注解的作用：是帮助编译器检查错误，使其可更早的发现问题，解决问题

第一步：

写一个注解类

@Target(ElementType.FIELD) // 注解的类型
@Retention(RetentionPolicy.RUNTIME) // 注解保留位置
public @interface InjectView {
//id就表示哪些控件，-1就表示取不到时候的默认值 （此处的书写有很多问题，应该多注意）
int id() default -1;
}

第二步：

写一个注解解析器

public class InjectViewParser {
public static void inject(Object object) {
try {
parse(object);
} catch (Exception e) {
e.printStackTrace();
}
}
public static void parse(Object object) throws Exception {
Class<?> clazz = object.getClass(); // 反射的方式拿到相应的类
View view = null;
Field[] fields = clazz.getDeclaredFields(); // 获取相应类型
for (Field field : fields) {
if (field.isAnnotationPresent(InjectView.class)) {
InjectView injectView = field.getAnnotation(InjectView.class);//获取注解类
int id = injectView.id();// 获取相应的值
if (id < 0) {
throw new Exception("id must not be null");
} else {
field.setAccessible(true); // 设置权限（为false，表示不能设置相应的属性）
**if (object instanceof View) {
view = ((View) object).findViewById(id);
} else if (object instanceof Activity) {
view = ((Activity) object).findViewById(id);
}**
field.set(object, view); // 设置相应的属性
}
}

}
}
}

第三步：

使用注解

@InjectView(id = 相应控件的id)

private Class class; // 未相应的控件设置相应的名

InjectViewParser.parse(object);// 在oncreate()方法中使用注解器

以上步骤就完成了注解的创建与使用