# 注解

## 概念

　Java提供了一种原程序中的元素关联任何信息和任何元数据的途径和方法。

[j](http://lib.csdn.net/base/java)ava注解是附加在代码中的一些元信息，用于一些工具在编译、运行时进行解析和使用，起到说明、配置的功能。  
注解不会也不能影响代码的实际逻辑，仅仅起到辅助性的作用。包含在 java.lang.annotation 包中

java自定义注解和运行时靠反射获取注解。

## Annotation的工作原理：

JDK5.0中提供了注解的功能，允许开发者定义和使用自己的注解类型。该功能由一个定义注解类型的语法和描述一个注解声明的语法，读取注解的API，一个使用注解修饰的class文件和一个注解处理工具组成。

Annotation并不直接影响代码的语义，但是他可以被看做是程序的工具或者类库。它会反过来对正在运行的程序语义有所影响。

Annotation可以冲源文件、class文件或者在运行时通过反射机制多种方式被读取。

# 分类

## 按运行机制

按运行机制（注解存在于程序的那个阶段）将注解分为三类：源码注解(只在源码存在)、编译注解(在class文件中也存在)、运行时注解(在运行阶段仍然起作用)



## 按来源

### Jdk自带注解

1：JDK自带的注解（Java目前只内置了三种标准注解：@Override、@Deprecated、@SuppressWarnings，以及四种元注解：@Target、@Retention、@Documented、@Inherited）

### 第三方注解

2：第三方的注解——这一类注解是我们接触最多和作用最大的一类



### 自定义注解

3：自定义注解——也可以看作是我们编写的注解，其他的都是他人编写注解