HZ BOOKS
华章科技

Android 插件化技术的集大成之作。

以浅显易懂的图文讲解Android系统的底层知识，详细介绍DroidPlugin、DL、Small、Zeus框架的设计思想和代码实现，探究插件混淆、so插件化、HTML 5降级等诸多被遗忘的角落。

移动开发

包建强◎著

Guide to Android Plug-In Techniques

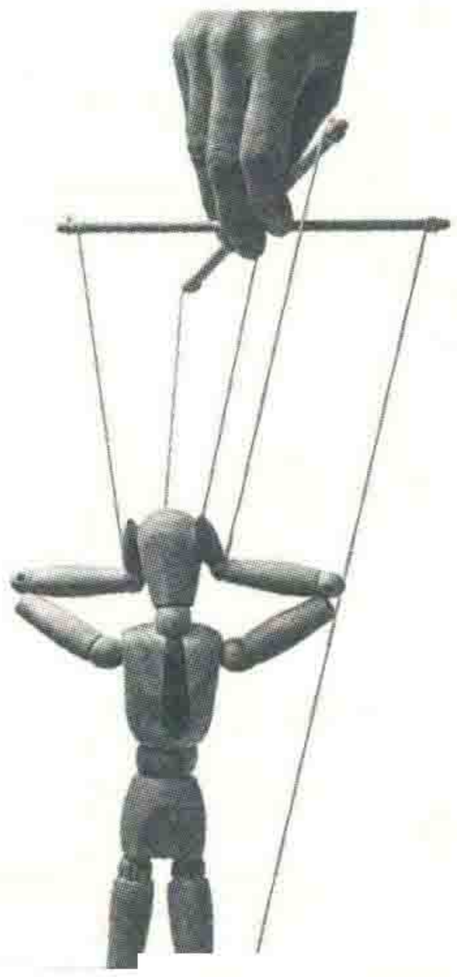
Android 插件化开发指南

机械工业出版社
China Machine Press

移动开发

Android 插件化开发指南

包建强◎著



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Android 插件化开发指南 / 包建强著 . —北京：机械工业出版社，2018.8
（移动开发）

ISBN 978-7-111-60336-8

I. A…　II. 包…　III. 移动终端 - 应用程序 - 程序设计 - 指南　IV. TN929.53-62

中国版本图书馆 CIP 数据核字（2018）第 145767 号

Android 插件化开发指南

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码：100037）
责任编辑：吴　怡　　责任校对：李秋荣
印　　刷：北京市兆成印刷有限责任公司　　版　　次：2018 年 8 月第 1 版第 1 次印刷
开　　本：186mm × 240mm　1/16　　印　　张：22
书　　号：ISBN 978-7-111-60336-8　　定　　价：79.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzit@hzbook.com

作者简介

包建强

毕业于复旦大学数学系。先后在多家互联网公司担任无线部门技术总监，现在从事区块链技术领域的研究，在Android、iOS、ReactNative等多门无线技术中跋涉过，在App的项目管理上也有多年的实践经验。他曾经出版了《App研发录》，并有一个坚持写了10年的技术博客：http://jax.cnblogs.com/，他的GitHub地址：https://github.com/BaoBaoJianqiang。

华章科技

HZBOOKS | Science & Technology

Preface 序 一

当接到包老师邀请写序时，我真是受宠若惊。大家都知道，作为一个程序员，写代码拿手那是自然的事情，文字工作实在不是我的强项，但是能给包老师的书写序，实在是荣幸之至，更何况盛情难却呢。

