深入理解 Android

序

携来百侣曾游，忆往昔峥嵘岁月稠。 -- 《沁园春。长沙》

对于Android，我也算是老人了，所谓，有文有真想。正由于这段玩票经历，使得我在毕业后，鬼使神差的成为移动平台的一名码工，再次有机会放肆的拥抱Android。

2010 开年，手上突然有了一把闲散时间，有机会进一步总结和学习 Android。于是想再一次为 Android 写一系列的东西，这些东西来自于一些开发经验，对源码的学习和对 Android 的浅薄认识，也算是鞭笞自己学习的一种手段。

其下所有内容，预计有十数篇，抑或更多。基本和技术相关，也许会配有一些其他相关比较闲淡的话题。可能会有一些具象实例，但更多的可能是自己的一些理解和认知。所有一切，源自于妄自挖掘，难免有疏漏或误解，观者淡定。

以此为序，有心者，望共勉。

Android 简史

人生若只初识，何事秋风悲画扇。 -- 《木兰辞》

要说当今移动平台的当红辣子鸡，Android说它是第二，也许没有别家敢认这个第一（好吧，iPhone，有意见就说。..）。了解Android开发平台的过去和现状，除了往下看，另有便捷的方式就是在WikiPedia中键入Android，在这里，特此鸣谢GFW友情放生。

诞生

早在 2005 年 7 月，Google 舞动着手中的美刀，收下了由 Andy Rubin（传说中的 Android 之父。..）等人创立的一家小公司，他们当时做的就是基于 Linux 内核的手机操作系统，也就是小时候的 Android。经过 Google 多年打磨，Android 在 07 年末，正二八经的粉墨登场开门接客。

自打一出生，Android便被钉板在富二代的角色上，不仅是因为老爹有钱的令人发指，也是因为其后有一帮金光闪耀的叔伯们保驾护航。这个叔伯群，便是响当当的开放手机联盟OHA（Open Handset Alliance）。这个联盟涵盖了中国移动、T-Mobile、Sprint这样的移动运营商，也包括HTC、Motolora、三星这些的手机制造商，同时还有以Google为代表的手机软件商，以Inter、Nvidia为标志的底层硬件厂商和Astonishing Tribe这样的商业运作公司（这公司是啥我也不晓得。..）。作为后援团，他们理论上的任务，是各尽其长，全力捧红Android，实际上的任务是齐心协力，借Android东风赚一个盆钵满响。

当然，Android自所以被万众瞩目一炮走红，不仅仅是丫实在太有背景，同时，它也有这太多的新鲜的概念。Android是一个开源的平台（恩，真正的全面开源，是在发布后很久以后了。..），它给那些捂着自家平台源码当宝的竞争对手们一记当头棒喝。Android自行研发了一套Java虚拟机，当时仅提供Java API的支持（NDK是更久以后的事情了。..），号称为专为高端智能设计。Android开发环境支持所有主流操作系统平台，包括Windows，Linux，Mac，即便到今天，在手机开发中也是极其罕见的。Android的带头人Google，本身是做网

络起家，Android内嵌大量Google网络应用，听上去就显得很酷。这所有的一切，共同造就了Android那鹤立鸡群，不染风尘的少侠形象。

造势

推出伊始，Google 还有一个很震撼的推广举动，就是举行所谓的 Android 程序挑战赛（Android Developer Challenge， ADC）。整个比赛分成两场，第一场（ADC1）比赛，在没有任何真机问世，SDK 还是个雏形的状况下，便鸣金开锣了。

比赛套路是无差别的群殴，基本概念是无论你来自何方（还是要满足美国法律要求和避嫌要求的），无论你想做些什么，也不管你是光杆司令还是流氓团伙，只要提交一个能在模拟器上跑起来的程序，就可参赛。而比赛只是对你作品进行参观评比，作品的所有权依然放在开发者的口袋中。

