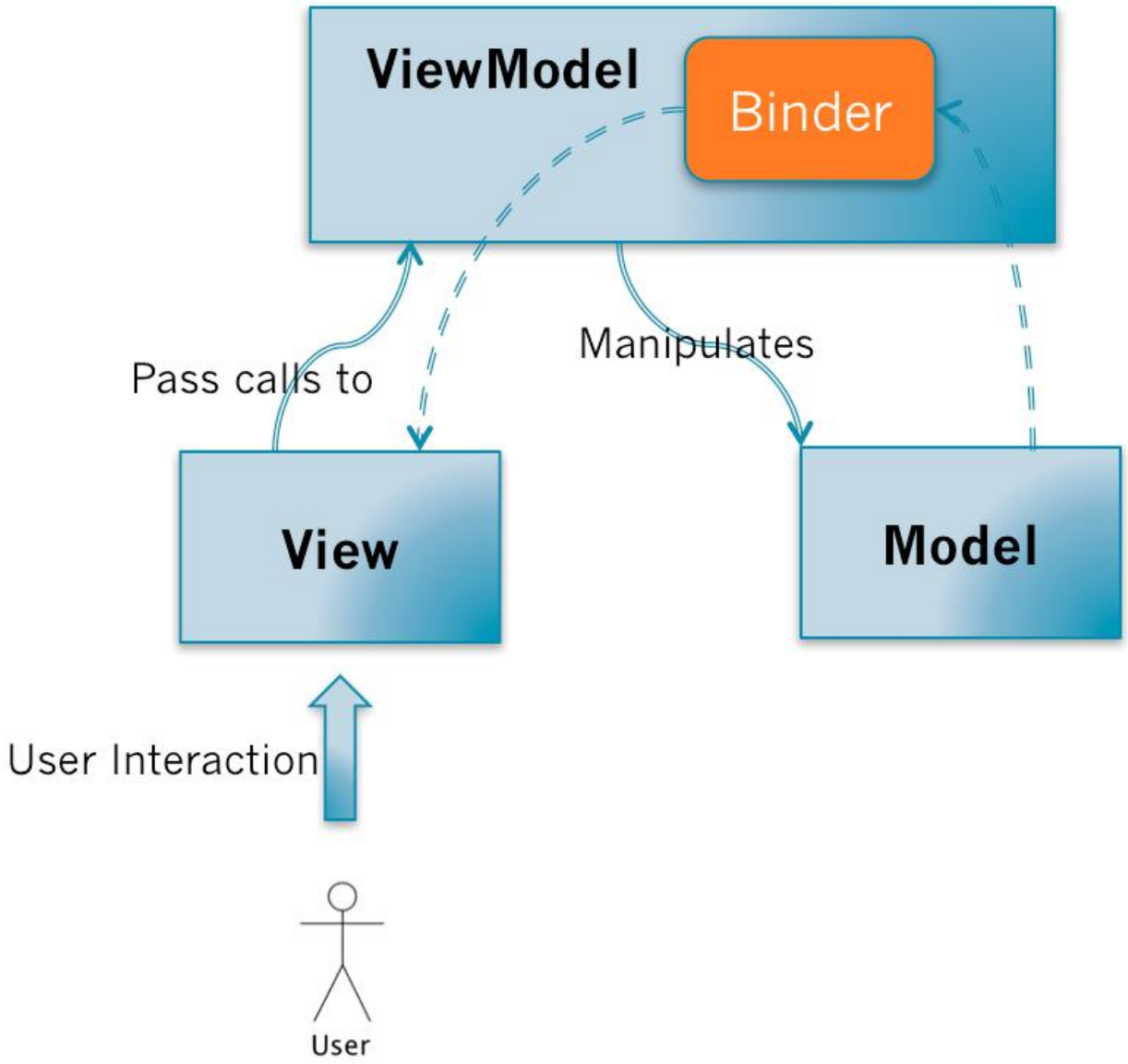
MVC

MVC 全名是 Model--View--Controller，是模型(model)－视图(view)－控制器(controller)的缩写，一种软件设计典范，用一种业务逻辑、数据、界面显示分离的方法组织代码，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。其中 Model 层处理数据，业务逻辑等；View 层处理界面的显示结果；Controller 层起到桥梁的作用，来控制 View 层和 Model 层通信以此来达到分离视图显示和业务逻辑层。



