［译］开发安卓Apps，我所努力学习到的三十多条宝贵经验

* [Android](http://yifeng.studio/tags/Android/)

[Android](http://yifeng.studio/categories/Android/)

热度 2599℃

网上阅读时猎取到的一篇外国工程师的文章，作者 [César Ferreira](https://medium.com/@cesarmcferreira?source=post_header_lockup) 结合自己在安卓领域的摸爬滚打经历总结了三十多条宝贵经验，非常值得参考学习，故翻译记录于此，并作了一些补充说明，帮助阅读理解。译文原文：

[Building Android Apps — 30 things that experience made me learn the hard way](https://medium.com/@cesarmcferreira/building-android-apps-30-things-that-experience-made-me-learn-the-hard-way-313680430bf9#.5exr8zumk)

学习领域有两种人，一种是自身刻苦钻研一步一步摸索的人，一种是采取捷径获取别人经验的人。下面是我一路学到的东西，和你分享：

1. 添加使用第三方类库前，请再三思考，真的很重要；（未来一些未知的错误也许就发生在这些类库中，关于第三方类库的选择，参考文章：[stormzhang－如何正确使用开源项目？](http://stormzhang.com/android/2016/05/08/how-to-choose-open-source-project/" \t "_blank)）
2. 用户看不到的地方，就不要去画它；（避免过度绘制，参考文章：[Optimizing Layouts in Android – Reducing Overdraw](http://riggaroo.co.za/optimizing-layouts-in-android-reducing-overdraw/" \t "_blank)）
3. 除非真的需要，否则不要使用数据库；
4. 应用中65K的方法数很快就能达到，我的意思是真的很快！不过 [multidexing](https://medium.com/@rotxed/dex-skys-the-limit-no-65k-methods-is-28e6cb40cf71" \t "_blank) 也许能帮到你；（最近刚总结过一篇：[Android 突破64K方法数的限制](http://yifeng.studio/2016/10/26/android-64k-methods-count/)）
5. [RxJava](https://github.com/ReactiveX/RxJava) 绝对是AsyncTasks等绝大多数类最好的替代品；（参考文章：[Party tricks with RxJava, RxAndroid & Retrolambda](https://medium.com/swlh/party-tricks-with-rxjava-rxandroid-retrolambda-1b06ed7cd29c" \l ".ctr2m3fpn" \t "_blank)）
6. [Retrofit](http://square.github.io/retrofit/) 是最优秀的网络框架；（没有之一）
7. 使用 [Retrolambda](https://medium.com/android-news/retrolambda-on-android-191cc8151f85" \t "_blank) 缩减你的代码；
8. 感受RxJava与Retrofit和Retrolambda一起使用的魅力；（参考文章：参考文章：[Party tricks with RxJava, RxAndroid & Retrolambda](https://medium.com/swlh/party-tricks-with-rxjava-rxandroid-retrolambda-1b06ed7cd29c" \l ".ctr2m3fpn" \t "_blank)）
9. 我使用 [EventBus](https://github.com/greenrobot/EventBus" \t "_blank) ，它很强大，但我不会过度使用，因为它会使代码库会变得很杂乱无章；
10. 根据应用功能分包，而不是所属类别；（项目目录结构划分，参考文章：[Package by features, not layers](https://medium.com/@cesarmcferreira/package-by-features-not-layers-2d076df1964d" \l ".em996bn3v" \t "_blank)）
11. 移除 Application 线程里的一切代码；（避免拖慢应用的初始化和启动速度）
12. 使用 [lint](http://developer.android.com/tools/help/layoutopt.html) 优化布局，以便你能一眼识别出冗余的视图并移除；
13. 如果你使用gralde，想尽一切办法加快编译速度；（参考文章：[How I save 5h/week on Gradle builds](https://medium.com/@cesarmcferreira/speeding-up-gradle-builds-619c442113cb" \l ".1d2hhi8hu" \t "_blank)）