当然，这还不算什么创新，NB 的是无比丰厚的奖金，整个 ADC 的奖金高达 1 千万美刀，每场各半，基本上首轮入围奖（前 50）已经超越了那时候一般程序比赛的头名奖金，这对很多小公司和个人而言，无疑是具有很强吸引力的。于是，各路打酱油好手蜂拥而至，各论坛、博客、网站也七嘴八舌的讨论开来，一时间，满城风雨。

ADC1 我也很厚颜无耻的参加了，结果当然可以预想，一毛钱都没摸上。回想整个过程，差距最大的并不是在技术上，而是认知上，我们玩的产品是人家几年前玩剩下的，说创新只是一抹笑谈。

当时觉着，Google 太 NB 了，ADC 这种车马未动粮草先行的招太华丽了，就这动静，不论比出来个啥结果，这 1 千万刀也掏值了。但时过境迁，现在回头来想，也许一切并不是看上去那么美。由于没有扎扎实实的真机摆出来，大家普遍抱着一种玩票的心理，真的敢不顾一切舍下身家性命押宝在 Android 上的尽在少数，这就锻造了 Android 平台很长一段时间的只见雷不见雨的局面。而等喧嚣过后，很多人热情消退，Google 真端出 Android 真机的时候，还需要重新热场再来一次，也许，真的有些得不偿失。

困境

所有的东西现在来说，都是事后诸葛亮，只能听个响不能当个真。而真实的状况是 ADC1 很快进入囧境，由于架构设计上的种种原因，Google 花了比预想多的多的时间做 Android 的优化工作，ADC1 比赛被迫不断延期，彻底沦为懒婆娘的裹脚布。各路曾热捧 Android 的媒体，也不失时机的倒戈，亲手在自己画上的感叹号后面，重重画上了个大大的问号。

祸不单行，同样是由于 Android 的性能问题，虽然各路高手把 Android 移植到了不同的手机平台上，但传说中的 GPhone 一直难产中，使得人们不免有了胎死腹中的猜测。

与此同时，其他对手可一点也没闲着。iPhone 很快宣布开放 SDK，以此来勾引纯情的开发人员。Symbian 被 Nokia 彻底收购，成为 Nokia 的自留地，开源计划也很快浮出水面。所有景象，对 Android 而言都犹如梦魇。

破茧

所有一切困境，都在 G1 发布后，渐渐消散了。2008 年 10 月，第一款搭配 Android 平台的真机，搭载着无限光荣与梦想的 HTC Dream 正式发售，这就是注定要载入史册的 G1。虽然比之当时绝代风华的 iPhone，粗陋的 G1 犹如村姑遇上公主一般，但无论如何，G1 让人们真真切切的看到了 Android。这就犹如你家买的跳票 N 久的期房，终于见着了个毛胚房，那种感觉，除了踏实，找不到更适合的词汇了。

好事当然也会成双，G1 它不是一个人在战斗。ADC1 总算是落下帷幕，Android Market 的也顺理成章的破茧而出，早期的应用，大都来自于 ADC1 的贡献。

Android也结束了伪开源的历史旅程，正式开发SDK的源码实现，搭配的是Apache的License，这种坦诚相见的感觉看上去很不错。

忠心耿耿的 HTC，更是再接再励，在 G1 后，陆续发布了 Magic（G2）和 Hero（G3）。尤其是 Hero 的现身，惹得一阵小惊艳，HTC 为 Hero 搭配的是基于 Android 改造 UI 的 Sense 系统，以华丽的界面风格赚足了眼球，也创了改造 Android 的先河。

在HTC高歌猛进的同时，猫在螳螂后的一群黄雀，也敌在动我也动了。摩托罗拉，三星，LG，华为，戴尔，联想等一干手机厂商纷纷跟进，各式各样的Android蜂拥而至。与此同时，其他嵌入式厂商也推陈出新，爱可视（Archos）发布了基于Android的平板设备，明基的Android上网本也是箭在弦上，而基于Android的手持电子书阅读设备也不断的被推出，庞大的Android联盟初现峥嵘。