我与包老师相识是在 2015 年的一个关于 DroidPlugin 的分享会上，那会儿我还在 360 公司做手机助手。2014 年年底到 2015 年年初的时候，在公司比较空闲，所以写了一个插件化框架叫 DroidPlugin，并在 2015 年 7 月份以公司的名义在 GitHub 上开源了出来，承蒙大家抬爱，在极短的时间内便收获了数千颗星。因为在项目的介绍中我写上了“免修改、免重打包、免安装运行”的宣传语，所以也被一些人冠以“360 黑科技”，在知乎上甚至有人认为这是 360 的什么阴谋。不管是不是什么阴谋，但这个项目让我认识了很多行业中的大牛（包老师就是其中一位），确实是我意料之外的事情。后来跟包老师、任玉刚老师几位经常结成“饭醉团伙”，自然也少不了讨论插件化技术。既然与包老师是因为 DroidPlugin 开源项目相识，而包老师这本书的内容涉及的很多技术也跟其相关，所以我还是主要介绍一下 DroidPlugin 项目中的相关技术。

如你所见，那会儿正是插件化技术大热的时候，各大公司也相继开源了自己的插件化框架，但是总结来看，所涉及的技术原理也大同小异。但是 DroidPlugin 在其中的确算是比较“奇葩”的一个，因为它实际不止是一个插件化框架，更多的算是一个用户态虚拟机，后来大多数的双开软件也都参考了它的源码或者原理。

要在 Android 上实现免安装、免修改运行一个 App，并不是一件容易的事情。因为 Android 系统设计和权限的限制，我们需要做很多的工作……这跟 Docker 不一样，虽然都是 Linux 系统，但在 Android 上我们不能要求 root 权限，而 Docker 则没有这些限制，因此 Linux 内核提供的 namespace、cgroup、chroot 等能力也自然可以应用在 Docker 中。同时因为国内各大厂商的定制系统，我们也做了大量的适配工作。虽然现在看起来这个框架还有不少缺陷，但在当时确实算是比较先进的一个框架。

DroidPlugin 使用了一些比较 hack 的技巧，但是总结起来也就是一句话“利用 hook 技

术实现欺上瞒下，从而达到免安装运行的目的”。因为 Android 系统出于安全考虑，系统服务与 App 进程采用分进程设计，它们之间通讯使用 binder 技术，系统服务实际上是不知道 App 进程中运行的具体代码，这为我们实现“欺上瞒下”提供了可能。所谓“欺上瞒下”中的“欺上”是指我们可以通过某种技术手段，拦截所有插件发向系统服务的通信，让系统不知道插件在我们的宿主 App 中运行；“瞒下”是指通过拦截并模拟系统服务发向插件的通信，让插件“以为”自己已经被安装。这样我们就可以模拟一个环境，让插件运行在宿主的模拟进程中。

要做到欺上瞒下，hook 技术不可缺。hook 这个词是我从 Window 安全技术中借用而来的，这实际上是一种函数拦截技术。在某个函数调用流程中插入我们自己的函数，实现对目标函数的参数、返回值的修改。比如某个函数调用流程为：a 调用 b、b 调用 c，那么我们可以动态拦截对 b 函数的调用，插入我们自己的函数 h，修改后的调用流程为：a 调用 h，h 调用 b，b 调用 c，因为 h 是我们新插入的函数，由我们自己编写逻辑，那么在 h 函数中调用 b 的时候，我们就可以修改其参数、返回值，甚至可以中断调用流程，不调用 b 函数而伪造一个返回结果给 a。在此过程中，a 是不知道它调用的 b 函数已经被我们修改了。这就达到了我们“欺上瞒下”的目的。

hook 技术的思想非常类似于设计模式中的代理模式和 Java Spring 框架中的 AOP（Aspect Oriented Programming，面向切面编程）。尽管它们的实现原理完全不同，但目的却差不多。DroidPlugin 中的 Hook 技术实际上是使用了 Java 中的动态代理技术，它只是在一些关键点通过动态代理生成的对象替换掉系统原来的对象，从而完成了对系统通信的 hook。

