# Java 自定义注解

- What?Why?How?

# 什么是注解?

- ***Annontation***是Java5开始引入的新特征，中文名称叫**注解**。
- 它提供了一种安全的类似注释的机制，用来将任何的信息或元数据（metadata）与程序元素（类、方法、成员变量等）进行关联。
- 为程序的元素（类、方法、成员变量）加上更直观更明了的说明，这些说明信息是与程序的业务逻辑无关，并且供指定的工具或框架使用。
- Annontation像一种修饰符一样，应用于包、类型、构造方法、方法、成员变量、参数及本地变量的声明语句中。
- Java注解是附加在代码中的一些元信息，用于一些工具在编译、运行时进行解析和使用，起到说明、配置的功能。
- 注解不会也不能影响代码的实际逻辑，仅仅起到辅助性的作用。包含在java.lang.annotation 包中。

# 注解的作用

- 生成文档。这是最常见的，也是java 最早提供的注解。常用的有@param @return 等
- 跟踪代码依赖性，实现替代配置文件功能。
- 在编译时进行格式检查。如@override 放在方法前，如果你这个方法并不是覆盖了超类方法，则编译时就能检查出。

# 注解的原理

反射

# 内置注解

- @Override：定义在java.lang.Override中，此注释只适用于修饰方法，表示一个方法声明打算重写超类中的另一个方法声明
- @Deprecated：定义在java.lang.Deprecated中，此注释可以修饰方法、属性、类，表示不鼓励程序员使用这样的元素，通常是因为它很危险或者存在更好的选择
- @SuppressWarnings：定义在java.lang.SuppressWarnings中，用来抑制编写编译时的警告信息

# 元注解

- 元注解的做哟个是负责注解其他注解，java中定义了四个标准的meta-annotation类型，他们被用来提供对其他annotation类型作说明
- 这些类型和它们所支持的类在java.lang.annotation包中
  - @Target：用来描述注解的使用范围（注解可以用在什么地方）
  - @Retention：表示需要在什么级别保存该注释信息，描述注解的生命周期
    - Source < Class < Runtime
  - @Document：说明该注解将被包含在javadoc中
  - @Inherited：说明子类可以继承父类中的该注解

# 自定义注解

- 使用@interfac自定义注解时，自动继承了java.lang.annotation.Annotation接口
- 使用规则：
  - @interface用来声明一个注解，格式：public @interface 注解名{}
  - 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（返回值只能是基本类型，Class,String,enum）
  - 可以头盖骨default来声明参数的默认值
  - 如果只有一个参数成员，一般参数名为value
  - 注解元素必须要有值，我们定义注解元素时，经常使用空字符串，0作为默认值