**注解**

## 注解对比XML

* 注解：便捷、易于维护修改、耦合度高
* XML：非便捷、不易于维护修改、耦合度低

## 注解的本质

注解的本质就是一个继承了 Annotation 接口的接口，通过反编译一个注解类可以看到结果。

解析一个类或一个注解有两种形式：

* 编译器直接扫描
* 运行期反射

## 元注解

* @Target：注解的作用目标(类/属性/方法)-ElementType.METHOD

ElementType.TYPE：允许被修饰的注解作用在类、接口和枚举上

ElementType.FIELD：允许作用在属性字段上

ElementType.METHOD：允许作用在方法上

ElementType.PARAMETER：允许作用在方法参数上

ElementType.CONSTRUCTOR：允许作用在构造器上

ElementType.LOCAL\_VARIABLE：允许作用在本地局部变量上

ElementType.ANNOTATION\_TYPE：允许作用在注解上

ElementType.PACKAGE：允许作用在包上

* @Retention：注解的生命周期-RetentionPolicy.RUNTIME

RetentionPolicy.SOURCE：当前注解编译期可见，不会写入 class 文件

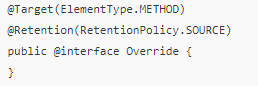
RetentionPolicy.CLASS：类加载阶段丢弃，会写入 class 文件

RetentionPolicy.RUNTIME：永久保存，可以反射获取

* @Documented：注解是否应当被包含在 JavaDoc 文档中
* @Inherited：是否允许子类继承该注解

## JAVA内置三大注解：

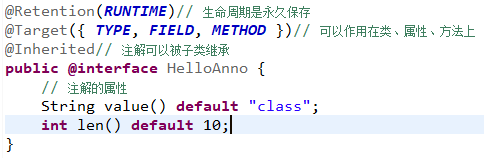
* @Override=标记重写方法，它只能作用于方法之上，编译结束后将被丢弃



* @Deprecated=标记已过期的(不推荐使用的)方法
* @SuppressWarnings=不输出警告信息

## 注解的定义

定义比较简单，在自定义注解上标记元注解，定义好注解的属性即可。



## 注解的解析

**对于一个类或者接口来说，Class 类中提供了以下一些方法用于反射注解**

 getAnnotation：返回指定的注解

 isAnnotationPresent：判定当前元素是否被指定注解修饰

 getAnnotations：返回所有的注解

 getDeclaredAnnotation：返回本元素的指定注解

 getDeclaredAnnotations：返回本元素的所有注解，不包含父类继承而来的

**通过反射来解析：**



## 使用范例

在WEB开发中，需要记录某个修改功能修改的数据前后的值。实现思路是在controller中的update方法加上自定义注解，被这个注解标记的方法将记录前后值的变化。通过AOP切面监控这个方法，一旦调用就将新旧值记录日志。

代码见quick—BusinessLog—LogAop—LogObjectHandler

## Spring对注解的使用