我们往往把 Android 中界面部分的实现也理解为采用了 MVC 框架，常常把 Activity 理解为 MVC 模式中的 Controller。

看似没有什么特别的地方，但是由几个需要特别关注的关键点：

1、View 是把控制权交移给 Controller，自己不执行业务逻辑。

2、Controller 执行业务逻辑并且操作 Model，但不会直接操作 View，可以说它是对 View 无知的。

3、View 和 Model 的同步消息是通过观察者模式进行，而同步操作是由 View 自己请求 Model 的数据然后对视图进行更新。

MVC 的优缺点

优点：

1、把业务逻辑全部分离到 Controller 中，模块化程度高。当业务逻辑变更的时候，不需要变更 View 和 Model， 只需要 Controller 换成另外一个 Controller 就行了（Swappable Controller）。

2、观察者模式可以做到多视图同时更新。

缺点：

- 1、Controller 测试困难。因为视图同步操作是由 View 自己执行，而 View 只能在有 UI 的环境下运行。在没有 UI 环境下对 Controller 进行单元测试的时候，

  Controller 业务逻辑的正确性是无法验证的：Controller 更新 Model 的时候，无法对 View 的更新操作进行断言。

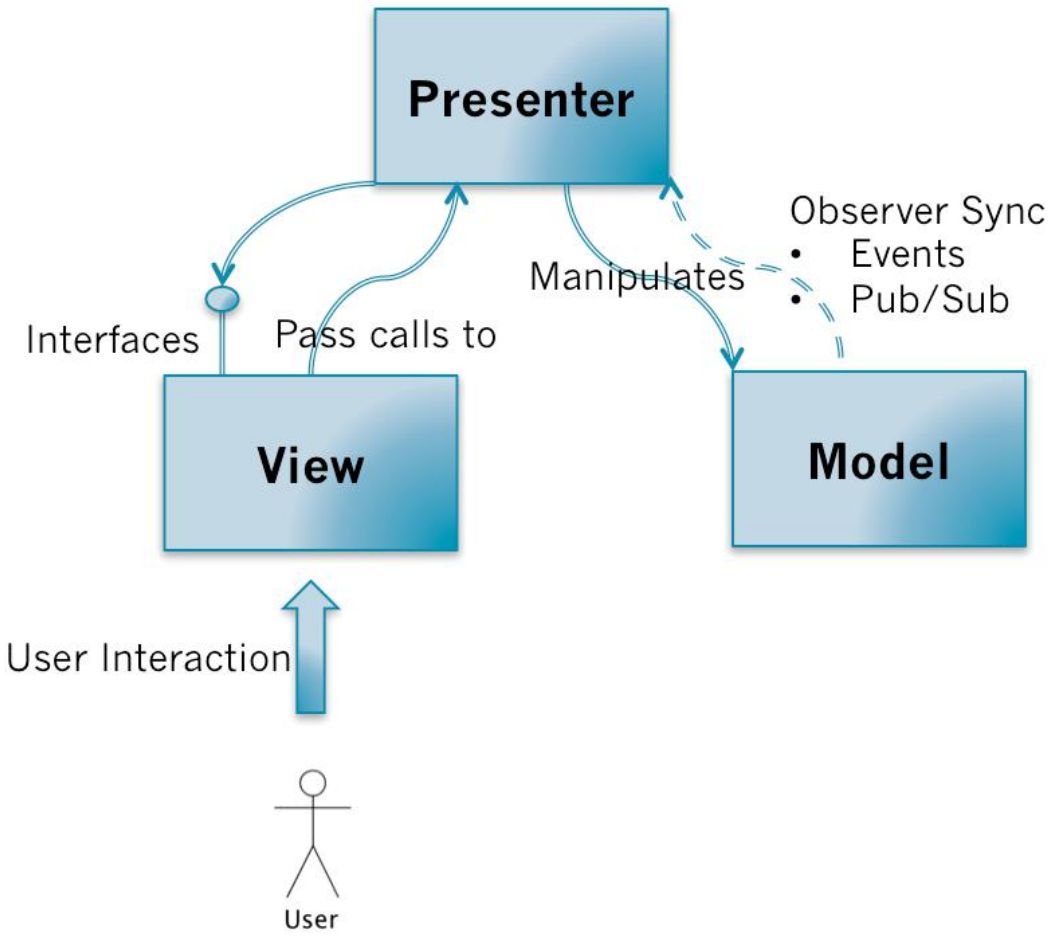
- 2、View 无法组件化。View 是强依赖特定的 Model 的，如果需要把这 View 抽出来作为一个另外一个应用程序可复用的组件就困难了。

