# （1）叙述基于xml的spring aop实现过程。

1: 添加spring类库。

2：编写目标类和切面类。

3：编写配置文件，将切面应用于目标。

4：编写测试程序Test。

# （2）叙述基于注解的spring aop实现过程。

1: 添加spring类库(5个)。

2：编写目标类和切面类。

3：编写配置文件，将切面应用于目标。

4：编写测试程序Test。

# （3）通知有哪几种类型，执行前后顺序？

五种：@Before、@Around、@After、@AfterReturning、@AfterThrowing

当方法符合切点规则不符合环绕通知的规则时候，执行的顺序如下

@Before→@After→@AfterRunning(如果有异常→@AfterThrowing)

当方法符合切点规则并且符合环绕通知的规则时候，执行的顺序如下

@Around→@Before→@After→@Around→@AfterRunning(如果有异常→@AfterThrowing)

# （4）方面类中如何获取拦截方法的方法名，参数值，执行结果等信息？

在bean.xml 中配置 <aop:aspectj-autoproxy></aop:aspectj-autoproxy>

拦截方法名 ：

@Before("pointCutMethod()")

public void doBefore(JoinPoint jp) {

System.out.println("doBefore: " + jp.getTarget().getClass().getName() + "." + jp.getSignature().getName());

}

拦截参数值：通过getArgs（）

public Object doAround(ProceedingJoinPoint pjp) throws Throwable {

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

Object retVal = pjp.proceed();

System.out.println("参数值"+ Arrays.toString(pjp.getArgs()));

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

return retVal;

}

拦截执行结果：

@AfterReturning(pointcut = "pointCutMethod()", returning = "result")

public void doAfterReturning(JoinPoint jp,String result) {

System.out.println("doAfterReturning:" + jp.getTarget().getClass().getName() + "." + jp.getSignature().getName()+"result:"+result);

}

# （5）运用spring aop对spring maven工程中service.UserSerive的login方法添加日志（可调用util下工具类）：

用户登录时，记录一条日志，日志包含时间、登录用户名，密码、登录是否成功等信息。提交所有改动代码

（包括方面类、配置文件，main方法，日志结果截图等。。）。

## 方面类

package aop;

import org.aspectj.lang.JoinPoint;

import org.aspectj.lang.ProceedingJoinPoint;

import org.aspectj.lang.annotation.\*;

import org.springframework.stereotype.Component;

import util.Log;

import java.text.SimpleDateFormat;

import java.util.Arrays;

import java.util.Date;

@Component

@Aspect

public class TestAnnotationAspect {

@Pointcut("execution(\* service.\*.\*(..))")

private void pointCutMethod() {

}

//声明后置通知

@AfterReturning(pointcut = "pointCutMethod()", returning = "result")

public void doAfterReturning(JoinPoint jp, String result) {

System.out.println("doAfterReturning:" + jp.getTarget().getClass().getName() + "." + jp.getSignature().getName()+"result:"+result);

SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");//设置日期格式

Log.writeLog("时间："+df.format(new Date())+",登录用户名："+ jp.getArgs()[0]+",密码："+jp.getArgs()[1]+",登录"+result);

}

}

配置文件

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

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/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd">

<!-- 配置自动扫描的包 -->

<context:component-scan base-package="service,aop">

</context:component-scan>

<!-- 自动为切面方法中匹配的方法所在的类生成代理对象。 -->

<aop:aspectj-autoproxy>

</aop:aspectj-autoproxy>

</beans>

main方法

package main;

import org.aspectj.lang.annotation.Before;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import service.IUserService;

import service.UserService;

public class Main {

static ApplicationContext ctx= new ClassPathXmlApplicationContext("beans.xml");

public static void main(String[] args) {

// TODO Auto-generated method stub

IUserService userService=(IUserService) ctx.getBean("iUserService");

//注意 在 Userservice 加上 @Service(value = "iUserService")

userService.login("abc","123");

}

}

日志结果截图