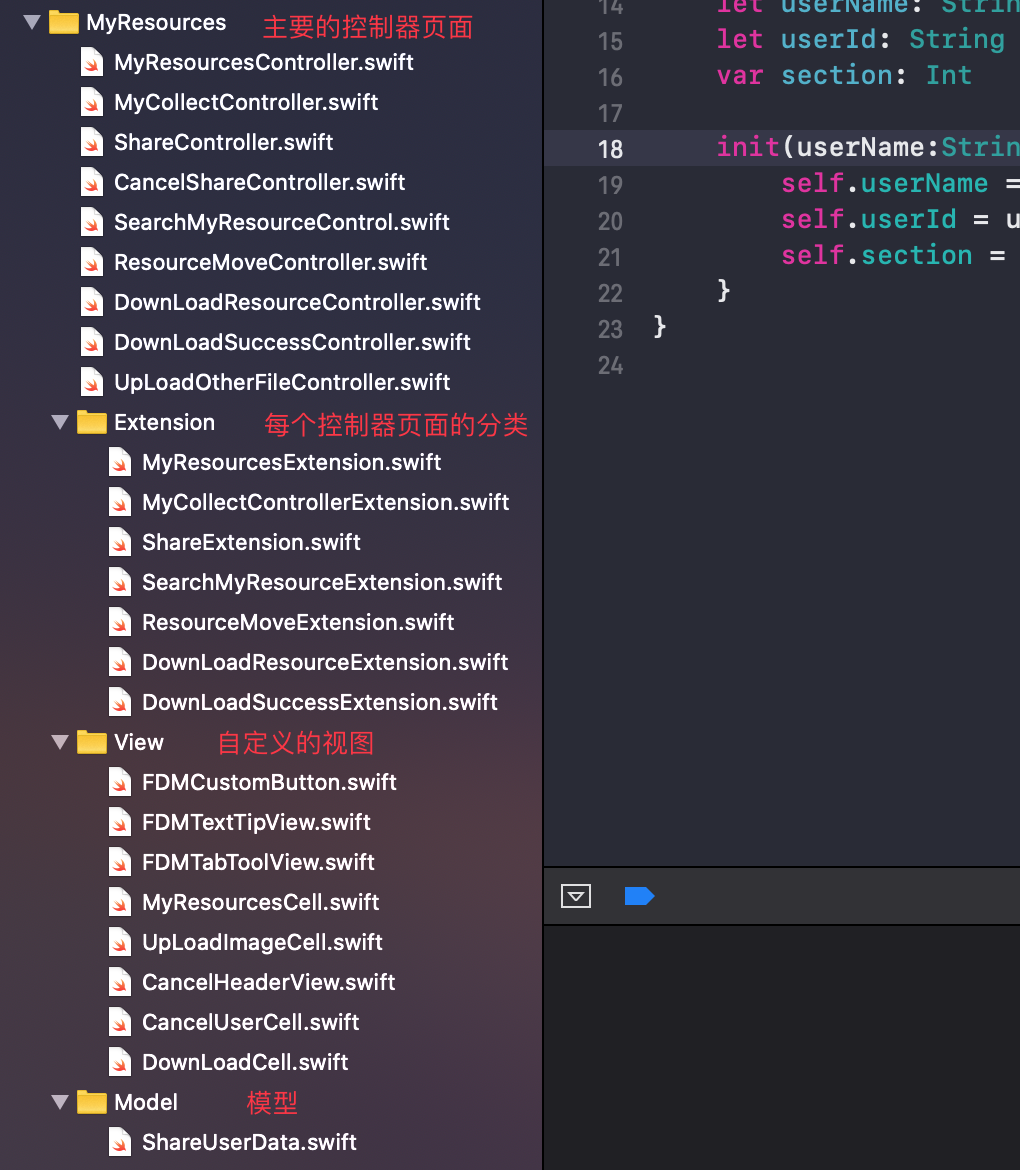
