QCon 全球软件开发大会
INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

BEIJING 2017

“无埋点”数据采集实践之路

网易乐得 / philon

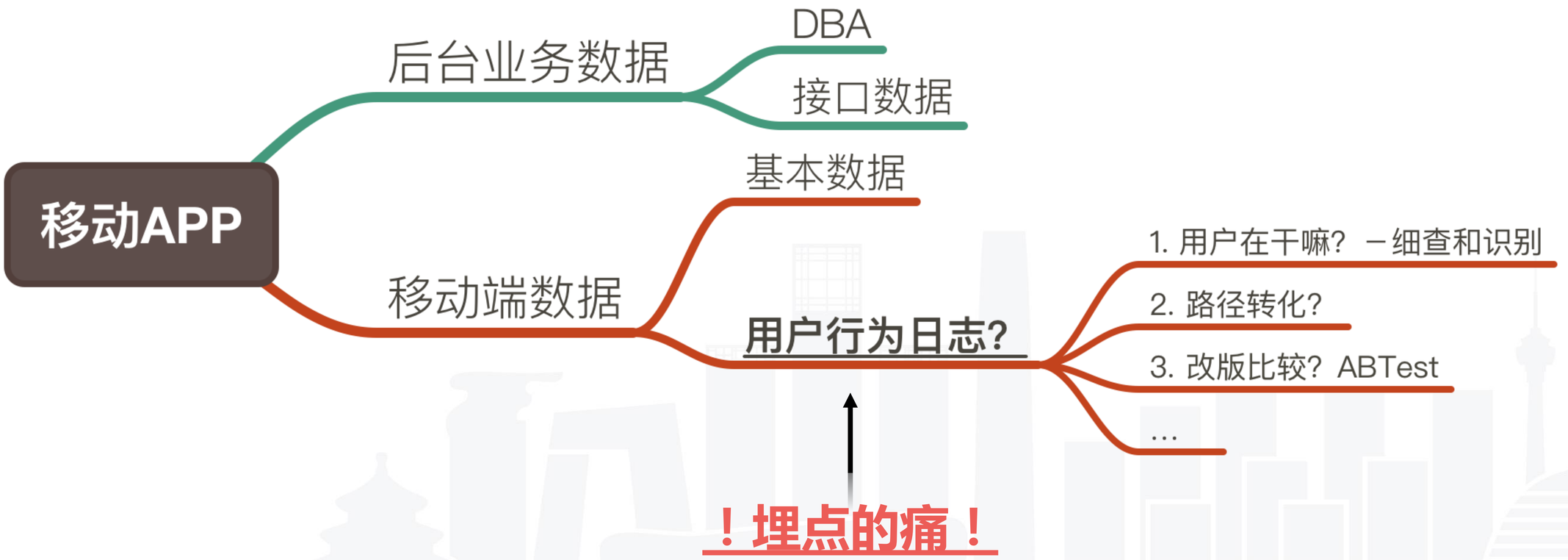
主办方 Geekbang. 极客邦科技 InfoQ

目录

- 一、埋点之“殇”