为了避免同质化，各个厂商纷纷对 Android 进行的改造，摩托罗拉推出了 Cliq，打得是 SNS 整合牌，三星的新系统也是被广泛期待，而中移动的 OMS 丑媳妇也要见婆娘了，打着整合移动服务牌@_@的 OMS，以丑陋的外貌、低下的 SDK 版本和雷死人不偿命的宣传文案（绝口不提 Android，只说自己做了大量很 NB 的工作，其实。..，哎，咋就那么小家子气呢。..）也算是招来大量眼球。

而还是没能耐住寂寞的 Google，联手 HTC，一同推出了至今只为止最重量级的 Android 手机：Nexus One。江湖有云：天下武功，无快不破。搭载了全新的 Android 2.1，1G 的 CPU，史上最清晰的手机屏幕的 Nexus One，快的是一塌糊涂迅雷不掩耳盗铃小叮当，在单机层面，第一次使得 Android 手机与 iPhone 掰手腕的能力（之前与 iPhone 的比较，都需要依靠集团力量，三英战吕布。..）。

在各家厂商努力的同时，Android本身也没有闲着，版本从 1.1，一步步进阶到了 2.1，SDK的升级，伴随着大量性能、接口的改进，和功能的丰富，Android变得越来越快，越来越省电，越来越丰富，越来越多Google服务被嵌入@_@。由于Android SDK是基于Java的，即便虚拟机做的很是NB，在某些情况下，性能也是无法与原生的C++代码相提并论，于是，从 1.5 版起，除了SDK，Android还拥有了NDK（Native Develop Kit），它提供了一些C++的库和编译环境（库是真的很少。..），开发人员可以基于C++写底层库，用Java写上层逻辑，通过混编的方式，兼得鱼和熊掌。

Android Market 的发展也甚为迅猛，虽然和其鼻祖 App Store 相比，应用的规模和盈利能力还显得比较幼齿，但其涨势凶猛，发展趋势远胜于前辈。国内一些比较著名的手机软件，也纷纷拥有了 Android 版本的小弟。

起飞

种种迹象表明，2010，也许就是姗姗来迟的 Android 元年。三星，moto，LG，HTC 等多家手机制造厂商，都为 2010 年将推出的半数以上的手机搭配了 Android。在国内，移动的 OPhone，丑媳妇要正式揭开盖头了，惨烈是惨烈了一点，但聊胜于无，除了水货，2010 毕竟至少多了条购买 Android 手机的道路。

软件开发方面，大家也从抱着双臂冷眼旁观的状态，进入到了一种伺机而动的战略准备阶段。前不久召开的 moto 开发者大会，惊现国内各领域的公司，试水开始，可见一斑。国内各个山寨的 Market 的，也越来越货源充足，下载量稳步上升，升温，就在当下。

