

《App研发录：架构设计、Cras》

图书基本信息

书名：《App研发录：架构设计、Crash分析和竞品技术分析》

13位ISBN编号：9787111516389

出版时间：2015-10-21

作者：包建强

页数：303

版权说明：本站所提供下载的PDF图书仅提供预览和简介以及在线试读，请支持正版图书。

更多资源请访问：www.tushu111.com

《App研发录：架构设计、Cras》

内容概要

本书是作者多年App开发的经验总结，从App架构的角度，重点总结了Android应用开发中常见的实用技巧和疑难问题解决方法，为打造高质量App提供有价值的实践指导，迅速提升应用开发能力和解决疑难问题的能力。本书涉及的问题有：Android基础建设、网络底层框架设计、缓存、网络流量优化、制定编程规范、模块化拆分、Crash异常的捕获与分析、持续集成、代码混淆、App竞品技术分析、项目管理和团队建设等。本书以“问题/解决方案”的形式给出疑难问题的解决方案，同时结合示例代码，深入剖析这些实用的编程技巧和模式，旨在帮助移动开发人员和管理人员提高编程效率，改进代码质量，打造高质量的App。

《App研发录：架构设计、Cras》

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。在Android、iOS、WP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。他是微软2008年MVP。曾经翻译出版《.NET探秘MSIL权威指南》，并有一个坚持写了6年的技术博客：<http://jax.cnblogs.com/>。

