Annatation

李玮玮

讲授思路

- Annatation简介

Annatation简介

- 预定义的注解
  - @Deprecated
  - @Override
- 自定义的注解

Annatation作用

- 帮助开发者提升代码的质量和可读性
- 自动化代码分析的准确性
- 功能
  - 作为特定的标记
  - 额外信息的载体
- Java 8 对 Annotation 引入了两项重要的改变
  - Type Annotation
  - Repeating Annotation

Annatation使用

- 元注解
- 注解的声明
- 注解的使用
- 重复注解

Annatation使用——元注解

- 不同于java中的注释
- 注解是一种类型，可以对方法、类、参数、包、域以及变量等添加标记(即附上某些信息)，之后通过反射将标记的信息提取出来以供使用
- java.lang.annotation包提供了4种元注解

Annatation使用——元注解

- java.lang.annotation包提供了4种元注解

@Target

@Retention

@Document

@Inherited

Annatation使用——元注解

- @Target：表示该注解用于什么位置,可选的参数是ElementType枚举中的成员
  - ElementType在SE1.8中又加入了两个成员 TYPE_PARAMETER、TYPE_USE

TYPE	类型声明(类,接口,enum)
FIELD	成员变量声明(对象,属性,enum的实例)
METHOD	方法声明
PARAMETER	参数声明
CONSTRUCTOR	构造器声明
LOCAL_VARIABLE	局部变量声明
ANNOTATION_TYPE	注解声明
PACKAGE	包声明

Annatation使用——元注解

- @Retention: 表示需要在什么级别保存该注解信息(生命周期)，可选的参数是RetentionPolicy枚举中的成员
  - 注意,只有声明为RUNTIME,才可以通过反射机制读取注解的信息

SOURCE	停留在java源文件,会将被编译器丢弃
CLASS	停留在class文件中,但会被VM丢弃
RUNTIME	内存中的字节码,VM在运行期间保留注解

- @Document: 将注解包含在Javadoc中
- @Inherited: 允许子类继承父类中的注解,只针对CLASS级别的注解有效

Annatation使用——注解的声明

- 语法——关键字@interface定义
- 默认继承Annotation接口
- 元素：表示一些值，类似接口中无参的方法
- 通过default提供默认值（不能为null）

```
@interface AnnotateTest {
int i();
float f() default 3.14f;
char[] c();
String[] s() default { "primitive type",
"String", "Class", "annotation",
"enumeration", "arrays" };
}
```

Annatation使用——注解的使用

- 必须对注解内的元素进行赋值（有默认值的元素除外）
- 注解只有一个元素且该元素的名称是value的话,在使用注解的时候可以省略"value="直接写需要的值即可

```
@Retention(RetentionPolicy.RUNTIME)
@interface AnnotateDemo {
int i();
String s() default "hello";
}

class AnnotateDemo {
@AnnotateDemo(i = 97)
public void a() {
}
}
```

Annatation使用——重复注解

- SE1.8引入了重复注解的特性
  - 允许在声明同一类型时多次使用同一个注解

```
@Repeatable(AnnotateDemoFactory.class)
@interface AnnotateDemo2 {
String s();
}
@interface AnnotateDemoFactory {
AnnotateDemo2[] value();
}
class AnnotateDemo {
@AnnotateDemo2(s = "hello")
@AnnotateDemo2(s = "world")
public void a() {
}
}
```

Thank You