Android APP程序书写和性能优化

|  |  |  |
| --- | --- | --- |
| 版本号 | 描述 | 日期(更新日期)/更新人 |
| 1.0 | APP程序书写和性能优化 | 2019/04/24 刘志保 |
|  |  |  |
| Github地址 |  | |
| Reference : |  | |

下面主要是经验分享.涉及理论就谈一下理论.

预备知识:

阅读Alibaba规范守则(第一版本,第二版本)

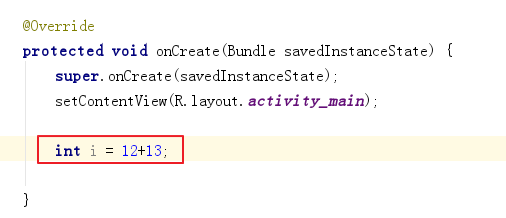
程序代码规范和优化,虽然上面介绍了两本阿里的书籍.但要在实际工作中做到书里面的规范似容易非容易.似容易是因为如果按照书写程序习惯了就很容易了,每写一条程序自然而然是规范的,而且还会犯下强迫症,变量命名不规范都不行.但是对于刚开始的来说,一下完全照着规范来很难,因为写着写着就忘记了,就会感觉按照规范写程序是”有时间”的开发人员做的事情.

使用约束方法,从下面几个习惯开始:

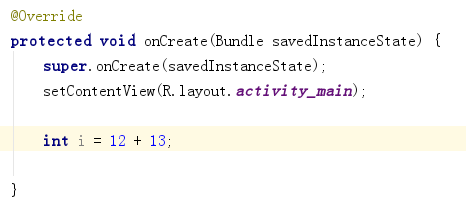
<1> : 写完一段程序后,养成随时随手ctrl+alt+L,将代码格式化一下.

比如 :

格式化之前:



格式化之后:



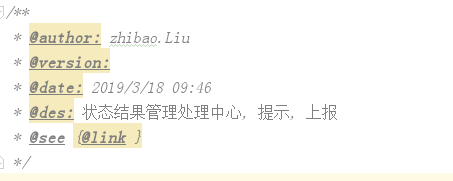
首先优先保持程序的美观性.

特别是一个Java文件里面程序非常多的时候,就一定要做.

这个操作简单吧.

增加Java doc模板,这个也是非常必要的,不要写了一个类,其他同事都不知道你这个类是干什么的,也没有类的基本介绍.

比如:



虽然看起来很简单,但是能减轻review代码和他人查看时的负担.

最烦最恶心的程序:

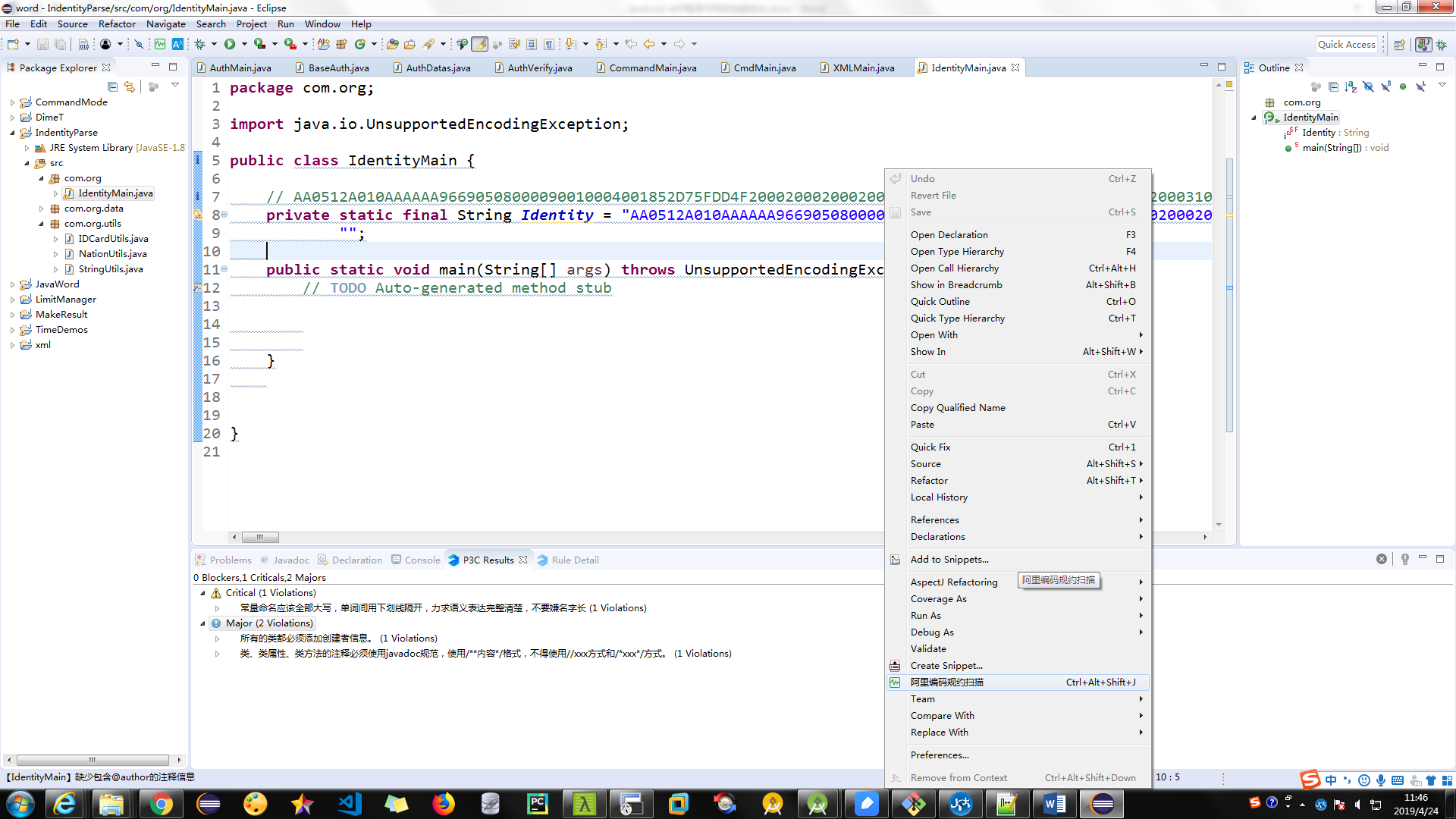
1. : 注释掉的代码还留着,总是觉得以后还能用的到,发挥余热;
2. : 连续空行TMD的多,空格,退格键乱添加,Tab键不换成空格键;
3. : 编码格式使用GBK,不使用utf-8;
4. : 注释到处乱写,时而写在程序行的上面一行,时而写在程序语句后面,时而写在程序语句后面一行,推荐注释都写在程序书写之前的一行位置,看完注释然后让人继续看程序的实现.注释写在程序语句后面同一行位置是最差的;
5. : 不要没事就打日志信息,能不打尽量不要打,不要搞得测试丢出一个日志信息,99%的日志信息是你的,别人的信息都被你的垃圾日志信息覆盖.
6. : 代码尽量复用,不要同一个逻辑功能,不同的人到网上各自抄一段不一样的,然后各自使用,一个人抄就可以了,后面的都用已经有的.
7. : 到网上抄代码,记得把方法名,变量名,注释修改一下,并且验证一下.
8. : 代码能够合并的尽量合并,共性合并,异性分离.