书籍目录

Contents?目 录

序一

序二

序三

前言

第一部分 高效App框架设计与重构

第1章 重构，夜未眠 3

1.1 重新规划Android项目结构 3

1.2 为Activity定义新的生命周期 5

1.3 统一事件编程模型 7

1.4 实体化编程 9

1.4.1 在网络请求中使用实体 9

1.4.2 实体生成器 11

1.4.3 在页面跳转中使用实体 12

1.5 Adapter模板 14

1.6 类型安全转换函数 16

1.7 本章小结 17

第2章 Android网络底层框架设计 19

2.1 网络低层封装 19

2.1.1 网络请求的格式 19

2.1.2 AsyncTask的使用和缺点 21

2.1.3 使用原生的ThreadPoolExecutor + Runnable + Handler 24

2.1.4 网络底层的一些优化工作 28

2.2 App数据缓存设计 32

2.2.1 数据缓存策略 32

2.2.2 强制更新 35

2.3 MockService 36

2.4 用户登录 38

2.4.1 登录成功后的各种场景 39

2.4.2 自动登录 41

2.4.3 Cookie过期的统一处理 44

2.4.4 防止黑客刷库 45

2.5 HTTP头中的奥妙 46

2.5.1 HTTP请求 46

2.5.2 时间校准 48

2.5.3 开启gzip压缩 51

2.6 本章小结 52

第3章 Android经典场景设计 53

3.1 App图片缓存设计 53

3.1.1 ImageLoader设计原理 53

3.1.2 ImageLoader的使用 54

3.1.3 ImageLoader优化 55

3.1.4 图片加载利器Fresco 56

3.2 对网络流量进行优化 58

3.2.1 通信层面的优化 58

3.2.2 图片策略优化 59

3.3 城市列表的设计 61

- 3.3.1 城市列表数据 61
- 3.3.2 城市列表数据的增量更新机制 63
- 3.4 App与HTML5的交互 64
 - 3.4.1 App操作HTML5页面的方法 64
 - 3.4.2 HTML5页面操作App页面的方法 65
 - 3.4.3 App和HTML5之间定义跳转协议 66
 - 3.4.4 在App中内置HTML5页面 67
 - 3.4.5 灵活切换Native和HTML5页面的策略 68
 - 3.4.6 页面分发器 68
- 3.5 消灭全局变量 70
 - 3.5.1 问题的发现 70
 - 3.5.2 把数据作为Intent的参数传递 71
 - 3.5.3 把全局变量序列化到本地 71
 - 3.5.4 序列化的缺点 75
 - 3.5.5 如果Activity也被销毁了呢 79
 - 3.5.6 如何看待SharedPreferences 80
 - 3.5.7 User是唯一例外的全局变量 80
- 3.6 本章小结 81
- 第4章 Android命名规范和编码规范 83
 - 4.1 Android命名规范 83
 - 4.2 Android编码规范 86
 - 4.3 统一代码格式 89
 - 4.4 本章小结 90
- 第二部分 App开发中的高级技巧
- 第5章 Crash异常收集与统计 93
 - 5.1 异常收集 93
 - 5.2 异常收集与统计 96
 - 5.2.1 人工统计线上Crash数据 96
 - 5.2.2 第一个线上Crash报表：Crash分类 97
 - 5.2.3 第二个线上Crash报表：Crash去重 99
 - 5.2.4 线上Crash的其他分析工作 104
 - 5.3 本章小结 105
- 第6章 Crash异常分析 107
 - 6.1 Java语法相关的异常 108
 - 6.1.1 空指针 108
 - 6.1.2 角标越界 109
 - 6.1.3 试图调用一个空对象的方法 110
 - 6.1.4 类型转换异常 110
 - 6.1.5 数字转换错误 111
 - 6.1.6 声明数组时长度为-1 111
 - 6.1.7 遍历集合同时删除其中元素 112
 - 6.1.8 比较器使用不当 114
 - 6.1.9 当除数为0 115
 - 6.1.10 不能随便使用的asList 116
 - 6.1.11 又有类找不到了（一）：ClassNotFoundException 116
 - 6.1.12 又有类找不到了（二）：NoClassDefFoundError 117
 - 6.2 Activity相关的异常 117
 - 6.2.1 找不到Activity 117
 - 6.2.2 不能实例化Activity 118

- 6.2.3 找不到Service 118
- 6.2.4 不能启动BroadcastReceiver 119
- 6.2.5 startActivityForResult不能回传 119
- 6.2.6 猴急的Fragment 120
- 6.3 序列化相关的异常 120
 - 6.3.1 实体对象不支持序列化 121
 - 6.3.2 序列化时未指定ClassLoader 121
 - 6.3.3 反序列化时发现类找不到：被ProGuard混淆导致的崩溃 122
 - 6.3.4 反序列化时发现类找不到：传入畸形数据 123
 - 6.3.5 反序列化时出错 123
- 6.4 列表相关的异常 123
 - 6.4.1 Adapter数据源变化但是没通知ListView 124
 - 6.4.2 ListView滚动时点击刷新按钮后崩溃 125
 - 6.4.3 AbsListView的obtainView返回空指针 125
 - 6.4.4 Adapter数据源变化但是没调用notifyDataSetChanged 126
- 6.5 窗体相关的异常 126
 - 6.5.1 窗口句柄泄露 126
 - 6.5.2 View not attached to window manager 128
 - 6.5.3 窗体在不恰当的时候获取了焦点 129
 - 6.5.4 token null is not for an application 130
 - 6.5.5 permission denied for this window type 131
 - 6.5.6 is your activity running 131
 - 6.5.7 添加窗体失败 133
 - 6.5.8 AlertDialog.resolveDialogTheme 134
 - 6.5.9 The specified child already has a parent 136
 - 6.5.10 子线程不能修改UI 137
 - 6.5.11 不能在子线程操作AlertDialog和Toast 141
- 6.6 资源相关的异常 143
 - 6.6.1 Resources\$NotFoundException 143
 - 6.6.2 StackOverflowError 144
 - 6.6.3 UnsatisfiedLinkError 144
 - 6.6.4 InflateException之FileNotFoundException 145
 - 6.6.5 InflateException之缺少构造器 145
 - 6.6.6 InflateException之style与android:textStyle的区别 146
 - 6.6.7 TransactionTooLargeException 147
- 6.7 系统碎片化相关的异常 147
 - 6.7.1 NoSuchMethodError 147
 - 6.7.2 RemoteViews 148
 - 6.7.3 pointerIndex out of range 149
 - 6.7.4 SecurityException之一：Intent中图片太大 150
 - 6.7.5 SecurityException之二：动态加载其他apk的activity 151
 - 6.7.6 SecurityException之三：No permission to modify thread 151
 - 6.7.7 view的getDrawableCache()返回null 152
 - 6.7.8 DeadObjectException 153
 - 6.7.9 Android 2.1不支持SSL 153
 - 6.7.10 ViewPager引发的血案 153
 - 6.7.11 ActivityNotFoundException 154
 - 6.7.12 Android 2.2不支持xlargeScreens 154
 - 6.7.13 Package manager has died 155

