# Android 架构说明

编写日期：2018.1.20

编写人：温苏彬

[Android 架构说明 0](file:////Users/vane/Documents/dankal-doc/草稿/Android架构技术文档.docx#_Toc504418528)

[架构 2](#_Toc504418529)

[MVP 模式 2](#_Toc504418530)

[官方应用架构指南 3](#_Toc504418531)

[现存的问题 3](#_Toc504418532)

[架构原则 3](#_Toc504418533)

[新的架构组件 4](#_Toc504418534)

[语言 6](#_Toc504418535)

[项目组件化方案 6](#_Toc504418536)

[版本控制工作流 7](#_Toc504418537)

[框架 7](#_Toc504418538)

[第三方解决方案 8](#_Toc504418539)

# 架构

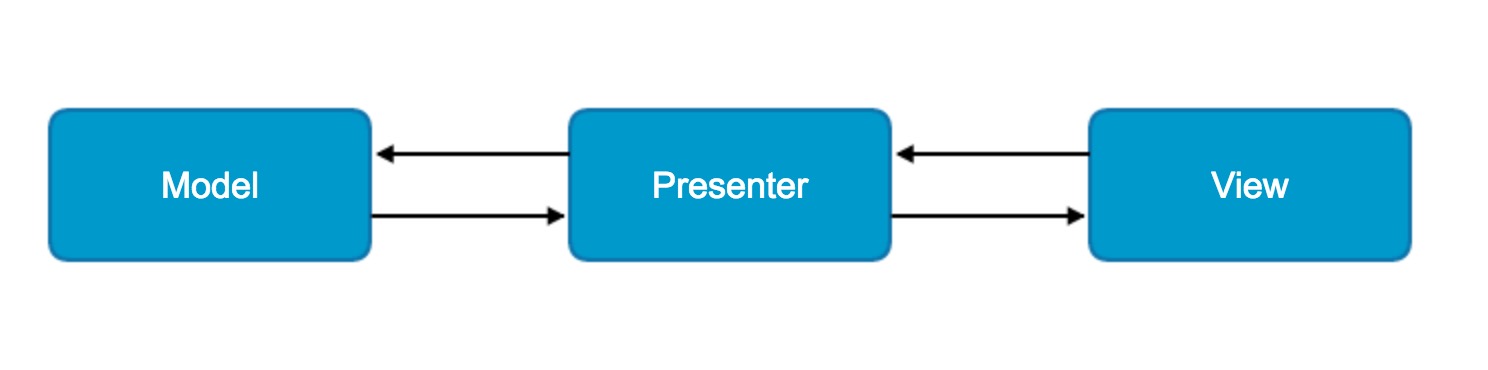
MVP、MVVM本质上都是一种代码架构思想，并没有明显的孰优孰劣，关键看应用的场景以及实现的细节。而项目结构的简单与否也决定是否使用组件化方案，盲目使用组件化只会增加项目复杂度。

## MVP 模式

对于一个应用而言我们需要对它抽象出各个层面，而在MVP架构中它将UI界面和数据进行隔离，所以我们的应用也就分为三个层次。

* View: 对于View层也是视图层，在View层中只负责对数据的展示，提供友好的界面与用户进行交互。在Android开发中通常将Activity或者Fragment作为View层。
* Model: 对于Model层也是数据层。它区别于MVC架构中的Model，在这里不仅仅只是数据模型。在MVP架构中Model它负责对数据的存取操作，例如对数据库的读写，网络的数据的请求等。
* Presenter:对于Presenter层他是连接View层与Model层的桥梁并对业务逻辑进行处理。在MVP架构中Model与View无法直接进行交互。所以在Presenter层它会从Model层获得所需要的数据，进行一些适当的处理后交由View层进行显示。这样通过Presenter将View与Model进行隔离，使得View和Model之间不存在耦合，同时也将业务逻辑从View中抽离。

下面通过MVP结构图来看一下MVP中各个层次之间的关系。



在MVP架构中将这三层分别抽象到各自的接口当中。通过接口将层次之间进行隔离，而Presenter对View和Model的相互依赖也是依赖于各自的接口。这点符合了接口隔离原则，也正是面向接口编程。在Presenter层中包含了一个View接口，并且依赖于Model接口，从而将Model层与View层联系在一起。而对于View层会持有一个Presenter成员变量并且只保留对Presenter接口的调用，具体业务逻辑全部交由Presenter接口实现类中处理。