而随着 G2 为首的 Android 水机价格火速下调，身边路边地铁边，可以看到越来越多的人，把玩着各式各样的 Android 手机，状况尤为喜人。

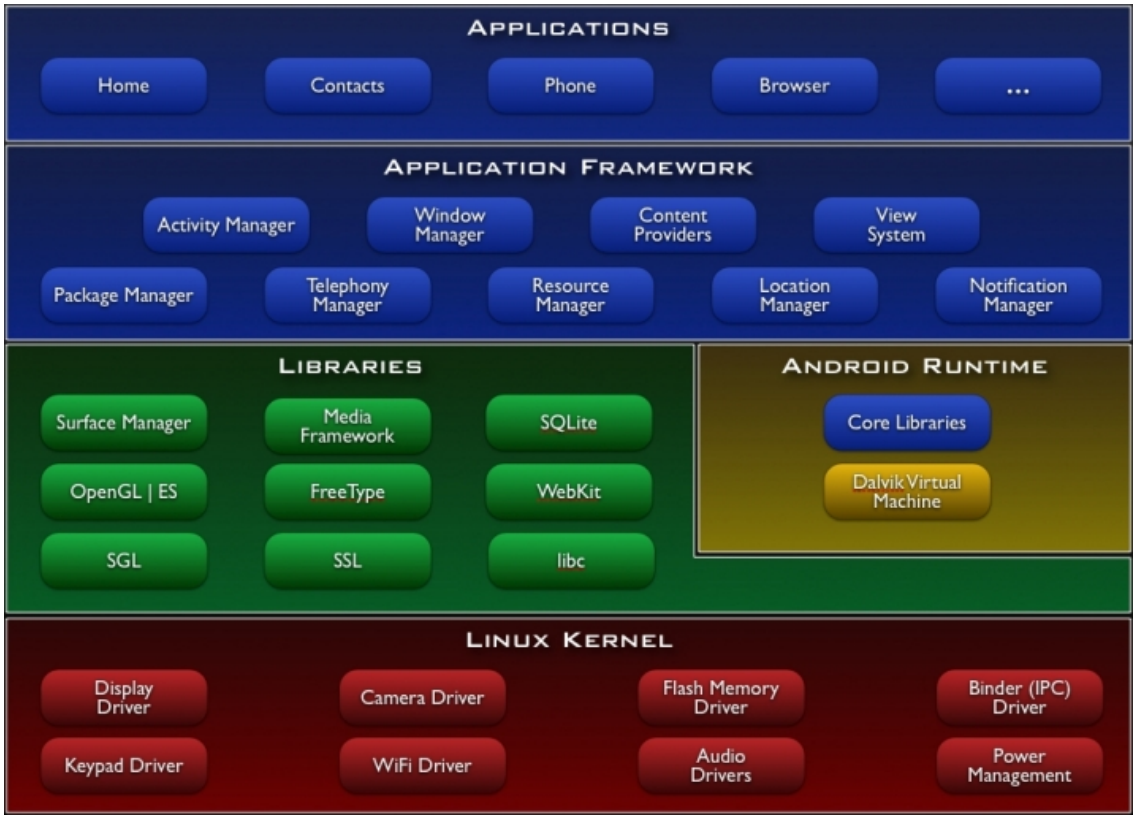
所以，2010，如果你有心，就做好准备吧。

- **Android 架构和特征**

千呼万唤始出来，犹抱琵琶半遮。 -- 《琵琶行》

虽贵为富二代，但 Android 要是没任何可圈点的地方，开不过 70 迈，在玲琅满目的手机平台竞争中，充其量也就做几个俯卧撑打一桶酱油，然后被落的远远的。说到底，出来混，靠的还是技术。

架构



从 SDK 文档中，偷来一幅 Android 平台的架构图，如上。在整个架构最底层红彤彤的部分，是 Linux Kernel 在移动平台的一个移植，它隐藏了硬件、网络等相关的细节，为上层提供了一个相对纯洁的统一接口。除非要做的是 Android 到不同设备的移植工作，否则对于大部分普通开发者而言，基本上是远观而不必亵玩的。Google 一直强调，Android 的底层实现异常 NB，可移植性超强，暂没有功夫研读，实属遗憾。

靠上一层，是一些核心的和扩展的类库，它们都是原生的 C++实现。在这一层，你可以看到很多熟悉的面孔，一如 SQLite、WebKit、OpenGL，开源的力量与贡献由此可见。

如果，该层类库需要被上层函数调用，就必须要通过 JNI 的导出相应的接口函数，否则就只能在层次内部自个把玩。

也是在这一层次上，还有为上层 Java 程序服务的运行时。Dalvik 虚拟机，是 Android 的 Java 虚拟机，之所以不采用 J2ME 的虚拟机，一方面是因为 J2ME 的设计是为了低端机器而优化，而 Dalvik 则是为了高端一些的机器进行优化，提供更好的性能。另一方面，从商业角度来看，必须绕开 J2ME 虚拟机，Android 才能彻底解放，想怎么开源就怎么开源，不再需要考虑 License 的问题。

