在Spring 2.0中，Pointcut的定义包括两个部分：Pointcut表示式(expression)和Pointcut签名(signature)。让我们先看看execution表示式的格式：  
括号中各个pattern分别表示修饰符匹配（modifier-pattern?）、返回值匹配（ret-type-pattern）、类路径匹配（declaring-type-pattern?）、方法名匹配（name-pattern）、参数匹配（(param-pattern)）、异常类型匹配（throws-pattern?），其中后面跟着“?”的是可选项。  
在各个pattern中可以使用“\*”来表示匹配所有。在(param-pattern)中，可以指定具体的参数类型，多个参数间用“,”隔开，各个也可以用“\*”来表示匹配任意类型的参数，如(String)表示匹配一个String参数的方法；(\*,String)表示匹配有两个参数的方法，第一个参数可以是任意类型，而第二个参数是String类型；可以用(..)表示零个或多个任意参数。  
现在来看看几个例子：  
1）execution(\* \*(..))  
表示匹配所有方法  
2）execution(public \* com. savage.service.UserService.\*(..))  
表示匹配com.savage.server.UserService中所有的公有方法  
3）execution(\* com.savage.server..\*.\*(..))  
表示匹配com.savage.server包及其子包下的所有方法  
除了execution表示式外，还有within、this、target、args等Pointcut表示式。一个Pointcut定义由Pointcut表示式和Pointcut签名组成，例如：

java 代码

1. execution(modifier-pattern?
2. ret-type-pattern
3. declaring-type-pattern?
4. name-pattern(param-pattern)
5. **throws**-pattern?)

java 代码

1. //Pointcut表示式
2. @Pointcut("execution(\* com.savage.aop.MessageSender.\*(..))")
3. //Point签名
4. **private** **void** log(){} 然后要使用所定义的Pointcut时，可以指定Pointcut签名，如  
   上面的定义等同与：

java 代码

1. @Before("og()")

java 代码

1. @Before("execution(\* com.savage.aop.MessageSender.\*(..))")

Pointcut定义时，还可以使用&&、||、!

运算，如：

java 代码

1. @Pointcut("execution(\* com.savage.aop.MessageSender.\*(..))")
2. **private** **void** logSender(){}
3. @Pointcut("execution(\* com.savage.aop.MessageReceiver.\*(..))")
4. **private** **void** logReceiver(){}
5. @Pointcut("logSender() || logReceiver()")
6. **private** **void** logMessage(){}

这个例子中，logMessage()将匹配任何MessageSender和MessageReceiver中的任何方法。  
还可以将一些公用的Pointcut

放到一个类中，以供整个应用程序使用，如：

java 代码

1. **package** com.savage.aop;
2. **import** org.aspectj.lang.annotation.\*;
3. **public** **class** Pointcuts {
4. @Pointcut("execution(\* \*Message(..))")
5. **public** **void** logMessage(){}
6. @Pointcut("execution(\* \*Attachment(..))")
7. **public** **void** logAttachment(){}
8. @Pointcut("execution(\* \*Service.\*(..))")
9. **public** **void** auth(){}
10. }

在使用这些Pointcut时，指定完整的类名加上Pointcut

签名就可以了，如：

java 代码

|  |
| --- |
| 1. **package** com.savage.aop; 2. **import** org.aspectj.lang.JoinPoint; 3. **import** org.aspectj.lang.annotation.\*; 4. @Aspect 5. **public** **class** LogBeforeAdvice { 6. @Before("com.sagage.aop.Pointcuts.logMessage()") 7. **public** **void** before(JoinPoint joinPoint) { 8. System.out.println("Logging before " + joinPoint.getSignature().getName()); 9. } 10. } |

当基于XML Sechma实现Advice时，如果Pointcut需要被重用，可以使用<aop:pointcut></aop:pointcut>来声明Pointcut，然后在需要使用这个Pointcut的地方，用pointcut-ref引用就行了，如：

xml 代码

|  |
| --- |
| 1. **<aop:config>** 2. **<aop:pointcut** id="log" 3. expression="execution(\* com.savage.simplespring.bean.MessageSender.\*(..))"**/>** 4. **<aop:aspect** id="logging" ref="logBeforeAdvice"**>** 5. **<aop:before** pointcut-ref="log" method="before"**/>** 6. **<aop:after-returning** pointcut-ref="log" method="afterReturning"**/>** 7. **</aop:aspect>** 8. **</aop:config>** |