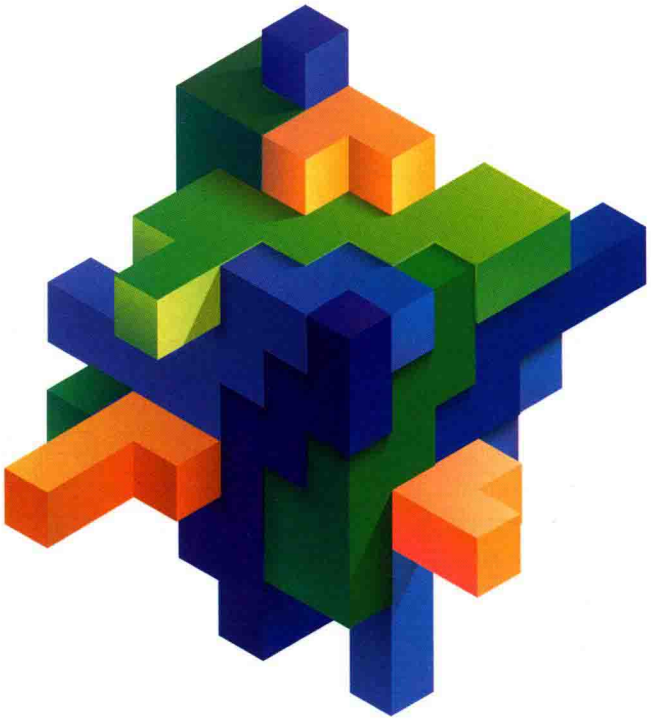
大公司的组件化指南，小公司的组件化革命
带你领略Android的组件化世界

Broadview®
www.broadview.com.cn



Android 组件化架构

苍王 著

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Android
组件化架构

苍王 著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书首先介绍组件化开发的基础知识，剖析组件化的开发步骤和常见问题，探究组件化编译原理和编译优化措施。其次在项目架构上，介绍如何组织团队来使用组件化开发，并将业务和人力进行解耦。最后深入介绍组件化分发技术及运用，探讨组件化架构的演进及架构的思维。

本书适合从事 Android 组件化技术研究，想在 Android 应用开发上进阶，以及有兴趣研究架构思维的 Android 开发者阅读。

图书在版编目（CIP）数据

Android 组件化架构 / 苍王著. —北京：电子工业出版社，2018.3
ISBN 978-7-121-33677-5

Ⅰ. ①A… Ⅱ. ①苍… 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（2018）第 028368 号

责任编辑：陈晓猛
印　　刷：三河市双峰印刷装订有限公司
装　　订：三河市双峰印刷装订有限公司
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：19.75　　字数：379.2 千字
版　　次：2018 年 3 月第 1 版
印　　次：2018 年 3 月第 1 次印刷
定　　价：79.00 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 zlts@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。

前　言

这是一本关于 Android 组件化的书籍

这是一本关于 Android 入门的书籍。

这是一本关于 Android 进阶的书籍。

这是一本关于 Android 编程原理的书籍。

这是一本关于 Android 架构的书籍。

我更愿意将这本书看作一本关于思维哲学的书籍。

书的用途，因人而异，有人用来垫书桌，有人将其作为工具，有人将其细细品味……

你用什么角度和什么态度来看待图书，它就会以什么形态展现在你眼前。

- 当你将它作为一本 Android 工具书时，它会指导你对 Android 的进阶学习。
- 当你将它作为一本软件架构书籍时，它会将工具和人的思想关联来调整你对架构的认知。
- 当你将它作为一本思维哲学书籍时，你有可能对 Android 开发有新的认识。

本书概要

第 1 章：组件化基础。

本章重点介绍组件化中开发的基础概念。首先介绍组件化中的依赖和解耦，然后介绍组件化中 AndroidManifest 的合成差异，最后深度认识 Application 的重要作用。

第 2 章：组件化编程。

本章介绍组件化中相关的开发编程技术，包括组件化通信、组件化存储、跨模块跳转、资

源冲突解决、多模块渠道、资源混淆、数据库运用、签名相关的运用及原理剖析。

第 3 章：组件化优化。

本章介绍如何使用 Gradle 对组件化中多种使用方式的优化，以及对编译适配的优化。随后介绍使用 Git 仓库来组织多人进行组件化开发，以及多人开发的项目解耦。

第 4 章：组件化编译。

本章介绍如何在组件化项目中缩短编译时间。首先介绍 Gradle 的打包流程，以及 Gradle 构建基础。随后介绍 Instant Run 的使用和适用场景。最后介绍 Freeline 增量编译，以及部分原理剖析。