- 6.7.14 SpannableString与富文本字符串 155
- 6.7.15 Can not perform this action after onSaveInstanceState 156
- 6.7.16 Service Intent must be explicit 157
- 6.8 SQLite相关的异常 157
 - 6.8.1 No transaction is active 158
 - 6.8.2 忘记关闭Cursor 158
 - 6.8.3 数据库被锁定 159
 - 6.8.4 试图再打开已经关闭的对象 159
 - 6.8.5 文件加密了或无数据库 159
 - 6.8.6 WebView中SQLite缓存导致的崩溃 160
 - 6.8.7 磁盘读写错误 161
 - 6.8.8 android_metadata表不存在 161
 - 6.8.9 android_metadata表中的locale字段 162
 - 6.8.10 数据库或磁盘满了 162
- 6.9 不明觉厉的异常 162
 - 6.9.1 内存溢出 163
 - 6.9.2 Verify Failed 163
- 6.10 其他情况的异常 163
 - 6.10.1 TimeoutException 164
 - 6.10.2 JSON解析异常 164
 - 6.10.3 JSONArray在初始化时为空 164
 - 6.10.4 第三方SDK抛出的Crash 165
 - 6.10.5 两个不同类型的View有相同的id 165
 - 6.10.6 LayoutInflater.from().inflate()使用不当导致的崩溃 166
 - 6.10.7 ViewGroup中的玄机 166
 - 6.10.8 Monkey点击过快导致的崩溃 167
 - 6.10.9 图片缩放很多倍 168
 - 6.10.10 图片宽高为0 168
 - 6.10.11 不能重复添加组件 168
- 6.11 本章小结 169
- 第7章 ProGuard技术详解 171
 - 7.1 ProGuard简介 171
 - 7.2 ProGuard工作原理 172
 - 7.3 如何写一个ProGuard文件 172
 - 7.3.1 基本混淆 172
 - 7.3.2 针对App的量身定制 175
 - 7.3.3 针对第三方jar包的解决方案 177
 - 7.4 其他注意事项 178
 - 7.5 本章小结 179
- 第8章 持续集成 181
 - 8.1 版本管理策略 181
 - 8.1.1 三种版本管理策略 181
 - 8.1.2 特殊情况的版本管理策略 183
 - 8.2 使用Ant脚本打包 184
 - 8.2.1 Android打包流程 184
 - 8.2.2 打包时的注意事项 189
 - 8.3 Monkey包的生成 190
 - 8.4 自动打包 191
 - 8.4.1 安装和配置各种软件 192

- 8.4.2 准备Ant打包脚本 193
- 8.4.3 配置CCNET 193
- 8.4.4 搭建IIS站点下载apk包 193
- 8.4.5 自动打包流程小结 193
- 8.5 批量打渠道包 194
 - 8.5.1 基于apk包批量生成渠道包 194
 - 8.5.2 基于代码批量生成渠道包 195
- 8.6 Android发版流程 197
- 8.7 分类打渠道包 198
 - 8.7.1 分门别类生成渠道包 198
 - 8.7.2 批量上传apk的两种方式 199
- 8.8 灵活切换服务器 199
- 8.9 单元测试 201
- 8.10 本章小结 203
- 第9章 App竞品技术分析 205
 - 9.1 竞品分析概述 205
 - 9.1.1 App竞品定义 205
 - 9.1.2 竞品分析要研究的几个方向 206
 - 9.1.3 竞品分析与拿来主义 206
 - 9.2 App安装包的结构 207
 - 9.2.1 Android安装包的结构 207
 - 9.2.2 iOS安装包的结构 208
 - 9.3 竞品技术一瞥：开机速度 208
 - 9.4 竞品技术二瞥：HTML5页面的打开速度 209
 - 9.4.1 把HTML5页面嵌入到Zip包中 209
 - 9.4.2 Zip包的增量更新机制 209
 - 9.4.3 制作Zip增量包 210
 - 9.4.4 使用WebView预先加载HTML5并缓存到本地 211
 - 9.5 竞品技术三瞥：安装包的大小 211
 - 9.5.1 从几件小事说起 211
 - 9.5.2 安装包为什么那么大 212
 - 9.5.3 png和jpg的区别及使用场景 212
 - 9.5.4 Splash、引导图和背景图 213
 - 9.5.5 iOS的1倍图、2倍图和3倍图 213
 - 9.5.6 在iOS中进行图片拉伸和旋转 214
 - 9.5.7 使用XML配置动画 214
 - 9.5.8 iOS使用storyboard还是xib 215
 - 9.5.9 字体文件的学问 215
 - 9.5.10 表情图片打包下载 217
 - 9.5.11 清除未使用图片 218
 - 9.5.12 Proguard不只是用来混淆的 218
 - 9.5.13 在iOS中使用pdf格式的图片 218
 - 9.5.14 iOS的包永远比Android包体积大吗 219
 - 9.5.15 从代码层面减少iOS包的体积 220
 - 9.6 竞品技术四瞥：性能优化 220
 - 9.6.1 App自动选取最佳服务器的策略 220
 - 9.6.2 使用TCP+Protobuf 222
 - 9.7 竞品技术五瞥：数据采集工具 223
 - 9.7.1 页面跳转器 223

