MVC,MVP,MVVM,MVPVM架构学习

1. 学习架构的目的

通过设计使程序模块化，做到模块内部的高聚合和模块之间的低耦合。这样做的好处是使得程序在开发的过程中，开发人员只需要专注于一点，提高程序开发的效率，并且更容易进行后续的测试以及定位问题。

1. 概述

MVC,MVP,MVVM以及MVPVM都是一种架构模式，为了解决图形界面中数据的展示以及通信而产生的应用架构模式。本文将阐述它们各自的特点，加深理解，以便后续能随心所欲的运用到项目开发中。

1. 发展历程

MVPVM

MVVM

MVP

MVC

1. MVC
   1. 概要:

在Android应用中,MVC分层大概如下所示:

View

Control

Model

Layout XML

Activity

DB

Fragment

View组件

Sharepref XML

JSON

模型层(Model)

针对业务模型，建立的数据结构和相关的类，它主要负责网络请求，数据库处理，I/O的操作。

视图层(View)   
对应于xml布局文件和java代码动态view部分

控制层(Controller)

由Activity来承担的，Activity本来主要是作为初始化页面，展示数据的操作，但是因为XML视图功能太弱，所以Activity既要负责视图的显示又要加入控制逻辑，承担的功能过多。

* 1. 应用场景:

从上述的分析中可以看出，MVC在Android应用中缺点比较明显，视图层和控制层过于紧密连接，分层不明显，业务逻辑和视图展示紧密联系在一起，造成控制层过分的臃肿(三者之间的关系一般都表现成如下所示的形式)，这种架构不适合在Android应用中，一般应用在WEB应用中。本文不再详细分析。

Model

Control/View

* 1. 缺点：

1. View持有Controller和Model的引用
2. 它没有把对UI逻辑的操作限制在单一的类里, 这个职能被Controller和View或者Model共享
3. MVP
   1. 概要

由于MVC中View层和Control层耦合度过高，在MVP里，Presenter完全把Model和View进行了分离，主要的程序逻辑在Presenter里实现。而且，Presenter与具体的View是没有直接关联的，而是通过接口进行交互，从而使得在变更View时候可以保持Presenter的不变，可以多次复用。

MVP(Model-View-Presenter)模式:

1. Model: 数据层. 负责与网络层和数据库层的逻辑交互.
2. View: UI层.

* 监听P层的数据更新通知, 刷新页面展示.
* 在点击事件触发时, 调用P层的对应方法, 并对方法执行结果进行展示.
* 界面元素布局和动画
* 反馈用户操作

1. Presenter:

* 实现view的事件处理逻辑，暴露相应的接口给view的事件调用
* 调用model的接口获取数据，然后加工数据，封装成view可以直接用来显示的数据和状态

View

Model

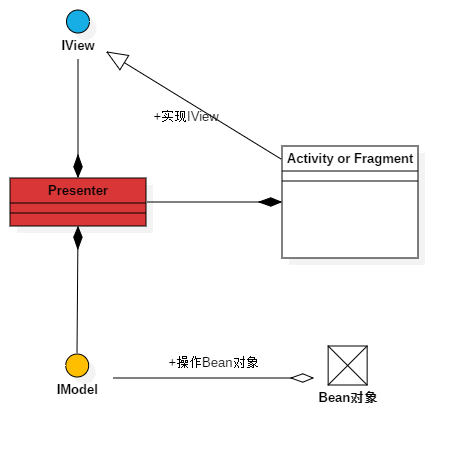
Presenter

* 1. 优点

1. 分离了视图逻辑和业务逻辑，降低了耦合
2. Activity只处理生命周期的任务，代码变得更加简洁
3. 视图逻辑和业务逻辑分别抽象到了View和Presenter的接口中去，提高代码的可阅读性
4. Presenter被抽象成接口，可以有多种具体的实现，所以方便进行单元测试
5. 把业务逻辑抽到Presenter中去，避免后台线程引用Activity导致Activity的资源无法被系统回收从而引起内存泄露和OOM
   1. 缺点

随着业务逻辑的增加，P和V的交互会越来越紧密，需要一种新的机制来管理两者之间的交互,为两者建立一一对应的关系

* 1. MVP实现DEMO总结



1. MVVM*(在MVP的基础上增加*Data B*inding的管理,未完待续)*
   1. 特点
   2. 优点
   3. 缺点
2. MVPVM*(未完待续)*
   1. 特点
   2. 优点
   3. 缺点
3. 参考文献:

# 谷歌开源示例: <https://github.com/googlesamples/android-architecture/tree/todo-mvp>

# MVP模式从入门到精通: <https://blog.csdn.net/qq_31852701/article/details/52946127>