# 元注解

元注解的作用就是负责注解其他注解。Java5.0 定义了 4 个标准的 meta-annotation 类型，它们被用来提供对其它 annotation 类型作说明。

## @Target

1. @Target **说明了 Annotation 所修饰的对象范围**：Annotation 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。
2. 用于描述注解的使用范围。

取值 (ElementType) 有：

CONSTRUCTOR: 用于描述构造器

FIELD: 用于描述域

LOCAL\_VARIABLE: 用于描述局部变量

METHOD: 用于描述方法

PACKAGE: 用于描述包

PARAMETER: 用于描述参数

TYPE: 用于描述类、接口 (包括注解类型) 或 enum 声明

## @Retention,

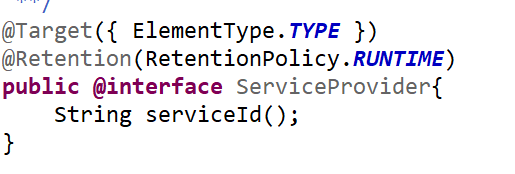
1. @Retention **定义了该 Annotation 被保留的时间长短**：某些 Annotation 仅出现在源代码中，而被编译器丢弃；而另一些却被编译在 class 文件中；编译在 class 文件中的 Annotation 可能会被虚拟机忽略，而另一些在 class 被装载时将被读取（请注意并不影响 class 的执行，因为 Annotation 与 class 在使用上是被分离的）。使用这个 meta-Annotation 可以对 Annotation 的 “生命周期” 限制。
2. 表示需要在什么级别保存该注释信息，用于描述注解的生命周期

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）**可以通过反射获取**



## @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。Documented 是一个标记注解，没有成员。

## @Inherited

1. @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**。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。
2. 注意：@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. 当@Inherited annotation 类型标注的 annotation 的 Retention 是

RetentionPolicy.RUNTIME，则反射 API 增强了这种继承性。如果我们使用 java.lang.reflect 去查询一个 @Inherited annotation 类型的 annotation 时，反射代码检查将展开工作：检查 class 和其父类，直到发现指定的 annotation 类型被发现，或者到达类继承结构的顶层。

# 自定义注解

## 定义规则

1. 使用 @interface 自定义注解时，自动继承了 java.lang.annotation.Annotation 接口，由编译程序自动完成其他细节。**在定义注解时，不能继承其他的注解或接口**。
2. @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。

## 格式

1. 定义注解格式：

　　public @interface 注解名 {定义体}

1. 注解参数的可支持数据类型：

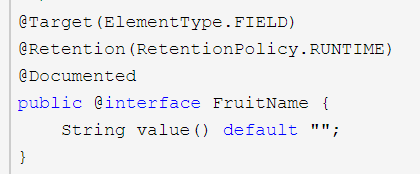
　　所有基本数据类型（int,float,boolean,byte,double,char,long,short)

　　String 类型、Class 类型、enum 类型、Annotation 类型

以上所有类型的数组

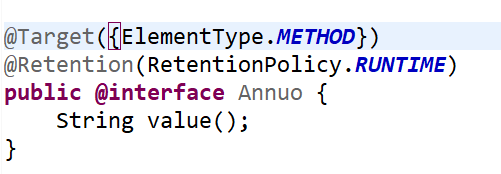
## 注解中参数的设定

1. **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 defaul 默认类型；
2. 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. 例如, String value(); 这里的参数成员就为 String;
3. 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.
4. 注解元素必须有确定的值，要么在定义注解的默认值中指定，要么在使用注解时指定，非基本类型的注解元素的值不可为 null。因此, 使用空字符串或 0 作为默认值是一种常用的做法。这个约束使得处理器很难表现一个元素的存在或缺失的状态，因为每个注解的声明中，所有元素都存在，并且都具有相应的值，为了绕开这个约束，**我们只能定义一些特殊的值，例如空字符串或者负数，一次表示某个元素不存在**，在定义注解时，这已经成为一个习惯用法。

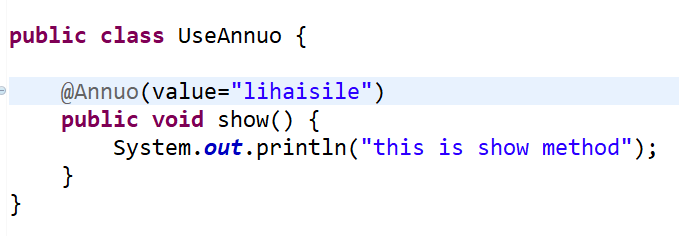


# 注解的使用

## 定义注解



## 使用注解



## 获取注解的值