第 5 章：组件化分发。

本章介绍如何在单页面中处理复杂的业务逻辑。首先介绍 Activity、Fragment、View 的生命周期，以及组件化分发架构的嵌入。随后介绍依赖倒置型的设计和层级问题的解决方法，其中插叙了编译期注解的高效生成代码的形式。最后介绍动态加载配置的形式。

第 6 章：组件化流通。

本章介绍如何在组件化中工程封装工具 SDK。首先介绍 Maven 基础和组件化中的缓存策略，随后介绍组件化中 SDK 的合成方式，最后介绍如何将 SDK 发布到流通平台中。

第 7 章：架构模板。

本章介绍如何制定组件化多人开发规范。首先介绍自定义 Android Studio 的模板及组件化模板的制作，随后介绍注解提示的使用。

第 8 章：架构演化。

本章介绍 Android 工程架构的演化，包括线程工程架构、组件化基础架构、模块化架构、多模板架构，以及进程化架构的原理基础。让读者能对 Android 架构有更加深入的了解。

读者对象

本书适合以下学习阶段的读者阅读：

- Android 进阶学习阶段；
- Android 组件化学习阶段；
- Android 架构初级学习阶段
- 移动端开发思维哲学学习阶段。

致谢

感谢父母对我的思想启蒙的培育；感谢我的妻子丸子对我写作的鼓励和生活的照顾；感谢我曾经就职的广州三星和现在在职的欢聚时代。感谢 Android 组件化架构 QQ 群中的映客科技 King、搜狐 56 夜闪冰、RetroX、亚伦，以及各位同学对我出版书籍内容上的建议。

勘误和互动

如果读者发现本书中文字、代码和图片的信息存在错误或者纰漏，欢迎反馈给我。若是对书中内容或者 Android 组件化架构存在疑问，可以在我的简书、掘金、QQ 群中与我互动，届时会在这些平台发布勘误的信息，并欢迎各位读者的提问和建议。

QQ 群：316556016

简书：http://www.jianshu.com/u/cd0fe10b01d2

掘金：https://juejin.im/user/565c6d3100b0acaad47e9050

GitHub：https://github.com/cangwang

苍王

2017 年 12 月 25 日于广州

读者服务

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。

- **下载资源**：本书如提供示例代码及资源文件，均可在下载资源处下载。
- **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/33677

目　录

第 1 章
组件化基础

万尺大楼始于足下，根基打得越深，构造的大楼越坚固。

深埋根下的事情，不显于表，并不是所有人都能看到大楼的整个构造过程，往往需要初始的建造图纸和空间的思维想象，才能理解以往事情发生的轨迹。

建造的图纸，很有可能只对内部人员提供，并且只标注关键信息（说明书），并不能完全描述每个建造细节。外部人员想要解析某些节点上的构造，就需要实地考察，通过自身经验来绘制局部的建造图纸，深化对局部的描述信息，才能定位具体的问题节点。

每种技术的基础都需要有基本学科理论的支撑，基本学科的理论基础是每时每刻都在运用的。

你认为了解的事情，在没其他基础支撑的时候，你看到的世界很可能还是表层，当你拥有更强有力的基础支撑后，你会在熟悉的场景中发现更多未知的特征节点，这些节点再次连接起来将是更深层的景象。

1.1 你知道组件化吗

组件化是什么？组件化的定义是什么？组件化是什么时候形成的？

在项目开发中，一般会将公用的代码提取出来用于制作基础库 Base module，将某些单独的功能封装到 Library module 中，根据业务来划分 module，组内的每个人分别开发各自的模块，如图 1-1 所示。