  因为不同程序的的 Domain Model 是不一样的

MVP

MVP 其实是 MVC 的一种演进版本，它更简单，将 MVC 中的 Controller 改为了 Presenter，View 通过接口与 Presenter 进行交互，降低耦合，方便进行单元测试。

View：负责绘制 UI 元素、与用户进行交互(Activity、View、Fragment 都可以做为 View 层)；
Model：对数据的操作、对网络等的操作，和业务相关的逻辑处理；
Presenter：作为 View 与 Model 交互的中间纽带，处理与用户交互的逻辑。可以把 Presenter 理解为一个中间层的角色，它接受 Model 层的数据，并且处理之后传递给 View 层，还需要处理 View 层的用户交互等操作。



关键点：

1、View 不再负责同步的逻辑，而是由 Presenter 负责。Presenter 中既有业务逻辑也有同步逻辑。

2、View 需要提供操作界面的接口给 Presenter 进行调用。（关键）

对比在 MVC 中，Controller 是不能操作 View 的，View 也没有提供相应的接口；而在 MVP 当中，Presenter 可以操作 View，View 需要提供一组对界面操作的接口给 Presenter 进行调用；Model 仍然通过事件广播自己的变更，但由 Presenter 监听而不是 View。

MVP（Passive View）的优缺点

-

优点：

-

1、便于测试。Presenter 对 View 是通过接口进行，在对 Presenter 进行不依赖 UI 环境的单元测试的时候。可以通过 Mock 一个 View 对象，这个对象只需要实现了 View 的接口即可。然后依赖注入到 Presenter 中，单元测试的时候就可以完整的测试 Presenter 业务逻辑的正确性。

2、View 可以进行组件化。在 MVP 当中，View 不依赖 Model。这样就可以让 View 从特定的业务场景中脱离出来，可以说 View 可以做到对业务逻辑完全无知。它只需要提供一系列接口提供给上层操作。这样就可以做高度可复用的 View 组件。

