush.EmployeeService.addEmployee())")

public void logAfter(JoinPoint joinPoint)

{

System.out.print("logAfter() is running!");

System.out.println(", after "

+ joinPoint.getSignature().getName() + " method");

System.out.println("\*\*\*\*\*\*");

}

}

**APP.java**

package com.sush.core;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.sush.EmployeeService;

public class App

{

public static void main(String[] args)

{

ApplicationContext context = new ClassPathXmlApplicationContext(

"applicationContext.xml");

System.out.println("---------------------------------------");

EmployeeService employeeService = context

.getBean("employeeService", EmployeeService.class);

employeeService.addEmployee();

employeeService.modifyEmployee();

employeeService.deleteEmployee();

}

}

**EmployeeService.java**

package com.sush;

public class EmployeeService

{

public void addEmployee()

{

System.out.println("Add Employee ");

}

public void modifyEmployee()

{

System.out.println("Modify Employee");

}

public void deleteEmployee()

{

System.out.println("Delete Employee");

}

}

**Application Context.xml**

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

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

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

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

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-4.3.xsd

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

http://www.springframework.org/schema/context/spring-context-4.3.xsd

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

http://www.springframework.org/schema/aop/spring-aop-4.3.xsd">

<aop:aspectj-autoproxy />

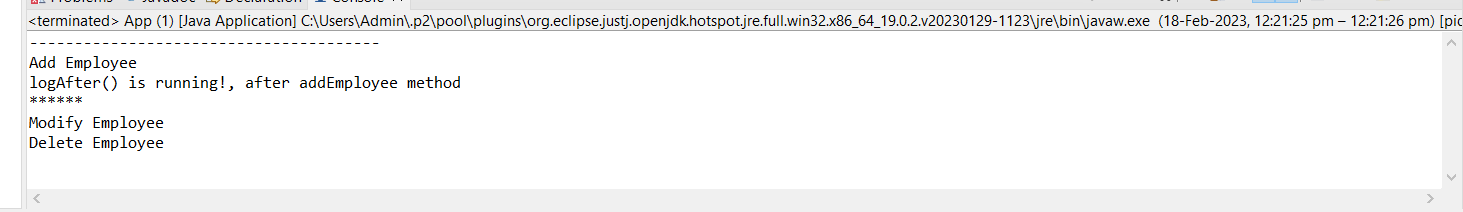
<bean id="employeeService" class="com.sush.service.EmployeeService"></bean>

<!-- Aspect -->

<bean id="logAspect" class="com.sush.Aspect.LoggingAspect" />

</beans>

**Output:**



**3. Write a program to demonstrate Spring AOP – Around advice.**

**LoggingAspect.java**

package com.sush;

import java.util.Arrays;

import org.aspectj.lang.ProceedingJoinPoint;

import org.aspectj.lang.annotation.Around;

import org.aspectj.lang.annotation.Aspect;

@Aspect

public class LoggingAspect

{

@Around("execution(\* com.sush.EmployeeService.addEmployee(..))")

public void logAround(ProceedingJoinPoint proceedingJoinPoint) throws Throwable

{

System.out.println("logAround() is running!");

System.out.println("hijacked method = " + proceedingJoinPoint.getSignature().getName());

System.out.println("hijacked arguments = " + Arrays.toString(proceedingJoinPoint.getArgs()));

System.out.println("Around before is running!");

proceedingJoinPoint.proceed(); //continue on the intercepted method

System.out.println("Around after is running!");

System.out.println("\*\*\*\*\*\*");

}

}

**APP1.java**

package com.sush;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App1

{

public static void main(String[] args)

{

ApplicationContext context = new ClassPathXmlApplicationContext(

"applicationContext.xml");

System.out.println("---------------------------------------");

EmployeeService employeeService = context

.getBean("employeeService", EmployeeService.class);

employeeService.addEmployee("Peter");

employeeService.modifyEmployee();

employeeService.deleteEmployee();

}

}

**EmployeeService.java**

package com.sush;

public class EmployeeService

{

public String addEmployee(String name)

{

System.out.println("addEmployee(String name) method is called");

return "Employee Peter information is added successfully";

}

public void modifyEmployee()

{

System.out.println("modifyEmployee() is called");

}

public void deleteEmployee()

{

System.out.println("deleteEmployee() method is called");