## 官方应用架构指南

不可能找到一个完美的方案使用于所有场景。这里的建议架构也只是对于大部分场景来说应该是一个好的开始。这里对官方架构指南做一个概述，具体参考(https://developer.android.google.cn/topic/libraries/architecture/guide.html#the\_final\_architecture)

### 现存的问题

相对了传统的桌面PC开发，Android 应用的结构更加复杂。一个典型的 Android 应用由多个应用组件组合构建而成，其中包括 Activities，Fragments，Services，ContentProviders 和 BroadcastReceivers。这些应用组件的创建和销毁是不完全可控的，它可能在任何时候由于用户或者系统的行为而触发。应用组件的生命周期不是完全由你掌控的，所以你不应该存储一些数据或者状态在你的应用组件中，应用组件之间也不应该彼此依赖。

### 架构原则

如果不能用应用组件去存储应用数据和状态，那应该怎样去设计应用的架构呢？

首先，通常的架构原则有几个重点：

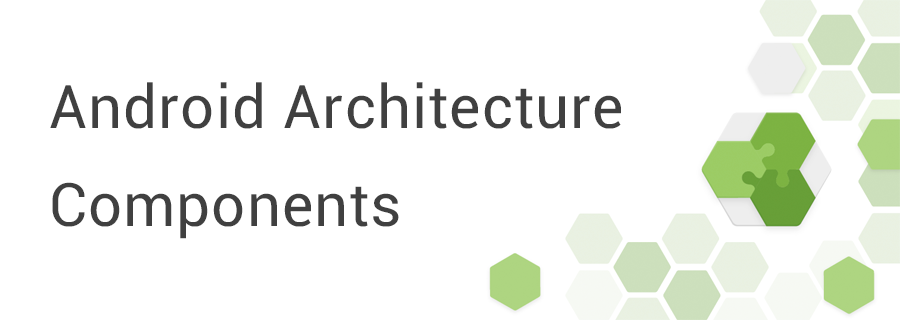
第一个重点是在应用中的关注点分离。比如在一个 Activity 或者 Fragment 中写所有的代码这明显是错误的。任何与 UI 或者交互无关的代码都不应该存在这些类中。保证他们尽可能的职责单一化将会使我们避免很多生命周期相关的问题。Android 系统可能会随时由于用户的行为或者系统状态（比如剩余内存过低）而销毁你的应用组件。所以应该最小化应用组件之间的依赖以提供一个健壮的体验。

第二个重点是我们应该采用数据模型驱动 UI 的方式，最好是一个可持久化的模型。持久化被建议的原因有两个：

1. 用户不会因为系统销毁我们的应用而导致丢失数据。
2. 我们的应用可以在网络状况不好甚至断网的情况下继续工作。

这里说的模型其实也是一种组件，他们就是专门负责为我们的应用处理和存储数据的。他们完全独立于 Views 和其他应用中的组件，所以他们不存在生命周期相关的问题。保证 UI 部分的代码足够简单，没有业务逻辑，使代码更容易去管理。

### 新的架构组件



**Android Architecture Components**是一个由官方推出的新库，它能够帮助你去构建一个健壮，易测，可维护的应用。

* Lifecycle

生命周期管理，把原先Android生命周期的中的代码抽取出来，如将原先需要在onStart()等生命周期中执行的代码分离到Activity或者Fragment之外。

* LiveData

一个数据持有类，持有数据并且这个数据可以被观察被监听，和其他Observer不同的是，它是和Lifecycle是绑定的，在生命周期内使用有效，减少内存泄露和引用问题。

* ViewModel

用于实现架构中的ViewModel，同时是与Lifecycle绑定的，使用者无需担心生命周期。可以在多个Fragment之间共享数据，比如旋转屏幕后Activity会重新create，这时候使用ViewModel还是之前的数据，不需要再次请求网络数据。