这其中最关键的是对 AMS（Activity Manage Service）和 PMS(Package Manage Service）以 及 Handler 的 hook。AMS 负 责 管 理 Android 中 Activity、Service、Content Provider、Broadcast 四大组件的生命周期以及各种调度工作，我们 hook 它可以实现对插件四大组件的管理及调度；PMS 负责管理系统中安装的所有 App，我们 hook 它是为了让插件以为自己已经被安装；Handler 是系统向插件传递消息的一个桥梁，我们 hook 它则是为了把系统发向宿主的消息转发给插件。至于在 DroidPlugin 中看到的对其他系统服务的 hook，在大多数情况下都是为了“欺上”——让系统感知不到插件的运行。DroidPlugin 还实现了一个简单的 AMS 服务、一个 PMS 服务，将 hook 后的 AMS 和 PMS 系统调用转发到我们自己的 AMS 和 PMS 服务中去，由 DroidPlugin 自己管理。

除此之外，DroidPlugin 的另外一个特色是“占位”操作。众所周知，在 Android 四大组件中除了 BroadcastReceiver 之外，其他如 Activity 三大组件都需要在 AndroidManifest.xml 中注册，这是静态的，是 Android 框架要求我们预先写死的，我们没有办法动态地向系统注册一个 Activity 或者是 Service。所以插件中的 Activity、Service、Content Provider 则是不可能向系统真正注册。所以我们使用了“占位”的技术，也就是说先在宿主中注册很多的“坑位”，比如对于 Activity 来说，就是 Activity1、Activity2、Activity3 等。在我们需要启动某插件 Activity 的时候，可以通过 hook 技术，将其替换为某个坑位，如 Activity1，

让系统服务去启动坑位 Activity；而在真正的系统服务 AMS 回调插件进程要求插件进程去启动坑位 Activity 的时候，我们再换回插件的 Activity，这样我们就实现了插件 Activity 的免注册运行。当然，因为 Activity 的 Launch Mode 等各种参数问题，我们还需要做很多的细节工作，才能完美。

为了某种完美，DroidPlugin 中做了大量的适配工作，这让其初看起来复杂又臃肿，但是当了解到其中的原理和关键代码后，你会发现如果仅仅是满足“插件化”需求的话，那么其中很多适配并不必需。从现在来看，DroidPlugin 实际上算是一个用户态虚拟机的雏形，从插件化的角度来说则有些“重”了。但是它对于大家深入研究 Android 技术本身，则或多或少会有些帮助。

现在市面上也有各种各样的开源插件化框架，其中很多都已经在各大公司自己的产品中长期稳定使用，满足了各种现实的需求，它们的稳定性、可用性都还是不错的。包老师在本书中也对其中很多插件进行了介绍剖析。

如果我们不满足于业务研发，希望可以了解一些 Android 底层知识，研读这些开源框架的源码则大有裨益。当然，包老师的这本书是国内第一本介绍插件化技术的书籍，作为我们学习插件化技术的入门书籍，则相当合适。

张勇，2018 年 6 月于北京

序　二 *Preface*

很荣幸能为本书写下三言两语。

我和插件化技术之间有着难解的情缘，到目前为止我已经工作好几年了，如果简化一下，就是下面这个样子：

1）开源 dynamic-load-apk 插件化框架（业余时间）。

2）开发百度手机卫士 & 出版《Android 开发艺术探索》(百度)。

3）开源 VirtualAPK 插件化框架（滴滴出行）。

这么一看，自从我工作以来，有一半时间都在从事插件化相关的开发工作。我喜欢 Android，也喜欢插件化。最开始我对插件化只是兴趣使然，在工作之余我喜欢做一些研究，所以有了 dynamic-load-apk。到后面我加入滴滴出行则是使命使然，在我的内心深处，我觉得 dynamic-load-apk 不够完美，我想开发一款完美的插件化框架，于是有了后面的 VirtualAPK。回想起插件化的发展，也就仿佛看到了一路走来的自己。

2014 年 3 月 30 号，我在 CSDN 上发布了一篇文章《Android apk 动态加载机制的研究》。这篇文章现在看起来很傻，技术也很落后，但是很多人都无法感受当时的情形。在 2014 年年初，别说插件化知识了，就连高质量的 Android 技术文章都比较匮乏。

