### 源码分析相关面试题

* [Volley源码分析](http://www.jianshu.com/p/ec3dc92df581)
* [注解框架实现原理](http://www.jianshu.com/p/20da6d6389e1)
* [okhttp3.0源码分析](http://www.jianshu.com/p/9ed2c2f2a52c)
* [onSaveInstanceState源码分析](http://www.jianshu.com/p/cbf9c3557d64)
* [静默安装和源码编译](http://www.jianshu.com/p/2211a5b3c37f)

### Activity相关面试题

* [保存Activity的状态](http://www.jianshu.com/p/cbf9c3557d64)

### 与XMPP相关面试题

* [XMPP协议优缺点](http://www.jianshu.com/p/2c04ac3c526a)
* [极光消息推送原理](http://www.jianshu.com/p/d88dc66908cf)

### 与性能优化相关面试题

* [内存泄漏和内存溢出区别](http://www.jianshu.com/p/5dd645b05c76)
* [UI优化和线程池实现原理](http://www.jianshu.com/p/c22398f8587f)
* [代码优化](http://www.jianshu.com/p/ebd41eab90df)
* [内存性能分析](http://www.jianshu.com/p/2665c31b9c2f)
* [内存泄漏检测](http://www.jianshu.com/p/1514c7804a06)
* [App启动优化](http://www.jianshu.com/p/f0f73fefdd43)
* [与IPC机制相关面试题](http://www.jianshu.com/p/de4793a4c2d0)

### 与登录相关面试题

* [oauth认证协议原理](http://www.jianshu.com/p/2a6ecbf8d49d)
* [token产生的意义](http://www.jianshu.com/p/9b7ce2d6c195)
* [微信扫一扫实现原理](http://www.jianshu.com/p/a9d1f21bd5e0)

### 与开发相关面试题

* [迭代开发的时候如何向前兼容新旧接口](http://www.jianshu.com/p/cbecadec98de)
* [手把手教你如何解决as jar包冲突](http://www.jianshu.com/p/30fdc391289c)
* [context的原理分析](http://www.jianshu.com/p/2706c13a1769)
* [解决ViewPager.setCurrentItem中间很多页面切换方案](http://www.jianshu.com/p/38ab6d856b56)

### 与人事相关面试题

* [人事面试宝典](http://www.jianshu.com/p/d61b553ff8c9)

### 本文配套视频：

* [配套视频](https://v.qq.com/x/page/l0397qgxmkc.html)

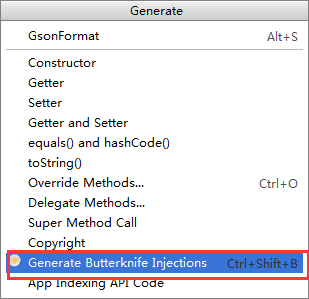
## 注解框架实现原理，手写ButterKnife实现自己的注解框架

初级程序员使用别人的框架，中级程序员不仅会使用别人的框架还知道内部的实现原理，高级程序员则按需编写自己的框架。添加该模块的目的就是想提交大家的逼格，让大家养成一个动手编写“自主知识产权”框架的意识。

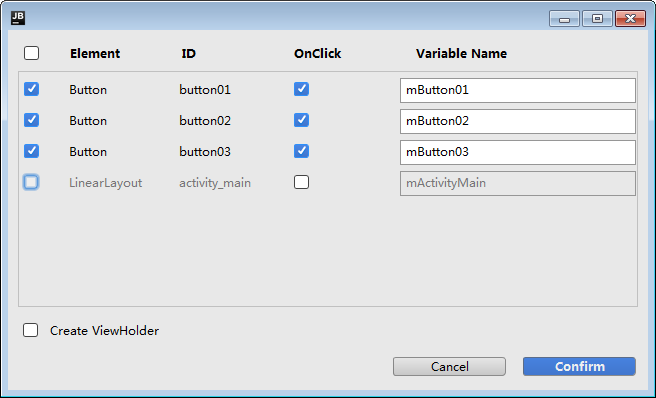
### 1. 编写 ButterKnife框架

业界比较出名的基于完全注解方式就可以进行 UI 绑定和事件绑定，无需 findViewById 和 setClickListener 等的IOC（Inverse Of Control 控制反转，就是将 UI 的初始化和事件绑定的“权利”交给框架来完成）框架有：

ButterKnife使用如下：



img



img

会出现如下代码：

@BindView(R.id.button01)  
 Button mButton01;  
 @BindView(R.id.button02)  
 Button mButton02;  
 @BindView(R.id.button03)  
 Button mButton03;  
  