1.1 “殇”在哪？

- 流量红利时代过去，开始精细化运营



1.2 移动App的埋点之殇

session日志时代

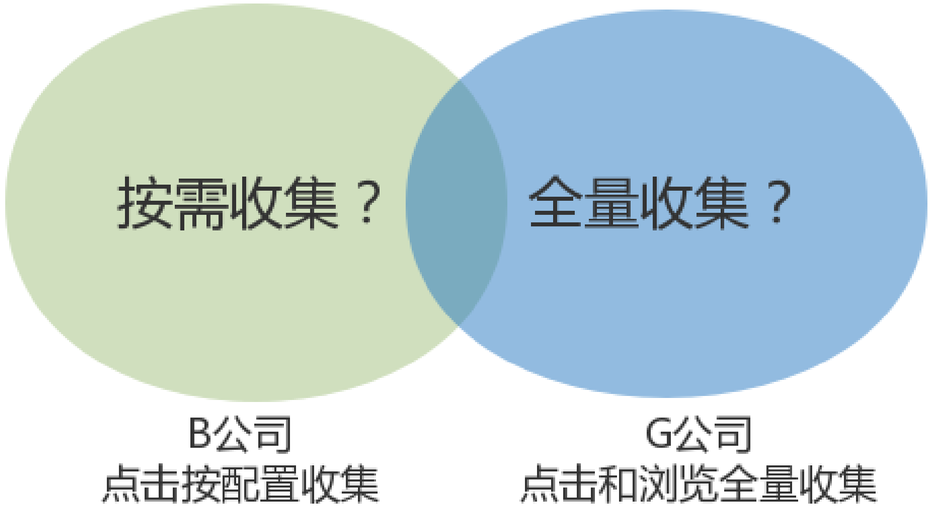
大数据时代

- 启动事件＋自动化页面事件＋手动埋点
- 缺陷：
  - 行为日志粒度较大
  - 手动埋点部分数据呈现周期太长
  - 无法动态收集业务数据

- 粒度更细：页面->点击->浏览量
- 如何做？？

· 二、收集策略的思考

2.1 基于页面点击的AOP全量收集

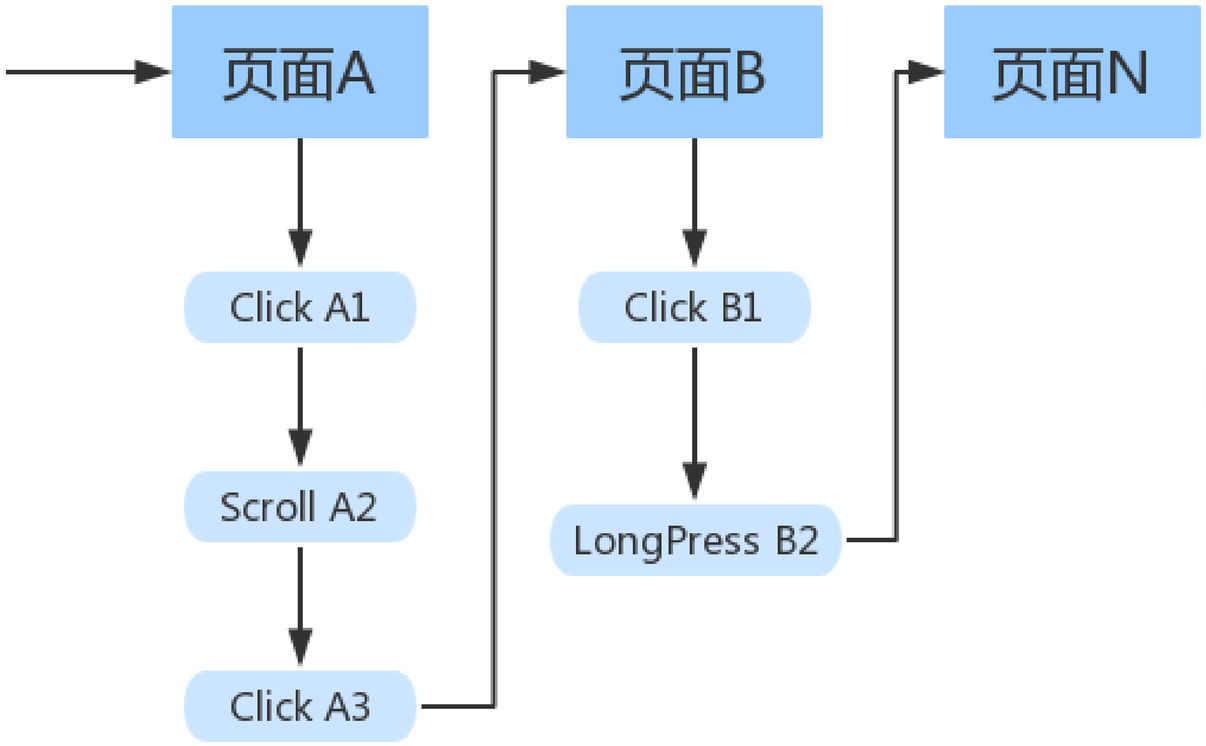


- 页面 + 点击！！
  - **页面事件和点击事件全量收集**
  - **列表浏览量按需配置收集**
- 实现：AOP

2.2 列表浏览量按需配置收集

- 浏览量收集原则：最小流量获得最大价值数据
  - **只关注列表元素浏览量**：行元素曝光量、停留时间
  - **只上传KVC配置的列表元素：**列表统计指标和内容结合分析
  - **只统计停留时间较长的行元素**：停止滑动时收集
- 实现：
  - iOS Hook：UIScrollView & UITableView
  - Android: AbsListView.OnScrollListener