说起高质量的 Android 技术文章，大家可能会想到：可以看《第一行代码 Android》和《Android 开发艺术探索》呀！但是很遗憾，那个时候这两本书都还没出版，不止是这两本书，很多大家所熟知的书都没有出版。当时 Android 技术圈沉醉于下拉刷新、侧滑菜单等这种炫酷特效，对于 AIDL 和 View 原理不曾研究过，你要问插件化？ 90% 的 Android 工程师都不知道这是个什么东西。除了技术文章和书籍比较匮乏以外，开源也比较匮乏。在 2014 年，插件化技术只是一个概念，虽然当时阿里已经有了 Atlas，但是并没有开源，所以在这种情形下，我发的那篇文章就显得很专业了，当时引起了技术圈的广泛关注，获得了 7 万多的阅读量。

在 2014 年下半年，我和田啸、宋思宇等同学发起了 dynamic-load-apk 这个开源项目，现在大家都知道了，dynamic-load-apk 在插件化历史中有着浓厚的一笔。dynamic-load-apk

支持动态加载代码和资源，资源访问可以直接通过 R 来进行，在四大组件方面支持 Activity 和 Service。虽然说 dynamic-load-apk 谈不上多完善，但是业内却有不少公司基于 dynamic-load-apk 进行二次开发来定制自己的插件化框架，从这个角度来说 dynamic-load-apk 是很成功的一款开源框架。

到 2016 年，插件化框架才真正迎来了大繁荣时代，可谓百家争鸣。不管是 Atlas、Small、DroidPlugin 还是携程的 DynamicAPK，都极大地促进了 Android 插件化框架的发展。我也是在 2016 年年初离开百度，来到了现在的滴滴出行。如果说在百度的工作是做应用开发的话，那么在滴滴出行的工作就完全是热修复和插件化开发了。经过长时间的开发和验证，滴滴出行在 2017 年 6 月 30 日开源了一个更为完善的插件化框架 VirtualAPK，而我则在这个框架的开源中发挥了至关重要的作用。

如果给我这几年的职业生涯写一个总结，那就是：两款 Android 插件化框架 + 一个 App+ 一本书，而我还将在 Android 的道路上继续耕耘。

任玉刚，2018 年 6 月于北京

序　三 *Preface*

听说包猪猪的第二本书要出版了，很为他高兴，作为一个旁观者，眼见着本书由一个想法萌芽逐渐充实，颇有感触。我脑海里就像过电影一般，浮现这几个月中包猪猪的种种状态：为解决一个 bug 而连续工作十几个小时的苦闷，第二天一早起来灵感触发迎刃而解的喜悦，一边照顾父母一边因书的进度被某一难题阻滞而焦灼……身边的人会跟他说："事情这么多，歇歇再写吧，别让自己太辛苦。"可是一个想快点跟业界分享自己想法和成果的程序员又怎会因琐事耽搁？曾是微软 MVP 的他，入行十几年，仍然秉承着刚入行时的激情与热忱，吭哧吭哧地调 bug 到凌晨，这本书或许是这个"内心有团火"的家伙希望送给读者最好的礼物——智慧的分享与教学相长的领悟。

我是学中文和法律的，作为圈外人在本书前作序，多少有班门弄斧之嫌。如论代码，各位读者在自己的领域各有所长。不过我这种对技术一窍不通的人，竟在看一本技术书时捧腹大笑，足见本书的吸引力。比如说，书中调侃张勇和任玉刚是插件化领域的男乔峰女慕容，以至于我一直想亲眼见一下这两个人。比如说他在前言中拿娃娃开涮，连着感谢了霹雳娇娃、赵越和 Dinosaur，殊不知那却是一个人，可能是为了看上去人多一些，以壮声势。

程序员的世界我不太懂，尤其是包猪猪，他经常做出一些令人啼笑皆非的事情。比如说，情人节他第一次给我送花，却阴差阳错地寄来了两束。晚上的时候他跟我说，两束花别浪费了，另一束花就快递给邓凡平吧——据说那也是 Android 行业内的一位大神，他因为伺候老婆坐月子而忘记准备情人节礼物了。于是，作为情人节只收别人礼物的我，在这一天，第一次给别人送礼物。