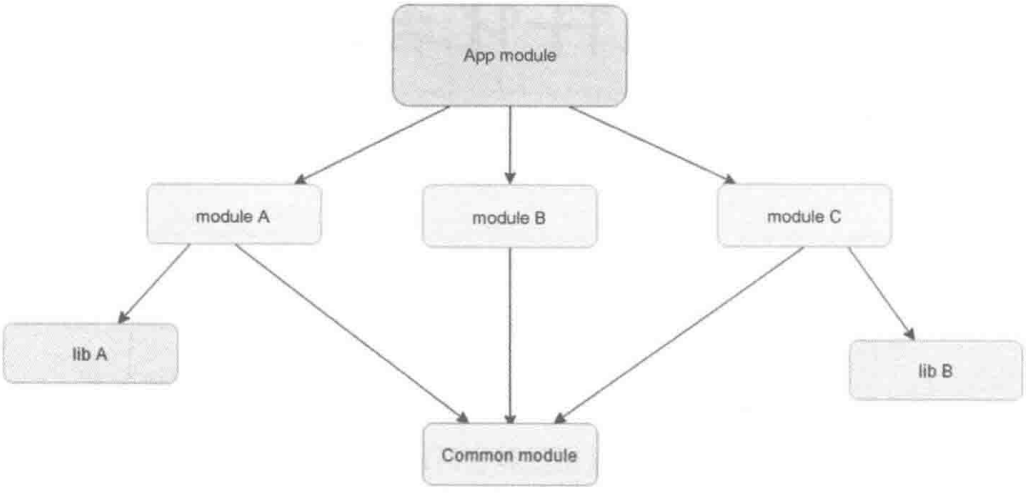


图 1-1 项目初始架构

随着时间的推移，项目迭代的功能越来越多。扩展了一些业务模块后，互相调用的情况就会增多，对某些库也增加了扩展和调用。工程的架构很可能如图 1-2 所示。

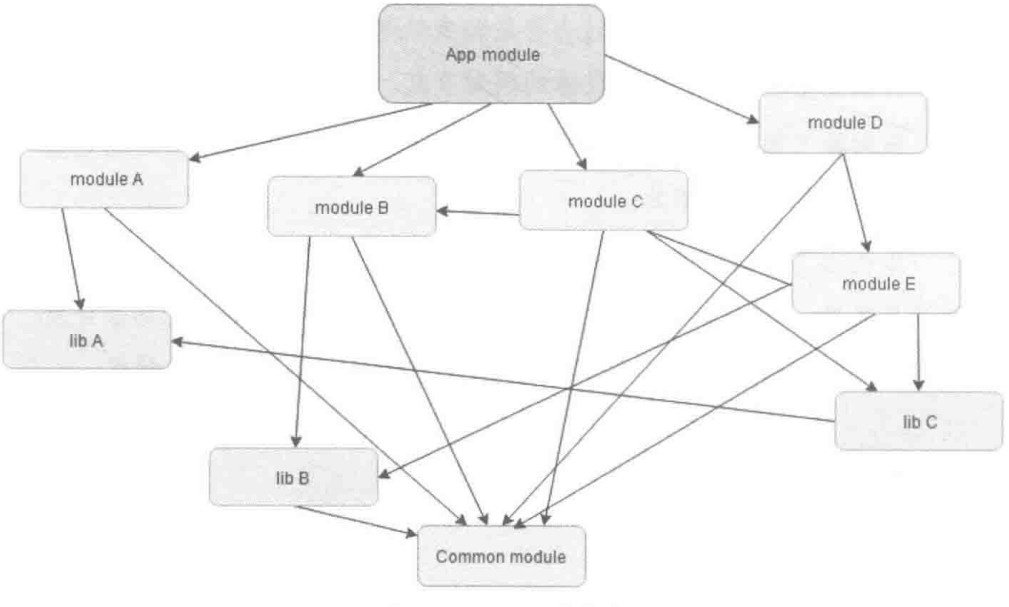


图 1-2 项目迭代架构

可以看出，各种业务之间的耦合非常严重，导致代码非常难以维护，更难以测试，扩展性和维护性非常差，这样的架构毫无章理可言，最后肯定会被替代。

这时新的规划规则出现了，这就是组件化、模块化，以及插件化。

多 module 划分业务和基础功能，这个概念将作为组件化的基础。

组件：指的是单一的功能组件，如视频组件（VideoSDK）、支付组件（PaySDK）、路由组件（Router）等，每个组件都能单独抽出来制作成 SKD。

模块：指的是独立的业务模块，如直播模块（LiveModule）、首页模块（HomeModule）、即时通信模块（IMModule）等。模块相对于组件来说粒度更大，模块可能包含多种不同的组件。