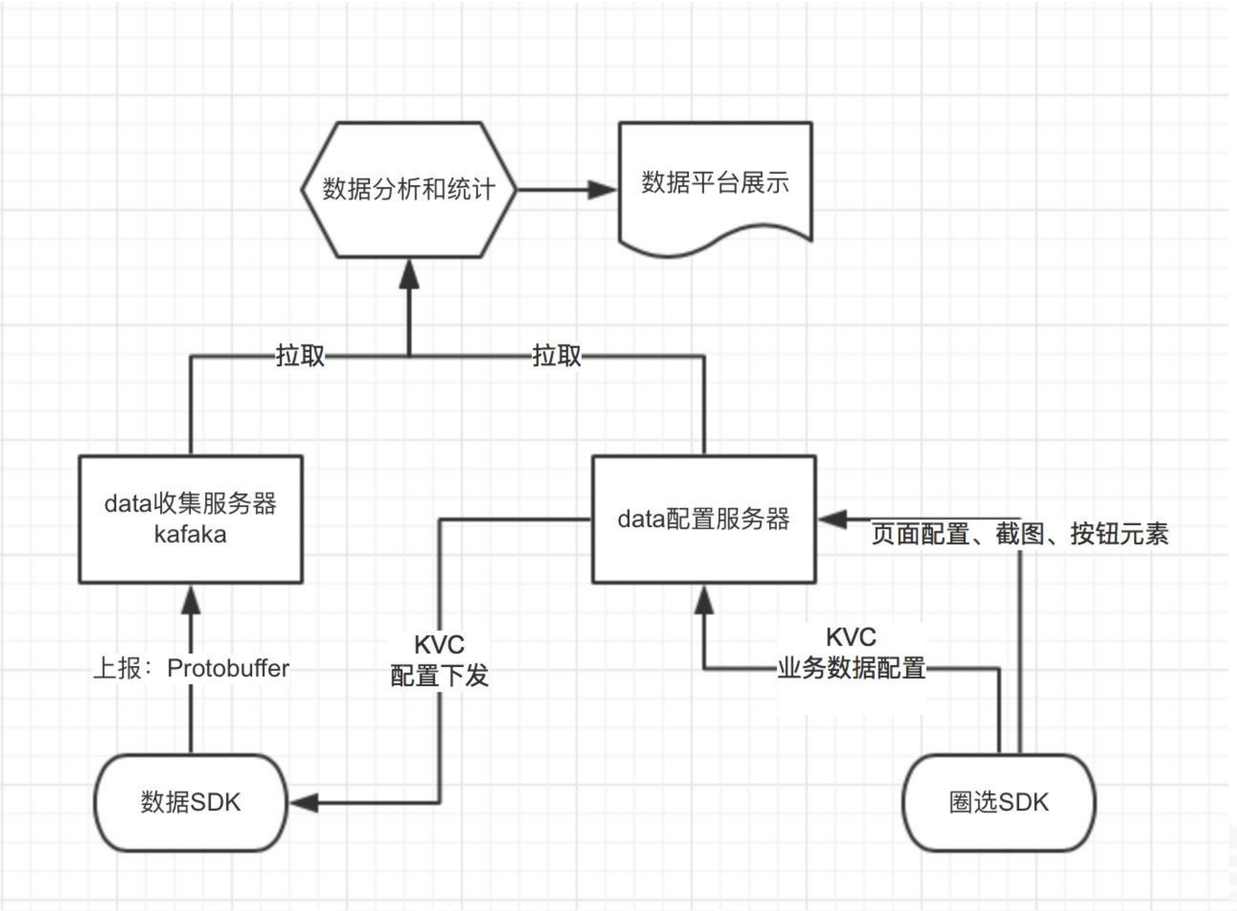
2.3 基于KVC的业务数据收集 - 1



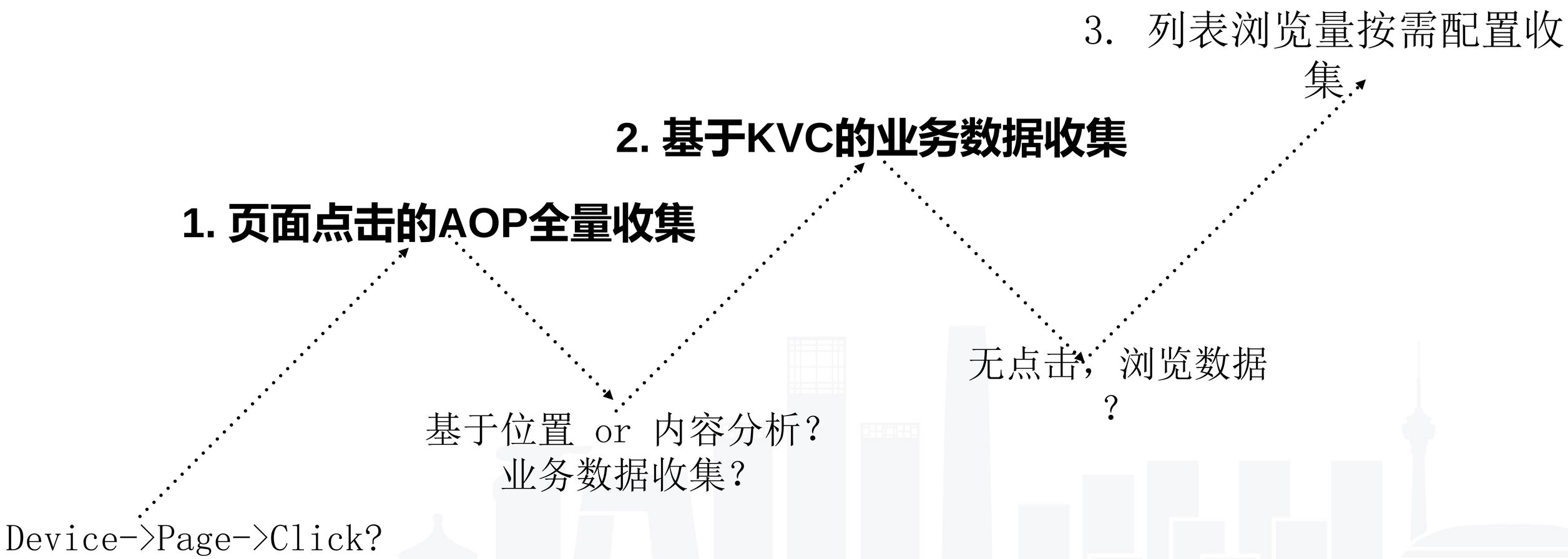
- 交互流 && 内容流
  - 交互内容化？
  - 基于元素位置&内容分析
- 实现：交互收集粘附内容数据
  - 自动获取：Button、Cell等 -> AOP
  - KVC配置获取 -> 反射

2.3 基于KVC的业务数据收集 - 2

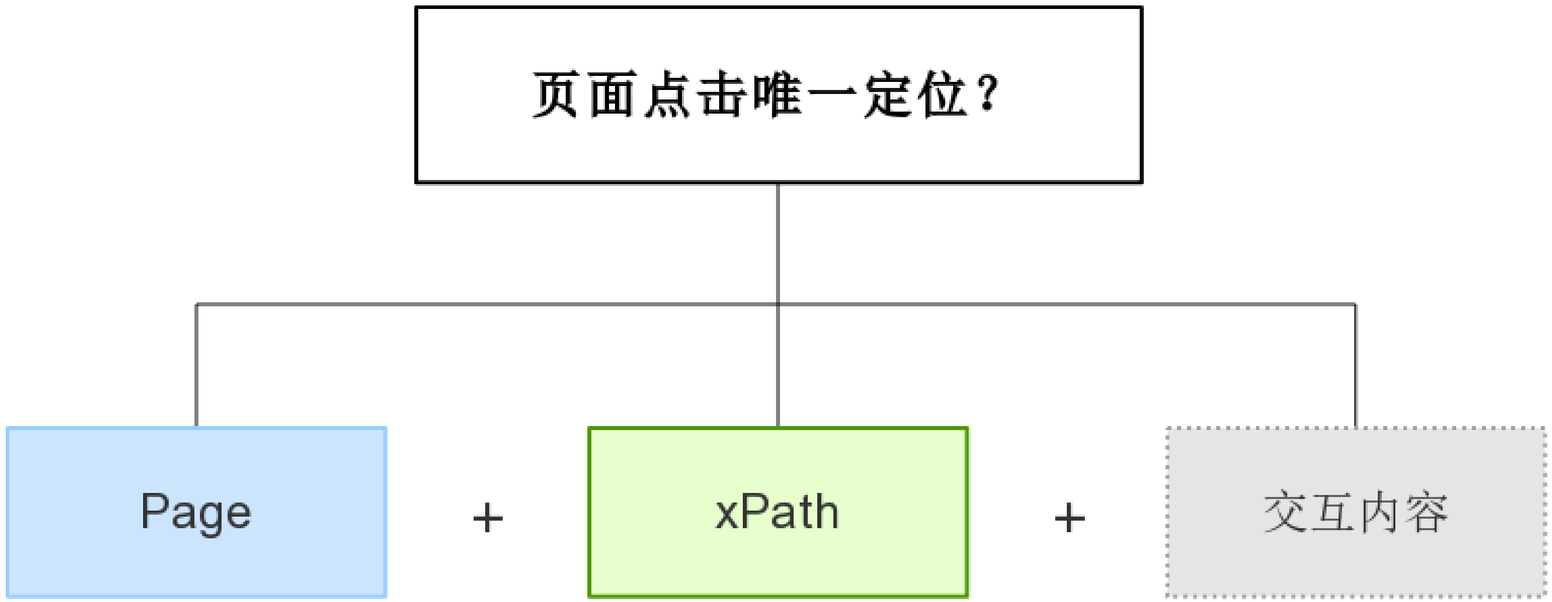
- 配置数据： xPath + 反射字符串
- 反射
  - iOS：KVC
    - [targetView valueForKeyPath:keyValuePath]
  - Android：自定义了一套反射机制
    - targetView.mParent.mParent.mAdapter.mAdapter.e[-1].mtag.productName