-

缺点：

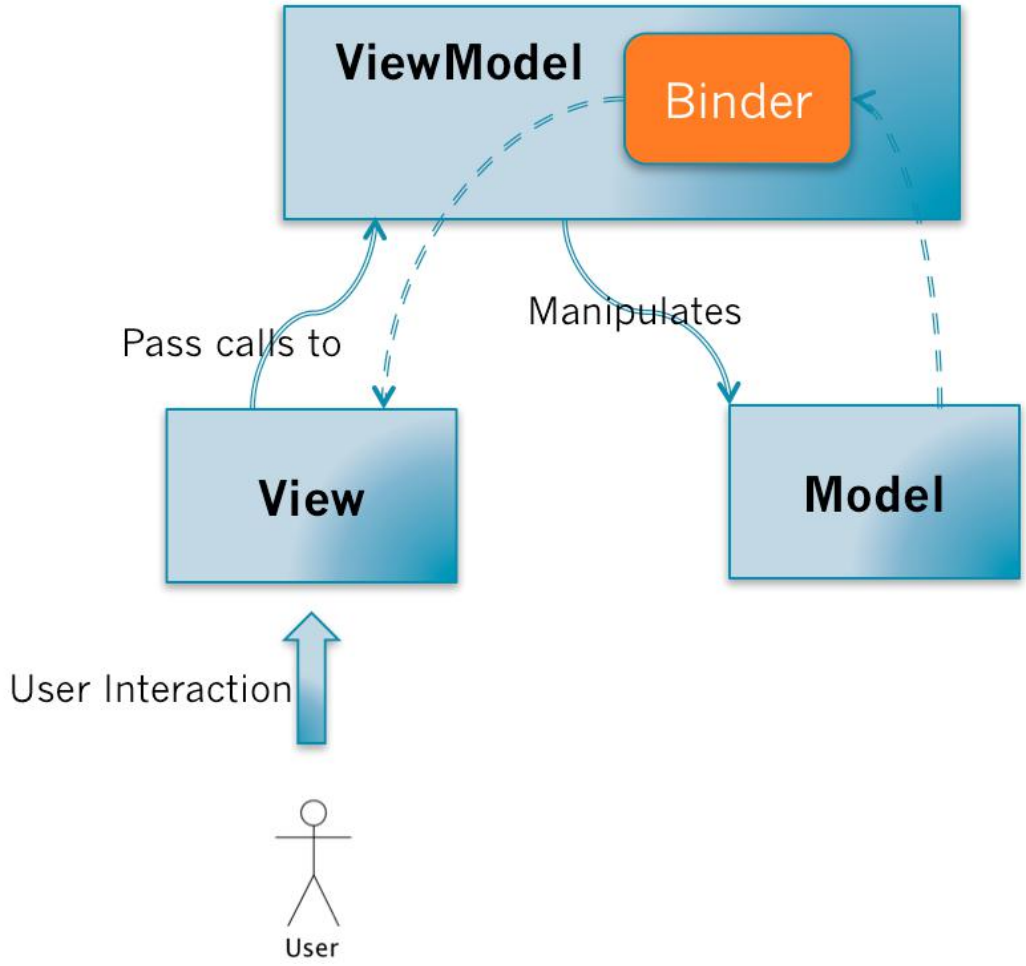
-

1、Presenter 中除了业务逻辑以外，还有大量的 View->Model，Model->View 的手动同步逻辑，造成 Presenter 比较笨重，维护起来会比较困难。

MVVM

MVVM 模式（Model--View--ViewModel 模式），和 MVP 模式相比，MVVM 模式用 ViewModel 替换了 Presenter ，其他层基本上与 MVP 模式一致，ViewModel 可以理解成是 View 的数据模型和 Presenter 的合体。

MVVM 采用双向绑定（data-binding）：View 的变动，自动反映在 ViewModel，反之亦然，这种模式实际上是框架替应用开发者做了一些工作（相当于 ViewModel 类是由库帮我们生成的），开发者只需要较少的代码就能实现比较复杂的交互。



MVVM 的调用关系

MVVM 的调用关系和 MVP 一样。但是，在 ViewModel 当中会有一个叫 Binder，或者是 Data-binding engine 的东西。以前全部由 Presenter 负责的 View 和 Model 之间数据同步操作交由给 Binder 处理。你只需要在 View 的模版语法当中，指令式地声明 View 上的显示的内容是和 Model 的哪一块数据绑定的。当 ViewModel 对进行 Model 更新的时候，Binder 会自动把数据更新到 View 上去，当用户对 View 进行操作（例如表单输入），Binder 也会自动把数据更新到 Model 上去。这种方式称为：Two-way data-binding，双向数据绑定。可以简单而不恰当地理解为一个模版引擎，但是会根据数据变更实时渲染。

关键点：

MVVM 把 View 和 Model 的同步逻辑自动化了。以前 Presenter 负责的 View 和 Model 同步不再手动地进行操作，而是交由框架所提供的 Binder 进行负责。

只需要告诉 Binder，View 显示的数据对应的是 Model 哪一部分即可。

MVVM 的优缺点

-

优点：

-

1、提高可维护性。解决了 MVP 大量的手动 View 和 Model 同步的问题，提供双向绑定机制。提高了代码的可维护性。

2、简化测试。因为同步逻辑是交由 Binder 做的，View 跟着 Model 同时变更，所以只需要保证 Model 的正确性，View 就正确。大大减少了对 View 同步更新的测试。

-

缺点：

-

1、过于简单的图形界面不适用，或说牛刀杀鸡。

2、对于大型的图形应用程序，视图状态较多，ViewModel 的构建和维护的成本都会比较高。

3、数据绑定的声明是指令式地写在 View 的模版当中的，这些内容是没办法去打断点 debug 的。

结语

可以看到，从 MVC->MVP->MVVM，就像一个打怪升级的过程。后者解决了前者遗留的问题，把前者的缺点优化成了优点。同样的 Demo 功能，代码从最开始的一堆文件，优化成了最后只需要 20 几行代码就完成。MV*模式之间的区分还是蛮清晰的，希望可以给对这些模式理解比较模糊的同学带来一些参考和思路。

收藏