https://www.cnblogs.com/zhangzongle/p/5944906.html

[Spring中的通知（Advice）和顾问（Advisor）](https://www.cnblogs.com/zhangzongle/p/5944906.html)

在Spring中，目前我学习了几种增强的方式，和大家分享一下

之前的话:

1.AOP  (Aspect  Oriented Programming  面向切面编程)

   在软件业，AOP为Aspect Oriented Programming的缩写，意为：面向切面编程，通过预编译方式和运行期动态代理实现程序功能的统一维护的一种技术。AOP是OOP的延续，是软件开发中的一个热点，也是Spring框架中的一个重要内容，是函数式编程的一种衍生范型。利用AOP可以对业务逻辑的各个部分进行隔离，从而使得业务逻辑各部分之间的耦合度降低，提高程序的可重用性，同时提高了开发的效率。

面向对象编程是从【静态角度】考虑程序的结构，而面向切面编程是从【动态角度】考虑程序运行过程。  
AOP底层，就是采用【动态代理】模式实现的。采用了两种代理：JDK动态代理和CGLIB动态代理。

基本术语（一些名词）：   
（1）切面(Aspect)  
切面泛指[\*交叉业务逻辑\*]。事务处理和日志处理可以理解为切面。常用的切面有通知(Advice)与顾问(Advisor)。实际就是对主业务逻辑的一种增强。

(2)织入（Weaving）  
织入是指将切面代码插入到目标对象的过程。代理的invoke方法完成的工作，可以称为织入。

（3） 连接点(JoinPoint)   
连接点是指可以被切面织入的方法。通常业务接口的方法均为连接点

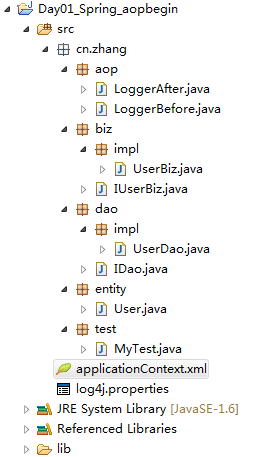
（4）切入点(PointCut)  
切入点指切面具体织入的方法  
注意：被标记为final的方法是不能作为连接点与切入点的。因为最终的是不能被修改的，不能被增强的。

(5)目标对象（Target）  
目标对象指将要被增强的对象。即包含主业务逻辑的类的对象。  
  
（6）通知（Advice）   
通知是切面的一种实现，可以完成简单的织入功能。通知定义了增强代码切入到目标代码的时间点，是目标方法执行之前执行，还是执行之后执行等。切入点定义切入的位置，通知定义切入的时间。

（7）顾问(Advisor)  
顾问是切面的另一种实现，能够将通知以更为复杂的方式织入到目标对象中，是将通知包装为更复杂切面的装配器。

AOP是一种思想，而非实现  
AOP是基于OOP，而又远远高于OOP，主要是将主要核心业务和交叉业务分离，交叉业务就是切面。例如，记录日志和开启事务。

**一:前置增强和后置增强**



源码介绍:

1.User.java

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

package cn.zhang.entity;

public class User {

private Integer id; // 用户ID

private String username; // 用户名

private String password; // 密码

private String email; // 电子邮件

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

}

[复制代码](javascript:void(0);)

2.IDao.java

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

package cn.zhang.dao;

//定义接口

import cn.zhang.entity.User;

public interface IDao {

//定义方法

public void save(User user);

}

[复制代码](javascript:void(0);)

3.UserDao.java

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

package cn.zhang.dao.impl;

//实现接口

import cn.zhang.dao.IDao;

import cn.zhang.entity.User;

public class UserDao implements IDao {

@Override

//实现方法

public void save(User user) {

System.out.println("save success!");

}

}

[复制代码](javascript:void(0);)

4.IUserBiz.java

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

package cn.zhang.biz;

//业务接口

import cn.zhang.entity.User;

public interface IUserBiz {

//待处理的方法

public void save(User user);

}

[复制代码](javascript:void(0);)

5.UserBiz.java

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

package cn.zhang.biz.impl;

