定义切入点

# 切入点指示符

**完整的AspectJ**中有很多**切入点指示符**，不过很多Spring AOP都不支持。Spring AOP主要是支持如下几个：**execution、**within、this、target、args、beans。最重要的就是**execution。**

**Spring AOP不支持的切入点指示符有：call、get、set、handler、cflow、if等。**

## execution

### 作用：用于匹配执行方法的连接点。是Spring AOP中最主要的切入点指示符。该切入点指示符的用法最为复杂。

### execution表达式的格式：

* 1. execution (modifiers-pattern? **ret-type-pattern** declaring-type-pattern? **name-pattern**(**param-pattern**) throws-pattern?) //带有问号的可省略

②modifiers-pattern：指定方法的修饰符，支持通配符，可省略。如public等

**③ret-type-pattern：指定方法的返回值类型，支持通配符。可以使用\*来匹配任何返回值类型。**

④declaring-type-pattern：指定方法所属的类的全名（包名+类名），支持通配符，该部分可省略。

**⑥name-pattern：指定匹配指定的方法名，支持通配符，可以用\*匹配所有的方法。**

**⑦param-pattern：指定方法声明中的形参列表，支持两个通配符。即\*和.. 。其中\*代表任何参数的方法，..匹配一个接受任意数量参数的方法（零个或一个或多个）**

**(\*):表示匹配方法中只有一个参数，参数类型为任意类型。**

**(\*,\*):表示匹配方法中有两个参数，参数类型为任意类型。(\*,..)至少一个参数；(\*,\*,..)至少两个参数；通过这种方式，可以限定方法参数的个数。**

**(String,\*,..):至少2个参数，第一个必须为String，第二个任意，后面可有可无。**

**(\*,String,..)：至少两个参数，第一个任意，第二个必须为String，后面可有可无。**

**(..,String) : 参数至少一个，最后一个必须为String。**

**注意：如果想要调用子包下的所有类，必须用..\*表示，\*不能代表多级目录。类名与方法名连在一起，但是方法名必须指定或用\*号匹配。**

**⑧throws-pattern ：指定方法声明抛出的异常，支持通配符，可省略。**

### 示例：

// @Pointcut("execution(\* cn.bupt.aspect\_spring.Singer.\*(\*,..))")

// @Pointcut("execution(\* cn.bupt.aspect\_spring.Singer.\*(\*,\*))")

@Pointcut("execution(\* cn.bupt.aspect\_spring.Singer.\*(String,..))")

**public** **void** pointCut1() {

}

## args

### 作用：用于对连接点的参数类型进行限制，要求参数类型是指定类型的实例。

（注意：Spring AOP只能匹配方法执行的连接点）

**args: 只是对待增强处理的方法的参数进行限制，包括参数个数和参数类型**。而对方法的存在的包、类、返回值不加以限制。注意与execution的区别。

### 示例：

@Pointcut("args(java.lang.String)")

**public** **void** argsPointcut(){}

调用：

@Before("argsPointcut()")

**public** **void** myBeforeDance(){

System.***out***.println("args 切入点指示符");

}

与execution的区别：

@Pointcut("execution(\* cn.bupt.aspect\_spring.Singer.\*(java.lang.String))")

**public** **void** pointCut1() {

}

**args** 是匹配动态传入的参数值与args指定参数的类型和个数一致的方法，而execution是匹配方法签名中的参数类型和个数与指定类型个数一致的方法。

## within

### 作用用于限定匹配特定类里的所有的连接点（就是特定类下的所有方法）。

### 示例：

// 匹配Singer类下的所有的连接点（Spring AOP中就是方法）

@Pointcut("within(cn.bupt.aspect\_spring.Singer)")

// 匹配cn.bupt.aspect\_spring包下所有类下的所有的连接点（Spring AOP中就是方法）

**// @Pointcut("within(cn.bupt.aspect\_spring.\*)")**

// 匹配cn.bupt.aspect\_spring包及其子包下的所有的连接点（Spring AOP中就是方法）

**// @Pointcut("within(cn.bupt.aspect\_spring..\*)")**

**public** **void** pointCut1() {

}

## this

### 作用：用于限定AOP代理必须是指定类型的实例，匹配该对象的所有连接点。

### 示例：AOP代理只能是Singer的实例，也就是只可以对Singer中的方法进行增强处理。与target有相似之处。

@Pointcut("this(cn.bupt.aspect\_spring.Singer)")

**public** **void** pointCut1() {

}

## target

### 作用：用于限定目标对象必须是指定类型的实例，匹配该对象的所有连接点。在Spring AOP中，就是匹配指定类型实例的所有的方法。

### 示例：只匹配Singer的实例。

@Pointcut("target(cn.bupt.aspect\_spring.Singer)")

**public** **void** pointCut1() {

}

## bean（Spring AOP特有的）

### 作用：用于限制只匹配指定Bean实例内的连接点，Spring AOP中只能使用方法执行作为连接点。就是只对指定的Bean的方法进行增强处理。

### 使用：定义Bean表达式时需要传入Bean的id或name，表示只匹配该Bean实例内的连接点。

### bean切入点指示符时Spring AOP特有的，AspectJ不支持bean切入点指示符。bean切入点表达式非常有用，可以明确指定为Spring容器中的哪个Bean织入增强处理。

如： 该**切入点**只对bean的id为singer内的方法进行增强处理。

@Pointcut("bean(singer)")