其他的规则可以参考前面介绍的两本书,反复熟读研究;

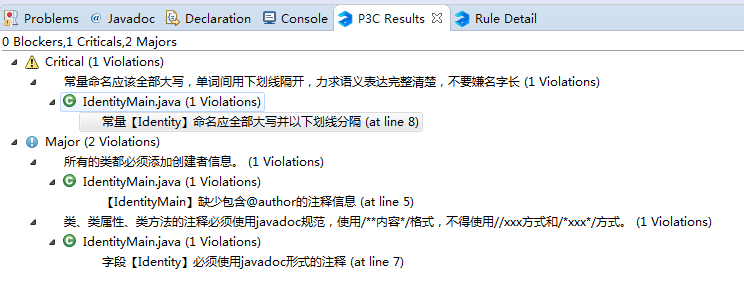
上面其中b,c,e这三种就是耍流氓抬扛的行为,平时绝对的杠精.

<2> : 阿里巴巴插件,无论是使用eclipse还是Android studio都可以下载安装阿里巴巴插件,下面主要以android studio为例.

1. Eclipse -> help->eclipse markplace,现在eclipse版本都有自己的软件市场(软件商店)了,搜索alibaba即可.
2. 安装以后使用:

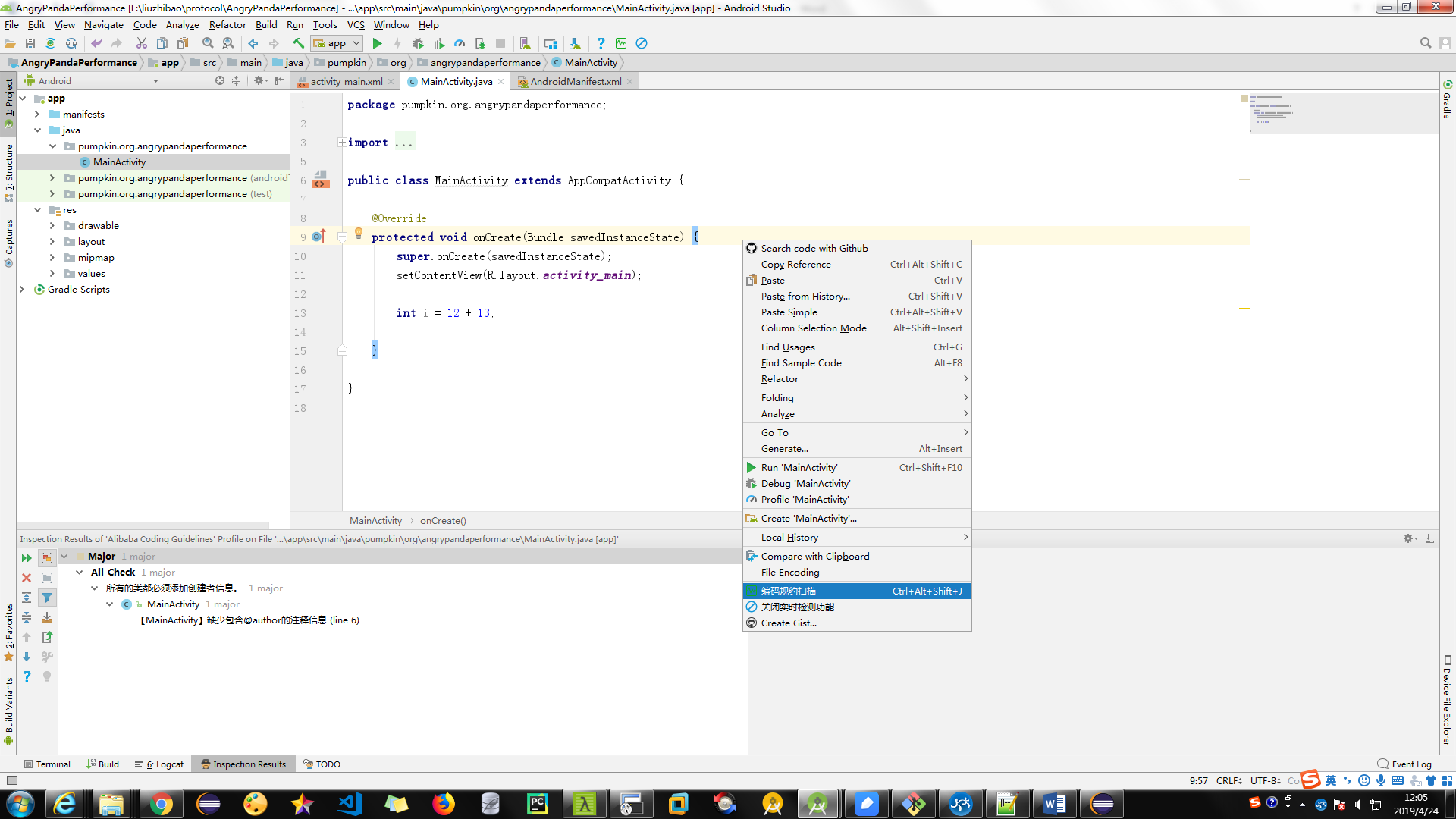


在代码空白处右击出快捷键,”阿里代码规则扫描”,稍等片刻即在下方显示不符合代码规则的信息提示:

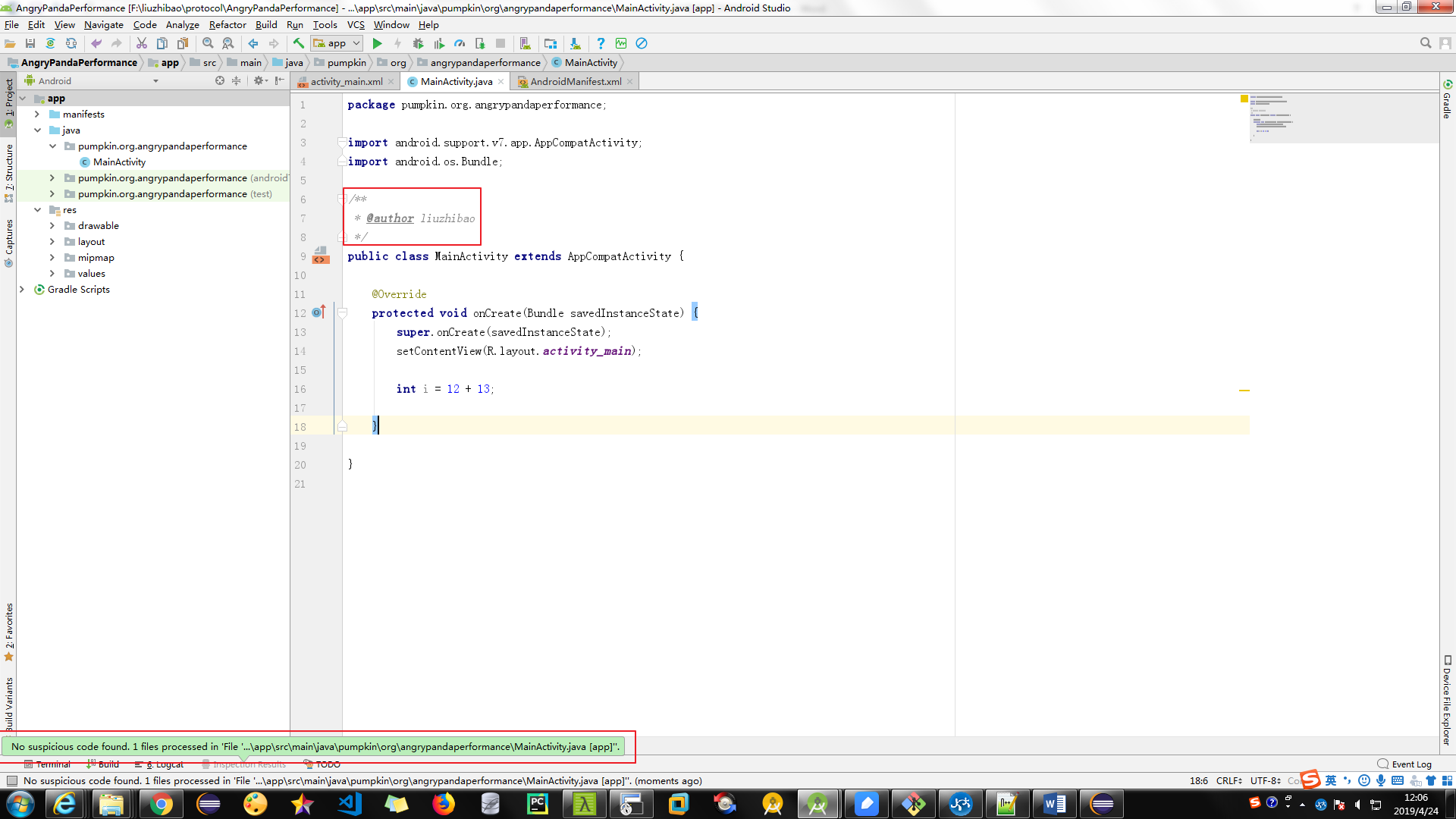


根据上面的提示,将对应不符合编码规则的进行调整.

1. : android studio 如下:



比如根据上面的提示修改,重新扫码:



它就会提示所有的代码基本上问题不大.当然实际当中肯定不止这么简单的扫描检查,它会检查所有的变量名,常量使用方式等待.

只有把程序写清楚了,整齐干净,逻辑理清了,才能谈性能.

性能分析有多种工具方式,一般都是多种手段结合起来搞.

<1> : 原生系统再带的分析性能的:

1. : 设置APP->开发者选项->监控->把这个下面的全部打开(如果是工程设备,会有很多项的,但是一般手机可能只有两项”严格模式”,”GPU”).
2. : android linux 的dumpsys命令,常见的如下:
3. : 查看内存:

查看整个系统的: dumpsys meminfo [进程PID号]

查看某个APK(包)的: dumpsys pkg\_name

1. : 查看线程(CPU消耗):

ps | grep '包名'

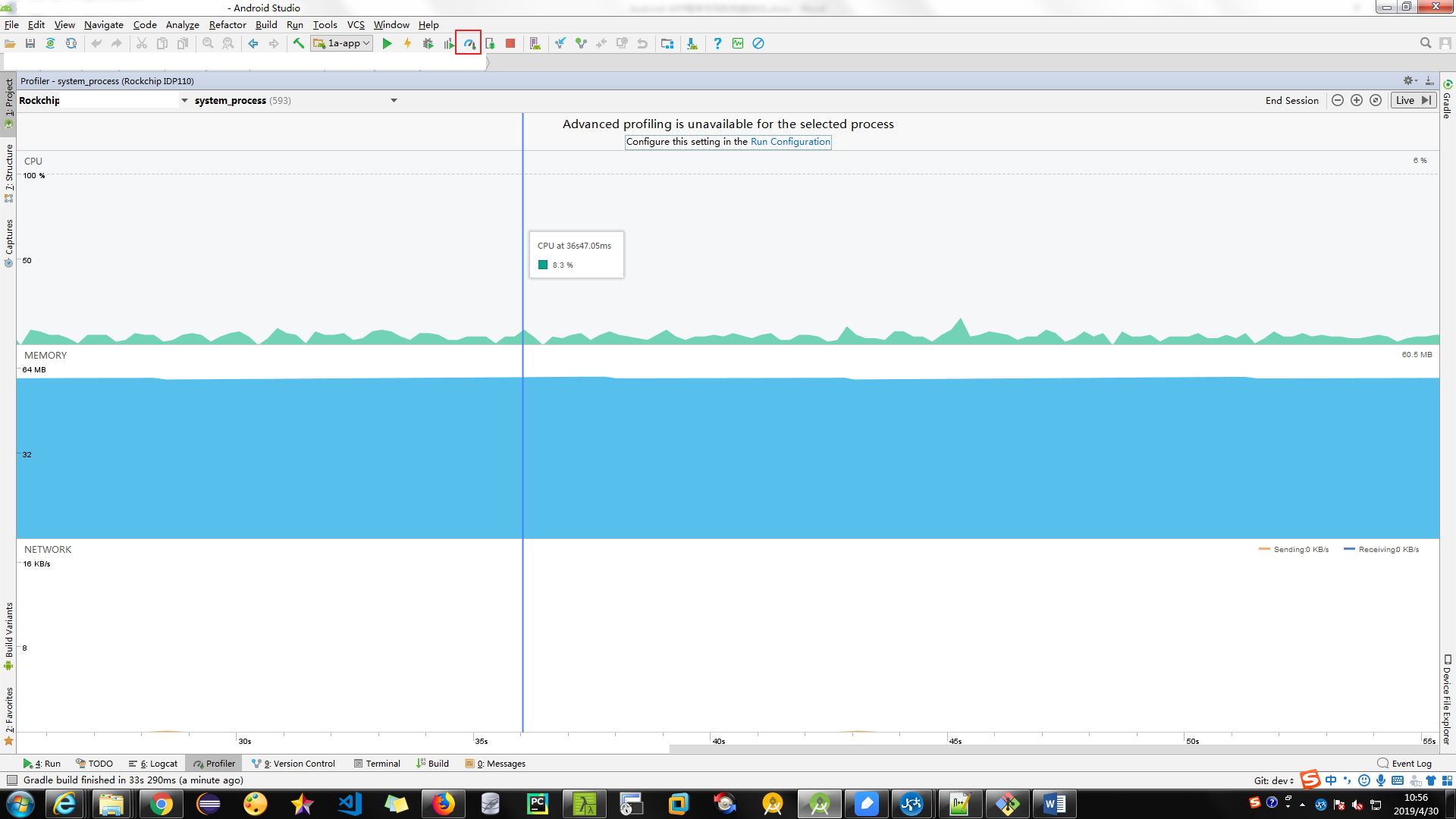
ps -t pid 查找指定pid的线程信息

1. : 查看组件:

