### 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。

* [HierarchyViewer配套视频](https://v.qq.com/x/page/y0393sa0jlp.html)
* [Lint配套视频](https://v.qq.com/x/page/d039381wbas.html)
* [内存抖动现象配套视频](https://v.qq.com/x/page/x0393gf7qp6.html)12-如何对android应用进行内存性能分析

### Android应用UI性能分析

在使用App时会发现有些界面启动卡顿、动画不流畅、列表等滑动时也会卡顿出现这种情况，可以考虑对UI性能分析。

首先要清楚卡顿的原因，有以下几种情况：

* 人为在UI线程中做轻微耗时操作，导致UI线程卡顿
* 布局Layout过于复杂，无法在16ms内完成渲染
* 同一时间动画执行的次数过多，导致CPU或GPU负载过重
* View过度绘制，导致某些像素在同一帧时间内被绘制多次，从而使CPU或GPU负载过重
* View频繁的触发measure、layout，导致measure、layout累计耗时过多及整个View频繁的重新渲染
* 内存频繁触发GC过多（同一帧中频繁创建内存），导致暂时阻塞渲染操作
* 冗余资源及逻辑等导致加载和执行缓慢
* 臭名昭著的ANR

### 如何分析？

分析UI卡顿我们一般都借助工具，通过工具一般都可以直观的分析出问题原因，从而反推寻求优化方案，具体如下细说各种强大的工具

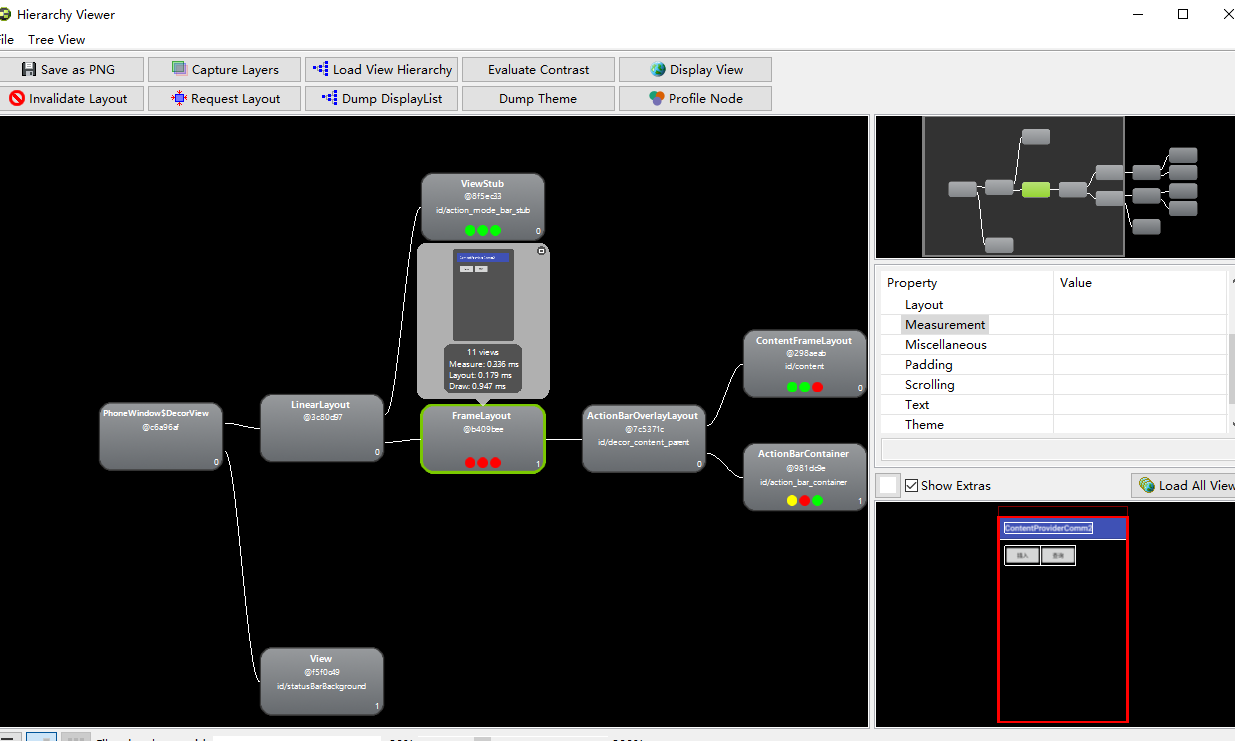
### 使用HierarchyViewer分析UI性能

我们可以通过SDK提供的工具HierarchyViewer来进行UI布局复杂程度及冗余等分析 通过命令启动HierarchyViewer

img

img

接下来Hierarchy window窗口打开：



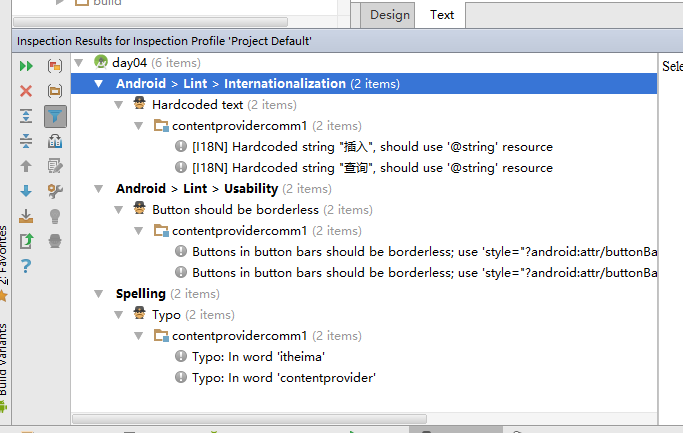
img

一个Activity的View树，通过这个树可以分析出View嵌套的冗余层级，以及每个View在绘制的使用时长也有表示。

### 使用Lint进行资源及冗余UI布局等优化

冗余资源及逻辑等也可能会导致加载和执行缓慢，这可以使用Link工具，来发现优化这些问题的