- 9.7.2 打点统计 226
 - 9.7.3 ABTest 230
 - 9.8 竞品技术六警：热修补 232
 - 9.8.1 Native页面和HTML5页面的相互切换 232
 - 9.8.2 在iOS中使用脚本编程 233
 - 9.9 竞品技术七警：曲径通幽 237
 - 9.9.1 一切皆可配置 237
 - 9.9.2 App后门 238
 - 9.9.3 Android包中META-INF目录的妙用 239
 - 9.9.4 classes.dex的拆与合 241
 - 9.10 竞品技术八警：模块化拆分 242
 - 9.10.1 iOS资源拆分与模块化 242
 - 9.10.2 Android模块化拆分 243
 - 9.11 竞品技术九警：第三方SDK 244
 - 9.11.1 HTML5篇 244
 - 9.11.2 iOS篇 245
 - 9.11.3 Android篇 245
 - 9.11.4 其他 246
 - 9.12 竞品技术十警：版本策略与App彩蛋 246
 - 9.12.1 版本策略 246
 - 9.12.2 App彩蛋 246
 - 9.13 本章小结 247
- 第三部分 项目管理和团队建设
- 第10章 项目管理决定了开发速度 251
- 10.1 项目管理中的三驾马车 251
 - 10.1.1 为什么不能没有测试团队 252
 - 10.1.2 产品经理应做的事 253
 - 10.1.3 开发人员的喜怒哀乐 254
 - 10.1.4 项目经理的职责 254
 - 10.2 优化团队结构，让敏捷流程跑得更快 255
 - 10.2.1 平行模式还是垂直模式 255
 - 10.2.2 让HTML5站点和MobileAPI的进度提前一个迭代 256
 - 10.2.3 如何进行模块化分工 256
 - 10.3 App敏捷开发流程 257
 - 10.3.1 四周时间的开发流程 257
 - 10.3.2 两周时间的开发流程 261
 - 10.3.3 一周时间的开发流程 262
 - 10.3.4 即时更新策略 263
 - 10.4 项目经理的百宝箱 263
 - 10.4.1 项目经理的任务评估表 263
 - 10.4.2 贴小纸条的艺术 264
 - 10.4.3 敏捷迭代中的会议纪要 265
 - 10.4.4 开站例会的技巧 266
 - 10.4.5 如何确保项目不延期 268
 - 10.4.6 迭代风险管理 268
 - 10.5 迭代中的测试工作 269
 - 10.5.1 冒烟测试 269
 - 10.5.2 探索性测试 271
 - 10.5.3 Monkey测试 271