//业务接口的实现类

import cn.zhang.biz.IUserBiz;

import cn.zhang.dao.IDao;

import cn.zhang.entity.User;

public class UserBiz implements IUserBiz {

//引入IDao接口

private IDao dao;

@Override

//实现方法

public void save(User user) {

dao.save(user);

}

//dao 属性的setter访问器，会被Spring调用，实现设值注入

public IDao getDao() {

return dao;

}

public void setDao(IDao dao) {

this.dao = dao;

}

}

[复制代码](javascript:void(0);)

6.LoggerAfter.java(后置增强)

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

package cn.zhang.aop;

//后置增强

import java.lang.reflect.Method;

import org.springframework.aop.AfterReturningAdvice;

public class LoggerAfter implements AfterReturningAdvice {

@Override

public void afterReturning(Object arg0, Method arg1, Object[] arg2,

Object arg3) throws Throwable {

System.out.println("后置增强代码");

}

}

[复制代码](javascript:void(0);)

7.LoggerBefore.java(前置增强)

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

package cn.zhang.aop;

//前置增强

import java.lang.reflect.Method;

import org.apache.log4j.Logger;

import org.springframework.aop.MethodBeforeAdvice;

public class LoggerBefore implements MethodBeforeAdvice {

private static final Logger log = Logger.getLogger(LoggerBefore.class);

@Override

public void before(Method arg0, Object[] arg1, Object arg2)

throws Throwable {

log.info("前置内容AAA");

System.out.println("前置增强代码");

}

}

[复制代码](javascript:void(0);)

8.applicationContext.xml（Spring配置文件）

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

<?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:aop="http://www.springframework.org/schema/aop"

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

xsi:schemaLocation="

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

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

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

http://www.springframework.org/schema/aop/spring-aop-4.1.xsd

">

<bean id="dao" class="cn.zhang.dao.impl.UserDao" />

<bean id="biz" class="cn.zhang.biz.impl.UserBiz">

<property name="dao" ref="dao"></property>

</bean>

<!-- 定义前置增强组件 -->

<bean id="loggerBefore" class="cn.zhang.aop.LoggerBefore" />

<!-- 定义后置增强组件 -->

<bean id="loggerAfter" class="cn.zhang.aop.LoggerAfter" />

<!-- 针对AOP的配置 -->

<aop:config>

<aop:pointcut id="pointcut"

expression="execution(public void save(cn.zhang.entity.User))" />

<!-- 将增强处理和切入点结合在一起，在切入点处插入增强处理，完成"织入" -->

<aop:advisor pointcut-ref="pointcut" advice-ref="loggerBefore" />

<aop:advisor pointcut-ref="pointcut" advice-ref="loggerAfter" />

</aop:config>

</beans>

[复制代码](javascript:void(0);)

当然,针对AOP的配置也可以使用代理对象 ProxyFactoryBean 代理工厂bean来实现，在测试类中:IUserBiz biz=(IUserBiz)ctx.getBean("serviceProxy");

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

<!-- 代理对象 ProxyFactoryBean 代理工厂bean -->

<bean id="serviceProxy" class="org.springframework.aop.framework.ProxyFactoryBean">

<property name="targetName" value="biz"></property>

<property name="interceptorNames" value="loggerBefore,loggerAfter"></property>

</bean>

9.MyTest.java

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

package cn.zhang.test;

//测试类

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import cn.zhang.biz.IUserBiz;

import cn.zhang.entity.User;

public class MyTest {

public static void main(String[] args) {

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

IUserBiz biz=(IUserBiz)ctx.getBean("biz");

User user=new User();

biz.save(user);

System.out.println("success!");

}

}

[复制代码](javascript:void(0);)

10.log4j.properties（日志的配置文件）

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

### direct log messages to stdout ###

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.Target=System.out

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L - %m%n

### direct messages to file mylog.log ###

log4j.appender.file=org.apache.log4j.FileAppender

log4j.appender.file.File=c\:mylog.log

log4j.appender.file.layout=org.apache.log4j.PatternLayout

log4j.appender.file.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L - %m%n