他做饭很好吃，有很多招牌菜。用他的话讲，炒菜是设计模式中的装饰器模式。比如说西红柿炒鸡蛋，放锅里炒了几分钟后，加点糖，就是味道甜甜的西红柿炒鸡蛋，再加些盐，就是酸甜可口的西红柿炒鸡蛋，也许还会再加些其他调料，但这道菜永远都是西红柿炒鸡蛋，只是味道不同罢了。技术做到这一步已经接近于完美，但他后面的奇葩行为，却颠覆了我对他的认知。

他研究做鱼，第一天没做好，第二天再买条鱼继续做，直到他认为完美。把钻研计算

机技术的执着用于烹饪，结果就是我一连吃了 5 天鱼汤，上火，满嘴都是泡。

第一次见包猪猪是在酒吧，一边听着不知道是哪里的古老而嘈杂的乐队嘶吼，另一边是他给我讲关于五个海盗分赃的小故事。隐约记得故事是这样的："5 个海盗抢到 100 个金币，他们决定依次由 A,B,C,D,E 五个海盗来分，他们订立了如下规则：当由 A 提分配方案时，剩下的海盗表决，如果 B,C,D,E 四人中有一半以上反对就把 A 扔下海，再由 B 分……如是这般，那么 A 海盗如何分，才能既保住性命又能获得更多的金币。"这是个有意思的小故事，轻松而又暗藏思维逻辑，我们可能有很多种解决方案，但是最优方案最后一定是"博弈与制衡"的平衡。此前，我一直以为程序员的世界充斥着代码，那是一套拥有独立计算机语言的系统，难以接近跟理解。可是包猪猪有种能力将难以理解或者比较复杂的事情用一种有趣的方式表达出来，诙谐有趣、通俗易懂又能在嘻嘻哈哈的氛围中有所感悟。

本书讲插件化，从插件化的历史讲起，说了不少这行的人跟事，还有八卦。接下来由基础知识开始讲起，后来又介绍了插件化解决方案及周边技术。文字所限，本书内容结构不赘述了，各位可以凭目录了解。代码方面我虽然读不懂，不过并不影响看书的心情，这是本书颇为神奇的地方。没有高深莫测的理论，没有艰深难懂的词汇，就像包猪猪站在我面前娓娓道来，举一些他觉得有意思的例子，让你觉得调皮又生动。对于对插件化不熟悉的读者，可能本书提及的有些词汇是新的，不太容易理解，感谢本书的编辑在成书过程中从旁指引，因而本书在前言部分增加了名词解释，并在各个章节多加了一些描述性的文字。

关于本书前言所提及未展开描述的部分，其实作者在写作前已经预留了位置，但是在成书前两天犹豫再三还是有所删减，因为总觉得讲得不透、不彻底是不好意思呈现给读者看的。因此，我跟很多读者一样很期待包猪猪过段时间能将那些本书没说透的东西再写本书好好讲一讲。比如说 Small，他半夜说梦话时经常念叨这个词。

谨以此书献给奋斗在一线的程序员们，作为家属深刻了解程序员的辛酸，希望本书对读者能够有所启发，运用到工作中可以提高效率，多一些休息时间陪伴家人、朋友，少一些熬夜加班、拼命赶工。

郭曼云，2018 年 6 月于北京

前　言 Preface

这是一本什么书

如果只把本书当作纯粹介绍 Android 插件化技术的书那就错了。本书在研究 Android 插件化之余，还详细介绍了 Android 系统的底层知识，包括 Binder 和 AIDL 的原理、四大组件的原理、App 的安装和启动流程、Context 和 ClassLoader 的家族史。没有罗列大量的 Android 系统中的源码，而是以一张张 UML 图把这些知识串起来。

本书详细介绍了 Android 中的资源机制，包括 aapt 命令的原理、resource 文件的组成以及 public.xml 的使用方式，顺带还提及了如何自定义一个 Gradle 插件化。

