链滴

Spring 面向切面

作者：jiangyue

原文链接：https://ld246.com/article/1478863077325

来源网站：链滴

在软件开发中，散布在应用中多处的功能被称为横切关注点 cross-cutting concern</trong>，通常横切关注点(如日志)从概念上是与应用的业务逻辑相分离，却又嵌入在业务逻辑中。把切关注点与业务逻辑相分离正是面向切面编程 AOP 所要解决的问题。

面向切面编程时，在一个地方定义通用功能，通过声明的方式定义这个功能以何种方式在何处应用，而无需修改被影响的类。横切关点可以被模块化为特殊类，称为切面 aspect。

在 AOP 术语中，切面的工作被称为通知，定义了切要完成的工作及何时执行。

Spring 切面支持 5 种类型的通知：前置通知 Before后置通知 After，返回通知 After-Returning，常通知 After-throwing，环绕通知 Around。

连接点是应用执行过程中能够插入切面的一个点，这个点可以是用方法时、抛出异常时、甚至修改一个字段时

切点是通知所要织入的一个或多个连接点。

切面是通知和切点的结合，通知和切点共同定义了切的全部内容，何时在何处完成何种工作。

引入为向现有类添加新的方法或属性。

织入是把切面应用到目标对象并创建新的代理对象的过程，切面在指定的接点(切点)被织入到目标对象中。

Spring 提供了 4 种类型的 AOP 支持：基于代理的经典 SpringOP，纯 POJO 切面，@AspectJ 注解驱动的切面</stron>，注入式 AspectJ 切面。

借助 Spring 的 aop 命名空间可以将纯 POJO 转换为切面，遗憾的是这种式只能通过 xml 配置。

注解驱动的 AOP 本质上依然是 Spring 基于代理的 AOP，编程模型几乎与 AspectJ 注解切面完一致，好处在于可以不使用 xml 配置。

如果 AOP 需求超过了简单的方法调用，如构造器或属性拦截，可以使用 AspectJ 来实现切面。

在 Spring AOP 中使用 AspectJ 的切点表达式语言来定义切点。进一步了解 AspectJ 和 AspectJ 切点表达式语言，可以看 Ramniva Laddad 编写的《AspectJ in Action》。

切点表达式 <code>execution (* package.class.function(..) && within(package.*)</code> 。第一个 <code>*</code> 表示任意返回类型，<code>..</code> 表示使用任意参数，<code>&&</code>表示"与"关系，xml 中可以使用<code>and</code>，类似 <code>||</code> 可以使用 <code>or</code>，<code>!</code> 可以使用 <code>not</code>。

Spring 中引入了一个新的 bean 指示器，用来指定 bean 的 ID，如 <code>execution (* package.class.function(..) && bean("beanID"))</code> 限制切点只匹配特定的 bean，也可以在 bean()指示器前添加<code>!</code>表示除特定 ID 以外的其他 bean。

<code>@AspectJ</code> 注解用来定义切面，在类的方法上使用定义通知的注解。AspectJ 供了 5 个注解来定义通知：<code>@After</code>，<code>@AfterReturning</code>，<code>@AfterThrowing</code>，<code>@Around</code>，<code>@Before</code>。定义通知的注解需要给定一个切点表达式作为它的值。

<code>@PointCut</code> 注解能够在一个<code>@AspectJ</code> 切面内定义可重用的切点。使用时在一个方法上添加注解，将切点表达式作为注解的值，方法为注提供依附，方法名 function() 可在其他需要切点表达式的地方代替切点表达式

在 JavaConfig 中，可以在配置类的级别上通过 <code>@EnableAspectJAutoProxy</code> 注解启动 <code>@AspectJ</code> 的自动代理功能。xml 中使用 <code><aop:aspectj-autoproxy /></code>。