在Android Studio 1.4版本中使用Lint最简单的办法： 就是将鼠标放在代码区点击右键->Analyze->Inspect Code–>界面选择你要检测的模块->点击确认开始检测，等待一下后会发现如下结果：



img

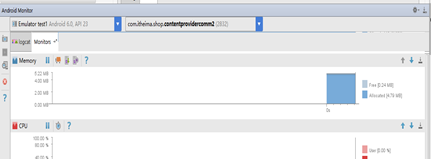
如果存在冗余的UI层级嵌套，会进行高亮显示， 我们根据提示可以点击跳进去进行优化处理掉的。

### 使用Memory监测及GC打印与Allocation Tracker进行UI卡顿分析

由于Android系统会依据内存中不同的内存数据类型分别执行不同的GC操作，常见应用开发中导致GC频繁执行的原因主要可能是因为短时间内有大量频繁的对象创建与释放操作，也就是俗称的内存抖动现象，或者短时间内已经存在大量内存暂用介于阈值边缘，接着每当有新对象创建时都会导致超越阈值触发GC操作

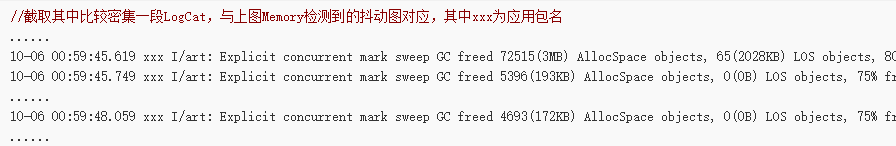
### 如何查看？

#### Android Studio 工具提供内存查看器：



img

#### 根据内存抖动现象，查看log日志进行分析：

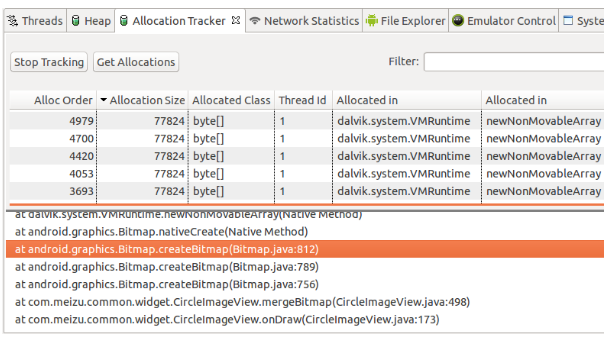


img

如何看到，这种不停的大面积打印GC导致所有线程暂停的操作必定会导致UI视觉的卡顿，所以我们要避免此类问题的出现，具体的常见优化方式如下：

* 检查代码，尽量避免有些频繁触发的逻辑方法中存在大量对象分配
* 尽量避免在多次for循环中频繁分配对象
* 避免在自定义View的onDraw()方法中执行复杂的操作及创建对象（譬如Paint的实例化操作不要写在onDraw()方法中等）
* 对于并发下载等类似逻辑的实现尽量避免多次创建线程对象，而是交给线程池处理。

有了上面说明GC导致的性能后我们就该定位分析问题了， 我们可以通过运行DDMS->Allocation Tracker标签打开一个新窗口，然后点击Start Tracing按钮，接着运行你想分析的代码，运行完毕后点击Get Allocations按钮就能够看见一个已分配对象的列表，如下：



img

根据内存分配情况，进行优化。

写篇文章和录制视频从选题，思考，组织，写作，编辑至少需要一个多小时，而您点赞和转发只需要0.1秒，就可以带给我更大的动力，何乐而不为呢？

* 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