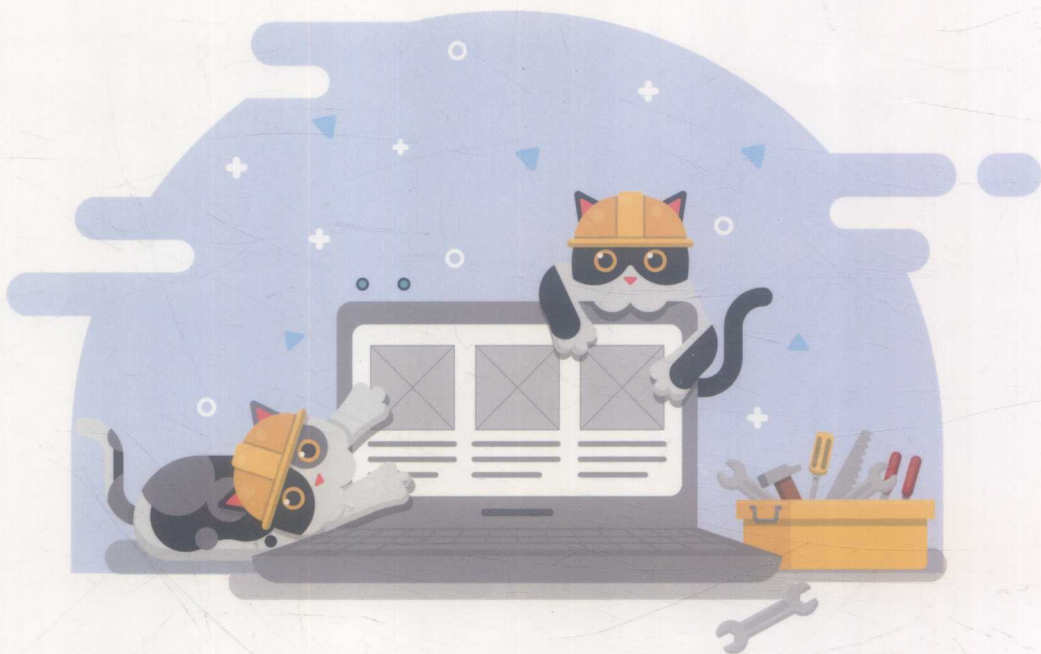


本书从实际工程问题入手，深入剖析源码！

从实践中逐步得出解决方案，是Android工程师
从量变到质变的随身锦囊！

Broadview®
www.broadview.com.cn



Android 工程化最佳实践

金凯 / 著



中国工信出版集团



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
<http://www.phei.com.cn>

Android 工程化最佳实践

金凯 / 著

电子工业出版社

Publishing House of Electronics Industry

北京•BEIJING

内 容 简 介

本书从工程实践角度详细阐述了 Android 的知识内容,全书分为基础知识和工程优化两部分。在工程优化部分专门增加了常用的 App 编译提速和瘦身的内容,对于大型分层项目的测试技巧也有所涉及。

本书涵盖 Android 开发的实际业务知识,涉及 Dialog、Intent、Fragment 等源码的核心细节分析,并扩展了一部分框架设计的内容,章节最后总结了开箱即用的开源库方案,实现从理论到实际的完整论述。最后还给出了抓包工具的使用技巧,帮助读者能方便地寻找到适合自己的工具集。

本书适合中、高级 Android 程序员阅读,也可以作为初级程序员进阶学习的参考书。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有,侵权必究。

