# **android mvvm 官方例子,Android下的MVVM架构**



[荣耀张大仙](https://blog.csdn.net/weixin_26713059" \t "https://blog.csdn.net/weixin_26713059/article/details/_blank)IMG_257于 2021-05-25 21:00:35 发布IMG_2581169IMG_259 收藏

文章标签： [android mvvm 官方例子](https://so.csdn.net/so/search/s.do?q=android+mvvm+%E5%AE%98%E6%96%B9%E4%BE%8B%E5%AD%90&t=blog&o=vip&s=&l=&f=&viparticle=" \t "https://blog.csdn.net/weixin_26713059/article/details/_blank)

写在文章开头：由于自己是一名学习Android开发的咸鱼玩家，如果理解不到位或者出错的地方，欢迎大家指出错误或者提出宝贵的意见；另外文章可能涉及到各个途径来源的资源，如有侵犯，立刻删除。

概述

1 [MVC](https://so.csdn.net/so/search?q=MVC&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_26713059/article/details/_blank)

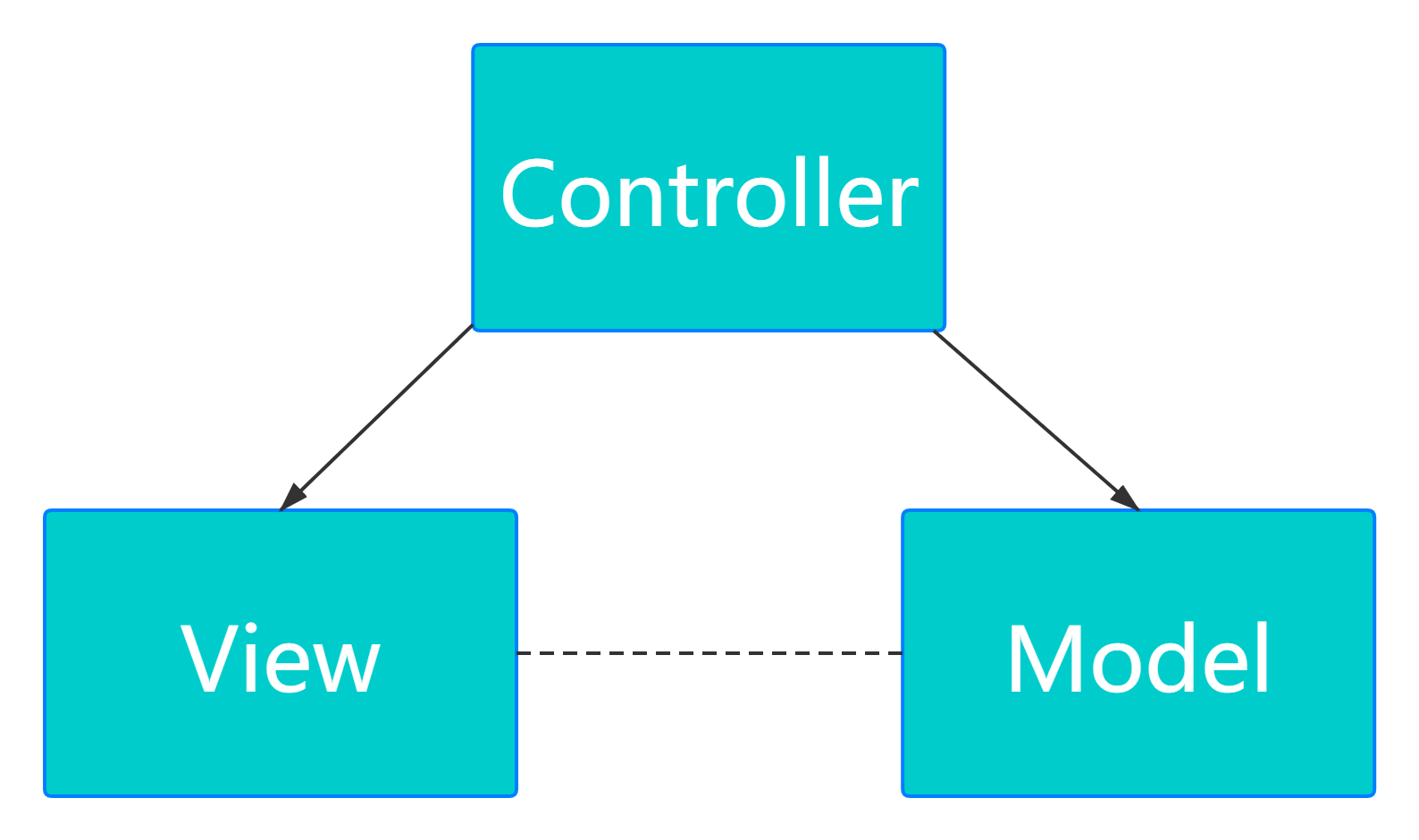
1.1 MVC含义

M([Model](https://so.csdn.net/so/search?q=Model&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_26713059/article/details/_blank)):模型，这个模块很抽象，但是总的来说，就是提供数据源的模块，它可以是本地数据库数据，可以是网络请求数据，可以是一切数据源。