此外，本书还介绍了 so 的加载原理，尤其是动态加载 so 的技术，可以帮助 App 进行瘦身；探讨了 HTML5 降级技术，可以实现任何一个原生页面和 HTML5 页面的互换；介绍了反射技术，以及 jOOR 这个有趣的开源框架；介绍了 Android 中的动态代理技术 Proxy.newProxyInstance 方法。

如果读者能坚持把这本书从头到尾读完，那么不仅掌握了插件化技术，而且也把上述所有这些知识点全都系统地学习了一遍。也许 Android 插件化会随着 Google 的限制而有所变化甚至消亡，但我在本书中介绍的其他知识，仍然是大有用武之处的。

如何面对 Android P 的限制

写作这本书的时候，Google 推出了 Android P preview 的操作系统，会限制对 @hide api 的反射调用。目前会通过 log 发出警告，用户代码仍然能够获取到正确的 Method 或 Field，在后续版本中获取到的 Method 或 Field 极有可能为空。

但是道高一尺，魔高一丈。Google 对这次限制，很快就被技术极客们绕过去了[⊖]，有两种解决方法：

⊖ 详细内容，请参见田维术的文章：http://weishu.me/2018/06/07/free-reflection-above-android-p/

1）把通过反射调用的系统内部类改为直接调用。具体操作办法是，在 Android 项目中新建一个库，把要反射的类的方法和字段复制一份到这个库中，App 对这个库的引用关系设置为 provided。那么我们就可以在 App 中直接调用这个类和方法，同时，在编译的时候，又不会把这些类包含到 apk 中。

其实早在 2015 年，hoxkx 就在他的插件化框架中实现了这种技术[⊖]。但是这种解决方案，仅限于 Android 系统中标记为 public 的方法和字段，对于 protected 和 private 就无能为力了。比如 AssetsManager 的 addAssetPath 方法，ActivityThread 的 currentActivityThread 方法。

2）类的每个方法和字段都有一个标记，表明它是不是 hide 类型的。我们只要在 jni 层，把这个标记改为不是 hide 的，就可以绕过检查了。

然而，魔高一丈，道高一丈二。Google 在 Android P 的正式版中势必会推出更严厉的限制方案，到时候，又会有新的解决方案面世，让我们拭目以待。

其实，开发者是无意和 Google 进行技术对抗的，这是毫无意义的。泛滥成灾的修改导致了 App 大量的崩溃，Google 实在看不下去了，所以才搞出这套限制方案；另一方面，插件化技术是刚需，尤其在中国的互联网行业，App 崩溃会直接影响使用，很可能导致经济损失，所以开发者才会不惜一切代价走插件化这条路。

再回到限制方案来，Google 也不是清一色不要开发者使用系统底层的标记为 hide 的 API，而是推出了一组黑灰名单，如下所示：

名　单	影　响
light-greylist 浅灰名单	仅打印警告日志，Google 尽可能在未来版本提供 public API
dark-greylist 深灰名单	第三方 App 不能访问，开发者可以申请把这份清单中的某些 API 加入到浅灰名单
blacklist 黑名单	第三方 App 不能访问

所以，另一种应对策略是，在插件化中使用浅灰名单中的 API，比如说 ActivityThread 的 currentActivityThread 方法。

Google 的这组清单还在持续调整中，据我所知，给各大手机厂商的清单与其在社区中发布的清单略有出入。在 Android P 的正式版本中，这份清单会最终确定下来。所以现在中国的各个插件化框架的开发人员，都在等 Android P 的正式版本发布后再制定相应的策略。留给中国队的时间不多了。

这本书的来龙去脉

这是一本酝酿了 3 年的书。

早在 2015 年 Android 插件化技术百家争鸣时，我就看好这个技术，想写一本书介绍这

⊖ 项目地址参见：https://github.com/houkx/android-pluginmgr

个技术，但当时的积累还不够。那年，我在一场技术大会上发表了《Android 插件化从入门到放弃》演讲，四十五分钟介绍了插件化技术的皮毛。后来这个演讲内容被整理成文章发布到网上，流传很广。