图书在版编目(CIP)数据

Android 工程化最佳实践 / 金凯著. —北京: 电子工业出版社, 2019.3

ISBN 978-7-121-35924-8

I. ①A… II. ①金… III. ①移动终端—应用程序—程序设计 IV. ①TN929.53

中国版本图书馆 CIP 数据核字(2019)第 011494 号

策划编辑: 陈晓猛

责任编辑: 宋亚东

印 刷: 山东华立印务有限公司

装 订: 山东华立印务有限公司

出版发行: 电子工业出版社

北京市海淀区万寿路 173 信箱 邮编: 100036

开 本: 787×980 1/16 印张: 25 字数: 623 千字

版 次: 2019 年 3 月第 1 版

印 次: 2019 年 3 月第 1 次印刷

定 价: 89.00 元

凡所购买电子工业出版社图书有缺损问题, 请向购买书店调换。若书店售缺, 请与本社发行部联系, 联系及邮购电话: (010) 88254888, 88258888。

质量投诉请发邮件至 zltz@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式: (010) 51260888-819, faq@phei.com.cn。

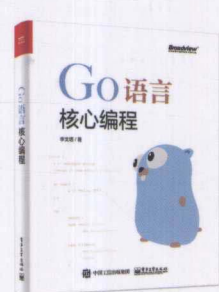
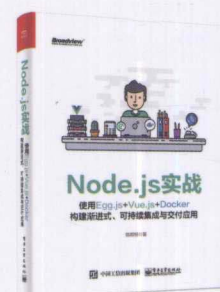
作者简介



金 凯

爱奇艺高级工程师。致力于摆脱程序思维，寻找各种简易的手段来方便程序的开发工作。目前的研究方向为Android业务基础框架和AI相关方面。

/ 好书分享 /



拒绝堆砌臃肿, 支持纯正原创

欢迎投稿: chenxm@phei.com.cn

前言

开发者的焦虑

当今，Android 和 iOS 手机无论在系统还是外形上都趋于雷同，App 功能的差异也是微乎其微，移动端发展趋于缓慢已成为开发者的共识。移动开发者面对小程序、React Native、Flutter 和 Weex 跨平台方案都显得焦躁不安，对于自己未来需要发展和生根的方向或多或少地会感到迷茫。

从市场的角度来说，公司缺少的并非基础的开发人员，更希望可以招聘到有大量实践经验的开发者。要进阶成为高级开发者，更需要阅读有深度、偏重实践的书籍。因此，我从 2015 年 8 月开始在 GitHub 上创建了“Android-Best-Practices”项目。这个项目的目的是将经实际验证的经验分享出来，力求达到“学完即用”的效果。在收到出版社的邀请后，我将相关的文章进行重写，同时新增大量内容，目的是让更多的开发者可以在短时间内得到更多的实际经验。

面对当前移动开发的生存环境，开发者的无力感会越发严重。作为移动应用开发者，应该了解与本职工作相关的知识，同时多涉猎其他邻域。如果你想快速巩固移动开发的知识后再投身到大前端中去，那么这本书很适合你。当阅读完本书的每一章后，你收获的是已经经过时间检验的知识，需要做的仅仅是去实践它、理解它。

写作风格

本书不会像其他书籍一样提供完整的代码下载，也不会花大量的篇幅分析源码。官方每隔几个月就会对源码的实现进行各种修改，而源码的设计思路则较为稳定。“授人以鱼不如授人以渔”，本书会将分析源码的过程提炼出来，主要阐述设计思路和关键的代码，甚至会为了可读性删除没必要的代码逻辑。这样做的目的是希望读者既可以快速理解源码的思路，又不会陷入代码的困境中。

本书内容

本书分为基础知识和工程优化两部分。基础知识部分讲述基础知识，读者可以找到合适的第三方库来提高业务开发速度；工程优化部分讲述工程实践方面的内容，仔细阅读此部分，可以提升自身内在的知识广度。

基础知识

在实际工程中，Android 本身的 API 大多不会被直接使用，需要做二次封装和扩展。基础知识部分将基础的知识提升到一个实践的层面，不再讲述如何使用它们，而是讲如何通过封装这些 API 解决实际中的各种问题。

第 1 章 Java 的反射是必知必会的内容，我们需要知道：

- 什么时候该用反射，Android 中的反射和 JDK 中的反射有何不同？
- 反射低效的原因是什么，如何让其变得高效。反射的缓存该如何实现？
- 为什么 Android P 不让反射被@hide 标记的 API，其影响是什么？
- 做一个好的反射封装，应注意什么问题，这种封装的意义是什么？