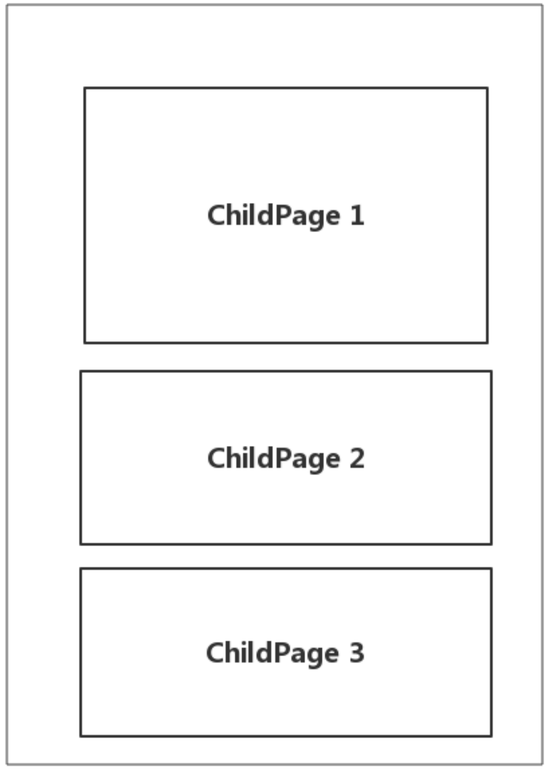
2.4 收集策略总结：三步曲



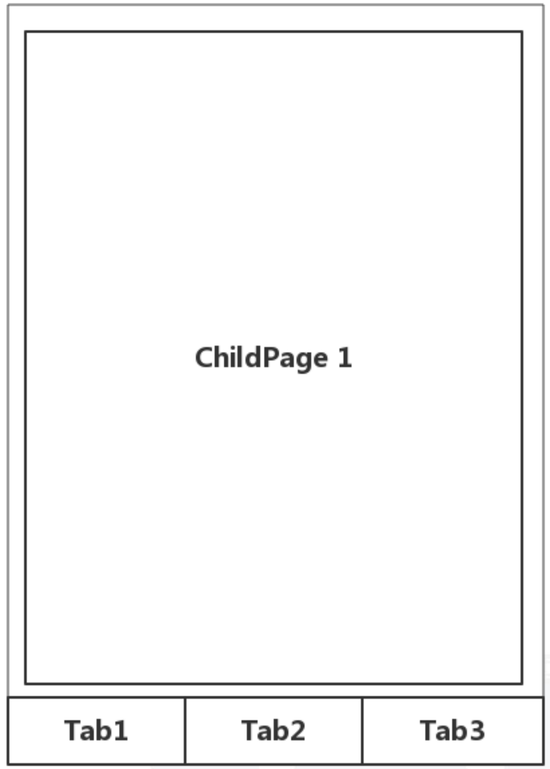
- 三、xPath相关


页面点击唯一定位？
Page
+
xPath
+
交互内容

3.1 Page定义1: 子页面处理



Page A

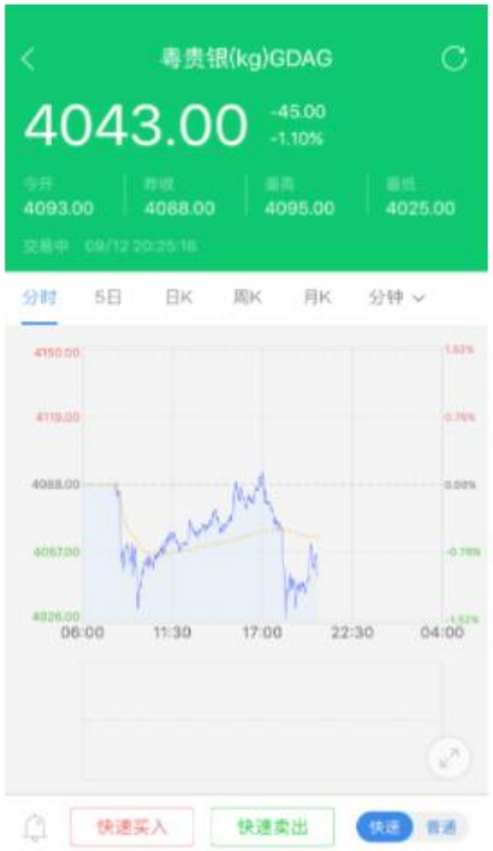


Page B

- Controller + Activity 类名定义是否足够
  - ChildController / Fragment 子页面 ？
    - 页面定义不统一
    - 子页面点击归属混乱
    - 页面路径分析受干扰
- 解决方案：
  - 上传页面级连关系和截图，手动定义配置中文名