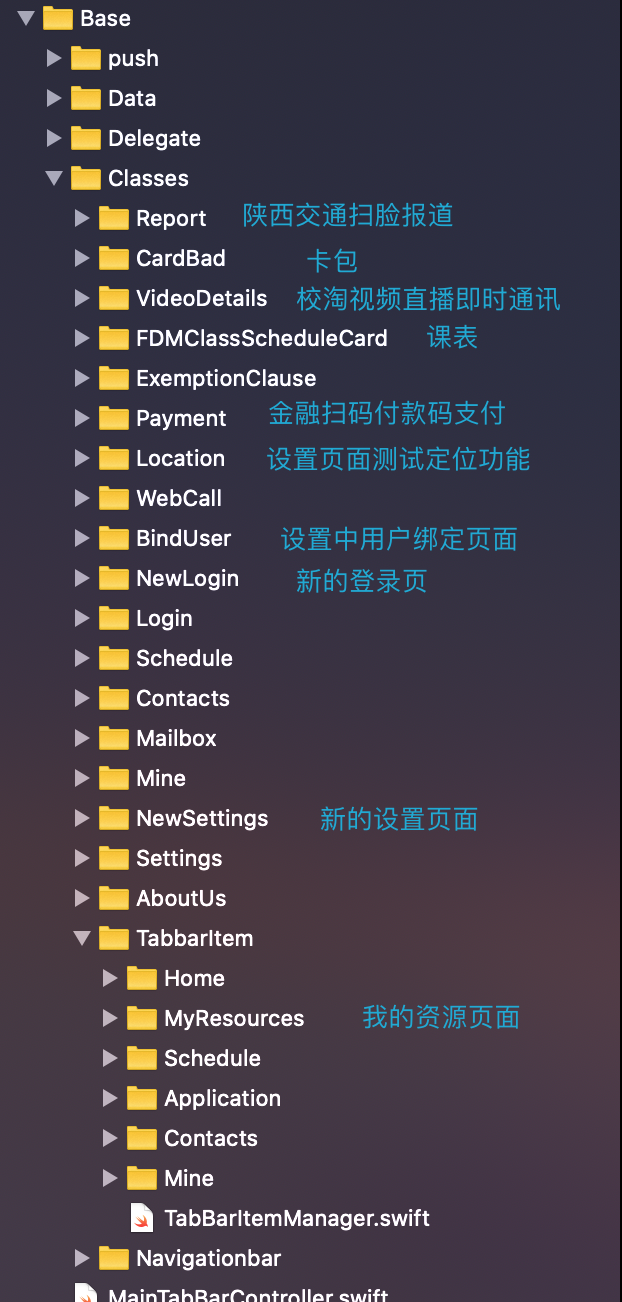
**iOS项目文档**