再往上，终于有 Java 出没了。首先是框架层，这里包含所有开发所用的 SDK 类库，另外还有一些未公开接口的类库和实现，它们是整个 Android 平台核心机制的体现。

而在最上面，就是应用层了，系统的一些应用和第三方开发的所有应用都是位于这个层次上，也许要纠结两者的差别，就是系统应用会用一些隐藏的类，而第三方的应用，总是基于 SDK 提供的东西来搞。

一般来说，Android 开发，就是在 SDK 的基础上，吭哧吭哧用 Java 写应用。但自从有了 NDK，一切有了写小变化。NDK 的出现意味着，最上面应用层的内容，可以穿越 Java 部署的框架层，直接和底层暴露出来的，或者自行开发的 C++库直接对话，当然在这些库中需要包含 JNI 的接口。

人说，这就不是 Android 也可以用 C++开发应用么，但其实，这样的说法不够确切，纯 C++应用，是无法被接受的。因为在 Android 中，大量的核心机制部署在框架层，它们都是用 Java 实现的，比如控件库，Activity 的调度之类的。因此，没了界面，没了调度，还是只用 C++做类库比较合适，否则一切都乱了套了。

特征

基于这样的架构，Android 有很多的设计显得很有意思。纵览整个 SDK 和核心机制的设计，工整漂亮，是 Android 给人的第一感觉。为了说明这一点，找一个反面教材是很有必要的，Symbian 同学毫无悬念的担当这个伟岸的角色。

写 Symbian 程序，感觉就像是在玩一个猜谜游戏。哪怕你是一个 Symbian 老手，当需要用到 Symbian 中某块陌生功能的时候，你可能依然束手无策。你往往需要猜并反复找寻，在这里我需要使用哪一种奇巧淫技呢，是该臆想某些事件，还是应该用一个神秘的 UID 寻找某个特定应用，诸如此类。

而做 Android 应用的时候，就像是做高考模拟试题，题看上去不一样，解答模式摸清楚，就一通百通，一了百了。监听某个系统事件，查一下 SDK 就好；访问某个应用的数据，看看它有没有提供 Content Provider 就可以。所有的一切，都是按套路出牌，只要你了解了套路，再陌生的牌也可以看得懂，出的顺。人说武林高手，都应该是无招胜有招，而一个好的应用框架，也应该做到举重若轻，可触类旁通。

而 Android 框架最文采飞扬的一点，就是引入了 Mash-Up 的思想。所谓 Mash-Up，就是把写应用搞成搭积木，要出效果的时候，东家一块西家一块现场拼起来就好。这里面关键有两点，一个是模块化，另一个就是动态性。所谓模块化，就是一个应用的功能要明确的被封成一个个边界清晰的功能点，每一个功能点都像是一个黑盒，由预先定义的规则描述出其交互方式；而动态性，就是这些独立的模块能够在运行的时候，按照需求描述，连接在一起，共同完成某项更大的功能。在这两点上，Android 都做得非常出色。

站在可 Mash-Up 构造应用这一点去看其他的一些 Android 中的核心功能设计，就显得很有章可循了。比如为什么要把文件私有化，为什么要让进程被托管，等等（当然也可以站在别的角度看出不同的效果，视角不同，视野自然不同。..）。

在 UI 机制方面，Android 也有很不错的表现。它采取 xml 格式的资源文件，描述所有界面相关的内容。资源文件不是什么新东西了，xml 格式也是老调重弹，但可贵的是 Android 做的更为的丰富和彻底，基本把界面相关的逻辑，全部从代码中剥离到了资源文件中，和 Symbian 那四不像的资源文件相比，真是强大了不知多少倍了。

- **Android 学习入门**

不积跬步，无以至千里。 -- 《劝学》

说，万事开头难。想开始 Android 的开发，最重要的应该是先把马步扎稳，套路摸清楚，后面的事情就顺当多了。打开怀抱，拥抱 Android，也许可以先做下面这些事。

