1. 元注解

@Target 没有@Target标明可以用于所有项上

@Retention RUNTIME才能通过反射获得他们，

@Document

@Inherited 指明当这个注解应用于一个类的时候，能够自动被它的子类继承

@Repeatable 指明这个注解可以在同一个项上应用多次

1. 所有注解都默认扩展自java.lang.annotation.Annotation接口
2. JDK提供的注解

@Deprecated 全部 标记为过时

@SupperssWarnings 除了包和注解外 组织某个给定类型的警告信息

@SafeVarargs 方法和构造器 断言varargs参数可安全使用

@Override 方法 检查该方法是否覆盖了某一个超类方法

@FunctionalInterface 接口 将接口标记为只有一个抽象方法的函数式接口

@PostConstruct 方法 被标记的方法应该在被构造之后立即被调用

@PreDestroy 方法 被标记的方法应该被移除之前立即被调用

@Resource 类、接口、方法、域 在类或接口上：标记为在其他地方要用到的资源。

在方法或域上：为“注入”而标记（类似di）

@Resources 类、接口 一个资源数组

@Generated 全部 标注这段代码是自动生成的

1. 适用于Package、Class、Constructor、Method、Field

//获取方法的(ActionListenerFor)注解  
ActionListenerFor actionListenerFor = method.getAnnotation(ActionListenerFor.class);

1. 适用于Package、Class、Constructor、Method、Field

方法是否有给定类型的注解

method.isAnnotationPresent(ActionListenerFor.class);

1. 适用于Package、Class、Constructor、Method、Field

获取某个可以重复注解类型的所有注解

method.getAnnotationsByType(ActionListenerFor.class);

1. 适用于Package、Class、Constructor、Method、Field

获得用于作用于该项的所有注解，包括继承来的注解。

method.getAnnotations();  
获得为该项声明的所有注解，不包含继承而来的注解。

method.getDeclaredAnnotations();

1. 获取该注解的类

Class annotationClass = actionListenerFor.annotationType();

//如果两个annotation是实现同一个接口,并且所有元素彼此相等,则返回true  
annotionTest.equals(annotionTest);

1. 注解的默认值是动态添加的，也就是说，即使你改变了注解接口的元素的默认值，之前已经创建的对象的注解的默认注也会改变。
2. 包注解只能出现在package-info.java中

例如：

@ActionListenerFor  
package com.test;

1. 注解接口的元素类型

//注解的所有类型  
//1.所有基本类型  
//2.String  
//3.Class可以是Class<? extends MyClass>  
//4.enum类型  
//5.注解类型  
//6.前面类型组成的数组  
//注解的参数  
//如果名为value则在使用时不用指定参数名  
//注解元素的值不能为null,你必须使用Void.class或者""代替  
enum Status {*UNCONFIRMED*, *CONFIRMED*, *FIXED*, *NOTABUG*};  
boolean showStopper() default false;  
String value() default "[none]";  
Class<?> testCase() default Void.class;  
Status status() default Status.*UNCONFIRMED*;  
Reference ref() default @Reference;  
String[] reportedBy() default {};