3.1 Page定义2：页面别名

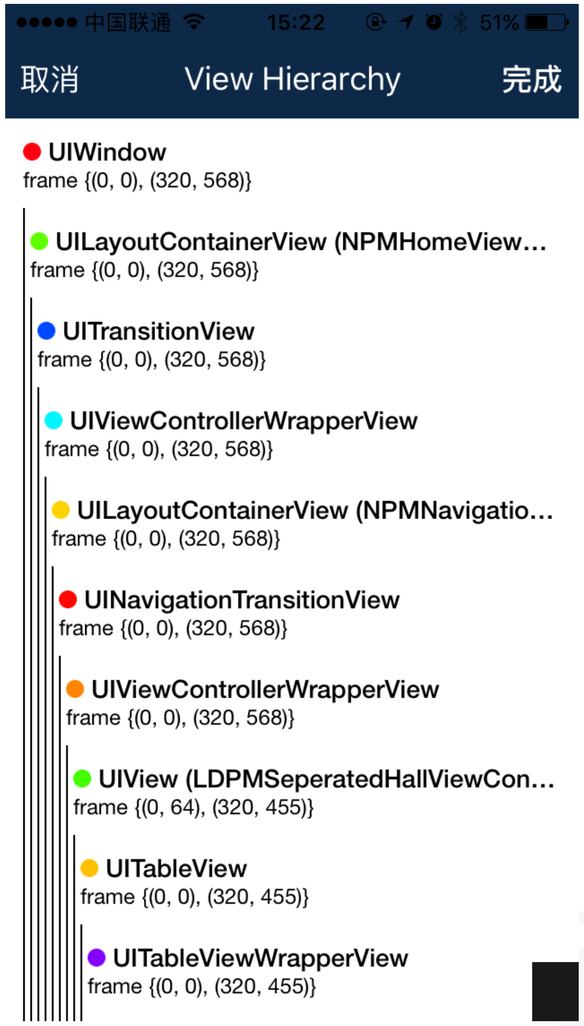
名称:白银详情页



名称:粤贵银详情页

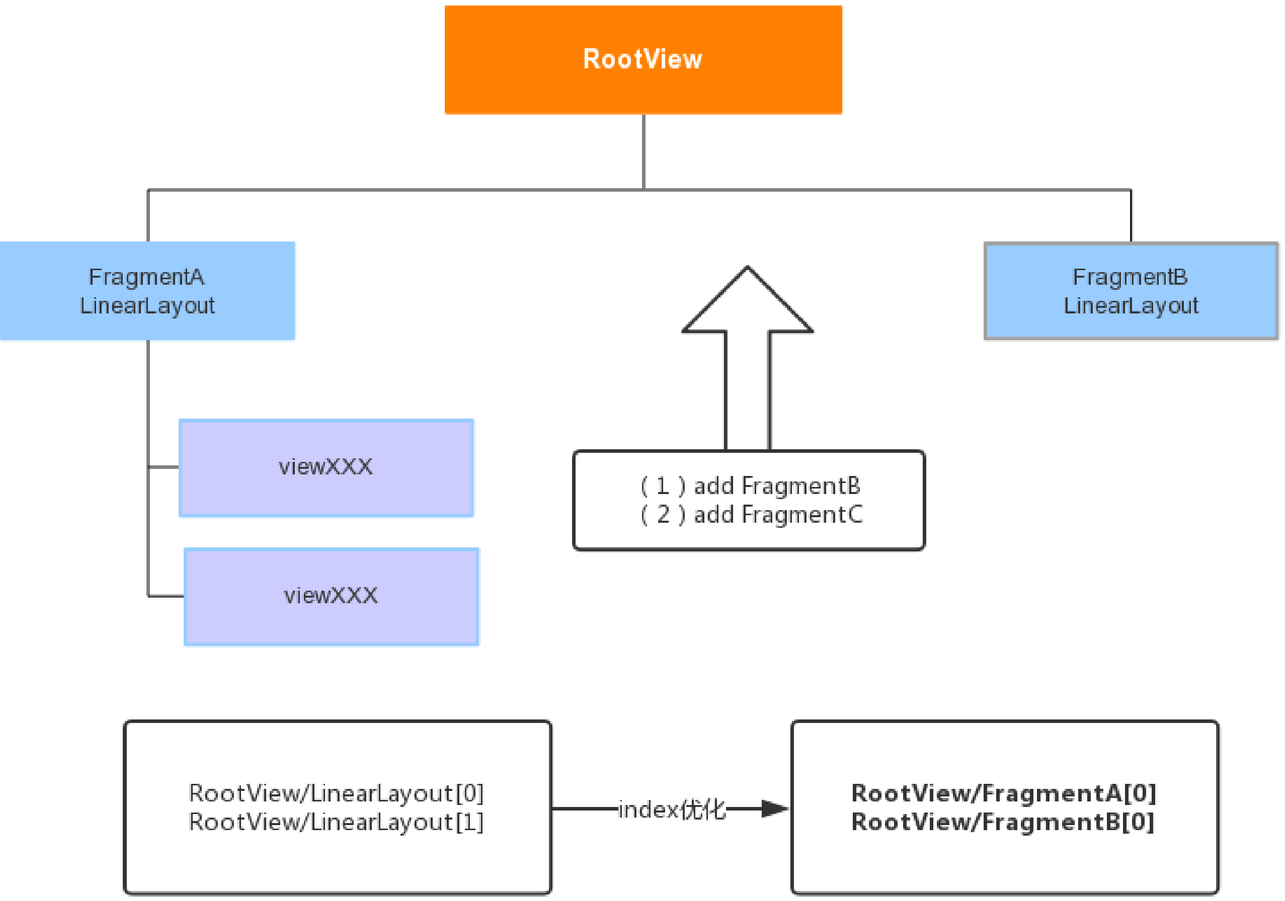
- Controller + Activity 类名定义是否足够？
  - Controller / Activity 不同页面复用？
    - 开发人员 代码复用
    - 精细化运营 就想分开看
- 解决方案：XXController/ XXActivity#别名 手动埋点
  - iOS: ViewController Category Property
  - Android: Activity or Fragment instance <-> aliasName [Map]