在**iOS Apple**这个项目上后期依然使用的**MVC**设计模式，为了分离代码功能模块减少**Controller**代码量，项目中使用了大量的 **extension** 进行扩展。

并且为了使项目更清晰，大部分代码都附有注释

**基本的模块分类和以前一样**



**大部分的架构都一样，以我的资源为例：**

这里我将整个我的资源功能模块分为了： **控制器模块，控制器扩展模块，自定义视图模块，模型模块**

**控制器模块**

1. 在控制器模块中，首先在顶部定义了需要使用的 view ，并在后续的使用中增加各种需要的视图与数据模型

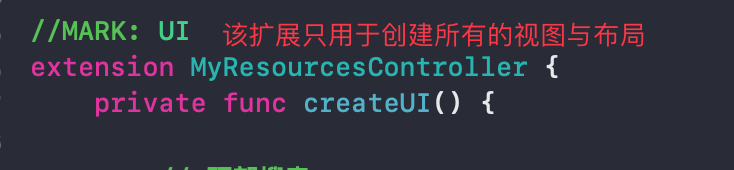


2. 几乎所有的控制器中都使用这样的分类【不同控制器可能会有createNetwork或其他获取网络数据的名称，但是功能都是相同的】



**createUI( )**

**用来创建基本的视图页面，并使用extension 作为一个单独的扩展**



在我的资源功能中，createUI 使用的创建视图方式为



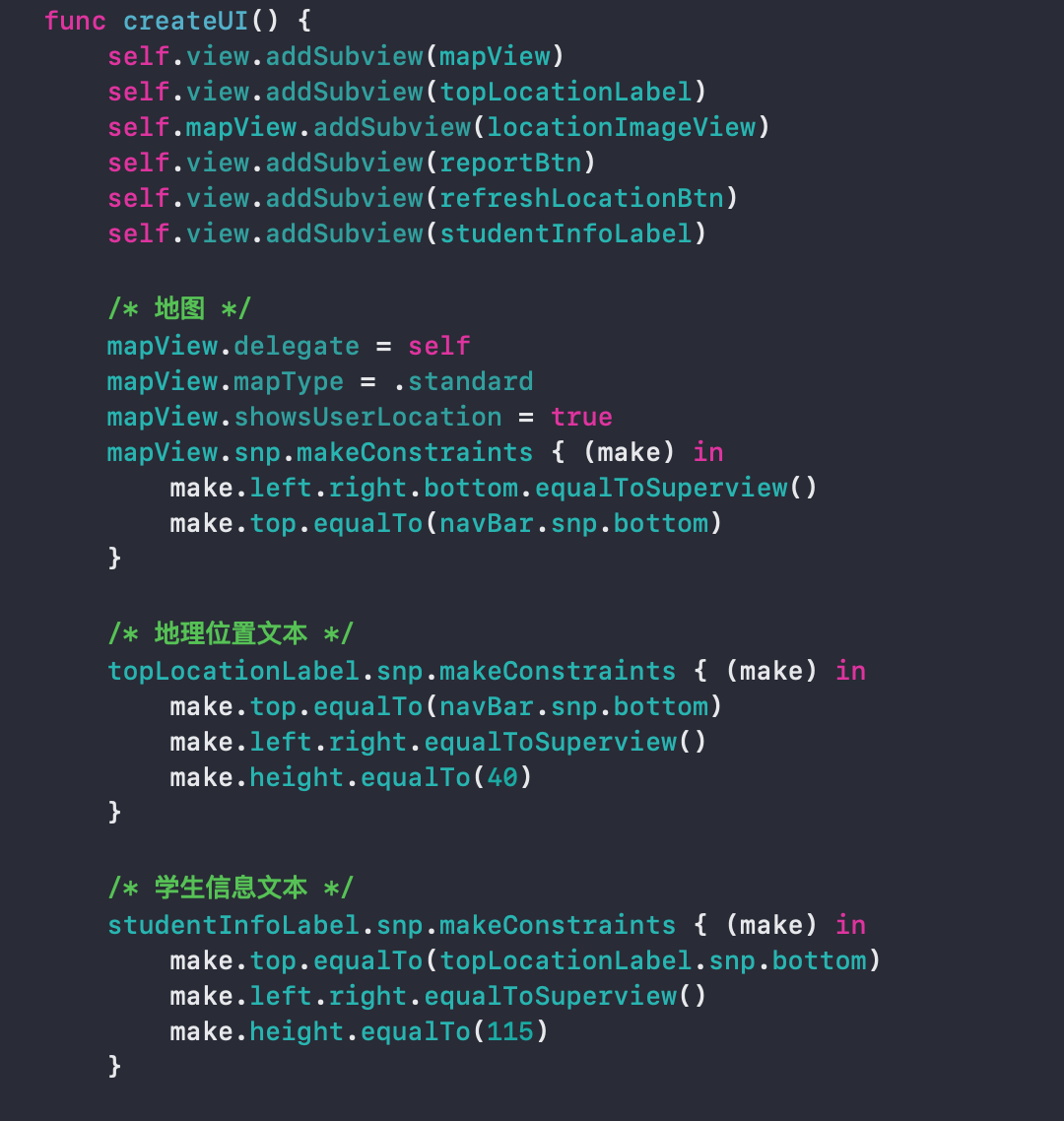
最开始使用该顺序进行搭建时，是因为通过 self.view的红色字体可以更加快捷的分辨出视图块，并且可以将属性的设置与布局的设置，块与块之间的设置通过 红色的字体进行分开。