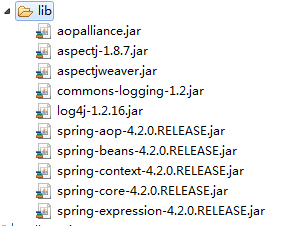
### set log levels - for more verbose logging change 'info' to 'debug' ###

log4j.rootLogger=info, stdout

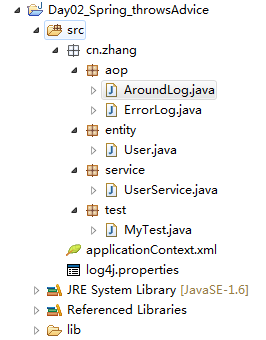
[复制代码](javascript:void(0);)

当然，别忘了引入我们需要的jar包啊！

常用的jar:



**二:异常抛出增强和环绕增强**



源码介绍:

1.User.java

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

package cn.zhang.entity;

public class User {

private Integer id; // 用户ID

private String username; // 用户名

private String password; // 密码

private String email; // 电子邮件

public User() {

super();

// TODO Auto-generated constructor stub

}

public User(Integer id, String username, String password, String email) {

super();

this.id = id;

this.username = username;

this.password = password;

this.email = email;

}

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

}

[复制代码](javascript:void(0);)

2.UserService.java

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

package cn.zhang.service;

public class UserService {

public void delete() {

//int i = 5 / 0;//制造一个错误，用于测试异常抛出增强

System.out.println("delete success!");

}

}

[复制代码](javascript:void(0);)

3.ErrorLog.java（异常抛出增强）

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

package cn.zhang.aop;

//异常抛出增强

import java.lang.reflect.Method;

import org.apache.log4j.Logger;

import org.springframework.aop.ThrowsAdvice;

public class ErrorLog implements ThrowsAdvice {

private static final Logger log = Logger.getLogger(ErrorLog.class);

public void afterThrowing(Method method, Object[] args, Object target,

RuntimeException e){

log.error(method.getName() + " 方法发生异常：" + e);

}

}

[复制代码](javascript:void(0);)

4.AroundLog（环绕增强）

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

package cn.zhang.aop;

//环绕增强

import java.lang.reflect.Method;

import java.util.Arrays;

import org.aopalliance.intercept.MethodInterceptor;

import org.aopalliance.intercept.MethodInvocation;

import org.apache.log4j.Logger;

public class AroundLog implements MethodInterceptor {

private static final Logger log = Logger.getLogger(AroundLog.class);

@Override

public Object invoke(MethodInvocation invocation) throws Throwable {

Object target=invocation.getThis();//获取被代理对象

Method method = invocation.getMethod();//获得被代理方法

Object[] args = invocation.getArguments();//获得方法参数

System.out.println("调用"+target+"的"+method.getName()+"方法。方法参数:"+Arrays.toString(args));

Object result;//调用目标方法，获取目标方法返回值

try {

result = invocation.proceed();

System.out.println("调用" + target + "的" + method.getName()

+ "方法。方法返回值:" + result);

return result;

} catch (Exception e) {

log.error(method.getName()+"方法发生异常:"+e);

throw e;

}

}

}

[复制代码](javascript:void(0);)

5.applicationContext.xml（Spring配置文件）

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

<?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:aop="http://www.springframework.org/schema/aop"

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

xsi:schemaLocation="

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

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

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

http://www.springframework.org/schema/aop/spring-aop-4.1.xsd

">

<bean id="service" class="cn.zhang.service.UserService" />

<!-- 异常抛出增强 -->

<!-- <bean id="error" class="cn.zhang.aop.ErrorLog"/> -->

<!-- 环绕增强 -->

<bean id="error" class="cn.zhang.aop.AroundLog"/>

<aop:config>

<aop:pointcut expression="execution(public void delete())"

id="pointcut" />

<aop:advisor advice-ref="error" pointcut-ref="pointcut" />

</aop:config>

</beans>

[复制代码](javascript:void(0);)

6.MyTest.java

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

package cn.zhang.test;