**public** **void** pointCut1() {

}

# 切入点表达式的组合

Spring支持是使用 与或非 三个逻辑运算符来组合切入点表达式。

## &&：要求连接点同时匹配两个切入点表达式。

## ||：只要连接点匹配任意一个切入点表达式；

## ！：要求连接点不匹配指定的切入点表达式。

示例：

@Pointcut("execution(\* cn.bupt..\*(..))**&&**args(java.lang.String)")

**public** **void** fuhePointcut() {

}

# 如何定义切入点

**所谓定义切入点，其实质就是为了一个切入点表达式起一个名字，从而允许多个增强处理中复用该名称对应的切入点。**

## 定义切入点包含两部分

### 切入点表达式（十分重要）

十分重要，主要是限定哪些连接点可以作为切入点。

利用@Pointcut(value=“”)

**主要是定义value属性**，至于argNames属性，不知道具体作用。

### 包含名字和任意参数的方法签名

方法的名字就是**切入点的名称**；

方法的参数：主要是JoinPoint、ProceedingJoinPoint、需要返回接收的参数。

方法只需要是空即可，即{} 。只是作为一名声明（签名）。

方法的权限修饰符：定义切入点的权限。通java中的普通方法一样，public、private等，调用切入点时，通调用方法一样，**pointcut=”包名.切面类名.方法名()”，主要不要忘了括号。**

### 示例

定义了一个名称为pointcut1的切入点，权限为public。

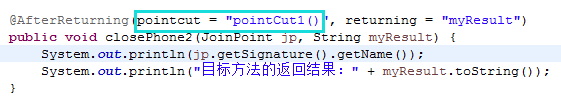


### 如何调用（复用）定义好的切入点：

**用value或省略调用**



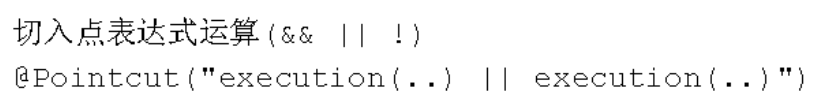
**用pointcut调用**

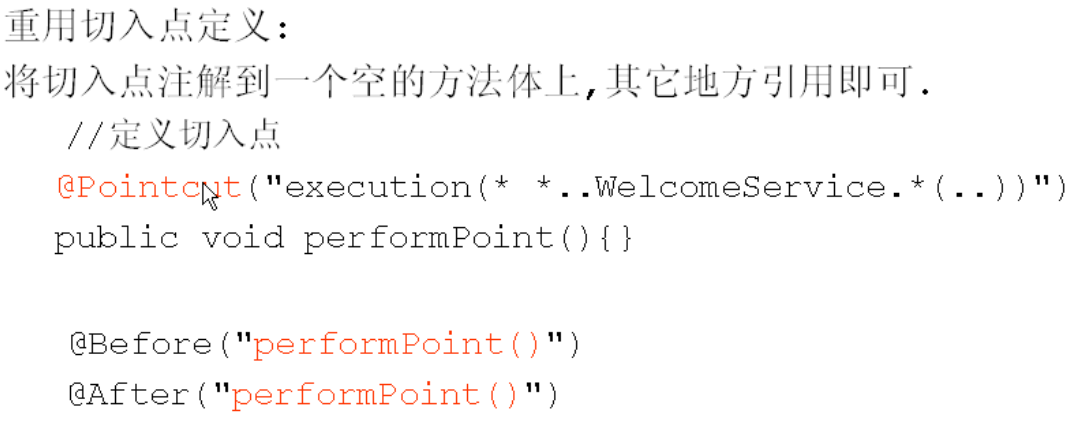


# AspectJ的表达式规则

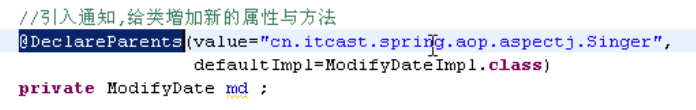


**表达式的与或非**：



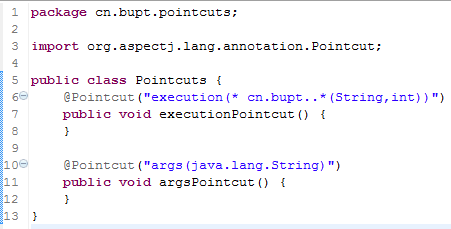




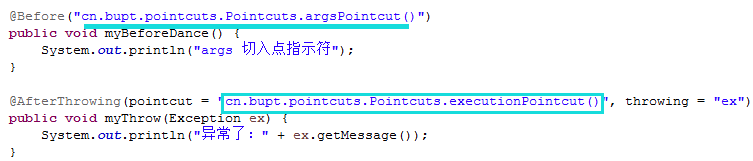


# 创建切入点类

将所有的切入点定义到一个类中，然后在切面类中直接调用切入点即可。



调用：



# 示例

利用**方法签名**编写**AspectJ切入点表达式**

　　最典型的切入点表达式时根据方法的签名来匹配各种方法：

## -execution \* com.yl.spring.aop.ArithmeticCalculator.\*(..):

匹配ArithmeticCalculator中声明的所有方法，第一个\*代表任意修饰符及任意返回值，第二个\*代表任意方法，..匹配任意数量的参数。若目标类与接口与切面在同一个包中，可以省略包名。

## -execution public \* ArithmeticCalculator.\*(..):

匹配ArithmeticCalculator接口的所有公有方法。

## -execution public double ArithmeticCalculator.\*(..):

匹配ArithmeticCalculator中返回double类型数值的方法。

## -execution public double ArithmeticCalculator.\*(double, ..):

匹配第一个参数为double类型的方法，..匹配任意数量任意类型的参数。

## -execution public double ArithmeticCalculator.\*(double, double):

匹配参数类型为double，double类型的方法。