黑客派

Java 注解的玩儿法。

作者：cando

原文链接：https://hacpai.com/article/1533114141009

来源网站：黑客派

元注解是指注解的注解，包括@Retention @Target @Document @Inherited 四种。

呃，看了一眼源码其实 8（我截止到今天用的是 8，2018/8/1）里面还有 @Repeatable。

先按顺序来分析：
<h3 id="1--Retention作用">1、@Retention 作用</h3>
定义注解的保留策略

@Retention(RetentionPolicy.SOURCE): 注解仅保存在源码阶段，编译和运行时都不会有。</li

<pre><code class="language-java highlight-chroma">/**
* Annotations are to be discarded by the compiler.
*/
SOURCE,
</code></pre>

@Retention(RetentionPolicy.CLASS): 注解会在 class 字节码中存在，运行时不可见。此策略
默认。

<pre><code class="language-java highlight-chroma">/**
* Annotations are to be recorded in the class file by the
compiler
* but need not be retained by the VM at run time. This
is the default
* behavior.
*/
CLASS,
</code></pre>

@Retention(RetentionPolicy.RUNTIME): 注解会在 class 字节码文件中存在，在运行时可以通
反射获取到

<pre><code class="language-java highlight-chroma">/**
* Annotations are to be recorded in the class file by the
compiler and
* retained by the VM at run time, so they may be read
eflectively.
*
* @see java.lang.reflect.AnnotatedElement
*/
RUNTIME
</code></pre>
<h3 id="2--Target--定义注解的作用目标">2、@Target： 定义注解的作用目标</h3>

```
<pre><code class="language-java highlight-chroma"><span class="highlight-kd">public</
pan> <span class="highlight-kd">enum</span> <span class="highlight-n">ElementType</
pan> <span class="highlight-o">{</span>
    <span class="highlight-cm">/** Class, interface (including annotation type), or enum decla
ation */</span>
  <span class="highlight-n">TYPE</span><span class="highlight-o">,</span>  <span class=
highlight-c1">// 类，接口，枚举，注解。
</span><span class="highlight-c1"></span>
    <span class="highlight-cm">/** Field declaration (includes enum constants) */</span>
  <span class="highlight-n">FIELD</span><span class="highlight-o">,</span> <span class=
highlight-c1">// 字段， 枚举常量。
</span><span class="highlight-c1"></span>
    <span class="highlight-cm">/** Method declaration */</span>
  <span class="highlight-n">METHOD</span><span class="highlight-o">,</span> <span cl
ss="highlight-c1">// 方法
</span><span class="highlight-c1"></span>
    <span class="highlight-cm">/** Formal parameter declaration */</span>
  <span class="highlight-n">PARAMETER</span><span class="highlight-o">,</span> <span
class="highlight-c1">// 方法参数
</span><span class="highlight-c1"></span>
    <span class="highlight-cm">/** Constructor declaration */</span>
  <span class="highlight-n">CONSTRUCTOR</span><span class="highlight-o">,</span> <
pan class="highlight-c1">// 构造函数
</span><span class="highlight-c1"></span>
    <span class="highlight-cm">/** Local variable declaration */</span>
  <span class="highlight-n">LOCAL_VARIABLE</span><span class="highlight-o">,</span>
span class="highlight-c1">// 局部变量
</span><span class="highlight-c1"></span>
    <span class="highlight-cm">/** Annotation type declaration */</span>
  <span class="highlight-n">ANNOTATION_TYPE</span><span class="highlight-o">,</spa
> <span class="highlight-c1">// 注解
</span><span class="highlight-c1"></span>
    <span class="highlight-cm">/** Package declaration */</span>
  <span class="highlight-n">PACKAGE</span><span class="highlight-o">,</span> <span cl
ss="highlight-c1">// 包
</span><span class="highlight-c1"></span>
    <span class="highlight-cm">/**
</span><span class="highlight-cm">     * Type parameter declaration
</span><span class="highlight-cm">     *
</span><span class="highlight-cm">     * @since 1.8
</span><span class="highlight-cm">     */</span>
 <span class="highlight-n">TYPE_PARAMETER</span><span class="highlight-o">,</span>
span class="highlight-c1">// 新特性,表示这个 Annotation 可以用在 Type 的声明式前
</span><span class="highlight-c1"></span>
    <span class="highlight-cm">/**
</span><span class="highlight-cm">     * Use of a type * * @since 1.8
</span><span class="highlight-cm">     */</span>
 <span class="highlight-n">TYPE_USE</span> <span class="highlight-c1">// 新特性,表示当
Annotation 可以用在所有使用 Type 的地方（如：泛型，类型转换等）。
</span><span class="highlight-c1"></span><span class="highlight-o">}</span>
</code></pre>
<script async src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js"></scr
pt>
<!-- 黑客派PC帖子内嵌-展示 -->
```