V(View):视图，能够实现数据源有目的性的展示，简单来说，它是一个负责提供用户所见的模块。

C(Controller):控制器，负责Model和View的“交流”，能够处理事件并作出响应，这里的事件可以是数据源的更改或特定显示，也可以是用户的行为。

1.2 MVC模型图



1.3 MVC带来了什么

a.降低耦合性，对于早期的应用程序，可能一个页面功能夹杂着获取数据源，修改数据源，显示数据源样式的代码，只要能实现功能，随心所欲地去编写代码。而MVC构架，强制分离了业务层和视图层，不用像早期代码一样牵一发而动全身。

b.增加了重用性，小到一个功能，大到一个项目为界限，每次编写新的应用程序需要重新对数据源进行操作，而MCV构架则是打破了这个界限，无特殊情况下(特殊情况考虑设计即可)，可以实现数据源的可重复利用，优化了成本和效率。

c.开发工程化，以一定的标准约束开发，无论是在开发效率、还是在工程构架、职责定位等一系列相关标准上，都是极具优势的。

2 [MVVM](https://so.csdn.net/so/search?q=MVVM&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_26713059/article/details/_blank)

2.1 MVVM含义

MVVM一定程度上算是MVC的改进版，分为Model、View、ViewModel。

VM(ViewModel):视图模型，其实我也不能很好地去翻译这个模块，就我个人的理解而言，ViewModel是处理数据源和业务逻辑的地方，不会去对View进行操作，更不会持有View的引用。

MVVM有着比MVC更明确的分工，通过数据绑定、依赖属性、命令、路由事件等一系列新特性对模块进行更细的划分，让功能归属更加明确，软件构架更加灵活。

2.2 MVVM模型图



2.3 MVVM带来了什么

a.各个模块的分工相比MVC会更加明确，这意味着分工界限，单元测试、可复用性等模块会更加优秀。

b.通过数据驱动，数据的变化会自动去更新View，View引起的事件触发也能自动反馈到数据层，因此我们可以把重心关注到数据处理模块。

c.一定程度上解决了MVC中由于业务逻辑的增加，Controller中控制Model和View的逻辑会越来越多,形成逻辑爆炸和维护爆炸的糟糕局面的问题。

Android下的MVC/MVVM

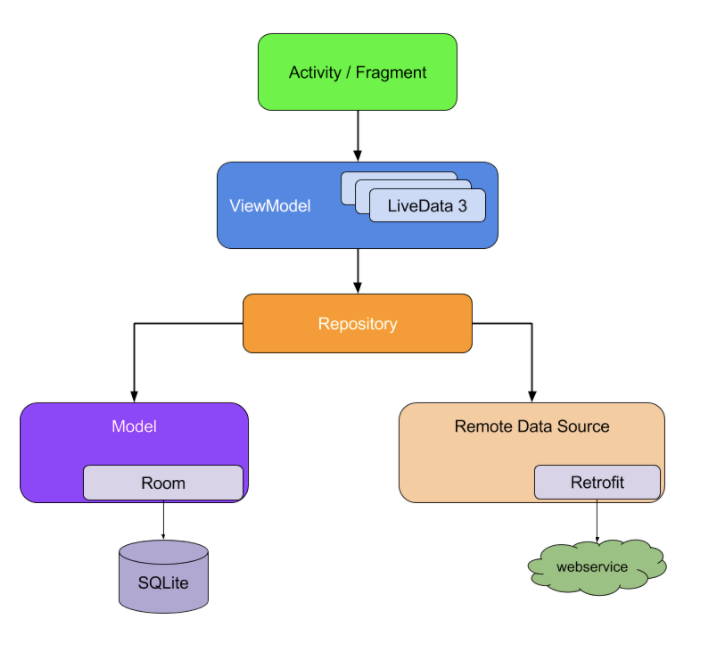
Android下的MVC

不得不说android[架构](https://so.csdn.net/so/search?q=%E6%9E%B6%E6%9E%84&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_26713059/article/details/_blank)的设计十分符合MVC的思想。xml文件就是View层，一般来说我们在Activity中进行View的处理和数据的处理，举个例子：自定义一个数据源存放在model类中，我们在Activity中获取这个数据源，并对数据源进行目的性处理，最后在UI控件上进行显示。

