**JAVA基础**

# 基本注解

**Annotation时要在其前面增加@符号，并把该Annotation当成一个修饰符使用，用于修饰它支持的程序元素**

* **三个基本的Annotation**

**1.@Override**

[**2.@Deprecated**](mailto:2.@Deprecated)

**@Deprecated用于表示某个程序元素(类、方法等)已过时，当其他程序使用已过时的类、方法时，编译器将会给出警告**

[3.@SuppressWarnings](mailto:3.@SuppressWarnings)

**被Annotation标识的程序元素(包括所有子元素)取消显示指定的编译器警告。**

|  |
| --- |
| // @Override代表重写的注解  @Override  **public** String toString() {  **return** "Student [name=" + name + "]";  } |
| @SuppressWarnings("deprecation")  **public** **void** test() {    Date date = **new** Date();  date.~~getDate~~();    } |

# 元Annotation

@Retention只能用于修饰一个Annotation定义

用于指定Annotation保留多长时间

@Retention包含一个RetentionPolicy类型的value成员变量，所以使用@Retention必须为该value成员变量指定值

@Retention的value成员变量值

1.RetentionPolicy.CLASS:编译器将把注解记录在class文件中。当运行Java程序时，JVM不再保留注解。这是默认值

2.RetentionPolicy.RUNTIME:编译器将把注解记录在class文件中。当运行Java程序时，JVM也会保留注解，程序可以通过反射获取该注解

3.RetentionPolicy.SOURCE:编译器直接丢弃这种策略的注解

* 使用@Target

@Target也是用于修饰一个Annotation定义，它用于指定被修饰的Annotation能用于修饰哪些程序元素

* @Target Annotation的value的成员变量取值:

1.ElementType.ANNOTATION\_TYPE:指定该策略的Annotation只能修饰Annotation

2.ElementType.CONSTRUCTOR:指定该策略的Annotation只能修饰构造器

3.ElementType.FIELD:指定该策略的Annotation只能修饰成员变量

4.ElementType.LOCAL\_VARIABLE:指定该策略的Annotation只能修饰局部变量 5.ElementType.METHOD:指定该策略的Annotation只能修饰方法定义

6.ElementType.PACKAGE:指定该策略的Annotation只能修饰包定义

7.ElementType.PARAMETER:指定该策略的Annotation只能修饰参数

8.ElementType.TYPE:指定该策略的Annotation可以修饰类、接口(包括注解类型)或枚举定义

与使用@Retention类似，使用@Target也可以直接在括号里指定value值，可以无须使用name=value形式

* 使用@Documented

@Documented用于指定被该元Annotation修饰的Annotation类将被javadoc工具提取成文档

# 自定义Annotation

定义新的Annotation类型使用@interface关键字(在原有的interface关键字前加上@符号)，它用于定义新的Annotation类型。

|  |
| --- |
| /\*\*  \* **@Retention定义保留注解的级别**  \* RetentionPolicy.SOURCE在源文件中可见  \* RetentionPolicy.CLASS在class文件中可见，默认值  \* RetentionPolicy.RUNTIME在运行时可见，可以通过反射获取  \* **@author** tzhang  \*  \* ElementType.METHOD表示当前注解只能用来修饰方法  \* ElementType.FIELD表示当前注解只能用来修饰属性  \*/  //@Target(ElementType.FIELD)  @Retention(RetentionPolicy.***RUNTIME***)  @Target({ElementType.***FIELD***, ElementType.***METHOD***})  @Documented  @Inherited  **public** **@interface** AnnoRetention {  String value();  **int** age() **default** 0;  } |
| @AnnoRetention("value=abc")  **public** **void** test1() {    } |
| @AnnoRetention(value="张三",age=10)  **private** String name; |

## 获取注解值

|  |
| --- |
| **import** java.lang.annotation.Annotation;  **import** java.lang.reflect.Method;  **public** **class** MainAnno {  **public** **static** **void** main(String[] args) **throws** ClassNotFoundException, NoSuchMethodException, SecurityException {  // 私有属性，可以直接通过反射去修改私有属性的值  Class clz = Class.*forName*("com.cognizant.Student");  // 通过反射获取方法  Method method = clz.getMethod("test1");  // 获取方法注解  Annotation[] annos = method.getAnnotations();    System.***out***.println("===");  **for**(Annotation a:annos) {  AnnoRetention ar = (AnnoRetention) a;  // 注解属性值  System.***out***.println(ar.value());  }  }  } |