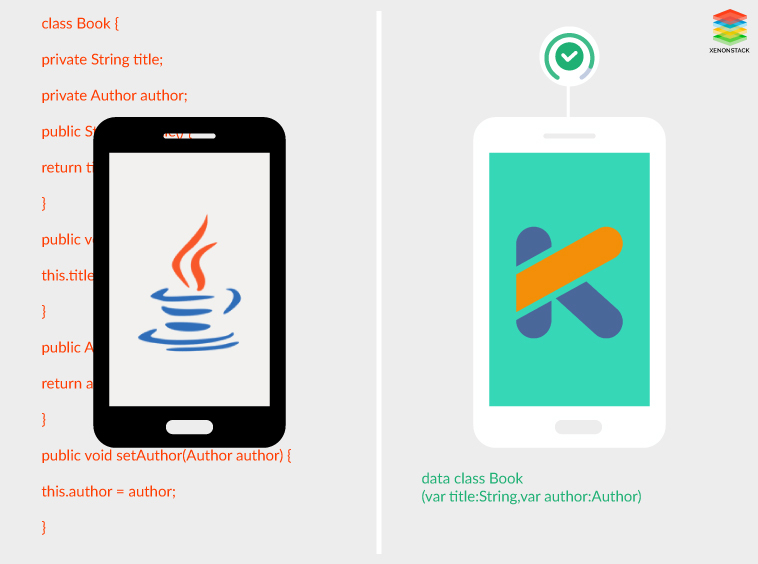
* Room

谷歌推出的一个Sqlite ORM库，不过使用起来还不错，使用注解，极大简化数据库的操作，有点类似Retrofit的风格。

##### 而最终实现的架构图应该是这样的：



# 语言



当我们考虑开发一个Android应用时，有可能立即出现一种编程语言：Java。尽管大多数Android应用程序都是用Java编写的，但在Android开发方面，Java并不是唯一的选择。我们可以使用任何可以在Java虚拟机（JVM）上编译和运行的语言编写Android应用程序。而一个真正引起Android社区关注的JVM兼容编程语言是Kotlin。蛋壳在开发Android程序应用时依然首选Java语言，尽管谷歌宣布kotlin成为Android开发的官方语言并且语言特性等占有优势，但Java语言更加成熟，除了基础类库完善，更有丰富高效的第三方组件。蛋壳Android开发组对kotlin持有积极学习和观望态度，以备后用。

# 项目组件化方案

**组件化**的目的是告别项目结构臃肿，让各个业务变得相对独立，业务组件在组件模式下可以独立开发，而在集成模式下又可以变为arr包集成到“app壳工程”中，组成一个完整功能的APP。组件化开发可以加快业务迭代速度，各个业务模块组件更加独立，不再出现业务耦合情况；可以减少重复开发和维护工作量；可以降低团队成员熟悉项目的成本，降低项目的维护难度。

具体实现细节和说明见《Android组件化开发方案》

# 版本控制工作流

工作流有各式各样的用法，但也正因此使得在实际工作中如何上手使用变得很头大。常见的版本控制工作流有 集中式g工作流、功能开发工作流、Gitflow 工作流、Forking 工作流。蛋壳开发团队在尝试过以上四种工作流后总结出了属于我们自己的工作流。该工作流基于Gitflow工作流做出适当流程修改，在保持项目网络清晰的条件下保证项目版本的完整和安全管理。更加详细请参考：<https://blog.bingo.ren/40.html>和《Git操作规范》

# 框架

(持续更新)

|  |  |
| --- | --- |
| **应用** | **名称** |
| **响应式编程** | RxJava、RxAndroid、RxBinding |
| **网络层** | Retrofit+OkHttp |
| **图片加载库** | Glide/Picasso |
| **Java Json解析/生成 库** | Fastjson |
| **视图绑定注解** | ButterKnife |
| **EventBus** | Otto/RxBus |
| **二维码** | zxing |
| **内存泄漏检测** | leakcanary |
| **热修复** | Tinker |
| **图表** | MPAndroidChart |
| **列表刷新加载库** | SwipeLayout/SmartRefreshLayout |
| **路由** | ARouter |
| **动态布局** | VirtualView系列 |

# 第三方解决方案

|  |  |
| --- | --- |
| **功能模块** | **方案** |
| IM通讯 | 环信 |
| 地图 | 百度地图/腾讯地图 |
| 推送 | 极光推送/百度推送 |
| 异常反馈 | Bugly |
| 图床 | 七牛云 |
| 第三方登录、分享 | Mob-ShareSDK |