<h3 id="3--Documented">3、@Documented</h3>
表示可以出现在 javadoc 中，
<h3 id="4--Inherited">4、@Inherited</h3>
该注解表示子类可以集成加载父类上的注解。但要注意：
<pre><code class="highlight-chroma"> 1.注解定义在类上面，子类是可以继承该注解的。

2.注解定义在方法上面，子类也可以继承该注解，但是如果子类复写了父类中定义了注解的方法，那子类将无法继承该方法的注解，也就是说，子类在复写父类中被@Inherited标注的方法时，会将该方上面的注解覆盖掉

3.Interface的实现类（implements实现）无法继承接口中所定义的被@Inherited标注的注解

</code></pre>

@Inherited 的总结来自：
 CShawnX:Java 和 Android 中的注解
<h3 id="5--Repeatable-可以重复注解">5、@Repeatable 可以重复注解</h3>
在之前，相同的注解在同一个位置只能使用一次， java8 引入了重复注解制。可以让一个注解在一个位置引用多次。例如：
<pre><code class="language-java highlight-chroma">@interfae Persons {
Person[] value();
}

@Repeatable(</sp
n>Persons.</spa
>class)

@interface Person
span class="highlight-o">{

String role default""<span class="
ighlight-o">;

}

@Person(
span class="highlight-n">role="artist")

@Person(
span class="highlight-n">role="coder")

@Person(
span class="highlight-n">role="PM")

public class SuperMan{

}

</code></pre>

表示这个 SuperMan 可以是三种角色，artist，coder，PM。
上面代码中，用@Repeatable 注解了注解 Person，而其括号内的 Persons.class 表示为一个注容器。
 按照规定，它里面必须要有一个 value 的属性，属性类型是一个被 @Repeatable 注解的注解数组，注意它是数组。
<h2 id="注解怎么玩儿-">注解怎么玩儿？</h2>
<blockquote>
注解是一系列元数据，它提供数据用来解释程序代码，但是注解并非是所解释的代码本身的一分。注解对于代码的运行效果没有直接影响。
</blockquote>
<blockquote>
注解有许多用处，主要如下：
</blockquote>
<blockquote>

提供信息给编译器： 编译器可以利用注解来探测错误和警告信息
编译阶段时的处理： 软件工具可以用来利用注解信息来生成代码、HTML 文档或者做其它相应理。
运行时的处理： 某些注解可以在程序运行的时候接受代码的提取

</blockquote>
<h3 id="玩儿法一">玩儿法一</h3>
Java 预置了一些注解，可以通过这些注解，对代码进行描述。达到某种目的。
 @Deprecat d、@Override、@SuppressWarnings、@SafeVarargs、@FunctionalInterface
<h3 id="玩儿法二">玩儿法二</h3>
那就是自己定义一些注解了。
 通过反射，获得相关的注解，并未目标增加一系列附加的处操作。因为正像官方描述：注解对于代码的运行效果没有直接影响。但是我们可以为其增强啊。
 getAnnotation（）或者 getAnnotations（）可以获得目标的注解对象或者所有注解的数组。再根这些注解对象以及其成员属性。
 可以据此为其编写外部能力。
应用场景
日志、测试类、IoC、功能增强等。