第 2 章 Log 是最简单的代码了，这里要讲的是日志的高级知识：

- 打印日志的时机是什么，什么日志应该留在 Release 版本中，这么做是否安全？
- 如何区分不同项目组的日志，如何避免日志间的相互干扰？
- 如何优化和封装日志，为什么说日志的分发功能是必须具备的？
- 微信是如何用 MMAP 提升日志的写盘成功率的，Xlog 的设计目的是什么？

第 3 章 Intent 是非常简单的对象，着重来讲：

- 如何封装和扩展 Intent，为什么它只能存储基本对象，Intent 的本质是什么？
- 如何在使用 Intent 时避免管理 Intent 的 Key，动态下发 Intent 的思路是什么？
- Intent 的最大数据存放量是多少，该如何传递一个大对象？
- Intent 传递 map 对象时会有什么隐患，linkedHashMap 可以直接存入 Intent 吗？

第 4 章 SharedPreferences 作为基本的存储对象，需要了解如下知识点：

- 用 SharedPreferences 存储 Java 对象有什么问题，如何清理其缓存？
- 为什么拆分 SharedPreferences 后可以有效地减少内存的消耗？
- SharedPreferences 的线程、进程安全吗，它是如何引起 ANR 的？
- 如果将 SharedPreferences 中的数据放云端，该如何设计这种云同步的功能？

第 5 章 Fragment 是开发者很容易出错的类，这里需要思考：

- Fragment 和 Activity 的生命周期是依次执行的吗？动态添加时的生命周期会如何执行？
- Square 不建议用 Fragment，但不用 Fragment 的话，该如何模块化地管理 UI 呢？
- Google 公司提出的 Lifecycle-Components 和 Fragment 有关系吗？使用的是什么原理？
- Fragment 的嵌套和回退栈是一个难点，Jetpack 中的 Navigation 是怎么简化它的？

工程优化

工具是开发者的武器，拥有一个好的武器可以战无不胜。在工程优化篇中，会提到如何利用工具和技巧帮助开发者优化项目。这部分的知识较为零碎，但学起来并不难，很适合读者用碎片时间进行学习。

第 6 章 Dialog 已经不推荐直接使用了，取而代之的是 dialogFragment，关于它我们需要知道：

- Dialog 和 alertDialog，alertDialog 和 dialogFragment 是什么关系？

- 如何实现一个可以随时展示的全局弹窗，这对于 App 的任务系统有什么好处？
- 为什么说完全可以通过 Style 实现自定义的 Dialog，Dialog 的背景该如何编写？
- alertDialog 引起内存泄露和崩溃的原因是什么，在开发中该如何防御？

第 7 章 Gradle 是常用工具，本章不准备讲它的基础知识，会直接介绍实际的例子：

- 如何自动化生成 versionCode 和 versionName，Gradle 还能做哪些自动化处理？
- 如何强制整个 App 的某个库均使用统一的版本号，如何更好地管理签名文件？
- 如何引入本地的 Maven 仓库，本地依赖的写法是什么，如何制作一个本地库？
- 如何使用 Exclude Lib、Jar、Module 等，对于产生类冲突的两个库该如何解决？

第 8 章 本章除讲述如何提升打 Debug 包的速度外，还介绍了如下内容：

- 为什么小范围的配置优化不如直接升级机器硬件来得快？
- Google 废弃 compile() 的原因是什么，新的依赖 API 有哪些，各有什么不同？
- 如何启用 Gradle 缓存，减少每次编译时的网络请求是提速的关键吗？
- Android 有几种签名方案，不同签名方案对应的多渠道打包策略有何不同？

