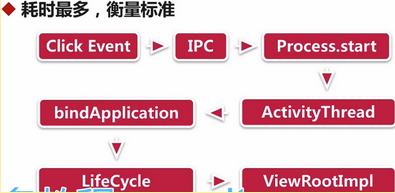
# 启动优化

## 1.启动的三种方式

### **（1）冷启动**

****

**优化方向: Application和Activity生命周期**

### **（2）热启动**

**最快，后台切到前台**

### **（3）温启动**

**较快LifeCycle**

## 2.启动时间测量方式

### **1.adb命令**

**adb shell am start -W packagename/packagename.MainActivity  
然后运行后出现三个时间  
ThisTime:最后一个Activity启动好使  
TotalTime:所有Activity启动耗时**

**说明：如果有启动页的话就是启动页和主页面启动的总耗时  
WaitTime:AMS启动Activity的总耗时**

**总结：**

* **时间耗时排序**

**ThisTime< TotalTime< WaitTime**

* **优缺点**
* **线下使用方便，不能带到线上**
* **非严谨、精确时间**

### **手动打点的方式-->（启动时埋点，启动结束埋点，二者差值） 1.开始时间为 application中执行attachBaseContext()方法为第一步 2.结束时间为 Feed的第一条展示的时间也就是Activity显示第一条数据的时间 （误区：认为 onWindowFocusChange这个方法是activity第一帧）**

**总结：**

1. **精确可带到线上，推荐使用**
2. **避开误区采用Feed第一条展示**

## 3.工具类使用

### **1.traceView**

**（1）traceView优缺点**

**优势**

* **图形的形式展示执行时间，调用栈等**
* **信息全面包含所有线程**

**缺点**

* **运行时开销严重，整体会变慢**
* **可能会带偏优化方向**

**（2）使用方式**

**Debug.startMethodTracing("")  
Debug.startMethodTracing()**

**生成文件在SD卡：Android /data/packagename/files**

****

**重点关注：call Chart 和 TopDown**

### **2.systrace**

**（1）特点**

* **结合Android内核的数据，生成html报告**
* **API 18以上使用，推荐TraceCompat**

**（2）使用**

**方式一**

1. **进入Android/Sdk/platform-tools/systrace目录下**
2. **可以配置一些参数，类似于通过Android Device Monitor抓取时步骤2配置的显示信息，若不选择则默认全部抓取**

**-o： 指定文件输出位置和文件名**

**-t： 抓取systrace的时间长度**

**-a： 指定特殊进程包名（自己加Label时必须加上）**

**eg: python systrace.py -t 10 -o /Users/Downloads/boot.html -a gaosi.com.learn**

1. **Chrome浏览器（必须）。在地址栏输入[chrome://tracing](chrome://tracing/)命令，然后将生成的trace.html文件拖进来，或者通过load按钮导入**

**方式二、**

**TraceCompat.beginSection("beginSection")**

**TraceCompat.endSection()**