0622技术整理

> Title: 0622-技术整理
>
> Author: Bucket
>
> content: Java 注解

注解

是Java语言用于工具处理的标注，一种在编译过程中会被使用或编译的内容。例如`@override`是由编译器检查是否实现了重写，或在加载.class文件时进行动态修改，或程序运行期间读取，例如`@PostConstruct`

定义注解

```
public @interface Report {
    int type() default 0;
    String level() default "info";
    String value() default "";
}
```

元注解

可以修饰其他注解，可直接使用。

- `@Target` 规定了Annotation能够应用的位置，如类或接口，字段，方法等。
- `@Retention` 定义了Annotation的生命周期。如果`@Retention`不存在，则该`Annotation`默认为`CLASS`。因为通常我们自定义的`Annotation`都是`RUNTIME`，所以，务必要加上`@Retention(RetentionPolicy.RUNTIME)`这个元注解。
- 经过`@Repeatable`修饰后，在某个类型声明处，就可以添加多个自定义注解。

处理注解

> √ 什么情况下需要应用这种方法处理注解?

使用反射API读取Annotation。

```
//方法1
Class cls = Person.class;
if (cls.isAnnotationPresent(Report.class)) {
    Report report = cls.getAnnotation(Report.class);
    ...
}
//方法2
Class cls = Person.class;
Report report = cls.getAnnotation(Report.class);
if (report != null) {
   ...
}
```

对一个方法中的参数也可以加多个注解。

```
// 获取Method实例:
Method m = ...
// 获取所有参数的Annotation:
Annotation[][] annos = m.getParameterAnnotations();
// 第一个参数（索引为0）的所有Annotation:
Annotation[] annosOfName = annos[0];
for (Annotation anno : annosOfName) {
    if (anno instanceof Range) { // @Range注解
        Range r = (Range) anno;
    }
    if (anno instanceof NotNull) { // @NotNull注解
        NotNull n = (NotNull) anno;
    }
}
```

使用注解

常用在测试中。

例如自定义注解MyTest，为需要检查的方法加上注解。建一个测试类/方法Check，在其中检查所有方法是否有此注解，对于有注解的方法进行检测，抛出异常或打印信息。