第 9 章 每家公司都希望 APK 越来越小，本章探讨的是切实有效的瘦身策略，主要解惑的点为：

- APK 是由哪些文件组成的？APK 和 Zip 是什么关系，签名相关的文件在哪个目录中？
- 瘦身开始前应该做哪些准备工作，为什么这个看似无关紧要的工作却很重要。
- APK 中的 9 图应该放在哪个文件夹中，到底要不要做屏幕分辨率的适配。
- ABI 是什么意思，从云端下载 so 时需要注意什么问题，具体的可行性如何？

第 10 章 测试用例的编写一直被我们忽视，本章是一个入门级的内容，涵盖了如下的知识点：

- mock 的目的是什么，常见的 mock 框架有哪些，我们一般会如何进行技术选型？
- 测试分为逻辑测试和 UI 测试，哪个才是移动测试的重点？
- 如果想要在 PC 端直接测试 Android 的逻辑，目前最方便的测试框架是哪个？
- 网络层是一个典型的单元测试和集成测试的场景，该如何写好异步代码的测试用例？

第 11 章 本章在前半部分会讲与 Debug 相关的内容，后半部分将介绍一些极其好用的插件，阅读后你将知道：

- 如何在 Debug 时动态打 Log，它和在代码中硬编码的 Log 有什么区别？
- 如何通过 Debug 工具检查内存泄露，怎样清楚地知道当前内存中的对象数目？
- Facebook 的 Sonar 是什么，这个平台可以带来怎样的想象力？
- 号称史上最强的数据库调试插件是什么，它真的值得购买吗，免费的替代品是什么？

第 12 章 使用代理工具是开发人员和测试人员的必会技能，本章会介绍 Whistle 的主要功能：

- 支持用正则制定过滤规则，返回的 JSON 能自动进行格式化。
- 支持为多个环境配置不同的代理，可以灵活地进行环境的切换。
- 可以直接编写 mock 规则，自动化地生成 mock 数据，并且能统一管理 mock 的 values。
- 支持调试 Webview，可以很方便地进行真机上 H5 页面的调试。

致谢

写书本身是一件很严肃的事情，尤其是技术方面的书籍。我是一个讨厌被限制的人，所以一直以来都是偷懒写“网文”，原本不愿意出版图书。下面要感谢一些朋友，没有他们我可能永远都无法迈出这一步，也无法得到一本真正属于自己的作品。

- 感谢博文观点团队的赏识和邀请，如果没有贵团队，我根本就不会有写书的想法和机会。
- 感谢本书的编辑陈晓猛，他的沟通能力和执行力是保障本书快速出版的关键。
- 感谢“不吃鱼同学”，要不是她的支持和鼓励，我就没有勇气开始做这件长达一年的事情。
- 感谢所有在微博关注我的朋友们，你们的支持和批评是鞭策我前进的动力。
- 感谢聚美、网易和美团的一些朋友们，你们给予了我很多的建议和灵感。
- 更要感谢即将开始阅读本书的你，这本书就是为你而写的，你的阅读是我最大的荣耀！

联系方式

如果你发现了本书的纰漏或是对内容有更好的建议，可以通过下列方式联系到我：

微博：@天之界线 2010；

邮箱：developer-kale@foxmail.com；

QQ 群：390272666；

GitHub：tianzhijixian。

轻松注册成为博文视点社区用户（www.broadview.com.cn），扫码直达本书页面。

- **提交勘误**：您对书中内容的修改意见可在 [提交勘误](#) 处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **交流互动**：在页面下方 [读者评论](#) 处留下您的疑问或观点，与我们和其他读者一同学习交流。