}

}

**ApplicationContext.xml**

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

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

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

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

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-4.3.xsd

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

http://www.springframework.org/schema/context/spring-context-4.3.xsd

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

http://www.springframework.org/schema/aop/spring-aop-4.3.xsd">

<aop:aspectj-autoproxy />

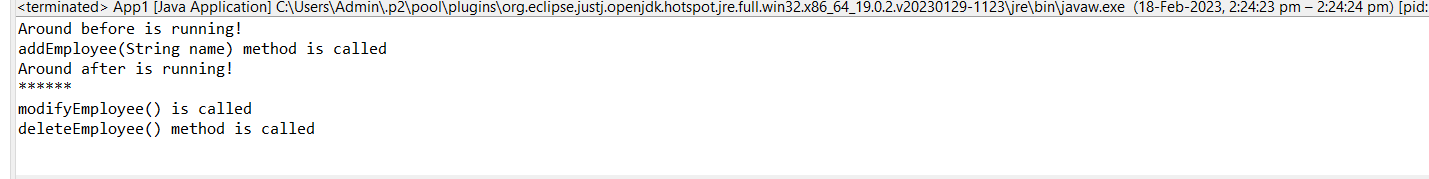
<bean id="employeeService" class="com.sush.EmployeeService"></bean>

<!-- Aspect -->

<bean id="logAspect" class="com.sush.LoggingAspect" />

</beans>

**Output:**



**4. Write a program to demonstrate Spring AOP – AfterReturning**

**LoggingAspect.java**

package com.sush;

import org.aspectj.lang.JoinPoint;

public class LoggingAspect

{

public void logAfterReturning(JoinPoint joinPoint, Object result)

{

System.out.print("logAfterReturning() is running!");

System.out.println(", after "

+ joinPoint.getSignature().getName() + " method");

System.out.println("Method returned value is = " + result);

System.out.println("\*\*\*\*\*\*");

}

}

**App2.Java**

package com.sush;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App2

{

public static void main(String[] args)

{

ApplicationContext context = new ClassPathXmlApplicationContext(

"applicationContext.xml");

System.out.println("---------------------------------------");

EmployeeService employeeService = context

.getBean("employeeService", EmployeeService.class);

employeeService.addEmployee();

employeeService.modifyEmployee();

employeeService.deleteEmployee();

}

}

**EmployeeService.java**

package com.sush;

public class EmployeeService

{

public String addEmployee()

{

System.out.println("addEmployee() is called");

return "Employee Peter information is added successfully";

}

public void modifyEmployee()

{

System.out.println("modifyEmployee() is called");

}

public void deleteEmployee()

{

System.out.println("deleteEmployee() is called");

}

}

**ApplicationContext.xml**

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

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

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

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

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-4.3.xsd

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

http://www.springframework.org/schema/context/spring-context-4.3.xsd

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

<http://www.springframework.org/schema/aop/spring-aop-4.3.xsd>">

<bean id="employeeService" class="com.sush.EmployeeService"></bean>

<!-- Aspect -->

<bean id="logAspect" class="com.sush.LoggingAspect" />

<aop:config>

<aop:aspect id="aspectLoggging" ref="logAspect">

<!-- @After -->

<aop:pointcut id="pointCutAfterReturning"

expression="execution(\* com.sush.EmployeeService.addEmployee()))" />

<aop:after-returning method="logAfterReturning"

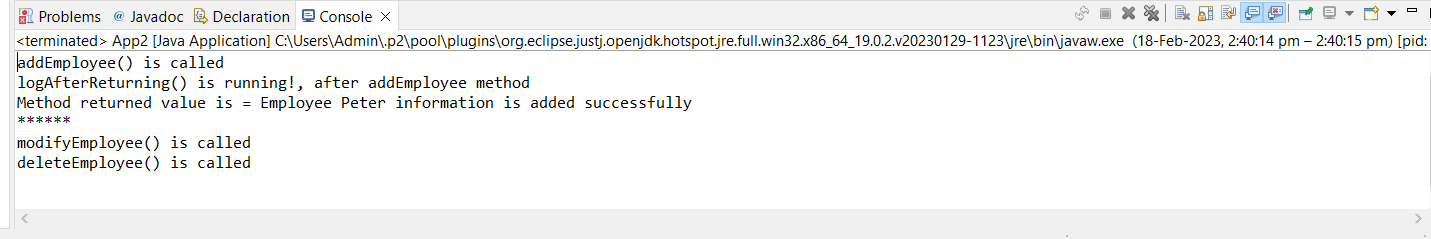
returning="result" pointcut-ref="pointCutAfterReturning" />

</aop:aspect>

</aop:config>

</beans>

**Output:**



**Practical No: 10**

**Aim: Assignment based Spring Boot and RESTful Web Services**

**1. Write a program to create a simple Spring Boot application that prints a message.**

**Code:**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.run(DemoApplication.class, args);

System.out.println("Hello! We are Running Spring Boot App");

}

}

**Output:**