环绕通知是最强大的通知类型，方法中需要有 ProceedingJoinPoint 参数(jp)，并添加 <code>@Around</code> 注解。在方法体中，<code>jp.proceed()</code> 表示切点，可以围绕它编写各种通知。如果不调用 proceed()，通知实际上会塞被通知方法的调动。

对于有参数的通知，需要在切点表达式的方法中指定接受参数的类型，并过 <code>args(paramName)</code>限定符匹配参数名称。例如 <code>execution(* package.class.function(int) && args(paramName))</code>。

Java 不是动态语言，但是可以利用 AOP 中的"引入"概念，通过切面为 Spring bean 添加新方。

面向注解的切面声明必须要为类添加注解，没有源码时，可通过 xml 配置明。

大多数 aop 配置元素需要在 <code><aop:config></code> 元素的上下文内使用，<code><aop:aspect ref="beanId"></code> 声明切面，<code><aop:pointcut id="pointId" expression="..." /></code> 定义切点，<code><aop:before(其他通知类型) pointcut-ref="pontcutId" method="methodName" /></code> 定义通知。

如果多个切面需要用到相同的切点，可以将 <code><aop pointcut></code> 元素放在 <code><aop:config></code> 元素的范围内。

<style><!--
h1,
h2,
h3,
h4,
h5,
h6,
p,
blockquote {
    margin: 0;
    padding: 0;
}
body {
    font-family: "Helvetica Neue", Helvetica, "Hiragino Sans GB", Arial, sans-serif;
    font-size: 13px;
    line-height: 18px;
    color: #737373;
    background-color: white;
    margin: 10px 13px 10px 13px;
}
table {
    margin: 10px 0 15px 0;
    border-collapse: collapse;
}
td,th {
    border: 1px solid #ddd;
    padding: 3px 10px;
}
th {
    padding: 5px 10px;
}

a {

color: #0069d6;

}

a:hover {

color: #0050a3;

```
text-decoration: none;
}
a img {
border: none;
}
p {
margin-bottom: 9px;
}
h1,
h2,
h3,
h4,
h5,
h6 {
color: #404040;
line-height: 36px;
}
h1 {
margin-bottom: 18px;
font-size: 30px;
}
h2 {
font-size: 24px;
}
h3 {
font-size: 18px;
}
h4 {
font-size: 16px;
}
h5 {
font-size: 14px;
}
h6 {
font-size: 13px;
}
hr {
```

```
margin: 0 0 19px;
border: 0;
border-bottom: 1px solid #ccc;
}
blockquote {
padding: 13px 13px 21px 15px;
margin-bottom: 18px;
font-family:georgia,serif;
font-style: italic;
}
blockquote:before {
content:"\201C";
font-size:40px;
margin-left:-10px;
font-family:georgia,serif;
color:#eee;
}
blockquote p {
font-size: 14px;
font-weight: 300;
line-height: 18px;
margin-bottom: 0;
font-style: italic;
}
code, pre {
font-family: Monaco, Andale Mono, Courier New, monospace;
}
code {
background-color: #fee9cc;
color: rgba(0, 0, 0, 0.75);
padding: 1px 3px;
font-size: 12px;
-webkit-border-radius: 3px;
-moz-border-radius: 3px;
border-radius: 3px;
}
pre {
```

display: block;
padding: 14px;
margin: 0 0 18px;
line-height: 16px;
font-size: 11px;
border: 1px solid #d9d9d9;
white-space: pre-wrap;
word-wrap: break-word;
}
pre code {
background-color: #fff;
color:#737373;
font-size: 11px;
padding: 0;
}
sup {
font-size: 0.83em;
vertical-align: super;
line-height: 0;
}
● {
-webkit-print-color-adjust: exact;
}
@media screen and (min-width: 914px) {
body {
width: 854px;
margin:10px auto;
}
}
@media print {
body,code,pre code,h1,h2,h3,h4,h5,h6 {
color: black;
}
table, pre {
page-break-inside: avoid;
}
}

--></style>