**Android自定义注解**

参考：<https://blog.csdn.net/fengxingzhe001/article/details/78520298>

<https://blog.csdn.net/hp910315/article/details/51199748>

<https://blog.csdn.net/qq_20521573/article/details/82321755>

<https://www.race604.com/annotation-processing>

<https://jsonchao.github.io/2019/01/13/Android%E4%B8%BB%E6%B5%81%E4%B8%89%E6%96%B9%E5%BA%93%E6%BA%90%E7%A0%81%E5%88%86%E6%9E%90%EF%BC%88%E4%B8%83%E3%80%81%E6%B7%B1%E5%85%A5%E7%90%86%E8%A7%A3ButterKnife%E6%BA%90%E7%A0%81%EF%BC%89/>

1. 元注解：负责注解其他注解。Java5.0定义了4个标准的meta-annotation类型
2. **@Target**说明了Annotation所修饰的对象范围

@Target**取值(ElementType)有：**

**PACKAGE**:用于描述包

**TYPE**:用于描述类、接口(包括注解类型) 或enum声明

**METHOD**:用于描述方法

**CONSTRUCTOR**:用于描述构造器

**FIELD**:用于描述域

**LOCAL\_VARIABLE**:用于描述局部变量

**PARAMETER**:用于描述参数

@Target(ElementType.TYPE)

public @interface Table {

/\*\*

\* 数据表名称注解，默认值为类名称

\* @return

\*/

public String tableName() default "className";

}

@Target(ElementType.FIELD)

public @interface NoDBColumn { }

1. **@Retention**定义了该Annotation被保留的时间长短， **表示需要在什么级别保存该注释信息，用于描述注解的生命周期。**

**@Retention取值（RetentionPoicy）有：**

SOURCE:在源文件中有效（即源文件保留）

CLASS:在class文件中有效（即class保留）

RUNTIME:在运行时有效（即运行时保留）

1. **@**Documented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。Documented是一个标记注解，没有成员。可以被例如javadoc此类的工具文档化。
2. @Inherited 元注解是一个标记注解，@Inherited阐述了某个被标注的类型是被继承的。@Inherited annotation类型是被标注过的class的子类所继承。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。
3. 自定义注解

@interface自定义注解时，自动继承了java.lang.annotation.Annotation接口。在定义注解时，不能继承其他的注解或接口。@interface用来声明一个注解，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：**所有基本数据类型（int，long，short，float，double，boolean，byte，char) ，String类型，Class类型，enum类型，Annotation类型和以上所有类型的数组。只能用public或默认(default)这两个访问权修饰注解。

/\*\*

\* 水果属性注解

\* @author peida

\*

\*/

@Target(ElementType.FIELD)

@Retention(RetentionPolicy.RUNTIME)

@Documented

public @interface FruitAttr {

/\*\*

\* 颜色枚举

\*/

public enum Color{ BULE,RED,GREEN};

/\*\*

\* 颜色属性

\* @return

\*/

Color fruitColor() default Color.GREEN;

/\*\*

\* 重量属性

\* @return

\*/

int fruitWeight() default 0;

}

public class Apple {

@FruitName("Apple")

private String appleName;

@FruitAttr(fruitColor=Color.RED)

private String appleColor;

@FruitAttr(fruitWeight=2)

private int appleWeight;

}

1. 运行时注解：
2. 定义注解；
3. 定义注解处理类；在运行时处理注解，并通过反射实现想要的功能；
4. 编译时注解：
5. 定义注解；
6. 通过继承AbstractProcessor自定义一个注解处理器，编译时扫描注解并生成模版类；
7. 运行时通过反射找到模版类；