在小型应用开发中，这种设计是十分方便和合理的，但是随着逻辑的增加，Acitivity的体量会越来越多，整个Acitivity代码爆炸，业务逻辑和View逻辑错综复杂。我想正是情景需要和技术更新，MVVM也应运而生。

Android下的MVVM

实际上Android并没有统一的MVVM框架这种东西，大家实现MVVM的方式都各不相同，但是由于谷歌推出的JetPack框架，里面的部分组件十分适用于MVVM构架，所以MVVM的实现可以逐渐向一个大方向靠近。



上面是谷歌官方给出的架构，数据源这里本质上并没有发生变化，这里我们会简单介绍一下DataBinding、ViewModel和LiveData组件：

这里并不会讲组件的具体用法，我也没能力去讲清组件的底层实现，详情可以移步谷歌官方文档，下面我会以自己的理解去讲解这些组件为什么适用于MVVM构架。

DataBinding组件

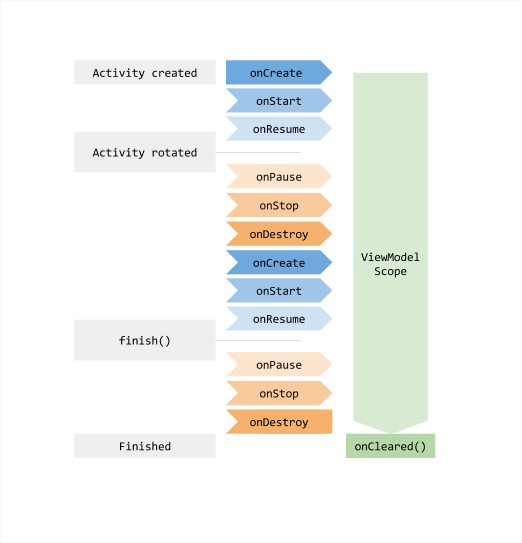
DataBinding组件从名字上很好理解，它充当了数据绑定的角色。数据能够单向或者双向绑定到UI控件上，之前我们提到MVVM很重要的一点是数据驱动，而DataBinding则是实现了数据驱动，降低了Android中布局和逻辑的耦合性，Activity也不需要用一堆样板代码来控制UI控件的实现。

这里值得提一句，DataBinding实现上可以在绑定数据的同时进行简单的逻辑处理，但是我认为DataBinding只需要做绑定数据操作，一是方便排错，更重要的是这样职责更加明确。

ViewModel组件

ViewMode组件并不是VM(ViewModel)，VM(ViewModel)是一种思想，一种构架，只要实现了VM(ViewModel)功能的类都可以称为VM(Viewmodel)。

在Android中，ViewModel组件充当了VM(ViewModel)的角色，为什么呢，首先我们可以看看ViewModel的生命周期：



可以看到，ViewModel贯穿了Activity的生命周期，这就解决了一个很常见的问题，由于配置改变而导致Acitivity重建的问题，由于ViewModel的生命周期跟Acitivity一样甚至更长，我们不用担心数据的丢失也不需要用额外的逻辑和内存去恢复数据。

其次使用ViewModel另一个好处就是解决了Activity和多个Fragment共享数据的问题，在以往的姿势上，我们通常会通过创建回调或者保存临时数据等一系列方式处理共享数据，当然后来出现的EvenBus使共享数据变得方便，不过在ViewModel中我们可以很自然地实现数据共享。

最后值得注意的是既然我们选择ViewModel组件作为VM(ViewModel)，那我们不管是从构架定义上，还是从长生命周期的对象持有短生命周期对象的引用容易导致内存泄漏的角度考虑，ViewModel里面都不应该出现View的引用。

ps：这里分享一个不知道在哪篇文章看到的小知识，判断ViewModel是否持有View控件的引用可以看ViewModel类中是否引入了android.xxx样式的资源，当然并不绝对。

LiveData组件

MVVM架构对层次进行了划分，那么各个模块始终需要一个东西来进行通信，官方推出的LiveData组件承担了这个职责。

“LiveData is a data holder class that can be observed within a given lifecycle. ”这是官方给出的定义，LiveData是可以在给定生命周期内观察到的数据持有者类。基于生命周期的可观察的数据持有类意味着使用LiveData组件能够同步生命周期，避免了内存泄漏，并且在Activity活跃状态总是能收到新数据、不活跃状态下LiveData会停止更新数据；同时UI和数据的变化能够进行实时反馈，不管是UI的变化还是数据的变化都能够引起LiveData的更新。

综上，LiveData组件无疑是负责MVVM模块间通信的一个好选择。

结语

关于Andorid下的MVVM就算是告一段落了，最后我想提一句，不管是MVC还是MVVM，都是为了能够更方便地解决某一问题或适用某一场景。构架是一个好构架，但是是否需要使用这个构架，以及构架中的职能该怎么分配，都需要自己去考量。