Spring AOP的核心就是jdk的动态代理技术。

在AopProxy代理对象的生成过程中，首先要从AdvisedSupport对象中取得配置的目标对象，需要根据配置的情况来决定使用什么方式来创建AopProxy代理对象.

连接点：执行点，和方位

用切点来定位执行点(即通过切点来定位接连点)，切点到连接点是1:n的关系,

连接点事方法执行前，执行后等包含方位信息的具体执行点，而切点只能定位到某个方法上，所以要定位到具体的连接点上，还需要提供方位信息。

增强(advice)除了描述一段程序代码之外，还拥有执行点的方位信息，结合执行点的方位信息和执行点信息，就可以找到特定的连接点了。(BeforeAdvice,AfterReturningAdvice)

所以，只有结合切点和增强两者，才能确定特定的连接点并实施增强逻辑。

织入是将增强添加到目标类具体的连接点上的过程。

切面由切点和增强(引介)组成。

原来的场景是在每一个方法的前面，或者是结束处，添加一段相同的代码，每一个方法上。

当使用jdk的动态代理时：(只能为接口创建代理实例)

**public** **class** PerformaceHandler **implements** InvocationHandler {

**private** Object target;

**public** PerformaceHandler(Object target){

**this**.target = target;

}

**public** Object invoke(Object proxy, Method method, Object[] args)

**throws** Throwable {

PerformanceMonitor.*begin*(target.getClass().getName()+"."+ method.getName());

Object obj = method.invoke(target, args);

PerformanceMonitor.*end*();

**return** obj;

}

}

TEST:

ForumService target = new ForumServiceImpl();

PerformaceHandler handler = new PerformaceHandler(target);

ForumService proxy = (ForumService) Proxy.newProxyInstance(

target.getClass().getClassLoader(),

target.getClass().getInterfaces(),

handler);

proxy.removeForum(10);

proxy.removeTopic(1012);

//注意一个问题(晓进的auth\_core中：一个是对接口方法做了一个日志的监控，另一个是缓存)

//我的测试功能：在控制台中将我的数据打印出来

<bean id=*"greetingAdvice"* class=*"com.baobaotao.advice.GreetingBeforeAdvice"* />

<bean id=*"target"* class=*"com.baobaotao.advice.NaiveWaiter"* />

<!--1. greetingAdvice -->

<!--

bean id="waiter" class="org.springframework.aop.framework.ProxyFactoryBean"

p:proxyInterfaces="com.baobaotao.advice.Waiter"

p:target-ref="target"

p:interceptorNames="greetingAdvice"/

-->

Spring通过Advisor(切面)将Pointcut和Advice两者组装起来。

增强提供了连接点的方位信息:如织入到方法前还是方法后等，而切点进一步的描述织入到哪些类的哪些方法上。

一般切面(仅包含Advice),切点的切面(Advice,pointCut),引介切面.

PointcutAdvisor有6个实现类：DefaultPointcutAdvisor,NameMatchMethodPointcutAdvisor…

实际上，他们都是通过拓展对应的Pointcut实现类，并实现PointcutAdvisor接口进行定义的。

切面代码：

**public** **class** GreetingAdvisor **extends** StaticMethodMatcherPointcutAdvisor {

**public** **boolean** matches(Method method, Class clazz) {

**return** "greetTo".equals(method.getName());

}

**public** ClassFilter getClassFilter(){

**return** **new** ClassFilter(){

**public** **boolean** matches(Class clazz){

**return** Waiter.**class**.isAssignableFrom(clazz);

}

};

}

}

//Advisor需要配合一个增强才会有作用，这边将一个前置增强装配到一个切面中去了，之后，配置切面就可以了。

<!-- 普通方法名匹配切面 -->

<bean id=*"waiterTarget"* class=*"com.baobaotao.advisor.Waiter"* />

<bean id=*"sellerTarget"* class=*"com.baobaotao.advisor.Seller"* />

<bean id=*"greetingAdvice"* class=*"com.baobaotao.advisor.GreetingBeforeAdvice"* />

<bean id=*"greetingAdvisor"* class=*"com.baobaotao.advisor.GreetingAdvisor"*

p:advice-ref=*"greetingAdvice"* />

<bean id=*"parent"* abstract=*"true"*

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

p:interceptorNames=*"greetingAdvisor"* p:proxyTargetClass=*"true"* />

<bean id=*"waiter"* parent=*"parent"* p:target-ref=*"waiterTarget"* />

<bean id=*"seller"* parent=*"parent"* p:target-ref=*"sellerTarget"* />

//以上是静态的切面配置

//当然还有动态切面

在Spring中，不管是静态切面还是动态切面，使用的都是动态代理技术，所谓的静态切面是指生成代理对象时就确定好了，增强是否需要织入到目标类的连接点上，而动态切面是指在运行期间根据方法的入参来决定是否需要织入到目标类的连接点上。

//注解的方式进行处理@AspcetJ

@Aspect

**public** **class** PreGreetingAspect{

@Before("execution(\* greetTo(..))")

**public** **void** beforeGreeting(){

System.***out***.println("How are you");

}

}

Waiter target = **new** NaiveWaiter();

AspectJProxyFactory factory = **new** AspectJProxyFactory();

factory.setTarget(target);

factory.addAspect(PreGreetingAspect.**class**);

Waiter proxy = factory.getProxy();

proxy.greetTo("John");

proxy.serveTo("John");

//当然，这边可以注意的一个问题就是，切点表达式函数，这是一个重点，我的观点就是，在进行方法的筛选时，用基本的几个表达式就可以，另注意，书上的一个表格。

//至于Schema的AOP,在没有JDK5.0的环境下进行的.这个暂时不做考虑。

以上，周五，于图书馆。(今天在恒隆将电脑升级成了11,另外，买了一个touch,是的，今天任性的请了一天的假。)