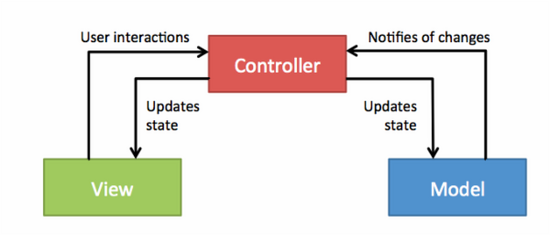
**1.MVVM模式介绍**

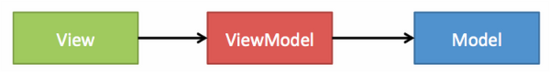
正如其名称一下，MVVM是一个UI设计模式。它是MV\*模式集合中的一员。MV\*模式还包含MVC(Model View Controller)、MVP(Model View Presenter)等。这些模式的目的在于将UI逻辑与业务逻辑分离，以让程序更容易开发和测试。为了更好的理解MVVM模式，我们可以看看其来源。

MVC是最初的UI设计模式，最早出现在Smalltalk语言中。下图展示了MVC模式的主要组成：



这个模式将UI分成Model(表示程序状态)、View(由UI控件组成)、Controller(处理用户交互与更新model)。MVC模式的最大问题是其令人相当困惑。它的概念看起来很好，但当我们实现MVC时，就会产生上图这种Model-View-Controller之间的环状关系。这种相互关系将会导致可怕的混乱。

最近Martin Fowler介绍了MVC模式的一个变种，这种模式命名为MVVM，并被微软广泛采用并推广。



这个模式的核心是ViewModel，它是一种特殊的model类型，用于表示程序的UI状态。它包含描述每个UI控件的状态的属性。例如，文本输入域的当前文本，或者一个特定按钮是否可用。它同样暴露了视图可以执行哪些行为，如按钮点击或手势。

我们可以将ViewModel看作是视图的模型(model-of-the-view)。MVVM模式中的三部分比MVC更加简洁，下面是一些严格的限制

1. View引用了ViewModel，但反过来不行。
2. ViewModel引用了Model，但反过来不行。

如果我们破坏了这些规则，便无法正确地使用MVVM。

这个模式有以下一些立竿见影的优势：

1. 轻量的视图：所有的显示逻辑都在ViewModel中。
2. 便于测试：我们可以在没有视图的情况下运行整个程序，这样大大地增加了它的可测试性。

现在你可能注意到一个问题。如果View引用了ViewModel，但ViewModel没有引用View，那ViewModel如何更新视图呢？哈哈，这就得靠MVVM模式的私密武器了。

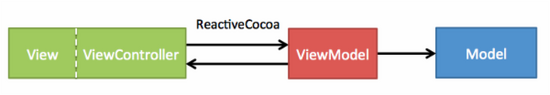
**MVVM和数据绑定**

MVVM模式依赖于数据绑定，它是一个框架级别的特性，用于自动连接对象属性和UI控件。例如，在微软的WPF框架中，下面的标签将一个TextField的Text属性绑定到ViewModel的Username属性中。

<TextField Text=”{DataBinding Path=Username, Mode=TwoWay}”/>

WPF框架将这两个属性绑定到一起。

不过可惜的是，iOS没有数据绑定框架，幸运的是我们可以通过ReactiveCocoa来实现这一功能。我们从iOS开发的角度来看看MVVM模式，ViewController及其相关的UI(nib, stroyboard或纯代码的View)组成了View:



……而ReactiveCocoa（ kvo,delegate,notification,action-target——>信号，代码集中，可先绑定响应关系，事件产生和事件处理依赖小，解耦）绑定了View和ViewModel。

**2.MVVM各部分职责**

Model 职责：

Model 将作为简单的对象，用来维持数据结构。同时，会使用Extension来扩展 Model，为其添加数据存取的功能。

View/ViewController职责：

布局、动画、设备转屏、View或Window的切换、显示UI

ViewModel 职责：

* 校验用户输入。
* 网络请求。
* 表示逻辑。
* 其他不能放入 Model，与 View 无关的逻辑。

“表示逻辑（presentation logic）”，以 MVVM 属术语来说，就是那些将 Model 数据转换为 View 可以呈现的东西的事情，例如将一个 NSDate 转换为一个格式化过的 NSString。ViewModel一词的确不能充分表达我们的意图. 一个更好的术语可能是 “View Coordinator”(感谢 [Dave Lee](https://twitter.com/kastiglione" \t "_blank) 提的这个 “View Coordinator” 术语, 真是个好点子). 你可以认为它就像是电视新闻主播背后的研究人员和作家团队. 它从必要的资源(数据库, 网络服务调用, 等)中获取原始数据, 运用逻辑, 并处理成 view (controller) 的展示数据. 它(通常通过属性)暴露给视图控制器需要知道的仅关于显示视图工作的信息(理想地你不会暴漏你的 data-model 对象). 它还负责对上游数据的修改(比如更新模型/数据库, API POST 调用).

顾名思义，viewModel即视图模型，对视图展示数据进行处理，一般流程是，接受vc的事件命令请求及处理相关数据，完事之后将标准展示数据处理好交给vc展示到view上，此谓视图模型。将视图模型分离出来，与视图类做法类似，留出操作接口，协议及代理，这样一来，对于数据层又可以重用，只要vc符合相关的协议，那么在不同的vc中就可以用同一个viewModel了。封装性和重用性得以体现，而且便于测试。

借用歪果人的回答就是：

Views display a certain shape of data. They have no idea where the data comes from.

ViewModels hold a certain shape of data and commands, they do not know where the data, or code, comes from or how it is displayed.

Models hold the actual data (various context, store or other methods)

Controllers listen for, and publish, events. Controllers provide the logic that controls what data is seen and where. Controllers provide the command code to the ViewModel so that the ViewModel is actually reusable.

这样一来，我们可以把数据有效性的验证，视图的展示逻辑，网络数据请求及处理，其他的数据处理逻辑集合到viewModel中。

**3.MVVM的优缺点**

优点：MVVM就是在MVC的基础上加入了一个视图模型viewModel，用于数据有效性的验证，视图的展示逻辑，网络数据请求及处理，其他的数据处理逻辑集合，并定下相关接口和协议。相比起MVC，MVVM中vc的职责和复杂度更小，对数据处理逻辑的测试更加方便，对bug的原因排查更加方便，代码可阅读性，重用性和可维护性更高。MVVM耦合性更低。MVVM不同层级的职责更加明确，更有利于代码的编写和团队的协作。

**低耦合：**View可以独立于 Model变化和修改，一个 ViewModel可以绑定到不同的 View上

**可重用性：**可以把一些视图逻辑放在一个 ViewModel里面，让很多 view重用这段视图逻辑

**独立开发：**开发人员可以专注于业务逻辑和数据的开发 ViewModel，设计人员可以专注于页面设计

**可测试：**通常界面是比较难于测试的，而 MVVM 模式可以针对 ViewModel来进行测试

缺点：MVVM相比MVC代码量有所增加。MVVM相比MVC在代码编写之前需要有更清晰的模式思路。

MVVM有优点也有缺点，但缺点在他们所带来的好处面前时不值一提的。他们的低耦合性，封装性，可测试性，可维护性和多人协作便利大大提高了开法效率。