- 10.6 高层对敏捷流程的干预 272
 - 10.6.1 重构与产品需求的平衡 272
 - 10.6.2 提高效率，拒绝6×12 273
 - 10.6.3 无线部门的座位安排 274
 - 10.6.4 静时 276
- 10.7 本章小结 277
- 第11章 日常工作中的问题解决 279
 - 11.1 使用二分法排查问题 279
 - 11.2 找到能稳定重现问题的人 281
 - 11.3 小流量包 282
 - 11.4 建立全国范围的测试群 283
 - 11.5 如何与用户沟通 284
 - 11.6 日志与App性能 286
 - 11.7 从新人入职作业入手 286
 - 11.8 本章小结 287
- 第12章 无线团队的组建和管理 289
 - 12.1 从面试谈起 289
 - 12.1.1 如今是卖方市场 289
 - 12.1.2 名校论不适用无线开发 290
 - 12.1.3 如何搞到更多的简历 290
 - 12.1.4 面试时需要考察的几个点 291
 - 12.2 无线团队必备的10份文档 292
 - 12.2.1 新员工入职文档 292
 - 12.2.2 加强版新员工入职文档 292
 - 12.2.3 测试机清单 293
 - 12.2.4 模块分工表 293
 - 12.2.5 页面逻辑流程文档 293
 - 12.2.6 MobileAPI接口分布图 295
 - 12.2.7 版本管理策略文档 295
 - 12.2.8 框架设计文档 295
 - 12.2.9 发版流程文档 296
 - 12.2.10 App启动流程图 296
 - 12.3 一对一沟通 297
 - 12.4 每周技术分享 298
 - 12.5 代码评审 299
 - 12.6 对Android团队Leader的定位 300
 - 12.7 Android应用开发所需技能自我评测 301
 - 12.8 App开发人员的学习路线 302
 - 12.9 本章小结 303

精彩短评

- 1、这些经验很受用
- 2、APP竞品技术分析的部分值得产品好好研读~~
- 3、这本书为啥只有7.5？技术书籍不是非得显得牛逼才高分的啊，这样朴素的，作者自己的经验总结不好吗？你们喜欢那种傻了吧唧罗列代码，连他们自己都不知道自己在讲什么的书？不过作者似乎在系统方面比较薄弱。Android的UI太开放，导致现在市面上混乱的UI场面。
- 4、前两章还有点干货，后面的比较水
- 5、一本够我嚼两三年的书，大把大把的干货，感谢作者
- 6、作者的技术水平真是呵呵！！
- 7、总归会在开发中遇到
- 8、利用周末完结了，写的很好，见识了正规的团队研发，同时还介绍了一些重要但是平时自己没有用到的东西，谢谢作者。
- 9、网络基础架构，性能优化不错。竞品分析和crash都是难得的资料
- 10、一线技术人员的总结与分享,可以学习与借鉴的很多.特别是团队组织协调方面.
- 11、不是一本纯粹的技术书，内容有点杂。
- 12、实用
- 13、今年看过的最好的一本Android/移动开发相关的技术图书。
- 14、一般，书里不少内容有点过时了，而且作者说太多外话了，不太适合初级研发看
- 15、干货，涉及app崩溃收集、混淆技术、持续集成、团队管理
- 16、真正讲一个app的研发，而不是单纯的码代码，书很快看完了，但是里面的思想全部掌握并运用还是任重道远啊。
- 17、技术牛逼的一本书
- 18、5天过完的一本书，其中第七章ProGuard的介绍还有点意思，其他章节不做评价。
- 19、非常不错的书，都是作者亲身开发经验总结出来的干货，三言两语点出关键。比起让人云里雾里不知所云的书不知道要高到哪里去了！虽然不是很厚，但是对于现阶段的初级Android程序员的自己太合适不过了。
- 20、还不错，相比技术对于移动开发的一些经验之谈值得借鉴
- 21、花了一周看完了，讲的比较系统。内容不很多，很多时候都是针对于作者自己项目中的问题讲解一番，不知是是否因为自己水平不够，并没有很多共鸣。
- 22、就一本安卓开发的书，怎么名字敢叫App研发录。靠，我是iOS开发者啊，看安卓有个毛用
- 23、干货不少，proguard那一章不错，毕竟以前没有哪本书系统地总结过。竞品分析那一章也不错，涨姿势了，知识面宽了。作者用的是Eclipse,没有对Android Studio和Gradle的讲解。
- 24、详尽地介绍了大公司的流程，还有一些最佳实践和坑等。
- 25、android
- 26、比较适合开发组长大概浏览，了解概念及一些技术点（插件化、本地测试、Native&H5交互、多渠道打包、开发后门），需要再自己后续研究。包括团队管理方向、新人文档、编码规范等，都是比较适合组长的知识点。Crash基本没看，看了也记不住，遇到时再根据堆栈信息去找bug就行，没必要现在把这些东西看一遍记下来
- 27、还不错，看这本书就像你和一个工作多年的技术经理在聊天，当然中间有一些作者的crash总结，可以没事拿来翻翻看看自己的app有没有同样的crash。看得出来作者有着非常丰富的带技术团队的经验，而且平时善于总结、归纳问题，确实对于我的开发工作有很多值得借鉴的，不过书中一些技巧、开源库不是最新最好的，或者不是最棒的解决办法，可能是作者很久之前写的吧
- 28、老司机经验之谈，新人可以管中窥豹，司机同行可以借机吐槽共鸣
- 29、android进阶很有用
- 30、1. 本书可以让你少趟一些坑。
2. 然而书中的“我觉得”有点多啊，看着信心不足。
3. 可见写博客是多么重要，那么多面试者总是说自己记了多少笔记，鬼才信呢~
- 31、“落地”，是阅读此书的最大感受，大到技术架构、竞品分析、发布流程，小到每个Crash、员工

