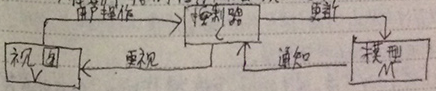
MVC、MVVM浅谈

1. MVC
2. 简介：模型－视图－控制器(Model-View-Controller,MVC)是xerox PARC在20世纪80年代为编程语言smalltalk－80发明的一种软件设计模式，至今已广泛用于用户交互应用程序中。在ios开发中MVC的机制被使用的淋漓尽致，充分理解ios的MVC模式，有助于我们程序的组织合理性。



1. 模型对象

模型对象封装应用程序的数据，并定义操控和处理该数据的逻辑和运算。例如：模型对象可能表示游戏中的角色或地薄中的联系人。用户在视图层中所创建或修改数据的操作，由控制器对象传达出去，最终会创建或更新模型对象。模型对象更改时(如网络请求了新数据)，它通知控制器对象，控制器对象。

1. 视图对象

视图对象是应用程序中用户可以看见的对象。视图对象知道如何将自己绘制出来，并可能对用户操作作出事件响应(UIControl及其子类)。视图对象的主要目的，就是显示来自应用程序模型对象的数据，并使该数据可被编辑。尽管如此，在MVC应用程序中，视图对象通常与模型对象分离。在ios应用程序开发中，所有的控件、窗口等都继承自UIView，对应MVC中的V。UIView及其子类主要负责UI的实现，而UIView所产生的事件可才用委托的方式交给视图控制器去实现。

1. 控制器对象

在应用程序的一个或多个视图对象和一个或多个模型对象之间，控制器对象充当媒介。控制器对象因此是同步管道程序，通过它，视图对象了解模型对象的更改反之亦然。控制器对象还可以为应用程序执行设置和协调任务，并管理其他对象的生命周期。控制器对象解释视图对象中进行的用户操作，并将新的或更改过的数据传给模型对象。当模型对象修改时，一个控制器对象会将新的模型数据传达给视图对象，以便视图对象可显示它对应不同的UIView，有相应UIViewController，对应MVC中的C，例如ios常用的UITableView，它所对应的controller就是UITableViewController

1. 注意事项
2. Model和View永远不能互相通信，只能通过Controller传递
3. Controller可以直接与Model对话(读写Model属性调用Model接口)，Model通过通知和kvo机制与Controller间接通信
4. Controller可以直接与View对话，通过outlet，直接操作view，outlet直接对应道view中的控件，view通过action向controller报告事件的发生，controller是view的直接数据源(数据可能是controller从model中取得并已经加工过了)。Controller是view的代理(delegate)，以同步view与controller
5. MVVM
6. 简介：在ios应用中日益增长的重量级视图控制器问题：在典型的MVC应用中，许多逻辑被放在controller里。有些确实属于controller，但更多的是表示逻辑，为了不让控制器日益复杂，便于测试管理，出现了MVVM。它其实是一个MVC的增效版，并将表示逻辑从controller移除放入新对象里，即viewModel里
7. Model层：存放数据，并定义数据操控和处理该数据的逻辑和运算
8. ViewModel层：view和model层的粘合剂，它是一个放置用户输入验证逻辑、视图显示逻辑、发起网络请求和其他各式各样代码的好地方。简而言之，即把 原来viewController层的业务逻辑和页面逻辑等剥离出来放到viewModel层
9. View层：即视图和视图控制器层，任务是从viewModel层取数据，然后显示。