2017 年 1 月，有企业要我去讲 2 天 Android 插件化技术。为此，我花了一个月时间，准备了四十多个例子。这是我第一次系统地积累了素材。

2017 年 6 月，我在腾讯课堂做 Android 线上培训，为了宣传推广我的课程，我写了一系列文章《写给 Android App 开发人员看的 Android 底层知识》，共 8 篇，没列太多代码，完全以 UML 图的方式向读者普及 Binder、AIDL、四大组件、AMS、PMS 的知识。本书的第 2 章就是在这 8 篇文章的基础之上进行扩充的。

2018 年 1 月，我父亲住院一周。我当时在医院每天晚上值班。老爷子半夜打呼噜，吵得我睡不着，事后我才知道，我睡着了打呼噜声音比他还大。半夜睡不着时就开始了本书的写作，每晚坚持写到凌晨两三点。直到父亲出院，这本书写了将近五分之一。

碰巧的是，这一年 5 月底我结婚，促使我想在 5 月初完成这本书的一稿，为此，我宅在家里整整写了 3 个月。仅以此书作为新婚礼物献给我亲爱的老婆，感谢你的理解，这本书才得以面世。

这两年我在忙什么

2016 年 5 月写完《App 研发录》后不久，我就从一线互联网公司出来，开始了长达两年的 App 技术培训工作。一改之前十几年在办公室闷头研究技术的工作方式，开始在全中国飞来飞去，给各大国企、传统公司、手机商的 Android 和 iOS 团队进行培训。这两年去过了近百家公司，有了很多与过去十几年不一样的感受。

App 开发人员的分布也呈金字塔型，在金字塔尖的自然是那些一线互联网的开发人员，他们掌握 Android 和 iOS 最先进的技术，比如组件化、插件化等技术，但这些人毕竟是少数。而位于金字塔底端的开发人员则是大多数，他们大都位于创业公司或者传统行业，相应的 App 侧重于业务的实现，对 App 的高端技术，用得不多，需要不断补充新知识。另外，我在腾讯课堂讲了几个月 App 开发课程的过程中，认识了很多学员，有几千粉丝，同样面临需要不断学习新知识。

写作这本书的目的，是向广大 Android 开发人员普及插件化技术。

这本书里讲些什么

战战兢兢写下这本书，有十几万字，仍不能覆盖 Android 插件化的所有技术。因为插件化技术千头万绪，流派众多，我想从最基本的原理讲起，配合大量的例子，希望能帮助完全不懂 Android 插件化技术的小白，升级为一个精通这类技术的高手。

面对业内各种成熟的插件化框架，我只选取了具有代表意义的 DroidPlugin、DL、Small 和 Zeus 进行介绍，这几个框架基本覆盖了插件化编程的所有思想，而且非常简单，像 Zeus 只有 11 个类，就支撑起掌阅 App 的插件化。而对于后期推出的 VirtualApk、Atlas、Replugin 等，在本书中并没有介绍，主要是因为这些框架都是大块头，代码量很多，我没有精力再去研究和学习了。但这些企业级插件化框架所用的技术，本书都有涉及。

本书的结构及内容

全书分为三大部分，共 22 章。第 1 部分“预备知识”包括第 1～5 章，是进行 Android 插件化编程的准备知识。第 2 部分“解决方案”包括第 6～16 章，详细介绍并分析了插件化编程的各种解决方案。第 3 部分“相关技术”包括第 17～21 章，介绍插件化编程的周边技术，并对纷繁复杂的插件化技术进行了总结。

第 1 章介绍的是 Android 插件化的历史，可以当作小说来读，茶余饭后，地铁站中就可以读完。

第 2 章介绍 Android 底层知识，涉及那些与 Android 插件化相关的知识，比如 Binder 和 AIDL，Android App 的安装流程和启动流程，ActivityThread，LoadedApk，Android 四大组件的运行原理。这一章篇幅较多，需要仔细研读。其中，讲到一个音乐播放器的例子，帮助大家更加深刻地认识 Android 的四大组件。