《App研发录：架构设计、Cras》

座位安排、开展技术分享、简历筛选，作者总在寻求落地的解决方案，细细阅读，你不觉得这种方案有多么高大上，不觉得背后有多么华丽的理论支撑，但是你会觉得，一切都是踏踏实实、真真切切可落地实践的，为工作带来不少的实践指导。

建议移动开发从业者都阅读一下本书，如果你是一名程序员，有必要了解下你的leader背后要考虑的大局，以及做出某些决策的原因，这能帮助你更好的为团队分忧，承担更多的责任，获得更多的晋升空间。如果你是一名技术管理者，反思一下你的决策是否能落地，是否只是空谈，或者你的决策与你想达到的目标其实是两码事...完整如下

<http://www.jianshu.com/p/ea48332c82af>

32、书名是《app研发录》，不过感觉书的定位好像很模糊，既有普通Android工程师需要的技术内容，又有一些类似于产品经理所关心的开发流程之类的内容，除了前面讲的一个网络框架的设计让人有印象之外，还有就是一章专门用来讲crash，我也是醉了，所以感觉整本书并没有很多干货，有些内容也是浅尝即止，有人把它跟《Android开发艺术探索》相提并论，我觉得是不够资格的。

33、个人不推荐购买，我见过的书里面出奇的居然讲业务，到稍微想看的知识点往往几句话带过，完全是浪费时间

34、强烈推荐，一口气看完，如果之前看过这本书，项目会少踩很多坑

35、还OK

精彩书评

1、之所以上述所说，是因为看这本书的时候，总感觉怪怪的。因为在地铁上看完了，作者书中基本都是他自己工作中遇到的问题和坑，虽说这样会让人感觉找到了解决方案，可以再进行深入的研究，可是某些地方介绍的有点片面，仅仅是引用部分博客就以偏概全了。还有可能是涉及的内容大部分都是我自己已经踩过的坑，所以觉得学到的东西不太多。再说说值得一看的地方吧，首先也如前面提到的，书中内容基本都是作者工作之谈，所以有很实用的内容，推荐阅读章节：1. App竞品技术分析个人认为这是本书的精华，很少看到有人愿意这样详尽的介绍自己的“机密”经验，感谢作者的无私分享。2. 五、六、七章的异常和ProGuard介绍很详细，网上都是非常琐碎的介绍，推荐新人看看这部分，尤其是ProGuard，虽然现在第三方已经给出了完善的解决方案，帮我们做了这部分事情，不过了解最基本的原理才能学得更透彻。3. 项目、团队管理部分这部分是时间的积累才能收获的，提前学习了解以后我们必须经历的路没什么不好的。先大概就总结这么多吧。

章节试读

1、《App研发录：架构设计、Crash分析和竞品技术分析》的笔记-第71页

既然GlobalVariables是单例类，那就应该增加synchronized同步块防止多线程并发访问，原文没有考虑到这一点。

```
public class GlobalVariables implements Serializable, Cloneable {
    private static GlobalVariables instance;

    private GlobalVariables() {
    }

    public static GlobalVariables getInstance() {
        if (instance == null) {
            synchronized (GlobalVariables.this) {
                Object object = Utils.restoreObject(CACHE_DIR + TAG);
                if (object == null) {
                    object = new GlobalVariables();
                    Utils.saveObject(CACHE_DIR + TAG);
                    instance = object;
                }
            }
        }
        return instance;
    }
}
```

