**6.2.3　探讨切入点表达式**

2009-09-02 17:04 马连浩等译 人民邮电出版社 字号：[**T**](javascript:setfont(12);) | [**T**](javascript:setfont(16);)

综合评级：

[想读(5)](http://home.51cto.com/apps/book/index.php?s=/Index/book/bid/771/status/0)  [在读(0)](http://home.51cto.com/apps/book/index.php?s=/Index/book/bid/771/status/1)  [已读(5)](http://home.51cto.com/apps/book/index.php?s=/Index/book/bid/771/status/2)   [品书斋鉴(1)](http://home.51cto.com/apps/book/index.php?s=/Index/book/bid/771/good/1)   [已有10人发表书评](http://home.51cto.com/apps/book/index.php?s=/Index/book/bid/771)

[一键收藏，随时查看，分享好友！](javascript:favorBox('open');)

《Spring高级程序设计》第6章AOP进阶，本章将详细讨论Spring中的AOP特性。尤其是，将从更加现实的角度来看待这个问题：探讨Spring中的框架服务如何让我们能透明地使用AOP，在示例程序中讲述AOP的实际用法，讨论如何使用Spring和AspectJ的集成来克服Spring AOP的限制。本节为大家介绍探讨切入点表达式。

AD：[干货来了,不要等！WOT2015 北京站演讲PPT开放下载！](http://wot.51cto.com/2015operation/)

**6.2.3　探讨切入点表达式**

共有十种类型的切入点表达式，每一种类型的语法都各不相同。即便是execution表达式，虽然表面看上去貌似简单明了，但是在完全利用其语法功能时也会变得非常复杂。

**1. Execution表达式**

Execution表达式语法在Spring中的用法跟AspectJ表达式的语法相同。代码清单6-15给出了正式的语法定义。

代码清单6-15　Execution切入点表达式的语法正式定义

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20090902/1708160.jpg> |
|  |

这其中的问号后缀（?）表示可选的表达式元素。换言之，我们大可不用理会它。让我们来分析我们用过的\* com.apress.prospring2.ch06.simple.TestBean2.\*(..)表达式：星号\*表示任何返回类型（ret-type-pattern，返回类型模式），后面跟着一个全限定类名（declaring-type-pattern，声明类型模式）。我们在这个类名后又跟另一个星号\*(..)，这表示一个任意名称、任意数量（包括零）和任意参数类型。因为我们没有指定修饰符模式（modifiers-pattern）或异常抛出模式（throws-pattern），Spring AOP将匹配任意修饰符和抛出任意异常的方法。

为了匹配com.apress.prospring2.ch06子包中任何类的任何方法（它可以包含任何参数、返回任何类型以及抛出任何异常），我们可能要将表达式写成\* com.apress.prospring2.ch06..\*.\*(..)。对这个表达式进行分析，我们得知它可以匹配一个任意名字和任意参数（最后的\*()）的方法，该方法在以com.apres.prospring2.ch06开头的包中的任意类（中间的\*）中返回任意类型（开头的\*），请注意包名与类名之间的..。而对那些更实际的例子，考虑一下代码清单6-16中的切入点表达式。

代码清单6-16　一个更实用的切入点表达式

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20090902/1708161.jpg> |
|  |

你大约能猜到serviceExecution @pointcut用来干什么。我们可以将它用于tx:advice并让com.apress.prospring2.ch06.services包中的所有类的所有方法变得具有事务性，只要不是简单读取方法或设置方法就行。为简单起见，可以将读取方法看作是一个以get开头的无参方法，而设置方法就是一个任何简单的以set开头的方法。

提示 我们更倾向于使用@Transactional注解来标识一个事务性方法。这甚至比使用@pointcuts更简单。

**2. within表达式**

within表达式的正式语法要比execution表达式的语法简单得多（见代码清单6-17）。

代码清单6-17　within切入点表达式的语法

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20090902/1708162.jpg> |
|  |