14. 使用 [Profile report](https://medium.com/the-engineering-team/speeding-up-gradle-builds-619c442113cb) 查看编译时间到底是在什么地方耗费的；
15. 尽量使用众所周知的成熟架构体系；（参考文章：[Architecting Android…The evolution](http://fernandocejas.com/2015/07/18/architecting-android-the-evolution/" \t "_blank)）
16. 测试消耗时间，但是一旦你掌握了测试的窍门就会发现，它比没有经过测试的代码更快更稳妥；（参考地址：[http://stackoverflow.com/questions/67299/is-unit-testing-worth-the-effort/67500#67500](http://stackoverflow.com/questions/67299/is-unit-testing-worth-the-effort/67500" \l "67500" \t "_blank))
17. 使用依赖注入使你的应用更加模块化，并且更容易测试；（参考文章：[Tasting Dagger 2 on Android](http://fernandocejas.com/2015/04/11/tasting-dagger-2-on-android/" \t "_blank)）
18. 关注 [Fragmened Podcast](http://fragmentedpodcast.com/" \t "_blank) 对你大有帮助；（Fragmented，一个专属安卓开发者的播客网站）
19. 永远不要使用私人邮箱作为安卓市场的发布者账号；（主要是Google Play，案例参考：[https://www.reddit.com/r/Android/comments/2hywu9/google\_play\_only\_one\_strike\_is\_needed\_to\_ruin\_you/](https://www.reddit.com/r/Android/comments/2hywu9/google_play_only_one_strike_is_needed_to_ruin_you/" \t "_blank)）
20. 坚持使用合适的输入类型；（针对输入框，参考链接：[Specifying the Input Method Type](https://developer.android.com/training/keyboard-input/style.html" \t "_blank)）
21. 学会借助分析学寻找通用模式和孤立问题；（设计模式，封装等）
22. 保持学习最新开源类库，并借助 [dryrun](https://github.com/cesarferreira/dryrun" \t "_blank)工具测试开源类库；（[Android Arsenal](https://android-arsenal.com/" \t "_blank)，一个搜索整合Android开源类库的网站）
23. Service服务应该做它们需要做的事情，并且尽可能快地终止；
24. 使用 [Account Manager](http://developer.android.com/reference/android/accounts/AccountManager.html) 提示用户名和邮箱地址；
25. 使用CI(持续集成)编译构建测试版和发布版应用；
26. 不要运行你自己的 CI server，防止 SSL 攻击而造成的磁盘空间、安全问题、服务更新都需要维持 server ，这是一件耗费时间的任务。使用circleci、travis 和 shippable，相比而言，性价比更高，更可靠；
27. 使用 [gradle-play-publisher](https://github.com/Triple-T/gradle-play-publisher" \t "_blank) 自动部署上传Apk文件等信息到应用商店；
28. 如果一个library比较大，而你只是用到其中的一小部分功能，那么你就应该寻找一个更小的替代品；（比如可以借助 [proguard](http://developer.android.com/tools/help/proguard.html" \t "_blank) 工具）
29. 不要大量使用超出你实际需要的依赖库。特别是当这些依赖库不是经常在变时，我们就要考虑到，这些类库从头编译（ CI Builds 就是一个很好的例子）或者检查之前编译好的独立类库是否需要更新所花费的时间相比简单地加载jar或者aar这样的二进制文件，高达四倍之多；
30. 开始考虑使用 SVG 代替 PNG 格式的图片；（参考地址：[Add Multi-Density Vector Graphics](http://developer.android.com/tools/help/vector-asset-studio.html" \t "_blank)）
31. 封装抽象化 library 的使用，这样当你需要使用新的 library 替代旧 library 时就会变得很容易；
32. 监听网络连接变化和连接类型（Wifi状态下数据更新更频繁？）；
33. 监听电源和电池电量变化（充电时数据更新更频繁？电池电量不足时暂停更新？）；
34. 展现给用户的UI就像一个笑话，如果你不得不解释一下的话，它就不是一个好笑话；
35. 性能测试很重要：Coding 实现慢，但要正确，然后验证优化，这不会影响任何测试内容。