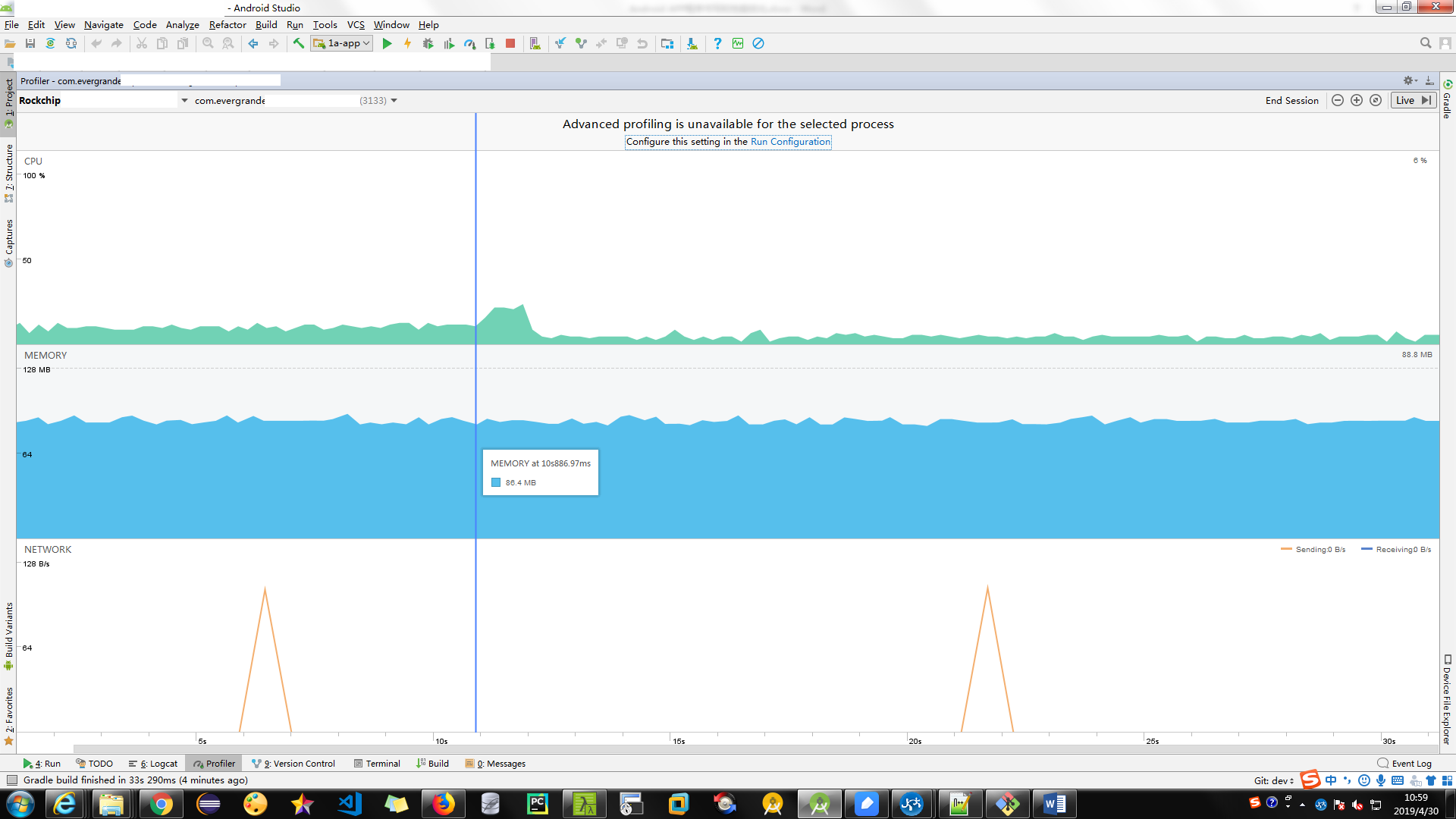
dumpsys activity

当然还有其他的,但是和性能的关系都不是很大,或者没有直接关系.