我们可以使用..和\*这样普通的通配符。例如，要声明一个可以匹配在com包及其子包的任意类中对任意方法调用的执行过程的切入点，应该写成within(com..\*)。

**3. this表达式**

this切入点表达式的语法跟within表达式的语法相似，唯一的区别是前者不能使用..或\*这样的通配符。this切入点的语义会匹配某一对象上所有的方法执行过程，该对象的类型匹配指定的表达式，但我们怎样匹配com包及其子包下所有的类呢？因此，唯一允许的语法是this(class-name)。例如，this(com.apress.prospring2.ch06.simple.TestBean2)。

**4. target表达式**

target表达式的语法跟this表达式的完全一样。因为target表达式定义了一个可能匹配某一特定表达式的类的对象上的所有方法执行过程的切入点，所以它不能使用通配符。因此，唯一可用的语法就是target(class-name)，例如，target(com.apress.prospring2.ch06.simple.SimpleBean)。

**5. args表达式**

args表达式的语法是args(type-pattern? (, type-pattern)\*)。换言之，我们可以指定零个、一个或多个type-pattern表达式。需要注意的是，当你在execution表达式中使用argument-pattern时，这个execution匹配会为形参的类型求值。args表达式匹配为传给方法的参数的实际类型求值。以代码清单6-18中的类举例来说明。

代码清单6-18　SimpleBean类

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20090902/1708163.jpg> |
|  |

切入点execution(\* SimpleBean.\*(CharSequence, String))可以匹配对方法x("A", "B")的调用，因为这个方法名和形参类型都匹配。然而，切入点execution(\* SimpleBean.\*(String, String))却不能匹配哪怕是使用两个String参数的方法调用（一个StringBuilder和一个String也不行）。如果我们想创建一个切入点来匹配当且仅当实参为String，String时对x(CharSequence, String)方法的调用，那么我们应该将表达式写成args(String, String)。

我们也可以在type-pattern中使用..通配符。如果要匹配某个方法的调用，而该方法由一个Integer类型的参数开始，以一个String类型的参数结尾，中间包含任意类型的任意数量个参数，那么我们就应该将表达式写成args(Integer, .., String)。

args表达式最常见的用法是用在参数构建中，我们将在6.2.6节介绍。

**6. @target表达式**

@target表达式是另一个需要一个完全类型名的简单表达式的例子。而且，这个类型名应该表示为一个@interface。因此，这种表达式的示例用法是@target(Magic)或@target(org.springframework. transaction.annotation.Transactional)。这里Magic和Transactional都是带有@Retention (Retention- Policy.RUNTIME)注解的@interface，而且它们的@Target注解包含ElementType.TYPE。这个切入点将匹配带有此注解的类型下所有方法的调用。如果要匹配带有特定注解的方法，应该使用@annotation表达式。

**7. @within表达式**

@within表达式需要一个完全@interface类型名，它将会匹配在具有特定注解的对象（或者方法）中对任意方法的调用。例如，我们可以这样写，@within(StartsTransaction)，其中StartsTransaction是一个@interface。

**8. @annotation表达式**

跟@target表达式类似，@annotation表达式将匹配任何具有特定注解的方法的执行。@annotation(Transactional)便是一个很好的例子，它会匹配具有Transactional注解的任意方法的执行。

**9. @args表达式**

@args表达式与args表达式相似，它与后者唯一的是它比较参数的注解而不是参数的类型。我们可以使用这个表达式来匹配所有对带有指定注解参数的方法的调用。我们也能使用跟在args表达式中相同的通配符。

**10. bean表达式**

这个表达式是真正Spring专有的。它将会匹配某一个bean中对所有方法的调用，而该bean的id或bean名能与特定名称相匹配。我们可以在bean名中使用星号通配符\*。为了匹配simple bean中所有方法的调用，我们可以写成bean(simple)；为了匹配id或其中一个名称以Service结尾的bean中所有方法的调用，则可以写成bean(\*Service)。