[**java注解**](http://www.cnblogs.com/Gordon-YangYiBao/archive/2012/08/07/2626340.html)

java中元注解有四个： @Retention @Target @Document @Inherited；

　　 @Retention：注解的保留位置

　　　　　　@Retention(RetentionPolicy.SOURCE)   //注解仅存在于源码中，在class字节码文件中不包含

　　　　　　@Retention(RetentionPolicy.CLASS)     // 默认的保留策略，注解会在class字节码文件中存在，但运行时无法获得，

　　　　　　@Retention(RetentionPolicy.RUNTIME)  // 注解会在class字节码文件中存在，在运行时可以通过反射获取到

　　@Target:注解的作用目标

　　　　　　　　@Target(ElementType.TYPE)   //接口、类、枚举、注解

　　　　　　　　@Target(ElementType.FIELD) //字段、枚举的常量

　　　　　　　　@Target(ElementType.METHOD) //方法

　　　　　　　　@Target(ElementType.PARAMETER) //方法参数

　　　　　　　　@Target(ElementType.CONSTRUCTOR)  //构造函数

　　　　　　　　@Target(ElementType.LOCAL\_VARIABLE)//局部变量

　　　　　　　　@Target(ElementType.ANNOTATION\_TYPE)//注解

　　　　　　　　@Target(ElementType.PACKAGE) ///包

     @Document：说明该注解将被包含在javadoc中

　  @Inherited：说明子类可以继承父类中的该注解

举例：

　　      @Retention(RetentionPolicy.RUNTIME)  
　　　　@Target({ElementType.METHOD})  
　　　　public @interface AnnatDemo{  
　　　　　　　 public int value();  
　　　　}

以上代码定义了@AnnatDemo注解，作用目标是 用于对方法注解，并且保留在运行时的环境中，我们可以利用反射 获得一个方法上的注解  调用定义的方法,

比如@AnnatDemo 作用于以下方法：

public interface IClientProtocolEx extends IProtocol {

　　int METHOD\_START=0;

　　@AnnatDemo(METHOD\_START)  
　　 public String say(String person);

}

那么可以利用以下代码进行反射：

        Class ipt=IClientProtocalEx.class;

　　　Method[] mts=ipt.getMethod();

         for(Method mt:mts)

　　　{

　　　　AnnatDemo ad=mt.getAnnotation(AnnatDemo.class);//如果方法上  没有该注解  则返回null

           int value=ad.value();

　　     System.out.println("value:"+value);

　　　}

注解是用于建设基础jar包的一部分   项目都有自己的框架，若运用恰当，注解则为其中良好的一部分！