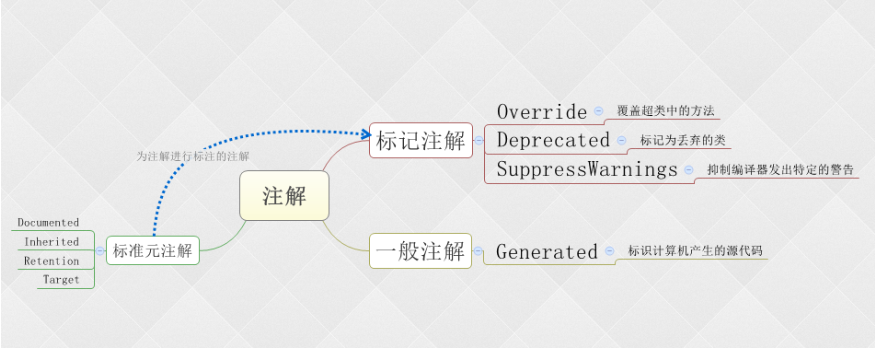
Android Studio Annotation注解技术

|  |  |  |
| --- | --- | --- |
| 版本号 | 描述 | 日期(更新日期)/更新人 |
| 1.0 | Java Annotation注解 | 2018/09/25 刘志保 |
|  |  |  |

Java Annotation注解分类:



上面的图可以看出，注解大体上分为三种：标记注解，一般注解，元注解

java中的常见注解

jdk的自带注解

* @Override：告诉编译器我重写了接口方法
* @Deprecated：告诉编译器这个方法过时了，不建议使用，Ide会在方法上划横线
* @SuppressWarnings("deprecation"):关闭方法中出现的警告

下面列出@SuppressWarnings注解参数的几个常见用法

例:@SuppressWarnings(value={ "rawtypes", "unchecked" })

1.deprecation：去除不暂成使用类的警告

2.serial：当在可序列化的类上缺少 serialVersionUID 定义时的警告

3.finally：任何 finally 子句不能正常完成时的警告

4.rawtypes：去除传参数时也要带泛型

5.unchecked：执行了未检查的转换时的警告，例如当使用集合时没有用泛型来指定集合保存的类型

6.unused：去除对未使用代码的警告

7:all:去除所有类型的警告

# 元注解

元注解的作用就是注解其他注解，一般我们使用自定义注解时，就需要用元注解来标注我们自己的注解，一共有以下四个元注解

1.**@Target**：说明了Annotation被修饰的范围，可被用于 packages、types（类、接口、枚举、Annotation类型）、类型成员（方法、构造方法、成员变量、枚举值）、方法参数和本地变量（如循环变量、catch参数）。在Annotation类型的声明中使用了target可更加明晰其修饰的目标:

例：@Target(ElementType.TYPE)

1.ElementType.CONSTRUCTOR:用于描述构造器

2.ElementType.FIELD:用于描述域（类的成员变量）

3.ElementType.LOCAL\_VARIABLE:用于描述局部变量（方法内部变量） 4.ElementType.METHOD:用于描述方法

5.ElementType.PACKAGE:用于描述包

6.ElementType.PARAMETER:用于描述参数

7.ElementType.TYPE:用于描述类、接口(包括注解类型) 或enum声明

2.**@Retention**：定义了该Annotation被保留的时间长短，有些只在源码中保留，有时需要编译成的class中保留，有些需要程序运行时候保留。即描述注解的生命周期

例：@Retention(RetentionPolicy.RUNTIME)

1.RetentionPoicy.SOURCE:在源文件中有效（即源文件保留）

2.RetentionPoicy.CLASS:在class文件中有效（即class保留）

3.RetentionPoicy.RUNTIME:在运行时有效（即运行时保留）

3.**@Documented**：它是一个标记注解，即没有成员的注解，用于描述其它类型的annotation应该被作为被标注的程序成员的公共API，因此可以被例如javadoc此类的工具文档化

4.**@Inherited**：它也是一个标记注解，它的作用是，被它标注的类型是可被继承的，比如一个class被@Inherited标记，那么一个子类继承该class后，则这个annotation将被用于该class的子类。

注意：一个类型被@Inherited修饰后，类并不从它所实现的接口继承annotation，方法并不从它所重载的方法继承annotation。

# 自定义注解

自定义注解格式：

public @interface 注解名 {定义体}

使用@interface定义一个注解，自动继承了java.lang.annotation.Annotation接口，其中的每一个方法实际上是声明了一个配置参数。方法的名称就是参数的名称，返回值类型就是参数的类型（返回值类型只能是基本类型、Class、String、enum）。可以通过default来声明参数的默认值。

注解参数的可支持数据类型：

1.所有基本数据类型（int,float,boolean,byte,double,char,long,short)

2.String类型

3.Class类型

4.enum类型

5.Annotation类型

6.以上所有类型的数组

定义注解成员的注意点: 第一,只能用public或默认(default)这两个访问权修饰.例如,String value();这里把方法设为default默认类型；

@Target(ElementType.FIELD)

@Retention(RetentionPolicy.RUNTIME)

@Documented public

@interface userName { String value() default ""; }

第二,参数成员只能用基本类型byte,short,char,int,long,float,double,boolean八种基本数据类型和 String,Enum,Class,annotations等数据类型，以及这一些类型的数组:

//定义一个枚举

public enum RequestMethod {

GET, HEAD, POST, PUT, PATCH, DELETE, OPTIONS, TRACE

}

@Target({ElementType.METHOD, ElementType.TYPE}) @Retention(RetentionPolicy.RUNTIME)

@Documented

@Mapping

public @interface RequestMapping {

String name() default "";

String[] path() default {};

RequestMethod[] method() default {};//枚举数组

}

第三,如果只有一个参数成员,最好把参数名称设为"value",后加小括号

# 注解的默认值

注解元素必须有确定的值，要么指定时给默认值，要么使用时给值。不过有时候我们需要确定表达一个元素不存在值，所以使用空字符串或者负数表示某个元素不存在，在定义注解时，这已经成为一个约定用法

@Target(ElementType.FIELD)

@Retention(RetentionPolicy.RUNTIME)

@Documented public

@interface User {

public int id() default -1;

public String name() default "";

public String address() default "";

}

# 对于@Inherited注解的补充



结论：父类的类上和方法上有自定义注解，并且被@Inherited标记，那么子类只有继承的情况下才会继承父类注解。重写，重载，实现父类方法这些都不会继承父类注解。