3.2 xPath定义



- 举例：
  - **JCZQBettingController**-UIScrollView-UITableView-UITableViewWrapperView-JCZQSPFCell-UITableViewCellContentView-UIImageView-UIButton&0-0-0-0-**0:0**-0-0-0
  - **DecorView**/LinearLayout[0]/FrameLayout[0]/RelativeLayout[0]/TabHost[0]/RelativeLayout[0]/FrameLayout[0]/DecorView[0]/LinearLayout[0]/FrameLayout[0]/LinearLayout[0]/CPRefreshableView[0]/ListView[0]/**LinearLayout[-1,1]**/LinearLayout[0]/LinearLayout[0]/FrameLayout[0]/LinearLayout[0]#**card_panel**
- xPath = viewPath+depthPath + (#id)
  - viewPath：点击view的**响应者链条**
  - depthPath：按class分类，取**同类元素的顺序Id + 列表处理**

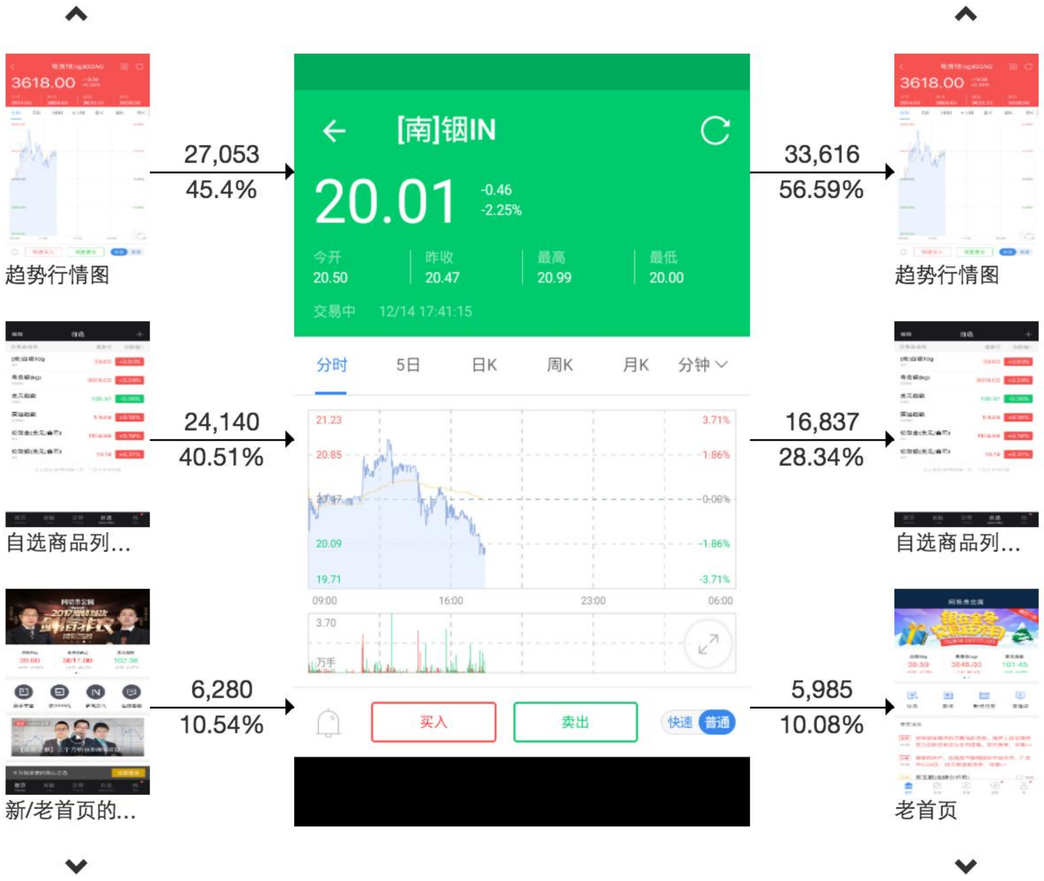
3.3 xPath优化 - 动态Add Fragment优化



- Activity承载多个Fragment
  - fragment第一个被初始化，index为0 xpath1=DecorView/ＸＸＸ/**FrameLayout[0]**
  - 同一个fragment第Ｎ个被初始化，index为[N-1] xpath2=DecorView/ＸＸＸ/**FrameLayout[N-1]**．
- 解决方案
  - fragment对应View节点名 ＝ fragment类名

3.4 xPath相关 - 可视化圈选

Page + xPath + Content <-> Title



按钮名	点击数	点击率
快速买入	10910	56.54%
快速卖出	7594	39.36%
提醒设置	392	2.03%
横屏显示	243	1.26%
快速/普...	157	0.81%