//测试类

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import cn.zhang.service.UserService;

public class MyTest {

public static void main(String[] args) {

ApplicationContext ctx = new ClassPathXmlApplicationContext(

"applicationContext.xml");

UserService service = (UserService) ctx.getBean("service");

try {

service.delete();

} catch (Exception e) {

System.out.println("错误了");

}

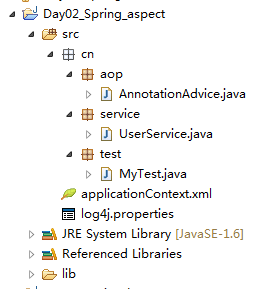
System.out.println("success!");

}

}

[复制代码](javascript:void(0);)

**三:注解增强方式实现前置增强和后置增强**



源码介绍:

1.UserService.java

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

package cn.service;

//业务处理类

public class UserService {

//方法

public void delete() {

System.out.println("delete success!");

}

}

[复制代码](javascript:void(0);)

2.AnnotationAdvice.java（注解增强）

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

package cn.aop;

//注解增强

import org.aspectj.lang.JoinPoint;

import org.aspectj.lang.annotation.AfterReturning;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Before;

@Aspect

public class AnnotationAdvice {

// 定义前置增强

@Before("execution(\* cn.service.UserService.\*(..))")

public void before(JoinPoint jp) {

System.out.println("调用"+jp.getTarget()+"的"+jp.getSignature().getName()+"方法，参数:"+jp.getArgs()+",参数个数:"+jp.getArgs().length);

System.out.println("before");

}

// 定义后置增强

@AfterReturning(pointcut="execution(\* cn.service.UserService.\*(..))",returning="returnValue")

public void afterReturning(JoinPoint jp,Object returnValue) {

System.out.println("调用"+jp.getTarget()+"的"+jp.getSignature().getName()+"方法，参数:"+jp.getArgs()+",返回值为:"+returnValue);

System.out.println("after");

}

}

[复制代码](javascript:void(0);)

注:

java.lang.Object[] getArgs()：获取连接点方法运行时的入参列表  
Signature getSignature() ：获取连接点的方法签名对象  
java.lang.Object getTarget() ：获取连接点所在的目标对象  
java.lang.Object getThis() ：获取代理对象本身

3.applicationContext.xml（Spring配置文件）

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

<?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:aop="http://www.springframework.org/schema/aop"

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

xsi:schemaLocation="

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

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

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

http://www.springframework.org/schema/aop/spring-aop-4.1.xsd

">

<bean id="service" class="cn.service.UserService" />

<bean id="error" class="cn.aop.AnnotationAdvice" />

<!-- 针对AOP的配置 -->

<aop:aspectj-autoproxy />

</beans>

[复制代码](javascript:void(0);)

4.MyTest.java

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

package cn.test;

//注解增强测试

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import cn.service.UserService;

public class MyTest {

public static void main(String[] args) {

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

UserService biz=(UserService)ctx.getBean("service");

biz.delete();

System.out.println("success!");

}

}

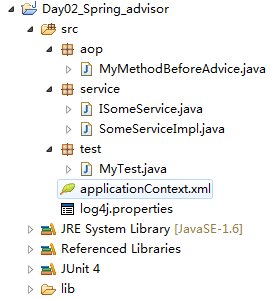
[复制代码](javascript:void(0);)

**四: 顾问（Advisor）实现前置增强**

通知Advice是Spring提供的一种切面(Aspect)。但其功能过于简单，只能  
将切面织入到目标类的所有目标方法中，无法完成将切面织入到指定目标方法中。

顾问Advisor是Spring提供的另一种切面。其可以完成更为复杂的切面织入功能。PointcutAdvisor是顾问的一种，可以指定具体   
的切入点。顾问将通知进行了包装，会根据不同的通知类型，在不同的时间点，将切面织入到不同的切入点。  
PointcutAdvisor接口有两个较为常用的实现类：  
\*:NameMatchMethodPointcutAdvisor 名称匹配方法切入点顾问  
\*:RegexpMethodPointcutAdvisor 正则表达式匹配方法切入点顾问  
<property name="pattern" value=".\*do.\*"></property> 表示方法全名（包名，接口名，方法名）  
运算符 名称 意义  
. 点号 表示任意单个字符  
+ 加号 表示前一个字符出现一次或者多次  
\* 星号 表示前一个字符出现0次或者多次  
=====默认Advisor自动代理生成器  
DefaultAdvisorAutoProxyCreator  
=====BeanName自动代理生成器  
BeanNameAutoProxyCreator

