Spring AOP Example – Pointcut , Advisor

By [mkyong](http://www.mkyong.com/author/mkyong/) | March 26, 2010 | Updated : August 30, 2012 | Viewed : 145,126 times

In last [Spring AOP advice examples](http://www.mkyong.com/spring/spring-aop-examples-advice/), the entire methods of a class are intercepted automatically. But for most cases, you may just need a way to intercept only one or two methods, this is what ‘Pointcut’ come for. It allow you to intercept a method by it’s method name. In addition, a ‘Pointcut’ must be associated with an ‘Advisor’.

In Spring AOP, comes with three very technical terms – **Advices, Pointcut , Advisor**, put it in unofficial way…

* Advice – Indicate the action to take either before or after the method execution.
* Pointcut – Indicate which method should be intercept, by method name or regular expression pattern.
* Advisor – Group ‘Advice’ and ‘Pointcut’ into a single unit, and pass it to a proxy factory object.

Review last [Spring AOP advice examples](http://www.mkyong.com/spring/spring-aop-examples-advice/) again.

*File : CustomerService.java*

Java

package com.mkyong.customer.services;

public class CustomerService

{

private String name;

private String url;

public void setName(String name) {

this.name = name;

}

public void setUrl(String url) {

this.url = url;

}

public void printName(){

System.out.println("Customer name : " + this.name);

}

public void printURL(){

System.out.println("Customer website : " + this.url);

}

public void printThrowException(){

throw new IllegalArgumentException();

}

}

*File : Spring-Customer.xml*

Markup

<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

http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">

<bean id="customerService" class="com.mkyong.customer.services.CustomerService">

<property name="name" value="Yong Mook Kim" />

<property name="url" value="http://www.mkyong.com" />

</bean>

<bean id="hijackAroundMethodBeanAdvice" class="com.mkyong.aop.HijackAroundMethod" />

<bean id="customerServiceProxy"

class="org.springframework.aop.framework.ProxyFactoryBean">

<property name="target" ref="customerService" />

<property name="interceptorNames">

<list>

<value>hijackAroundMethodBeanAdvice</value>

</list>

</property>

</bean>

</beans>

*File : HijackAroundMethod.java*

Java

package com.mkyong.aop;

import java.util.Arrays;

import org.aopalliance.intercept.MethodInterceptor;

import org.aopalliance.intercept.MethodInvocation;

public class HijackAroundMethod implements MethodInterceptor {

@Override

public Object invoke(MethodInvocation methodInvocation) throws Throwable {

System.out.println("Method name : "

+ methodInvocation.getMethod().getName());

System.out.println("Method arguments : "

+ Arrays.toString(methodInvocation.getArguments()));

System.out.println("HijackAroundMethod : Before method hijacked!");

try {

Object result = methodInvocation.proceed();

System.out.println("HijackAroundMethod : Before after hijacked!");

return result;

} catch (IllegalArgumentException e) {

System.out.println("HijackAroundMethod : Throw exception hijacked!");

throw e;

}

}

}

Run it

Java

package com.mkyong.common;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.mkyong.customer.services.CustomerService;

public class App {

public static void main(String[] args) {

ApplicationContext appContext = new ClassPathXmlApplicationContext(

new String[] { "Spring-Customer.xml" });

CustomerService cust = (CustomerService) appContext

.getBean("customerServiceProxy");

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

cust.printName();

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

cust.printURL();

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

try {

cust.printThrowException();

} catch (Exception e) {

}

}

}

Output

Bash

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Method name : printName

Method arguments : []

HijackAroundMethod : Before method hijacked!

Customer name : Yong Mook Kim

HijackAroundMethod : Before after hijacked!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Method name : printURL

Method arguments : []

HijackAroundMethod : Before method hijacked!

Customer website : http://www.mkyong.com

HijackAroundMethod : Before after hijacked!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Method name : printThrowException

Method arguments : []

HijackAroundMethod : Before method hijacked!

HijackAroundMethod : Throw exception hijacked!

The entire methods of customer service class are intercepted. Later, we show you how to use “**pointcuts**” to intercept onlyprintName() method.

Pointcuts example

You can match the method via following two ways :

1. Name match
2. Regular repression match

1. Pointcuts – Name match example

Intercept a printName() method via ‘pointcut’ and ‘advisor’. Create a **NameMatchMethodPointcut** pointcut bean, and put the method name you want to intercept in the ‘**mappedName**‘ property value.

Markup

<bean id="customerPointcut"

class="org.springframework.aop.support.NameMatchMethodPointcut">

<property name="mappedName" value="printName" />

</bean>

Create a **DefaultPointcutAdvisor** advisor bean, and associate both advice and pointcut.

Markup

<bean id="customerAdvisor"

class="org.springframework.aop.support.DefaultPointcutAdvisor">

<property name="pointcut" ref="customerPointcut" />

<property name="advice" ref="hijackAroundMethodBeanAdvice" />

</bean>

Replace the proxy’s ‘interceptorNames’ to ‘customerAdvisor’ (it was ‘hijackAroundMethodBeanAdvice’).

Markup

<bean id="customerServiceProxy"

class="org.springframework.aop.framework.ProxyFactoryBean">

<property name="target" ref="customerService" />

<property name="interceptorNames">

<list>

<value>customerAdvisor</value>

</list>

</property>

</bean>

Full bean configuration file

Markup

<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

http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">

<bean id="customerService" class="com.mkyong.customer.services.CustomerService">

<property name="name" value="Yong Mook Kim" />

<property name="url" value="http://www.mkyong.com" />

</bean>

<bean id="hijackAroundMethodBeanAdvice" class="com.mkyong.aop.HijackAroundMethod" />

<bean id="customerServiceProxy"

class="org.springframework.aop.framework.ProxyFactoryBean">

<property name="target" ref="customerService" />

<property name="interceptorNames">

<list>

<value>customerAdvisor</value>

</list>

</property>

</bean>

<bean id="customerPointcut"

class="org.springframework.aop.support.NameMatchMethodPointcut">

<property name="mappedName" value="printName" />

</bean>

<bean id="customerAdvisor"

class="org.springframework.aop.support.DefaultPointcutAdvisor">

<property name="pointcut" ref="customerPointcut" />

<property name="advice" ref="hijackAroundMethodBeanAdvice" />

</bean>

</beans>

Run it again, output

Bash

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Method name : printName

Method arguments : []

HijackAroundMethod : Before method hijacked!

Customer name : Yong Mook Kim

HijackAroundMethod : Before after hijacked!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Customer website : http://www.mkyong.com

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Now, you only intercept the printName() method.

**PointcutAdvisor**  
Spring comes with **PointcutAdvisor** class to save your work to declare advisor and pointcut into different beans, you can use **NameMatchMethodPointcutAdvisor** to combine both into a single bean.

Markup

<bean id="customerAdvisor"

class="org.springframework.aop.support.NameMatchMethodPointcutAdvisor">

<property name="mappedName" value="printName" />

<property name="advice" ref="hijackAroundMethodBeanAdvice" />

</bean>