- content ?
- 关于圈选通配
  - 列表：直接通配
  - 非列表：对比通配

3.5 Page归属优化 - 弹窗点击

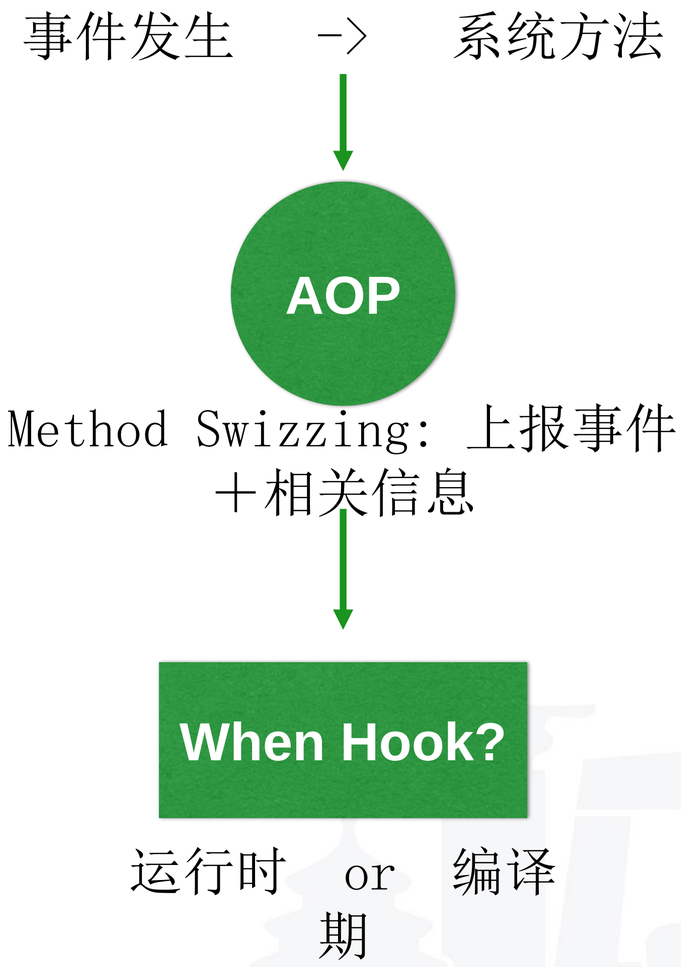
- 为什么要收集？
  - 点击所属Page？
  - 不同page相同弹窗区分？
- 策略：
  - 找当前最上层显示的Controller / Activity
    - 多个子页面处理？
  - 自动获取弹窗标题和content
- 实现：Hook（iOS） + 静态插桩（Android）

- 四、无埋点技术实现

博客：

http://www.jianshu.com/c/ee326e3
556

4.1 “无埋点” 核心实现



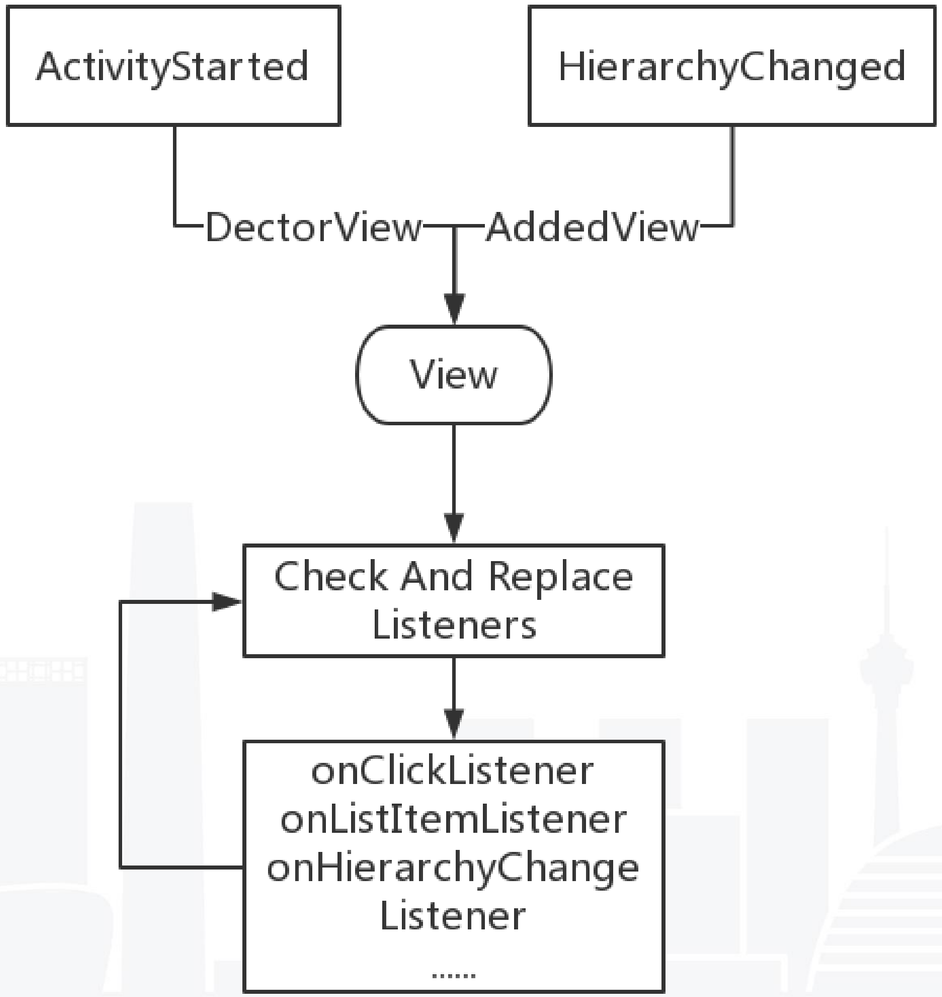
- 考虑：
  - 运行时Hook的性能消耗
  - 运行期多次Hook的隐忧
  - 编译期对构建工具的依赖
- iOS: 运行时
  - OC Runtime -> swift隐忧 -> 编译期？
- Android: 运行时 ＋编译期
  - 前期：运行时代理替换 ActivityStarted时机
  - 后期：编译期插桩 gradle插件