组件化开发的好处：

（1）避免重复造轮子，可以节省开发和维护的成本。

（2）可以通过组件和模块为业务基准合理地安排人力，提高开发效率。

（3）不同的项目可以共用一个组件或模块，确保整体技术方案的统一性。

（4）为未来插件化共用同一套底层模型做准备。

模块化开发的好处：

（1）业务模块解耦，业务移植更加简单。

（2）多团队根据业务内容进行并行开发和测试。

（3）单个业务可以单独编译打包，加快编译速度。

（4）多个 App 共用模块，降低了研发和维护成本。

模块化和组件化的缺点在于旧项目重新适配组件化的开发需要相应的人力及时间成本。

组件化和模块化的本质思想是一样的，都是为了代码重用和业务解耦。区别在于模块化是业务导向，组件化是功能导向。

项目体积越来越大后，必定会有超过方法数 65535 的一天，要么选择 MultiDex 的方式分包解决，要么使用插件化的方式来完成项目。

组件化和模块化的划分将更好地为项目插件化开路。插件化的模块发布和正常发布有着非常大的差异，已经脱离了组件化和模块化的构建机制，插件化拥有更高效的业务解耦。

1.2 基础组件化架构介绍

用语言来形容一个抽象的架构并不容易理解。我们用一个非常基础的组件化架构图来解释组件化基础，如图 1-3 所示。

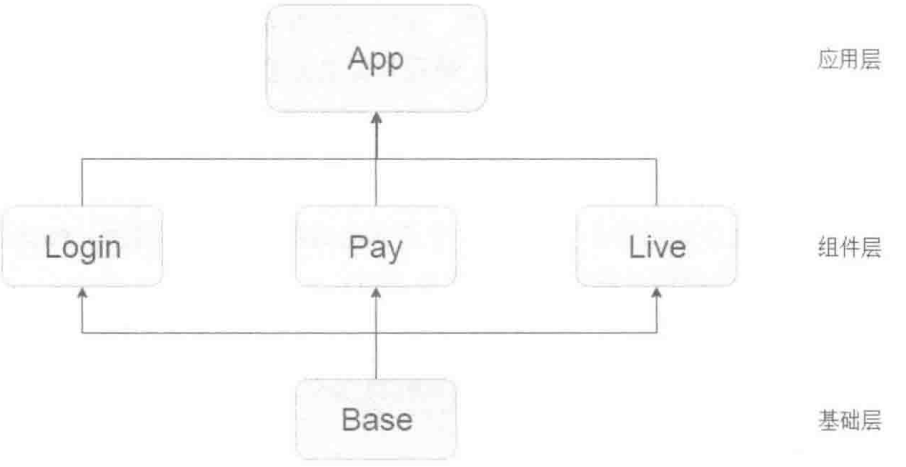


图 1-3 基础的组件化架构

上面的架构图从上到下分为应用层、组件层和基础层。

（1）基础层包含一些基础库和对基础库的封装，包括图片加载、网络加载、数据存储等。

（2）组件层包含一些简单的业务，比如登录、数据观看、图片浏览等。

（3）应用层用于统筹全部组件，并输出生成 App

虽然看起来这个基础组件化的架构非常简单、简陋，但是已经包含了组件化的内涵在里面。在这个基础的架构上，将会衍生出更多精细的架构，在之后的章节中会深入分析。

1.2.1 依赖

Android Studio 独有设计——module 依赖。

module 的依赖包括对第三方库的依赖，也包含对其他 module 的依赖。通过依赖我们可以访问第三方和其他被依赖 Module 的代码和资源。

基本的三种依赖方式如图 1-4 所示。

（1）Jar dependency：通过 Gradle 配置引入 lib 文件夹中的所有.jar 后缀的文件，还能引用.aar 后缀的文件。顺便提一下，这是 lib module 打包的独有格式文件类型。

（2）Base module：相当于一个基础模块库（lib module）来作为依赖，对应的是 module dependency，实质上是将其打包成 aar 文件，方便其他库进行依赖。