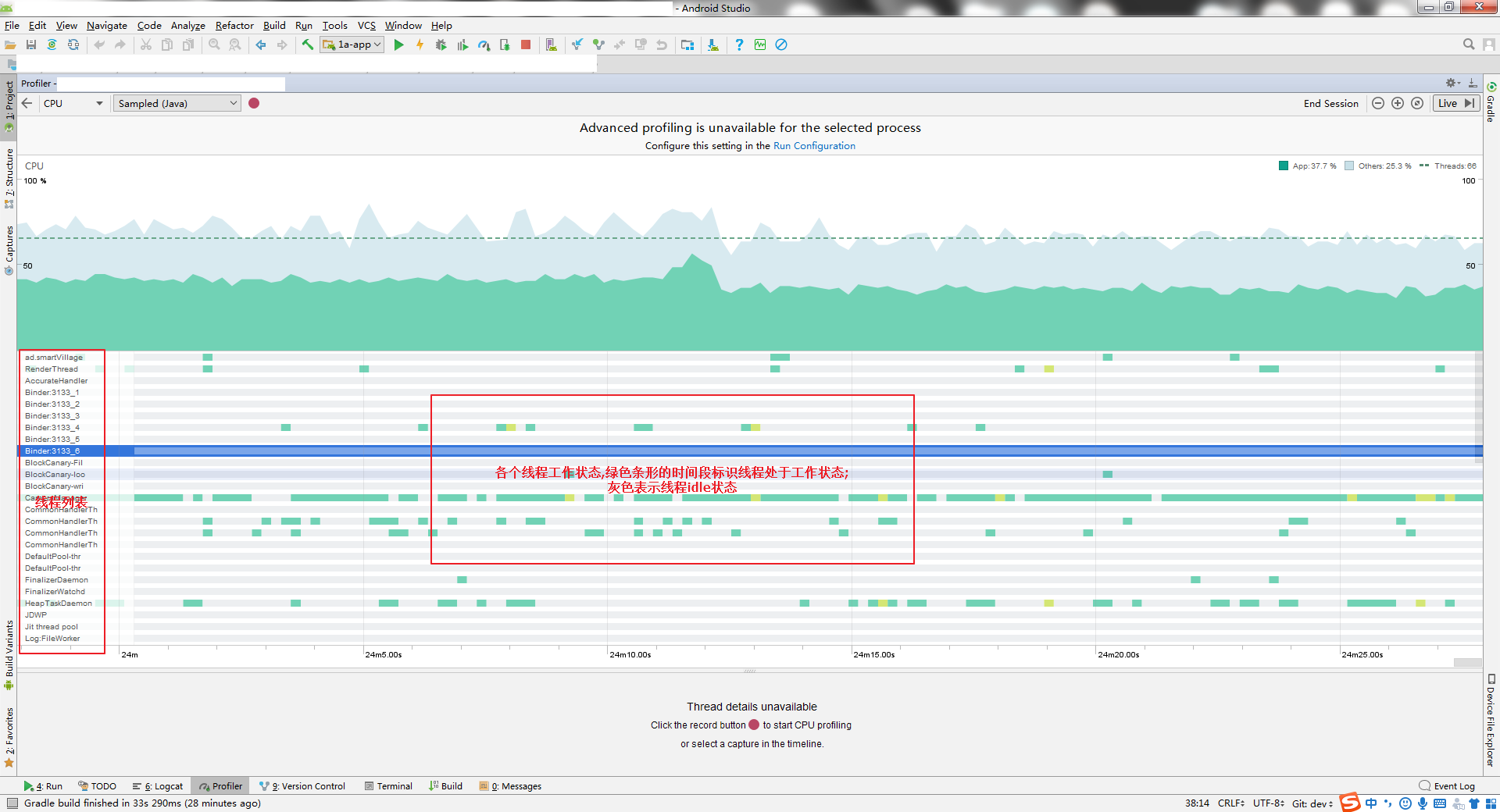
<2> : Android studio 点击菜单profile,如下红框:



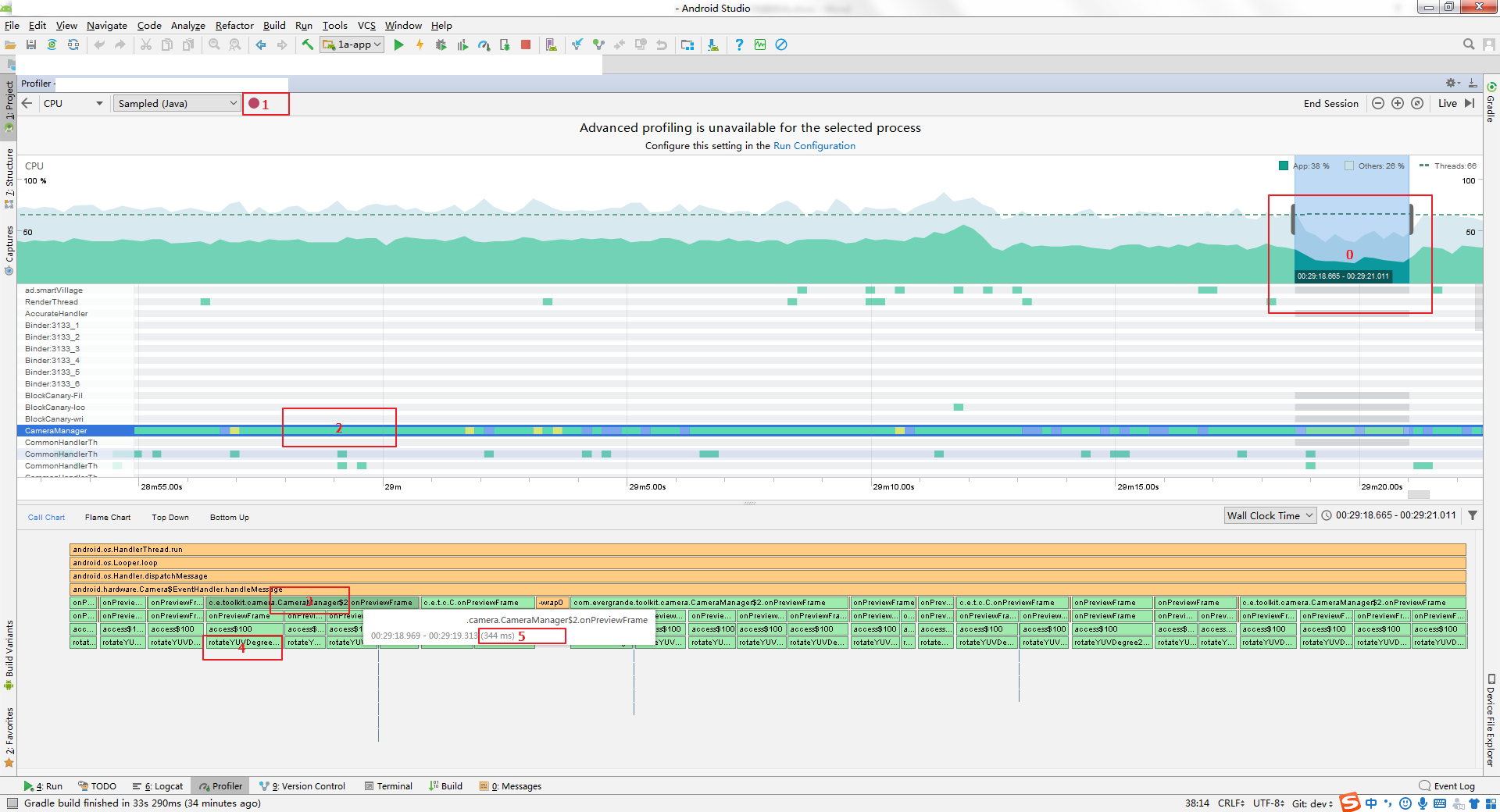
然后选中自己的app:



