注解(Annotation)简介

Annotation(注解)是 JDK5.0 及以后版本引入的一个特性。注解是 java 的一个新的类型（与接口很相似），它与类、接口、枚举是在同一个层次，它们都称作为 java 的一个类型（TYPE）。它可以声明在包、类、字段、方法、局部变量、方法参数等的前面，用来对这些元素进行说明，注释。它的作用非常的多，例如:进行编译检查、生成说明文档、代码分析等。

JDK 提供的几个基本注解

a. @SuppressWarnings
该注解的作用是阻止编译器发出某些警告信息。
它可以有以下参数:
deprecation：过时的类或方法警告。
unchecked：执行了未检查的转换时警告。
fallthrough：当 Switch 程序块直接通往下一种情况而没有 Break 时的警告。
path：在类路径、源文件路径等中有不存在的路径时的警告。
serial：当在可序列化的类上缺少 serialVersionUID 定义时的警告。
finally：任何 finally 子句不能完成时的警告。
all：关于以上所有情况的警告。

b. @Deprecated
该注解的作用是标记某个过时的类或方法。

c. @Override
该注解用在方法前面，用来标识该方法是重写父类的某个方法。

元注解

a. @Retention
它是被定义在一个注解类的前面，用来说明该注解的生命周期。
它有以下参数：
RetentionPolicy.SOURCE：指定注解只保留在一个源文件当中。
RetentionPolicy.CLASS：指定注解只保留在一个 class 文件中。
RetentionPolicy.RUNTIME：指定注解可以保留在程序运行期间。

b. @Target
它是被定义在一个注解类的前面，用来说明该注解可以被声明在哪些元素前。
它有以下参数：
ElementType.TYPE：说明该注解只能被声明在一个类前。
ElementType.FIELD：说明该注解只能被声明在一个类的字段前。
ElementType.METHOD：说明该注解只能被声明在一个类的方法前。
ElementType.PARAMETER：说明该注解只能被声明在一个方法参数前。

ElementType.CONSTRUCTOR：说明该注解只能声明在一个类的构造方法前。
ElementType.LOCAL_VARIABLE：说明该注解只能声明在一个局部变量前。
ElementType.ANNOTATION_TYPE：说明该注解只能声明在一个注解类型前。
ElementType.PACKAGE：说明该注解只能声明在一个包名前。

注解的生命周期

一个注解可以有三个生命周期，它默认的生命周期是保留在一个 CLASS 文件，
但它也可以由一个@Retetion 的元注解指定它的生命周期。

a. java 源文件
   当在一个注解类前定义了一个@Retetion(RetentionPolicy.SOURCE)的注解，那么说明该注解只保留在一个源文件当中，当编译器将源文件编译成 class 文件时，它不会将源文件中定义的注解保留在 class 文件中。
b. class 文件中
   当在一个注解类前定义了一个@Retetion(RetentionPolicy.CLASS)的注解，那么说明该注解只保留在一个 class 文件当中，当加载 class 文件到内存时，虚拟机会将注解去掉，从而在程序中不能访问。
c. 程序运行期间
   当在一个注解类前定义了一个@Retetion(RetentionPolicy.RUNTIME)的注解，那么说明该注解在程序运行期间都会存在内存当中。此时，我们可以通过反射来获得定义在某个类上的所有注解。

注解的定义

一个简单的注解：

```
public @interface Annotation01 {
    //定义公共的final静态属性
    .....
    //定以公共的抽象方法
    ......
}
```

a. 注解可以有哪些成员
   注解和接口相似，它只能定义 final 静态属性和公共抽象方法。
b. 注解的方法
   1.方法前默认会加上 public abstract
   2.在声明方法时可以定义方法的默认返回值。
   例如：

```
String color() default "blue";
String[] color() default {"blue", "red",......}
```

   3.方法的返回值可以有哪些类型
   8 种基本类型，String、Class、枚举、注解及这些类型的数组。
c. 使用注解（参照下面的注解使用）

注解的使用

注解的使用分为三个过程。

定义注解-->声明注解-->得到注解

a. 定义注解（参照上面的注解定义）

b. 声明注解

1. 在哪些元素上声明注解

如果定义注解时没有指定@Target 元注解来限制它的使用范围，那么该注解可以使用在 ElementType 枚举指定的任何一个元素前。否则，只能声明在@Target 元注解指定的元素前。

一般形式：

@注解名()

2. 对注解的方法的返回值进行赋值

对于注解中定义的每一个没有默认返回值的方法，在声明注解时必须对它的每一个方法的返回值进行赋值。

一般形式:

@注解名(方法名=方法返回值，、、、、、)

如果方法返回的是一个数组时，那么将方法返回值写在{}符号里

@注解名(方法名={返回值 1，返回值 2，、、、、、}，、、、、、、)

3. 对于只含有 value 方法的注解，在声明注解时可以只写返回值。

c. 得到注解

对于生命周期为运行期间的注解，都可以通过反射获得该元素上的注解实例。

1、声明在一个类中的注解

可以通过该类 Class 对象的 getAnnotation 或 getAnnotations 方法获得。

2、声明在一个字段中的注解

通过 Field 对象的 getAnnotation 或 getAnnotations 方法获得

3、声明在一个方法中的注解

通过 Method 对象的 getAnnotation 或 getAnnotations 方法获得

总结

注解可以看成是一个接口，注解实例就是一个实现了该接口的动态代理类。

注解大多是用做对某个类、方法、字段进行说明，标识的。以便在程序运行期间我们通过反射获得该字段或方法的注解的实例，来决定该做些什么处理或不该进行什么处理。