一、注解定义

* Java提供了一种源程序中的元素关联任何信息和任何元数据的途径和方法

二、JDK内置系统注解

* @Override ：用于修饰此方法覆盖了父类的方法;

@Deprecated ：用于修饰已经过时的方法;

@suppressWarings("deprecation") ：用于通知java编译器忽略特定的编译警告。

三、常见第三方注解

* Spring
* @Autowired:可以对成员变量、方法和构造函数进行标注，来完成自动装配的工作。通过 @Autowired的使用来消除 set ，get方法。
* @Service：用于标注业务层组件
* @Repository:用于标注数据访问组件，即DAO组件
* Mybatis(@InsertProvider @UpdateProvider @Options)

四、注解的分类

元注解：注解的注解

* 按照运行机制分：

1）源码注解（注解只在源码中存在，编译成.class文件就不存在了）；

2）编译时注解（注解在源码和.class文件中都存在）；

3）运行时注解（在运行阶段还起作用，甚至影响运行逻辑的注解 ）

* 按照来源分：

1）来自JDK的注解；

2）来自第三方的注解；

3）自定义的注解

五、自定义注解----是一个注解类

1、定义：

1. 使用@interface关键字定义注解 public @interface Description {
2. 成员以无参无异常方式声明 String desc();
3. 可以用default为成员指定一个默认值 int age() default 18;}
4. 成员类型是受限的，合法的类型包括java的基本类型及String,Class,Annotation,Enumeration
5. 如果注解只有一个成员，则成员名必须取名为value()，在使用时可以忽略成员名和赋值号（=）
6. 注解类可以没有成员，没有成员的注解称为标识注解
7. 元注解（对注解的注解）@Target（定义注解的作用域范围）@Retention（定义注解的生命周期）@Inherited（子类可以继承父类的注解）@Document（生成doc的时候会生成注解信息）

如图：



2、使用注解：



3、解析注解

1. 解析注解定义：通过反射获取类、函数或成员上的运行时注解信息，从而实现动态控制程序运行的逻辑。
2. 注解的继承：子类只会继承作用域为类的注解，作用于方法上的注解不会被继承。
3. @Inherited 只能实现类的继承，而接口是无法继承的。即接口的注解无法影响到实现接口的类上面。另外，父类的方法的注解也无法被子类继承。
4. 常用方法：

获取类上的注解信息（获取方法上的注解信息与之类似）

Class c = Class.forName(com.imooc.Child);

c.isAnnotationPresent(Description.class);//判断注解是否存在

Column co =(Column) c.getAnnotation(Description.class);//获取注解实例 ，co.value()可以获取值

4、应用注解步骤：

1. 自定义注解
2. 将注解应用于类或者方法；
3. 解析注解中的值
4. 注解可以帮助变量拥有两个名称，一个是注解中的名字，一个是本身拥有的名字。