先看看CPU详细信息:



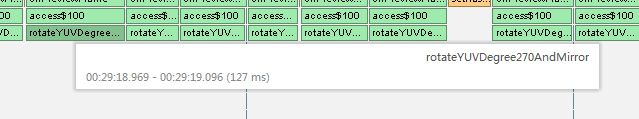
然后分析一段看看那些耗时:



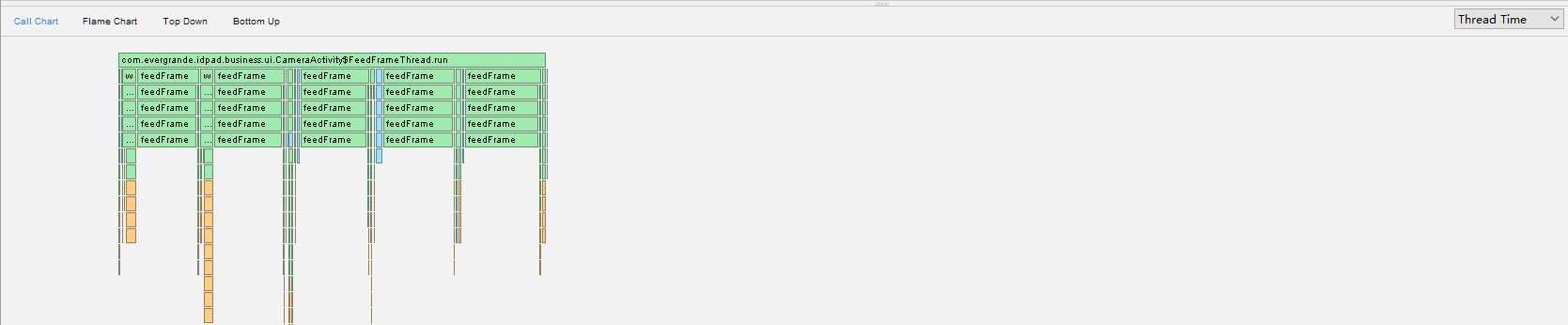
注意上面红色数字标注.

1. 点击开始记录,0是记录了一段.
2. 选择APP中的线程如上2,选中以后下面就会给出线程在这个记录时间段内所有的信息,比如看3位置,相机预览,鼠标点击3位置,就会显示预览回调整个过程使用了344ms的时间.

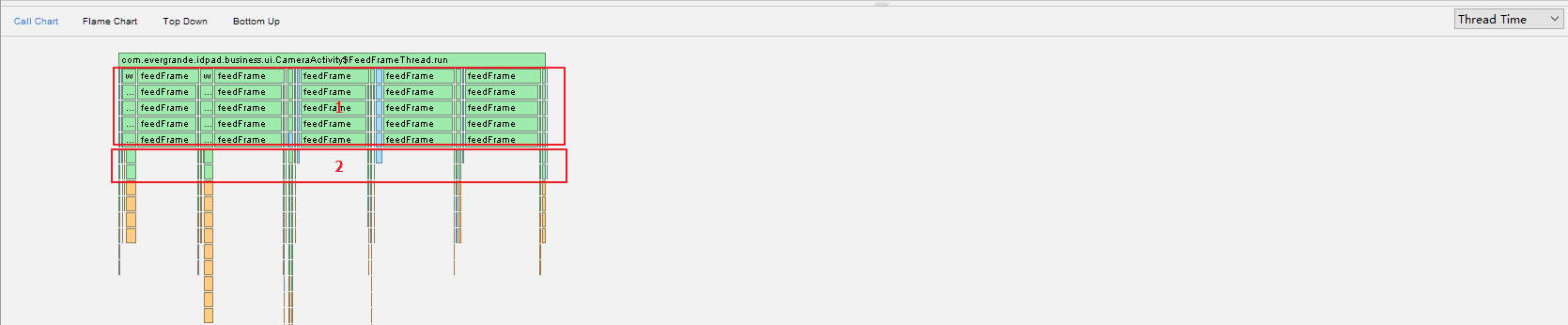
再看4位置:



