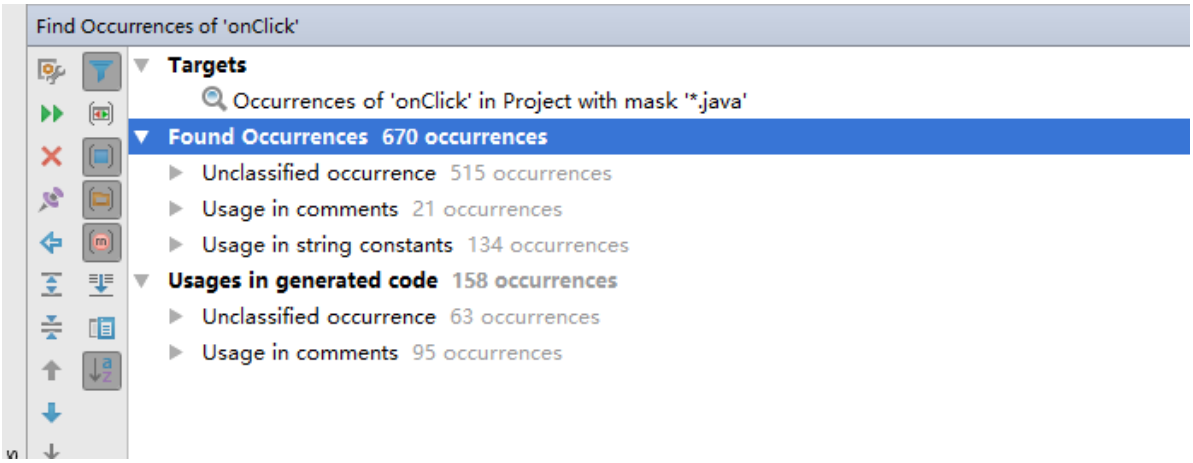
回想前几个月前，公司的项目在百度手机助手上线，在快速点击的时候会跳转两次Activity或者两个Dialog等等，为了能够顺利通过百度的测试，老大叫我将所有onClick全部要优化处理，避免用户快速多次点击，于是乎，我写了下面的代码

```
public abstract class NoDoubleClickListener implements View.OnClickListener {

    private int MIN_CLICK_DELAY_TIME = 1000;

    private long lastClickTime = 0;

    public abstract void onNoDoubleClick(View v);

    @Override
    public void onClick(View v) {
        long currentTime = Calendar.getInstance().getTimeInMillis();

        if (currentTime - lastClickTime > MIN_CLICK_DELAY_TIME) {
            lastClickTime = currentTime;
            onNoDoubleClick(v);
        }
    }
}
```

然后我打算这样来弄

```
  btn.setOnClickListener(new NoDoubleClickListener() {
            @Override
            public void onNoDoubleClick(View v) {
               //something
            }
 });
```

可是面临一个问题，有这么多，改到何年马月啊?



OK，这个是我以前碰到的一个好蛋疼的问题，再比如，在不侵入业务代码的情况下监听所有的点击事件并记录所有的点击数，用于统计热点页面和其他一些分析工作，你怎么办呢？现在介绍一个如何Hook掉View的onClick方法，相对与上一篇，这个很简单了。

一、第一步寻找Hook点：

去看setOnClickListener里面做了什么?

```
 btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
          }
   });
```

```
  /**
     * Register a callback to be invoked when this view is clicked. If this view
is not
     * clickable, it becomes clickable.
     *
     * @param l The callback that will run
     *
     * @see #setClickable(boolean)
     */
    public void setOnClickListener(@Nullable OnClickListener l) {
        if (!isClickable()) {
            setClickable(true);
        }
        getListenerInfo().mOnClickListener = l;
    }
```

```
 ListenerInfo getListenerInfo() {
        if (mListenerInfo != null) {
            return mListenerInfo;
        }
        mListenerInfo = new ListenerInfo();
        return mListenerInfo;
    }
```

