Spring AOP Example

In Spring framework, there are four ways we can achieve AOP.

Spring Proxy based AOP.

Spring AspectJ style AOP.

Spring XML-based AOP.

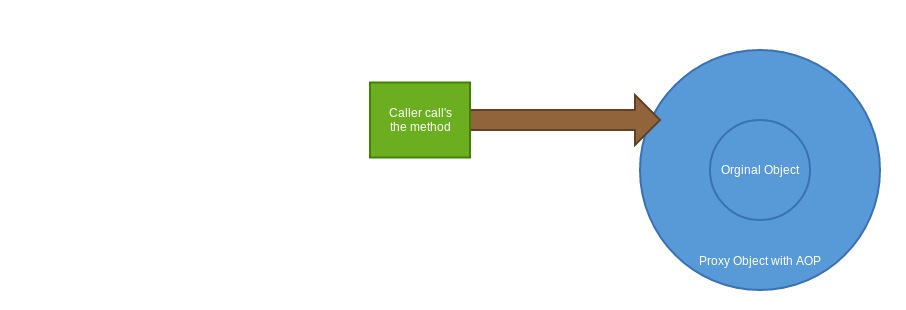
Incorporate AspectJ AOP.

Please note that Spring proxy based AOP is an Old style it was used in earlier days in Spring. No one now uses this.

First three strategies of AOP is Spring own, but number 4 is based on AspectJ programming, Spring just incorporates AspectJ.

We will discuss the AspectJ style(2) as it is most used Style.

**Spring AOP Programmatic Diagram**



**AOP Example :**

**Problem :** We all know Camera has an Auto mode, If we set the mode to Auto camera will set the focus , set the correct exposure and then you will just click the button to capture a snap. Then the camera will display the snap on its display screen. So User of the camera only needs to click the button rest of the action taken care by the camera so it is an ideal situation where we can use AOP.

1. Download the following jars from [**http://www.java2s.com/Code/Jar/a/Downloadaspectjrt173jar.htm**](http://www.java2s.com/Code/Jar/a/Downloadaspectjrt173jar.htm)
2. **aspectj.jar**
3. **aspectjrt.jar**
4. **aspectj-weaver.jar**
5. **aopalliance.jar**

2. put those jars into the classpath

3. Create a class called **CameraAutoModeAop.java,** here we will define all the work camera do automatically in other words we write all the Advice here and tell the advice when to execute before or after a method.

package com.example.aop;

import org.aspectj.lang.annotation.AfterReturning;

import org.aspectj.lang.annotation.AfterThrowing;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Before;

import org.aspectj.lang.annotation.Pointcut;

@Aspect

public class CameraAutoModeAOP {

@Pointcut("execution(\*\* com.example.aop.Camera.takePicture(..))")

public void takePicture() {}

@Before("takePicture()")

public void adjustExposure() {

System.out.println("Setting correct Exposure");

}

@Before("takePicture()")

public void adjustFocus() {

System.out.println("Adjusting Focus");

}

@AfterReturning("takePicture()")

public void displayPicture() {

System.out.println("Display Picture");

}

@AfterThrowing("takePicture()")

public void invaildSettings() {

System.out.println("Unable to process Picture");

}

}

Please note we use **@Aspect annotation** to denote this class is an aspect, also we use **@Pointcut annotation** to denote where the advice will be triggered.

Pay attention to **@Pointcut annotation** here we use an **AspectJ-specific syntax** as a parameter of the @Pointcut annotation.

***execution(\*\* com.example.aop.Camera.takePicture(..))***

**Here execution means where the advice will be executed.**

**\*\* denotes method irrespective of any return type.**

**com.example.aop.Camera : denote the fully qualified class.**

**takePicture(..) : denotes the Advice will be fired if it founds any method named takePicture irrespective of its argument list.**

So as whole this Expression says, execute the advice on a method takePicture under the class com.example.aop.Camera . takePicture method can have any return type and any argument list.

4. Create the camera class and takePicture method.

package com.example.aop;

public class Camera {

public void takePicture()

{

System.out.println("Picture has been taken successfully");

}

}

5. Create the Spring configuration 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"

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

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

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

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

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

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

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

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

<context:component-scan base-package="com.example.aop" />

<aop:aspectj-autoproxy />

<bean id="camera" class="com.example.aop.Camera" />

<bean class="com.example.aop.CameraAutoModeAOP" />

</beans>

Please note that we introduce new dtd entries to support aop tags.

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

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

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

Also , we use **<aop:aspectj-autoproxy />** , it is very important tag by using it we enable Spring AspectJ style AOP functionality.

**6.** To run the Example we use a main class which lookup the Camera bean from ApplicationContext and call the takePictureMethod() to fire AOP.

package com.example.aop;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Main {

public static void main(String[] args) {

ApplicationContext ctx = new ClassPathXmlApplicationContext(

"configFiles/aop.xml");

Camera camera = (Camera) ctx.getBean("camera");

camera.takePicture();

}

}

7. Output:

Setting correct Exposure

Adjusting Focus

Picture has been taken successfully

Display Picture