**3.1.3  Pointcut切点**

Pointcut（切点）：决定Advice通知应该作用于哪个连接点，也就是说通过Pointcut切点来定义需要增强的方法的集合，这些集合的选取可以按照一定的规则来完成。在这种情况下，Pointcut通常意味着标识方法，例如，这些需要增强的地方可以是被某个正则表达式进行标识，或根据某个方法名进行匹配等。

为了方便用户使用，Spring AOP提供了具体的切点供用户使用，关于切点，可以看看Spring AOP中的类继承体系，如图3-5所示。

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20091222/142831902.jpg> |
|  |

在Pointcut的基本接口定义中可以看到，需要返回一个MethodMatcher；对于Point的匹配判断功能，具体是由这个返回的MethodMatcher来完成的，也就是说，由这个MethodMatcher来判断是否需要对当前方法调用进行增强，或者说对当前调用方法是否需要应用配置好的advice通知。在Pointcut的类继承关系中，我们以正则表达式切点 JdkRegexpMethodPointcut的实现原理为例，来具体了解切点Pointcut的工作原理。JdkRegexpMethodPointcut类完成通过正则表达式对方法名进行匹配的功能。在JdkRegexpMethodPointcut的基类 StaticMethodMatcherPointcut的实现中可以看到，设置的MethodMatcher为StaticMethodMatcher，同时JdkRegexpMethodPointcut也是这个MethodMatcher的子类，它的类层次关系如图3-6所示。

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20091222/142907405.jpg> |
|  |

如图3-6所示，在Pointcut中，通过这样的类继承关系，MethodMatcher对象实际上是可以被配置成JdkRegexpMethodPointcut来完成方法的匹配判断的。在JdkRegexpMethod-

Pointcut中，可以看到matches方法，这个方法是MethodMatcher定义的接口方法；在JdkRegexpMethodPointcut的实现中，这个matches方法就是使用正则表达式来对方法名进行匹配的地方。关于在AOP框架中对matches方法的调用，我们会在下面具体分析Spring AOP实现中看到。要了解matches在AOP框架中的调用位置，比较简单的方法就是从matches方法作为起始点，去对它的方法调用关系进行追溯，可以看到对matches方法的调用关系如图3-7所示。

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20091222/142940163.jpg> |
|  |

在对matches方法的调用关系中可以看到，是JdkDynamicAopProxy的invoke方法中触发了对matches方法的调用，熟悉Proxy使用的读者，一定会想到，这个invoke方法应该就是Proxy对象进行代理回调的入口方法，这个invoke回调的实现是使用JDK动态代理完成AOP功能的一部分，关于这部分实现原理，在下面AOP的实现分析中有详细的阐述。我们这里的重点是了解Pointcut的实现原理，比如去具体看看matches本身的实现，对于JdkRegexpMethod-

Pointcut的matches方法，它的是实现如代码清单3-5所示。

代码清单3-5  JdkRegexpMethodPointcut使用matches完成匹配

1. **protected** **boolean** matches(String pattern, **int** patternIndex) {
2. Matcher matcher = **this**.compiledPatterns[patternIndex].
3. matcher(pattern);
4. **return** matcher.matches();
5. }

在JDKRegexpMethodPointcut中，通过JDK来实现正则表达式的匹配，在代码清单3-5中可以看得很清楚；如果要详细了解使用JDK的正则表达式匹配功能，可以参考JDK的API文档。我们接着看看其他的Pointcut。

在Spring AOP中，还提供了其他的MethodPointcut，比如通过方法名匹配进行Advice匹配的NameMatchMethodPointcut。它的matches方法实现很简单，匹配的条件是方法名相同或者方法名相匹配，如代码清单3-6所示。

代码清单3-6  NameMatchMethodPointcut的matches

1. **public** **boolean** matches(Method method, Class targetClass) {
2. **for** (String mappedName : **this**.mappedNames) {
3. **if** (mappedName.equals(method.getName()) ||
4. isMatch(method.getName(),
5. mappedName)) {
6. **return** **true**;
7. }
8. }
9. **return** **false**;
10. }
11. **protected** **boolean** isMatch(String methodName, String mappedName) {
12. **return** PatternMatchUtils.simpleMatch(mappedName, methodName);
13. }