看完了上面，就能猜到我们设置的Listener最终是被赋值给ListenerInfo的mOnClickListener成员了，ListenerInfo的实例可以说是信息的载体，那么很简单，只要把mOnClickListener替换掉，在ListenerInfo中还有mOnLongClickListener，mOnFocusChangeListener两个成员，分别对应了长按事件与焦点变化事件，所以处理长按事件与焦点变化事件与此类似。

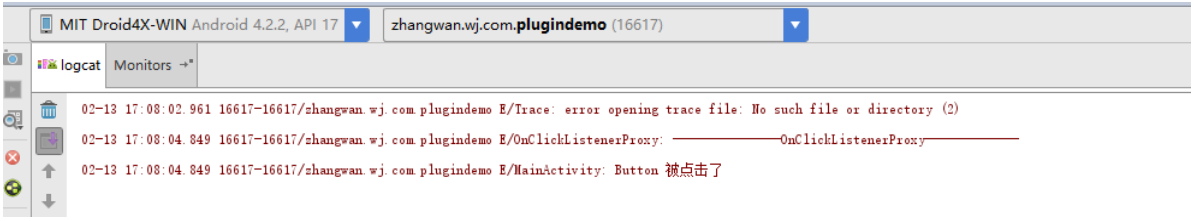
```
public class HookViewClickUtil {

    public static HookViewClickUtil getInstance() {
        return UtilHolder.mHookViewClickUtil;
    }

    private static class UtilHolder {
        private static HookViewClickUtil mHookViewClickUtil = new
HookViewClickUtil();
    }

    public static void hookView(View view) {
        try {
            Class viewClazz = Class.forName("android.view.View");
            //事件监听器都是这个实例保存的
```

```
            Method listenerInfoMethod =
viewClazz.getDeclaredMethod("getListenerInfo");
            if (!listenerInfoMethod.isAccessible()) {
                listenerInfoMethod.setAccessible(true);
            }
            Object listenerInfoObj = listenerInfoMethod.invoke(view);

            Class listenerInfoClazz =
Class.forName("android.view.View$ListenerInfo");

            Field onClickListenerField =
listenerInfoClazz.getDeclaredField("mOnClickListener");

            if (!onClickListenerField.isAccessible()) {
                onClickListenerField.setAccessible(true);
            }
            View.OnClickListener mOnClickListener = (View.OnClickListener)
onClickListenerField.get(listenerInfoObj);
            //自定义代理事件监听器
            View.OnClickListener onClickListenerProxy = new
OnClickListenerProxy(mOnClickListener);
            //更换
            onClickListenerField.set(listenerInfoObj, onClickListenerProxy);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }


    //自定义的代理事件监听器
    private static class OnClickListenerProxy implements View.OnClickListener {

        private View.OnClickListener object;

        private int MIN_CLICK_DELAY_TIME = 1000;

        private long lastClickTime = 0;

        private OnClickListenerProxy(View.OnClickListener object) {
            this.object = object;
        }

        @Override
        public void onClick(View v) {
            //点击时间控制
            long currentTime = Calendar.getInstance().getTimeInMillis();
            if (currentTime - lastClickTime > MIN_CLICK_DELAY_TIME) {
                lastClickTime = currentTime;
                Log.e("OnClickListenerProxy", "OnClickListenerProxy");
                if (object != null) object.onClick(v);
            }
        }
    }
}
```

使用起来也是非常简单，首先在MainActivity的View渲染完毕的时候进行注入，即在getWindow().getDecorView().post()中。

```
public class MainActivity extends Activity {

    private Button  btn;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final View btn = findViewById(R.id.btn);

        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Log.e("MainActivity","Button 被点击了");
            }
        });

        getWindow().getDecorView().post(new Runnable() {
            @Override
            public void run() {
                HookViewClickUtil.hookView(btn);
            }
        });
    }
}
```

执行结果：



OK，到此完成了，至于怎么获取页面的所有View,调用 HookViewClickUtil.hookView(view)，就不多说了。