Java注解

在Java中注解（Annotation）是一个重要的知识点，其描述：

*Java 注解用于为 Java 代码提供元数据。作为元数据，注解不直接影响你的代码执行，但也有一些类型的注解实际上可以用于这一目的。Java 注解是从 Java5 开始添加到 Java 的。*

可以将Java注解理解成标签（对事物行为的某些角度的评价与解释）：如果将代码理解成事物，注解就是对于代码中某些鲜活个体的贴上去的标签。

在Java中，如同Class和Interface，注解也属于一种类型，Java SE 5.0版本中开始引入，其定义如下：

*@Documented*

*@Retention(RetentionPolicy.RUNTIME)*

*@Target(ElementType.ANNOTATION\_TYPE)*

*public @interface Documented {*

*}*

它的形式跟接口类似，但是多个一个@符号。

# 元注解

元注解是注解到注解上的注解，或者说元注解是一种基本注解，但是它能够应用到其他注解上面。有以下元注解：

* @Retention，保留期，当应用到一个注解上的时候，其解释说明这个注解的存活时间
* RetentionPolicy.SOURCE，在源码阶段保留，编译时被丢弃
* RetentionPolicy.CLASS，保留到在编译进行阶段，不会加载到JVM
* RetentionPolicy.RUNTIME，保留到程序运行时，加载到JVM中

示例：

*@Retention(RetentionPolicy.RUNTIME)*

*public @interface TestAnnotation { }*

在程序运行时被获取到。

* @Documented，将注解中的元素包含到Javadoc中
* @Target，指明注解运用的地方，限定使用场景，其取值：
* ElementType.ANNOTATION\_TYPE，给注解进行注解
* ElementType.CONSTRUCTOR，给构造方法进行注解
* ElemenType.FIELD，给属性进行注解
* ElementType.LOCAL\_VARIABLE，给局部变量进行注解
* ElementType.METHOD，给方法进行注解
* ElementType.PACKATE，给包进行注解
* ElementType.PARAMETER，给方法中的参数进行注解
* ElementType.TYPE，给类型中进行注解，比如类、接口和枚举
* @Inherited，继承的意思，并不是注解本身可以继承，而是说如果一个超类被@Inhertied注解过的注解进行注解，如果它的子类没被任何注解应用的话，那么这个子类就继承了超类的注解，使用示例：

*@Inherited*

*@Retention(RetentionPolicy.RUNTIME)*

*@interface Test {}*

*@Test*

*public class A {}*

*public class B extends A {}*

注解Test为@Inherited修饰，类A被Test注解，类B继承A时也拥有Test这个注解

* @Repeatable，可重复，通常是注解的值可以同时取多个
* @FunctionalInterface

Java 1.8版本中引入的函数式接口注解，其是一个具有一个方法的普通接口

*@FunctionalInterface*

*public interface Runnable {*

*/\*\**

*\* When an object implementing interface <code>Runnable</code> is used*

*\* to create a thread, starting the thread causes the object's*

*\* <code>run</code> method to be called in that separately executing*

*\* thread.*

*\* <p>*

*\* The general contract of the method <code>run</code> is that it may*

*\* take any action whatsoever.*

*\**

*\* @see java.lang.Thread#run()*

*\*/*

*public abstract void run();*

*}*

进行线程开发中常用的Runnable就是一个典型的函数式接口，被@FunctionalInterface注解，使用函数式接口注解，很容易转换成Lambda表达式。

# 注解属性

注解属性也叫做成员变量，注解只有成员变量，没有方法，注解的成员变量在注解定义中以无形参的方法来声明，其方法名定义了该成员变量的名字，其返回值定义该成员变量的类型：

*@Target(ElementType.TYPE)*

*@Retention(RetentionPolicy.RUNTIME)*

*public @interface TestAnnotation {*

*int id() default -1;*

*String msg() default “hi”;*

*}*

上面代码定义了TestAnnotation这个注解拥有id和msg两个属性，使用时进行赋值，如下：

*@TestAnnotation(id=3,msg="hello annotation")*

*public class Test { }*

# 注解的提取

通过反射获取注解，可以通过Class对象的isAnnotationPresent方法判断是否应用某个注解

*public boolean isAnnotationPresent(Class<? extends Annotation> annotationClass) {}*

然后通过getAnnotation方法获取Annotation对象

*public <A extends Annotation> A getAnnotation(Class<A> annotationClass) {}*

其使用示例如下：

*@TestAnnotation()*

*public class Test {*

*public static void main(String[] args) {*

*boolean hasAnnotation = Test.class.isAnnotationPresent(TestAnnotation.class);*

*if ( hasAnnotation ) {*

*TestAnnotation testAnnotation =*

*Test.class.getAnnotation(TestAnnotation.class);*

*System.out.println("id:"+testAnnotation.id());*

*System.out.println("msg:"+testAnnotation.msg());*

*}}}*

程序的运行结果：

*id:-1*

*msg:*

如果一个注解在运行时被成功提取，那么@Retention(RentionPolicy.RUNTIME)是必须的。

https://blog.csdn.net/briblue/article/details/73824058