4.2 iOS Method Swizzing

- Hook协议方法（系统类的Delegate方法）： referProtocol
  - class_replaceMethod(Class cls, SEL name, IMP imp, **const char *types**)
  - **protocol_copyMethodDescriptionList**();
- 其他AOP框架（Aspects && Wax && JSPatch ）的兼容
  - _objc_msgForward
  - hook方法**反替换**
  - [self **forwardInvocation:invocation**]

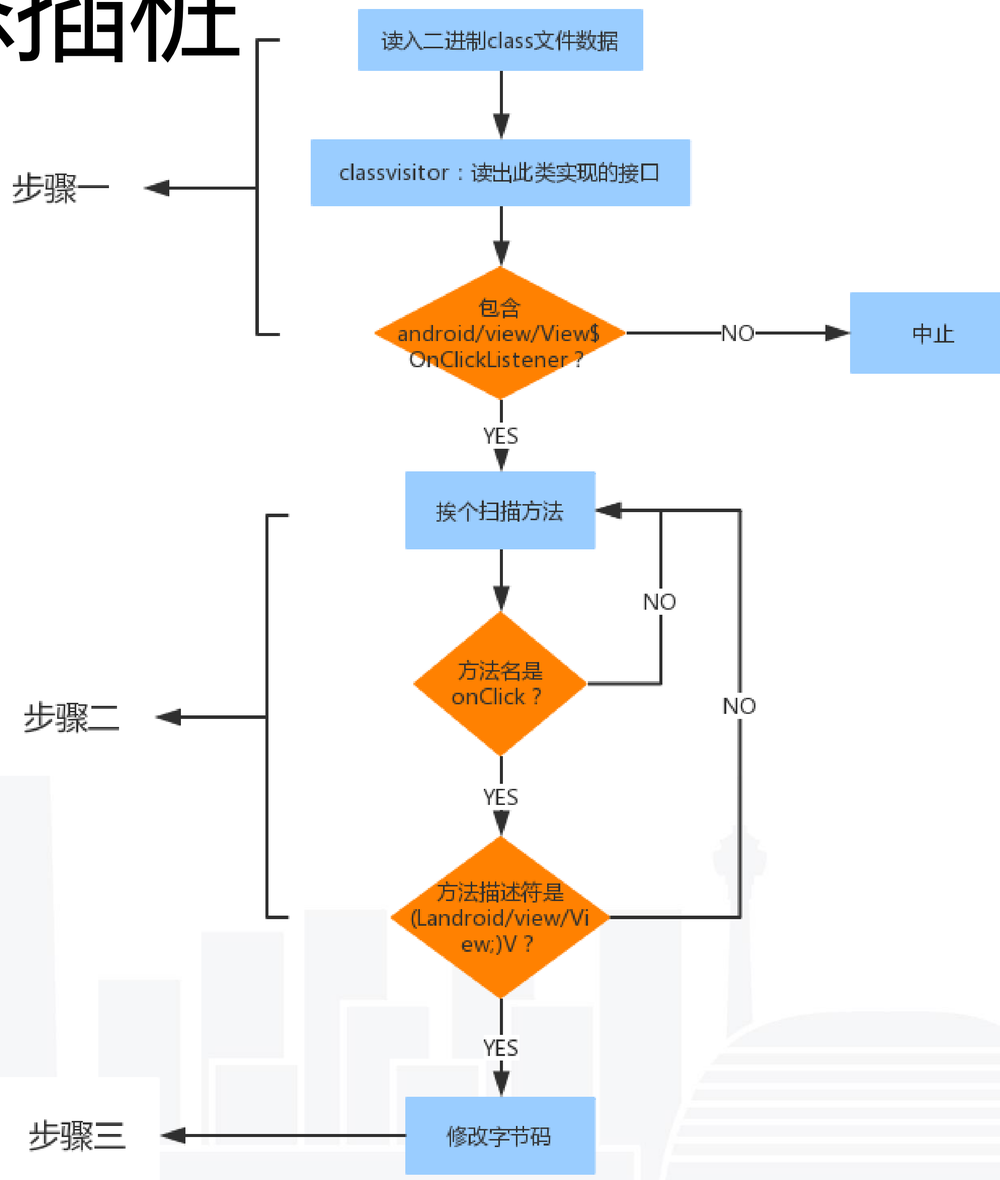
4.3 Android 动态代理替换

- 点击Hook时机：
  - viewTree生成
  - 回调ActivityStarted 或 HierarchyChanged方法
  - 反射Listener方法 / 其他事件方法
- 缺点：[辅助方案]
  - 性能消耗（ vs iOS ）
  - 业务代码重新setListener，Hook失效
  - 弹窗无法代理

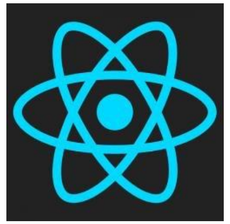


4.4 Android 静态插桩

- 插桩：Hook Android打包流程 + ASM字节码注入
  - 插桩入口获取 (com.android.tools.build:gradle)
    - **v.androidplugin < 1.5.0 : hook dx.jar**
    - v.android gradle plugin >= 1.5.0: transformapi
- 优缺点：[主方案]
  - 收集数据全，无反射，效率高 [优]
  - 插件框架忧虑：
    - 造成插件和主项目间的依赖关系
    - 主项目配置和各个插件一致，较为繁琐



4.5 ReactNative的支持



- 纯RN项目
  - navigator + tabBar
- 混合开发 (主)
  - Single Bridge
  - Mutil ViewController Instance

- 页面事件：手动埋点（ Show / Hide ）
  - 纯RN：Root ViewController / Activity 单例 + Associated Page Property
  - 混合模式：instance + page property
- 页面点击事件: Hook RN method
  - iOS: RCTTouchHandler -> touch对象
  - Android: NativeViewHierarchyManager -> reactTag

Thanks！ Q&A

庞辉

不丹

扫一扫上面的二维码图案，加我微信