这样写才是真正的单例类。

2、《App研发录：架构设计、Crash分析和竞品技术分析》的笔记-第96页

5.2和5.3章节就是作者自己在自high，这种分析数据库crash信息的基础sql语言毫无意义，也没技术含量，还浪费两个章节的篇幅，无语。

也就是5.1节大概提了一下，在Application的onCreate函数里注册UncaughtExceptionHandler还有些意义。

3、《App研发录：架构设计、Crash分析和竞品技术分析》的笔记-第159页

作者提到：对于多进程App而言，还是需要ContentProvider。其实，ContentProvider也是CS架构，只是Binder封装了SQLiteOpenHelper的一系列操作，将db封装成单例，那ContentProvider操作数据库也是单例的。

4、《App研发录：架构设计、Crash分析和竞品技术分析》的笔记-第114页

指出两处错误：

1. Comparator：我认为应该是基于快排的产物。
2. compare方法里面，p1和p2应该是d1和d2

还有，其实比较算法可以简化啊，这种简单的升序或者降序，直接：

```
return d1-d2;
```

不就可以了。

5、《App研发录：架构设计、Crash分析和竞品技术分析》的笔记-第232页

不太清楚竞品分析里干嘛揉入大量IOS的介绍，而且还都是最浅显的基本使用，浪费篇幅，看起来也没啥兴趣。

6、《App研发录：架构设计、Crash分析和竞品技术分析》的笔记-第89页

12条：为了节省内存，请使用ArrayList<自定义实体>，而不是HashMap。理由是：HashMap占用的内存多。

不太清除作者为什么会有这种结论，或者作者是否研究过HashMap的源码。

HashMap采用的数据结构是：数组+链表。之所以HashMap的查找时间复杂度是O(1)，是因为数组的下标是key的hash值，所以查找速率快。但是，不意味着数据多，HashMap分配的内存就要大。

7、《App研发录：架构设计、Crash分析和竞品技术分析》的笔记-第174页

指出一个ProGuard使用的错误，保留Parcelable序列化的类不被混淆,应该使用

```
-keepclassmembers class * implements android.os.Parcelable {  
    static android.os.Parcelable$Creator CREATOR;  
}
```

而不是用-keep

8、《App研发录：架构设计、Crash分析和竞品技术分析》的笔记-第94页

1. 比较无语，第六章已经介绍了不能在子线程中更新UI，作者还自己搞了个线程,在线程里显示了一个Toast，真bug。

2. 首先实现一个UncaughtExceptionHandler,应该是单例类,示例代码如下：

```
public class CrashHandler implements UncaughtExceptionHandler {  
    /** CrashHandler实例 */  
    private static CrashHanlder mInstance;  
    /** 系统默认的UncaughtException处理类 */  
    private Thread.UncaughtExceptionHandler mDefaultHandler;  
  
    private CrashHandler(Context context) {  
        // 保留系统默认处理类,便于自己不处理时进行调用  
        mDefaultHandler = Thread.getDefaultUncaughtExceptionHandler();  
        // 设置自己为当前线程的未捕获异常处理类  
        Thread.setDefaultUncaughtExceptionHandler(this);  
    }  
    public static CrashHandler getInstace(Context context) {  
        if (mInstance == null) {
```

```
synchronized(CrashHandler.class) {
    if (mInstate == null)
        mInstate = new CrashHandler(context);
}
}
}

@Override
public void uncaughtException(Thread thread, Throwable ex) {
}
}
```

在Application中注册:

```
public class App extends Application{
    @Override
    public void onCreate() {
        super.onCreate();
        //注册crashHandler
        CrashHandler crashHandler = CrashHandler.getInstance(getApplicationContext());
    }
}
```

9、《App研发录：架构设计、Crash分析和竞品技术分析》的笔记-第151页

作者说这一章花费了巨大的精力，但是其实很多Exception都是一笔带过，并没有深入讲解。

例如APK动态注册BroadcastReceiver，为什么会出现SecurityException?

10、《App研发录：架构设计、Crash分析和竞品技术分析》的笔记-开篇题记

APP研发录读书学习，感谢豆瓣读书笔记开发者。

《App研发录：架构设计、Cras》

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:www.tushu111.com