页面入口：<http://www.broadview.com.cn/35924>



目 录

第 1 章 探寻高效易用的反射 API	1
1.1 反射的能力	2
1.1.1 得到 Class 对象	2
1.1.2 操作 Field	4
1.1.3 调用 Method	8
1.1.4 动态代理	10
1.2 反射封装库——JOOR	19
1.2.1 反射的流程	19
1.2.2 VirtualApp 中的反射	20
1.2.3 一行代码建立对象	22
1.2.4 简化 Field 的相关操作	23
1.2.5 简化方法调用	24
1.2.6 封装动态代理	25
1.3 注意事项	25
1.3.1 反射的性能问题	25
1.3.2 反射的使用时机	27
1.3.3 如何降低反射的性能损耗	28
1.3.4 反射的危险性	29
1.3.5 反射和混淆的关系	30
1.4 总结	31
第 2 章 打造高扩展性的 Log 系统	32
2.1 基本概念	32
2.2 命令行操作 Log	33
2.2.1 输出日志	33
2.2.2 过滤日志	35
2.3 Android Studio 中的 Log	38
2.3.1 设置模板	39
2.3.2 正则过滤	39
2.3.3 热部署 Log	40
2.4 微信的 Xlog	42
2.4.1 设计和开发目标	42
2.4.2 编译、引入和使用	43

2.4.3 对 Log 文件进行优化	45
2.5 美团的 Logan	47
2.6 扩展 Log 的功能	48
2.6.1 TAG 的自动化	48
2.6.2 文本内容的设计	50
2.6.3 开关的设计	50
2.7 封装 Log 库	54
2.7.1 Timber	54
2.7.2 LogDelegate	55
2.7.3 Logger	56
2.7.4 扩展 Timber 的功能	57
2.7.5 分发日志	58
2.8 实用日志	60
2.8.1 操作耗时日志	60
2.8.2 页面跳转日志	61
2.8.3 网络请求日志	61
2.9 总结	63
第 3 章 万变不离其宗的 Intent	64
3.1 源码分析	64
3.1.1 静态变量的写法	64
3.1.2 Intent 的深拷贝	65
3.1.3 makeMainActivity	66
3.1.4 Intent 的 Chooser	67
3.1.5 用 URI 代替 Intent	69
3.1.6 存取值的底层实现	71
3.1.7 区分显式和隐式 Intent	74
3.1.8 抛弃 Bundle 的传值策略	75
3.2 序列化方案	77
3.2.1 Serializable/Externalizable	77
3.2.2 Android 中的 Parcelable	80
3.2.3 Google 的 Protocol Buffer	82
3.2.4 Twitter 的 Serial	83
3.3 常见问题	85
3.3.1 父类的序列化	85
3.3.2 类型转换异常	85
3.3.3 重复启动的问题	86
3.3.4 传递大对象	86
3.4 简单的传值库——Parceler	88

3.4.1	降低 Key 的维护成本	88
3.4.2	自动维护 Intent 的 Key	89
3.4.3	Jetpack 中的自动化	90
3.4.4	自动保存状态	92
3.4.5	处理 ClassCastException	93
3.4.6	IntentLauncher	94
3.4.7	统一存取的 API	94
3.5	总结	95
第 4 章	SharedPreferences 的再封装	96
4.1	源码分析	96
4.1.1	缓存机制	97
4.1.2	SharedPreferencesImpl	98
4.1.3	值操作	99
4.1.4	提交操作	101
4.2	异常处理	104
4.2.1	name 为 null	104
4.2.2	管理好 Key 的取名	105
4.2.3	清空操作失效	106
4.2.4	磁盘写入异常	106
4.2.5	出现 ANR	107
4.2.6	存序列化对象	110
4.2.7	多 App 和多进程访问异常	110
4.3	性能优化	111
4.3.1	避免储存大量数据	111
4.3.2	尽可能提前初始化	112
4.3.3	避免 Key 过长	112
4.3.4	多次操作, 批量提交	112
4.3.5	缓存 Editor 对象	113
4.3.6	不存放 HTML 和 JSON	114
4.3.7	拆分高频和低频操作	115
4.4	封装 SharedPreferences	115
4.4.1	PreferenceDataStore	115
4.4.2	通过接口提高内聚	117
4.4.3	得到 SharedPreferences	118
4.4.4	多用户存储设计	118
4.4.5	统一管理 Key	119
4.4.6	自动判断返回值类型	119
4.4.7	决定是否使用 Apply	120