实例:



源码介绍:

1.ISomeService.java

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

package service;

//接口

public interface ISomeService {

//待实现的方法

public void doFirst();

public void doSecond();

}

[复制代码](javascript:void(0);)

2.SomeServiceImpl.java

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

package service;

//接口实现类

public class SomeServiceImpl implements ISomeService {

//实现接口定义的方法

@Override

public void doFirst() {

System.out.println("方法A");

}

@Override

public void doSecond() {

System.out.println("方法B");

}

}

[复制代码](javascript:void(0);)

3.MyMethodBeforeAdvice.java

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

package aop;

//前置增强

import java.lang.reflect.Method;

import org.springframework.aop.MethodBeforeAdvice;

public class MyMethodBeforeAdvice implements MethodBeforeAdvice{

@Override

public void before(Method method, Object[] args, Object target)

throws Throwable {

System.out.println("目标方法执行之前执行");

}

}

[复制代码](javascript:void(0);)

4.applicationContext.xml（Spring配置文件）

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

<?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"

xsi:schemaLocation="

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">

<!-- 目标对象 -->

<bean id="someService" class="service.SomeServiceImpl"></bean>

<!-- 切面:通知 -->

<bean id="beforeAdvice" class="aop.MyMethodBeforeAdvice"></bean>

<!-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* -->

<!-- 1.\*:NameMatchMethodPointcutAdvisor 名称匹配方法切入点顾问 -->

<!-- 切面:顾问 顾问(Advisor)要包装通知（Advice） -->

<bean id="beforeAdvisor" class="org.springframework.aop.support.NameMatchMethodPointcutAdvisor">

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

<!--指定需要增强的方法：这里是doFirst()方法，而doSecond()方法则不会增强 -->

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

<!-- 也可以使用mappedNames指定多个方法

<property name="mappedNames" value="doFirst,doSecond"></property>

-->

</bean>

<!-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* -->

<!-- 2.\*:RegexpMethodPointcutAdvisor 正则表达式匹配方法切入点顾问 -->

<!-- 切面： 顾问 顾问(Advisor)要包装通知（Advice） -->

<!-- <bean id="beforeAdvisor"

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

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

<property name="pattern" value=".\*doF.\*t"></property>

</bean> -->

<!-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* -->

<!-- 代理工厂Bean -->

<bean id="serviceProxy" class="org.springframework.aop.framework.ProxyFactoryBean">

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

<property name="interceptorNames" value="beforeAdvisor"></property>

</bean>

</beans>

[复制代码](javascript:void(0);)

5.MyTest.java

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

package test;

//对顾问（Advisor）测试

import org.junit.Test;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import service.ISomeService;

public class MyTest {

@Test

public void testOne(){

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

ISomeService service = (ISomeService)ctx.getBean("serviceProxy");

service.doFirst();

service.doSecond();

}

}

[复制代码](javascript:void(0);)

分类: [Spring](https://www.cnblogs.com/zhangzongle/category/886748.html)

[**好文要顶**](javascript:void(0);) [**关注我**](javascript:void(0);) [**收藏该文**](javascript:void(0);) **[https://common.cnblogs.com/images/icon_weibo_24.png](javascript:void(0);)** **[https://common.cnblogs.com/images/wechat.png](javascript:void(0);)**

[https://pic.cnblogs.com/face/826791/20160105153405.png](http://home.cnblogs.com/u/zhangzongle/)

[天尽头的那片海](http://home.cnblogs.com/u/zhangzongle/)  
[关注 - 45](http://home.cnblogs.com/u/zhangzongle/followees)  
[粉丝 - 74](http://home.cnblogs.com/u/zhangzongle/followers)

[+加关注](javascript:void(0);)

2