开发环境

辣手摧花成性的 GFW，无情的把 Android 开发者官网关在了墙外。不过没关系，猛击这里，同样可以殊途同归（Shit，一直在用代理，刚试了下，发现竟然也被盾了，如果不行，那就只能翻墙了。..）。

如果旅途顺利，你可以在路径 sdk/index.html 下找到安装说明，成功配置好 Android 的开发环境（【注】：在以后，如果要给开发者页面上的链接，都会给一个像 sdk/index.html 这样的相对路径，你可以在前面加上官网地址，或者本地 SDK 的 doc 地址拼凑成完整的路径，在一个盾牌横行的朝代，只能用这样委屈求全的方法保证能更好的使用。..）。

在 2.0 之前，每一次版本更新，你都要自己去下个全新的 SDK，然后按照说明，小心翼翼的一步步修改 eclipse 的设置，甚是麻烦。在 2.0 后，这个模式有所改善，你会先下到一个类似于下载器的插件，通过它可以来管理和升级 SDK，不仅简化了整个升级模式，还使得你可以更好的在各个不同的 SDK 版本间游走，利国利民。

Eclipse + ADT（Android Development Tool），是正牌的 Android 开发环境。你可以在 Windows，Linux，Mac 下做开发，甚为自由。比之 Symbian 的开发环境，ADT 显得尤为强大，它对 SDK 提供的一堆优秀的命令行工具进行了 UI 上的封装，提供了图形界面（命令行控当然同样幸福，具体参见：guide/developing/tools/index.html）。通过 ADT，你可以用运行和管理模拟器，使用调试器进行调试，过滤和查看 Log，浏览模拟器上的文件信息，模拟拨号、短信等手机才有的事件，等等。

文档

我知道，有很多人在学习一个新平台开发的时候，都习惯去买一些《xxx 21 天精通》之类的书籍。但其实，最好的入门学习资料，就是 SDK 文档。因为只有做平台的自己，才能最了解平台中的各个玄机，各方面的轻重缓急，从而能够更好的对症下药药到病除。

在 Android 的 SDK 中，guide/index.html 是由浅入深的教学文档，reference/packages.html 是标准的 API 文档。对于教学文档，我的意见是，一字不拉的通读一遍甚至多遍，至少做到能对 Android 摸着头脑，并且碰到问题的时候，能够快速想起在哪里可以找到，回来深刻阅读。

而 API 文档方面，Android 做的算是还不错了。基本上每个类，每个接口，都有标准而详尽的说明，在一些尤为重要的类中，还具有大量的学习性的内容，不和 Symbian 似的，

有太多的太监类，只有光秃秃的一个函数，一行文档说明都没有。整个文档结构是按照 Java 包来组织的，本身 Java 包命名的结构性和可读性很强，找起来也颇为方便。

很多人对 SDK 文档有抵触情绪，我想，有两方面的原因。一则是 SDK 文档普遍缺少文学性，麻木不仁的八股文，难以下咽。Android 在这方面做得算是乏善可陈，虽然算不上文采华丽，但还是挺适合阅读的。另一则，就会是语言方面的原因了。SDK 文档多为英语，偶尔像 MSDN 这样有中文的，也停留在机器翻译的水平上，阅读起来颇为难受。特地在网上搜了下，找到一些翻译 SDK 的中文文档，比如这里。虽然是基于 1.5 r1 版本 SDK 所著，稍显过时，但翻译的还是有小用心的，作为辅助，也不失为一份好资料，特代表广大看官向这些为人民福利着想的同志致敬。

Tutorials

光说不练假把式，除了读，在入门阶段，写也是一项不能少的运动。同样是在 SDK 中，Android 提供了一组 Tutorials 和一些列的 Samples，详见：resources/index.html。