4.4.8	存放序列化对象	121
4.4.9	支持数据格式转换器	121
4.5	思维扩展	122
4.5.1	偏好界面的实现方案	122
4.5.2	监听数据的改变	123
4.5.3	利用 Tray 实现多进程访问	124
4.5.4	React Native 中的使用	125
4.6	总结	126
第 5 章	寻找 Fragment 的继任者	127
5.1	使用场景	127
5.1.1	日夜间模式	127
5.1.2	缓存界面数据	129
5.1.3	作为搜索页	130
5.1.4	作为 Presenter	131
5.2	源码分析	132
5.2.1	Transaction 简介	132
5.2.2	提交操作	135
5.2.3	commitAllowingStateLoss	137
5.2.4	Add 操作的原理	139
5.2.5	Replace 操作的本质	142
5.2.6	Fragment 的可见性监听	145
5.2.7	ViewPager 中的懒加载	147
5.3	常见问题	149
5.3.1	Activity 为空	149
5.3.2	startActivityForResult	150
5.3.3	ViewPager 的 getItem	152
5.3.4	FragmentPagerAdapter	153
5.3.5	显示一个对话框	154
5.3.6	重叠显示的问题	155
5.3.7	Fragment 的 StateLoss	155
5.4	Fragment 的替代品	157
5.4.1	Jetpack 的 Navigation	157
5.4.2	Square 的 Flow	161
5.4.3	简化版的 Fragment	162
5.5	Shatter 库	163
5.5.1	建立 Shatter 类	164
5.5.2	设计 ShatterManager	165
5.5.3	分发生命周期	165

5.5.4 使用方式	168
5.6 总结	172
第 6 章 让 alertDialog 为我所用	173
6.1 Dialog	173
6.1.1 Dialog 和 Window	173
6.1.2 Show 和 Dismiss 方法	176
6.2 alertDialog	178
6.2.1 alertController	178
6.2.2 alertDialog.Builder	180
6.3 dialogFragment	182
6.3.1 Fragment 和 Dialog	184
6.3.2 Show 和 Dismiss 方法	185
6.4 实际问题	188
6.4.1 无法弹出输入法	188
6.4.2 如何支持层叠弹窗	188
6.4.3 容易引起内存泄露	189
6.4.4 修改尺寸、背景和动画	191
6.4.5 点击后会自动关闭	192
6.4.6 在关闭或开启时出现崩溃	194
6.5 封装 dialogFragment	195
6.5.1 用现成的 alertParams	195
6.5.2 让 Builder 类支持继承	197
6.5.3 建立 dialogFragment 框架	200
6.6 easyDialog	201
6.6.1 基本用法	201
6.6.2 自定义一个 Dialog	202
6.6.3 BottomSheetDialog	206
6.6.4 设置全局样式	209
6.6.5 支持动态样式	216
6.6.6 避免丢失监听器	216
6.7 可全局弹出的 Dialog	218
6.8 总结	219
第 7 章 Gradle 的使用技巧	220
7.1 全局配置	220
7.1.1 设定 UTF-8	220
7.1.2 依赖 Google 仓库	220
7.1.3 支持 Groovy	221

7.1.4	定义全局变量	222
7.1.5	配置 Lint 选项	223
7.2	操控 Task	224
7.2.1	更改输出的 APK 的名字	224
7.2.2	更改 AAR 输出的位置	224
7.2.3	跳过 AndroidTest	225
7.2.4	找出耗时的 Task	226
7.2.5	抽离 Task 脚本	227
7.3	动态化	228
7.3.1	动态设置 buildConfig	228
7.3.2	填充 Manifest 中的值	229
7.3.3	让 buildType 支持继承	230
7.3.4	让 Flavor 支持继承	231
7.3.5	内测版本用特定的 Icon	232
7.3.6	不同渠道不同包名	234
7.3.7	自动填充版本信息	234
7.4	远程依赖	236
7.4.1	配置 Maven 仓库	236
7.4.2	依赖相关的 API	236
7.4.3	组合依赖	237
7.4.4	依赖传递	238
7.4.5	动态版本号	238
7.4.6	强制版本号	239
7.4.7	exclude 关键字	240
7.4.8	依赖管理	240
7.5	本地依赖	241
7.5.1	引用 AAR	241
7.5.2	依赖 Module/Jar	242
7.5.3	自建本地仓库	243
7.5.4	本地依赖 React Native	245
7.5.5	重新打包第三方 Jar	245
7.6	资源管理	246
7.7	总结	247
第 8 章	缩减 APK 的编译时间	248
8.1	分析项目现状	248
8.1.1	Gradle Profile	248
8.1.2	BuildTimeTracker	248
8.1.3	Dexcount GradlePlugin	249