第 3 章讲反射，详细介绍了构造函数、方法、字段、泛型的反射语法。这章介绍了 Java 领域很火的一个开源库 jOOR，可惜，它对 Android 的支持并不是很好，所以这章还介绍了我们自己封装的 RefInvoke 类，这个类将贯穿本书，基本上所有源码例子都会使用到它。

第 4 章讲代理模式。这个模式在 Android 中最著名的实现就是 Proxy.newProxyInstance 方法。基于此，我们 Hook 了 AMS 和 PMS 中的一些方法。

第 5 章是第 4 章的延续，仍然是基于 Proxy.newProxyInstance 方法，Hook 了 Activity 的启动流程，从而可以启动一个没有在 AndroidManifest 中声明的 Activity，这是插件化的核心技术之一。

第 6 章介绍了如何加载插件 App，以及如何对插件化项目的宿主 App 和插件 App 同时进行调试。说到插件化编程，离不开面向接口编程的思想，这章也花了很多笔墨介绍这个思想，以及具体的代码实现。

第 7 章介绍了资源的加载机制，包括 AssetManager 和 Resources，并给出了资源的插件化解决方案，从而为 Activity 的插件化铺平了道路。另外还介绍了换肤技术的插件化实现。

第 8 章介绍了最简单的插件化解决方案，通过在宿主 App 的 AndroidManifest 中事先声明插件中的四大组件。为了能让宿主 App 随意加载插件的类，这章介绍了合并 dex 的技术方案。

第 9 章到第 12 章介绍了 Android 四大组件的插件化解决方案。四大组件的生命周期各不相同，所以它们各自的插件化解决方案也都不同。

第 13 章、第 14 章介绍了 Android 插件化的静态代理的解决方案。这是一种“牵线木偶”的思想，我们不用 Hook 太多 Android 系统底层的代码。

第 15 章再次讲到资源，这次要解决的是宿主和多个插件的资源 id 值冲突的问题。这一章介绍了多种解决方案，包括思路分析、代码示例。

第 16 章介绍一种古老的插件化解决方案，通过动态替换 Fragment 的方式。

第 17 章介绍了 App 的降级解决方案。一旦插件化方案不可用，那么我们仍然可以使用 H5，来替换任何一个 App 原生页面。

第 18 章介绍了插件的混淆技术。有时候宿主 App 和插件 App 都会引用 MyPluginLibrary 这个类库，这个公用类库是否要混淆，相应有两种不同的混淆方案。

第 19 章介绍了增量更新技术。这是插件化必备的技术，从而保证插件的升级，不需要从服务器下载太大的包。

第 20 章介绍了 so 的插件化解决方案。这章详细介绍了 so 的加载原理，以及从服务器动态加载 so 的方案，基于此，有两种 so 的插件化解决方案。

第 21 章介绍了 gradle-small 这个自定义 Gradle 插件。这章是对第 15 章的补充，是另一种解决插件资源 id 冲突的方案。

第 22 章作为整本书的结尾，系统总结了 Android 插件化的各种解决方案。如果读者能坚持读到这最后一章，可以帮助读者巩固这些知识。

关于本书名词解释

本书有很多专业术语，刚接触 Android 插件化的读者可能不容易理解。有一些专业术语还有别称或者简称，我在这里罗列出最常见的一些术语：

- HostApp，本书中有时也写作“宿主 App”。用于承载各种插件 App，是最终发版的 App。我们从 Android 市场上下载的都是 HostApp。
- Plugin，本书中有时也写作“插件”、“插件 App”。
- Receiver，是 BroadcastReceiver 的简称，Android 四大组件之一。
- AndroidManifest，也就是 AndroidManifest.xml。
- Hook，就是使用反射修改 Android 系统底层的方法和字段。
- AMS，是 ActivityManagerService 的简称，在 App 运行时和四大组件进行通信。
- PMS，是 PackageManagerService 的简称，用于 App 的安装和解析。

关于本书的源码

本书精心挑选了 70 多个例子，都可以直接下载使用，正文中都列出了代码名称，在相应网站可以找到。附录 B 还列出了所有源码对应的章节。