（3）第三方依赖通过 Library dependency 完成仓库索引依赖，这里的仓库可以配置为网络库和本地库。

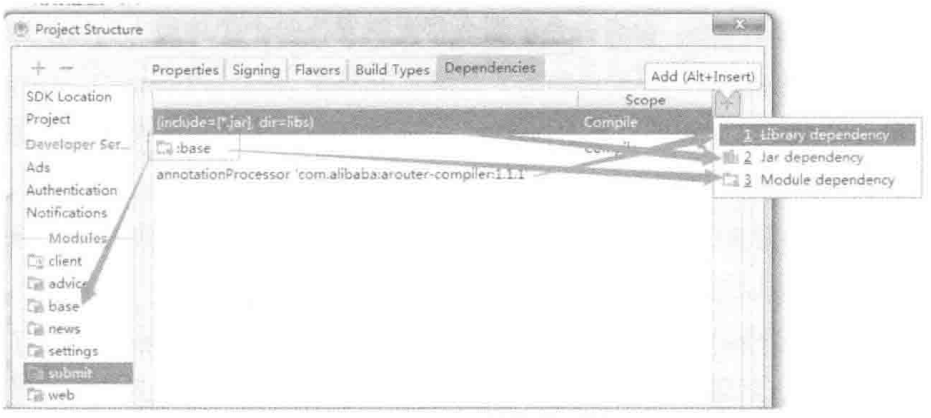


图 1-4 依赖方式

全部工具作用的配置最后还是会转化为代码的形式存在于我们配置的 build.gradle 中。

```
dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    compile project(':base')
    annotationProcessor 'com.alibaba:arouter-compiler:1.1.1'
}
```

这里可以很清晰地看到，其代码分别对应三种不同的资源加入方式。

一般情况下，Android Studio 定义使用 dependencies 包含全部资源引入，使用 compile 字段提示来引入库。

这里值得注意的是：

（1）读入自身目录使用的是 fileTree。

（2）读入其他资源 module 使用的是“project”字段，而“:base”中冒号的意思是文件目录内与自己相同层级的其他 module。

（3）有些人会觉得奇怪，为何“annotationProcessor”字段不是 Android Studio 原生的，而是使用 Gradle 自定义的字段？这里的作用和 compile 类似。

1.2.2 聚合和解耦

为何要介绍依赖？

依赖→关系

有依赖才能产生关系。一个工程体系我们可以想象成一个群体，如果一个群体中的每个个

体彼此都非常陌生，没有沟通，没有关系，那么将无法发挥每个个体最大的做事能力。所以我们需要为个体提供沟通交流的渠道。

Android Studio 正是以依赖的方式给每个 module 之间提供了沟通和交流的渠道，从而形成聚合。

如果一个个体和非常多的个体交流，则使用依赖关系。一个个体为非常多的模块提供服务，这样效率是很高的，而且通信成本很低。

假如有一天，这个个体出逃了，或者上司生气了，说以后都不需要这个个体了，要移除这个个体和其他人的关系，后果将是灾难性的，其他一堆人会跟你抱怨："他为我们做了太多事情，离开他我们找不到其他人做事了"。这是非常令人苦恼的。

所以作为这个运行系统的设计者和统筹者，我们应该设计一个为移除或者替换某个个体的行为付出最少代价的方案。

这就是：关系→解耦。

既然需要解耦，那么我们就需要设计更加适合交流沟通的系统，也可以考虑为一个群体设计更加适合的交流沟通的方式。

聪明的工程师和架构师在 Android Studio 中发现了适合时代趋势的组件化架构。

聚合和解耦是项目架构的基础。

以上的列子简单介绍了聚合和解耦的概念，工程的架构和集体的连接关系有着非常相像的地方。架构的实质也可以想象为人与人之间关系的连接。在开发过程中，有的人看到的或许只是程序本身，并没想到很多架构方面的内容，而架构上的思想和人类活动是非常相似的。架构工程就是一个活生生的群体集合，如何让每个个体产生最大的作用；如何让个体间的交流通畅；如何让每个个体付出最小的消耗来完成任务，并获得更大的集体利益；如何统筹更大规模的集体——这些就是架构师毕生探讨的论题，也是架构师的乐趣所在。

组件化架构就是在文件层级上有效地控制沟通和个体独立性的做法。

1.3 重新认识 AndroidManifest

当我们开始学习 Android 的时候，第一个认识的除了 Activity，估计就是 AndroidManifest 了。

AndroidManifest 是什么？有什么作用？

估计很多人很快就理解了 AndroidManifest 其实就是 Android 项目的声明配置文件。manifest 的字面意思是货单、旅客名单（直接翻译有点搞笑，意思是我们要载着配置文件去旅行了）。

既然这份货单用来声明 Android 工程里的一些文件信息，那么什么文件会入选这个货单呢？