# 元注解(meta-annotation)是什么？

元注解的作用就是负责注解其他注解

四个标准元注解：

1. @Target,  
   2.@Retention,  
   3.@Documented,  
   4.@Inherited

这些类型和它们所支持的类在java.lang.annotation包中

# @Target

用于描述注解的使用范围（即：被描述的注解可以用在什么地方）

**取值(ElementType)有：**

　　　　1.CONSTRUCTOR:用于描述构造器  
　　　　2.FIELD:用于描述域  
　　　　3.LOCAL\_VARIABLE:用于描述局部变量  
　　　　4.METHOD:用于描述方法  
　　　　5.PACKAGE:用于描述包  
　　　　6.PARAMETER:用于描述参数  
　　　　7.TYPE:用于描述类、接口(包括注解类型) 或enum声明

## 例1

**import** java.lang.annotation.Target;

**import** java.lang.annotation.ElementType;

@Target(ElementType.TYPE)//注解Table 可以用于注解类、接口(包括注解类型) 或enum声明

public @interface Table {

/\*\*

\* 定义注解属性tableName，默认值为className

\* @return

\*/

public String tableName() default "className";

}

## 例2

**import** java.lang.annotation.Target;

**import** java.lang.annotation.ElementType;

@Target(ElementType.FIELD)//注解NoDBColumn仅可用于注解类的成员变量

public @interface NoDBColumn {

}

Retetion：决定注解存在于resource，class 或 runtime

Inherited：决定注解是否能被继承

Documented： 决定注解是否能生成API文档

## 例3

**import** java.lang.annotation.Target;

**import** java.lang.annotation.ElementType;

@Target({ElementType.*TYPE*,ElementType.*FIELD*})//当@Target取值为超过一个是应用{}wrap起来

**public** **@interface** NameAnnotation{

/\*\*

\* 定义该annotation的成员，默认值为""

\* **@return**

\*/

String value() **default** "";//当成员名用value()时，引用时 @NameAnnotation(value="\*")中的 value= 可不写

}

# @Retention

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（即：被描述的注解在什么范围内有效）

**取值（RetentionPolicy）有：**

　　　　1.SOURCE:在源文件中有效（即源文件保留）  
　　　　2.CLASS:在class文件中有效（即class保留）  
　　　　3.RUNTIME:在运行时有效（即运行时保留）

## 例1

**import** java.lang.annotation.ElementType;

**import** java.lang.annotation.Retention;

**import** java.lang.annotation.RetentionPolicy;

**import** java.lang.annotation.Target;

@Target(ElementType.*FIELD*)

@Retention(RetentionPolicy.*RUNTIME*)//该注解运行时有效，运行前在源或编译时情况本人不清楚

**public** **@interface** IdAnnotation {

/\*\*

\* 定义该annotation的成员，默认值为-1

\* **@return**

\*/

**int** value() **default** -1;

## 例2

**import** java.lang.annotation.ElementType;

**import** java.lang.annotation.Retention;

**import** java.lang.annotation.RetentionPolicy;

**import** java.lang.annotation.Target;

@Target(ElementType.*FIELD*)

@Retention(RetentionPolicy.*RUNTIME*)

**public** **@interface** Column {

/\*\*

\* 以下为定义该annotation成员，annotation成员可见性只能为public或默 \*认

\* **@return**

\*/

String name() **default** "fieldName";

**public** String setFuncName() **default** "setField";

**public** String getFuncName() **default** "getField";

**public** **boolean** defaultDBValue() **default** **false**;

}

# @Inherited

@Inherited阐述了某个被标注的类型是被继承的。如果一个使用了@Inherited修饰的annotation类型被用于一个class，则这个annotation将被用于该class的子类。

## 例

**import** java.lang.annotation.ElementType;

**import** java.lang.annotation.Inherited;

**import** java.lang.annotation.Retention;

**import** java.lang.annotation.RetentionPolicy;

**import** java.lang.annotation.Target;

@Target(ElementType.*FIELD*)

@Retention(RetentionPolicy.*RUNTIME*)

@Inherited//使用了@Inherited修饰的annotation类型被用于一个class，则这个annotation将被用于该class的子类

**public** **@interface** AgeAnnotation {

/\*\*

\* 定义注解的成员，默认值为-1

\* **@return**

\*/

**int** age() **default** -1;

}

# @Documented

**@**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，因此可以被例如javadoc此类的工具文档化

## 例

**import** java.lang.annotation.Documented;

**import** java.lang.annotation.ElementType;

**import** java.lang.annotation.Retention;

**import** java.lang.annotation.RetentionPolicy;

**import** java.lang.annotation.Target;

@Target(ElementType.*FIELD*)

@Retention(RetentionPolicy.*RUNTIME*)

@Documented//该注解可以被生成API帮助文档

**public** **@interface** Column {

/\*\*

\* 以下定义注解的成员

\* **@return**

\*/

**public** String name() **default** "fieldName";

**public** String setFuncName() **default** "setField";

**public** String getFuncName() **default** "getField";

**public** **boolean** defaultDBValue() **default** **false**;

}

# Note：

当成员名用value()时，引用时 @AnnotationName(value="\*")中的 value= 可不写

annotation成员可见性只能为public或默认