Tutorials 很简单，Hello World 只是在教你如何在 eclipse 中，在 ADT 的帮助下，创建一个 Android 项目。相比之下，Hello Views 复杂了些，它集中展示了几种标准的 Android Layout 样式是如何构建的，很多时候，你都是在这些样式下扩展所需的 UI。

Hello Localization，是教你如何使用资源的，做完这个，就可以了解 Android 的资源有多杀～。最后收官的是一个更为完整的 Notepad Tutorial，它展示了很多 Android 的核心机制，比如基于 Intent 的 Activity 整合，Activity 的生命周期等。迈过这个 Tutorial，欢迎你，进入 Android 的大门。

当然，做完 Tutorials，对于 Android 而言，只是管中窥豹略见一斑。在 SDK 中，还提供了一系列的 Samples。可以根据自己的需求，挑选合适的 Sample 编译运行和学习。但其中，有一个是不论你做什么，都需要必看必读必熟悉的，就是 API Demos。在这个 Sample 中，集中展示了 Android 重点功能的 API 使用，把这个 Sample 用熟悉，需要做什么的时候过去找一下就可以很快的入手了。

源码

到这里，很多看官一定很不屑，前面所谓的学习入门介绍，只不过是围着 SDK 打转。其实，事实也是如此，SDK 中包含的内容是真的非常重要，我只是期望通过一些简短的介绍，激起一些初学者的重视，如是而已。

当然，SDK 每一个平台都有，没什么稀罕的。但 Android 有另一个非常稀罕而值钱的看家法宝，就是源代码。从 Android Source 的主站上：http://source.android.com/，你可以获得整个平台的源码以及相关介绍。非常苦口婆心的期望大家都去 down 一份源码放在机器上，哪怕你不需要进行修改编译，放在机器上当百科全书也是远胜于任何一本 Android 教学书籍的。本系列文章后续很多内容，都是从源码中学习到的一些浅薄见识。

对于大部分开发者都有学习价值的源码，主要在源码的 frameworks 和 packages 目录下。前者包含的是平台核心的一些实现，比如你需要自定义一个控件，也许你就可以翻到一个系统控件的实现中去，看看它是怎么来做的。后者包含一些系统的应用实现，比如你想做个播放器，也许你可以先去参考参考系统自带的是具体怎么做的。这样的实现，即便不算是最华美，至少也是最标准，其价值不容小视。

另外，你也可以把它当个代码库来使，不会使用某个类，grep 一把，也许就能获得一份最漂亮的 Sample。当然，如果你有时候对某些系统机制表示费解，抑或有一些 bug 不知道源头在哪，都可以跟着源码顺藤摸瓜的搞清楚。这样的好东西，可不是每个平台都能够享用的。

其他

论坛，其实对于开发和学习都是很重要的资源。毕竟，所有的资料都是死的，只有人是活得，能够最大限度的因地制宜解决问题。

只不过，标准的官方论坛，放在 Google Group 上，已经惆怅的被盾了。中文论坛方面，没有特别优秀和活跃的，这一方面是由于 Android 的发展现状还不算很磅礴，另一方面是由于 Android 的开发相对于 Symbian 而言，奇技淫巧少了很多，没有那么多好问的。也许你可以去去 csdn 这样的传统论坛，或者 eoe 这样专门的论坛。有的时候，还是多少能获得一些帮助的。

书籍方面，真没有什么推荐，豆瓣上搜索一下，你可以看到，目前的书籍，基本上还是集中在 SDK 使用层面上，很少有解析的很透彻，做的很深入的。而 SDK 的使用，看 SDK 的文档就足够了，如果实在对 e 文不感冒，买一两本评价不太差的中文书籍，放着翻翻也还是挺好。

更进一步，也许可以读读一些经验性的文档，去 Google Code 上搜索一些代码回来看看。比如，SDK 文档中，有个经验性文档的集合：resources/articles/index.html，就可以翻看一下。

最后，更多的一切还需要自己在工程和思考中，慢慢总结。相信，好的代码，会垂青一个勤于动手和思考的人。