 @butterknife.OnClick({R.id.button01, R.id.button02, R.id.button03})  
 public void onClick(View view) {  
 switch (view.getId()) {  
 case R.id.button01:  
 Toast.makeText(this, "butterknife-button01", Toast.LENGTH\_SHORT).show();  
 break;  
 case R.id.button02:  
 Toast.makeText(this, "butterknife-button02", Toast.LENGTH\_SHORT).show();  
 break;  
 case R.id.button03:  
 Toast.makeText(this, "butterknife-button03", Toast.LENGTH\_SHORT).show();  
 break;  
 }  
 }

点击每个按钮会弹出响应的Toast，如下：



img

这个就是ButterKnife的用法。

### 接下来，我们开始编写自己的框架实现 findViewById 和 setOnClickListener 功能。

1. 编写自定义注解类 ViewInject 和 Click;

ViewInject 注解类用于添加在 Filed 上。Click 注解类用于添加到 Method 上。

【文件】ViewInject.java

/\*\*  
\* @Retention 用于声明该注解生效的生命周期，有三个枚举值可以选择<br>  
 \* 1. RetentionPolicy.SOURCE 注释只保留在源码上面，编译成class的时候自动被编译器抹除  
 \* 2. RetentionPolicy.CLASS 注释只保留到字节码上面，VM加载字节码时自动抹除  
 \* 3. RetentionPolicy.RUNTIME 注释永久保留，可以被VM加载时加载到内存中  
 \* 注意：由于我们的目的是想在VM运行时对Filed上的该注解进行反射操作，因此Retention值必须设置为RUNTIME  
 \*  
 \* @Target 用于指定该注解可以声明在哪些成员上面，常见的值有FIELD和Method，  
 由于我们的当前注解类是想声明在Filed上面  
 \* 因此这里设置为ElementType.FIELD。  
 \* 注意：如果@Target值不设置，则默认可以添加到任何元素上，不推荐这么写。  
 \*  
 \* @interface 是声明注解类的组合关键字。  
 \*/  
  
@Target({java.lang.annotation.ElementType.FIELD})  
@Retention(RetentionPolicy.RUNTIME)  
public @interface ViewInject {  
 public abstract int value();  
}

【文件】Click.java

@Target({java.lang.annotation.ElementType.METHOD})  
@Retention(RetentionPolicy.RUNTIME)  
public @interface OnClick  
{  
 public abstract int[] value();  
}

编写核心方法 ViewUtilsTest.inject(Ativity);

public class ViewUtilsTest {  
 public static void inject(final Activity activity)  
 {  
 /\*\*  
 \* 通过字节码获取activity类中所有的字段，在获取Field的时候一定要使用  
 \* getDeclaredFields(),  
 \* 因为只有该方法才能获取到任何权限修饰的Filed，包括私有的。  
 \*/  
  
 Class clazz = activity.getClass();  
 Field[] declaredFields = clazz.getDeclaredFields();  
 //一个Activity中可能有多个Field，因此遍历。  
 for (int i = 0; i < declaredFields.length; i++) {  
 Field field = declaredFields[i];  
 //设置为可访问，暴力反射，就算是私有的也能访问到  
 field.setAccessible(true);  
 //获取到字段上面的注解对象  
 ViewInject annotation = (ViewInject)field.getAnnotation(ViewInject.class);  
 //一定对annotation是否等于null进行判断，因为并不是所有Filed上都有我们想要的注解  
 if (annotation == null)  
 {  
 continue;  
 }  
 //获取注解中的值  
 int id = annotation.value();  
 //获取控件  
 View view = activity.findViewById(id);  
  
 try  
 {  
 //将该控件设置给field对象  
 field.set(activity, view);  
 } catch (IllegalArgumentException e) {  
 e.printStackTrace();  
 } catch (IllegalAccessException e) {  
 e.printStackTrace();  
 }  
  
 }  
 //获取所有的方法（私有方法也可以获取到）  
 Method[] declaredMethods = clazz.getDeclaredMethods();  
 for (int i = 0; i < declaredMethods.length; i++) {  
 final Method method = declaredMethods[i];  
 //获取方法上面的注解  
 OnClick annotation = (OnClick)method.getAnnotation(OnClick.class);  
 if (annotation == null) {  
 //如果该方法上没有注解，循环下一个  
 continue;  
 }  
 //获取注解中的数据，因为可以给多个button绑定点击事件，因此定义注解类时使用的是int[]作为数据类型。  
 int[] value = annotation.value();  
 for (int j = 0; j < value.length; j++) {  
 int id = value[j];  
  
 final View button = activity.findViewById(id);  
  
 button.setOnClickListener(new View.OnClickListener()  
 {  
 public void onClick(View v)  
 {  
 try  
 {  
 //反射调用用户指定的方法  
 method.invoke(activity,button);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 });  
 }  
 }  
 }  
}

咱们自己的注解框架就实现好了，效果如下：



img

* 欢迎关注微信公众号,长期推荐技术文章和技术视频

微信公众号名称：Android干货程序员

img

img