是一个图像翻转的算法过程,耗时127ms,上面的想看哪个耗时,鼠标点击一下即可.上面显示也比较形象,比如3耗时比较长,3的绿色长条就比较长,4相对就短.方便让开发人员第一时间看到谁的耗时最长.如果双击选中的长条位置,它会自动帮开发者跳转到程序代码位置处,指示具体的程序位置.



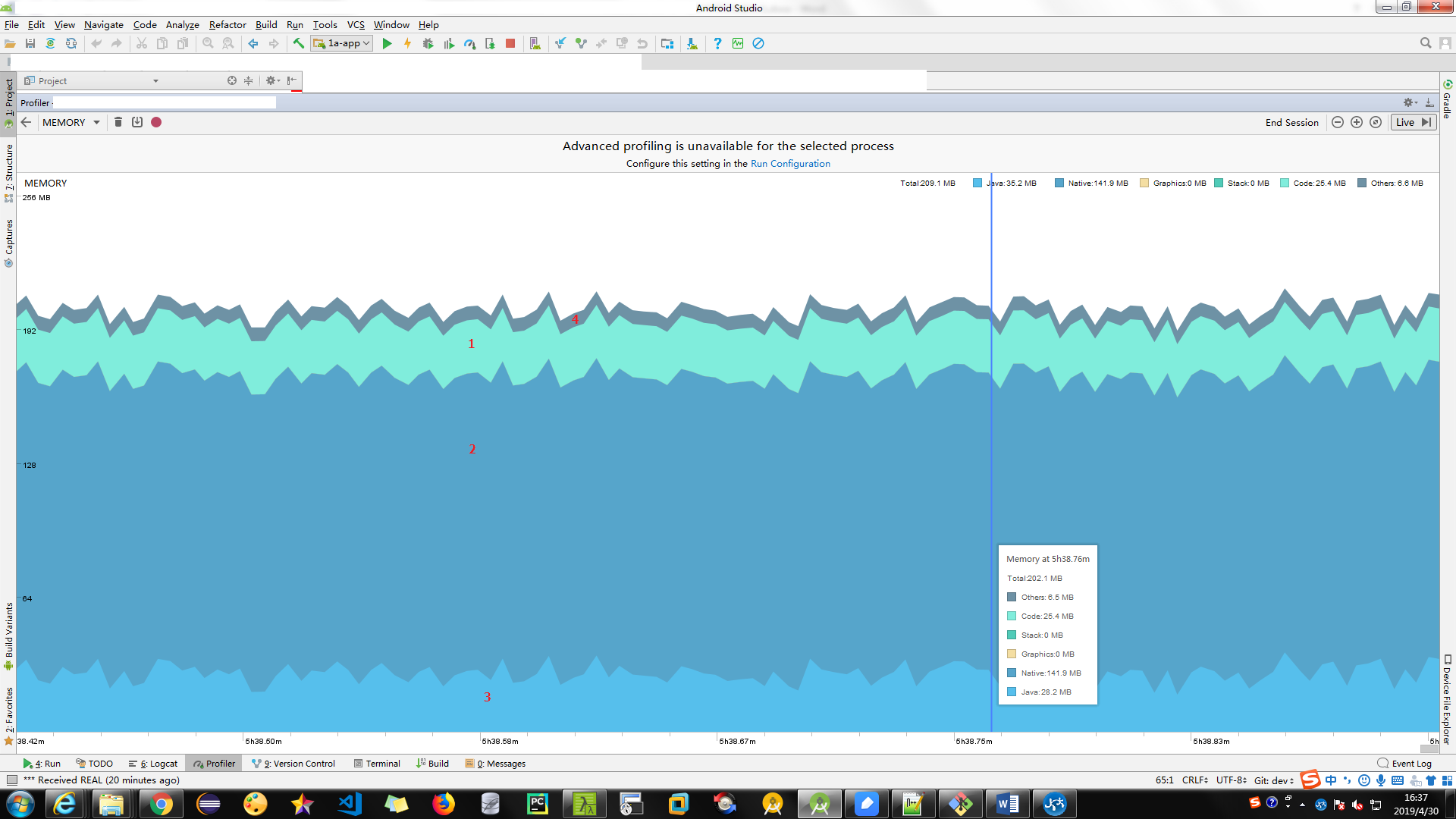
分成两个区:



可以看出FeedFrameThread线程,主要耗时是在1区的feedFrame中和2区的,从上面可以看出工作时间消耗最长的是1区.

耗时最厉害的部分,需要对程序进行优化.

再看看内存消耗Memory :



上面标记的区间,主要看1,2,3区,其中1区不需要特别关注,APP运行起来以后,基本上是一个固定值,1区条形锯齿非常多不是1区产生的,而是下面的3区产生的.而3区一般是大部分APP需要优化的,其实就是Java程序.