8.1.4	经验小结	249
8.2	编译环境优化	250
8.2.1	升级硬件设备	250
8.2.2	升级软件	250
8.2.3	优化工程配置	251
8.2.4	配置 Studio 的可用内存	251
8.2.5	提升 JVM 的堆内存	251
8.2.6	开启并行编译	252
8.2.7	启用 Demand 模式	252
8.2.8	配置 DexOptions	252
8.3	善用缓存	253
8.3.1	减少动态方法	253
8.3.2	硬编码 BuildConfig 和 Res	254
8.3.3	拆分脚本	254
8.3.4	拆分代码	255
8.3.5	写死库的版本号	256
8.4	精简工程	256
8.4.1	差异化加载 Plugin	256
8.4.2	使用 WebP 和 SVG	257
8.4.3	精简语言和图片资源	258
8.4.4	善用 no-op	258
8.4.5	Exclude 无用库	260
8.4.6	删减 Module	260
8.4.7	去掉 MultiDex	261
8.4.8	删除无用的资源	261
8.5	综合技巧	262
8.5.1	构建开发时的 Flavor	262
8.5.2	优化 MultiDex	262
8.5.3	跳过无用的 Task	263
8.5.4	关闭 AAPT 的图片优化	264
8.5.5	调试时关闭 Crashlytics	264
8.5.6	谨慎使用 AspectJ	265
8.6	多渠道打包工具	268
8.6.1	MultiChannelPackageTool	268
8.6.2	美团的 Walle	269
8.6.3	腾讯的 VasDolly	270
8.7	总结	271

第 9 章 APP 终极瘦身实践.....272

9.1 安装包的构成.....272

9.1.1 Assets.....273

9.1.2 Lib.....273

9.1.3 Resources.arsc.....274

9.1.4 META-INF.....274

9.1.5 Res.....274

9.1.6 Dex.....275

9.2 优化 Assets 目录.....275

9.2.1 删除无用的字体.....275

9.2.2 减少 IconFont 的使用.....276

9.2.3 动态下载资源.....276

9.2.4 压缩资源文件.....278

9.3 优化 Lib 目录.....279

9.3.1 配置 ABI Filters.....279

9.3.2 根据 CPU 引入 so.....281

9.3.3 动态加载 so.....283

9.3.4 避免复制 so.....284

9.3.5 谨慎处理 so.....284

9.4 优化 Resources.arsc.....285

9.4.1 删除无用的映射.....285

9.4.2 进行资源混淆.....286

9.5 优化 META-INF.....287

9.5.1 MANIFEST.MF.....287

9.5.2 CERT.SF.....288

9.5.3 CERT.RSA.....289

9.5.4 优化建议.....290

9.6 优化 Res 目录.....290

9.6.1 通过 IDE 删除无用资源.....290

9.6.2 打包时剔除无用资源.....291

9.6.3 删除无用的语言.....291

9.6.4 控制 Raw 中的资源大小.....291

9.6.5 减少 Shape 文件.....292

9.6.6 减少 Menu 文件.....293

9.6.7 减少 Layout 文件.....294

9.6.8 动态下载图片.....296

9.6.9 分目录放置图片.....296

9.6.10 合理使用图片资源.....300