事实证明该方法在修改布局时可以很快速的分辨出视图的层级，比如谁是父视图谁是子视图。再根据每个视图的注释，可以很快速的找的需要修改的视图进行修改。

但是该方法的弊端也很严重。

在使用snapkit进行布局，是要求先将视图进行添加在进行布局。否则就会出现崩溃的情况，使用上图方法时，如果遇到了需要依靠下一个视图布局当前视图的情况就会出现问题。

所以在后期修改了创建的顺序。如图:



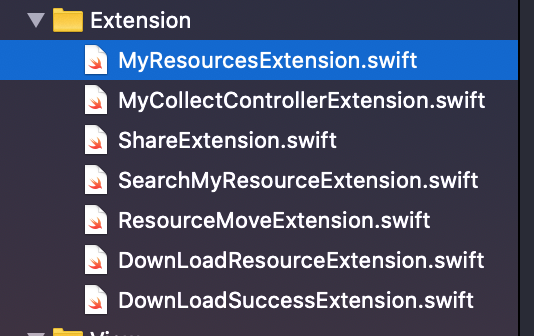
**createAction( )**

**用来创建一些手势操作监听等**



createNetwork（） 获取基本网络数据

接下来是扩展模块



该模块主要为一些数据或视图的 代理方法与网络请求方法

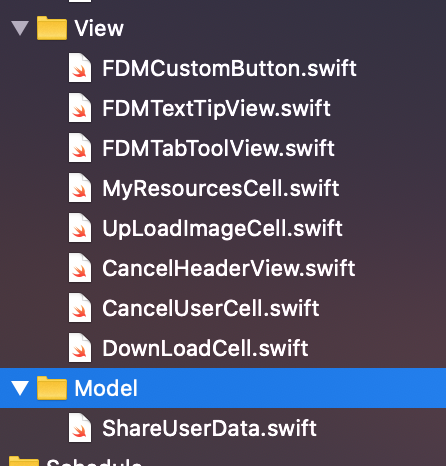
通过extension 对网络请求与代理进行分割。不过由于不同的功能，在功能少的情况下有一些代理的扩展会被放到控制器中，



后期为了更详细的模块分开也会将 action 与 network 单独分出来两个文件， 例如:



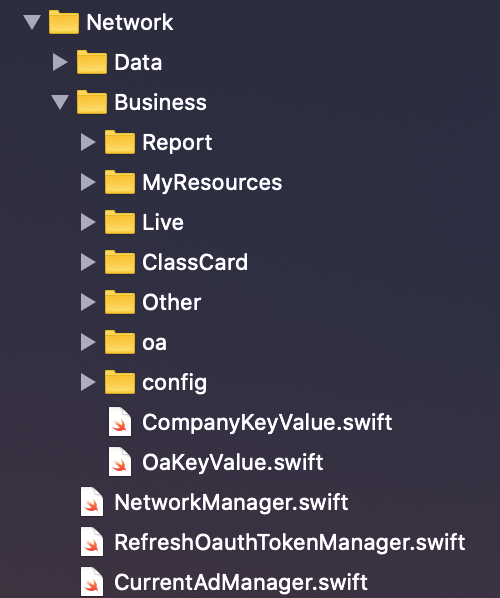
View文件夹与 model 文件夹



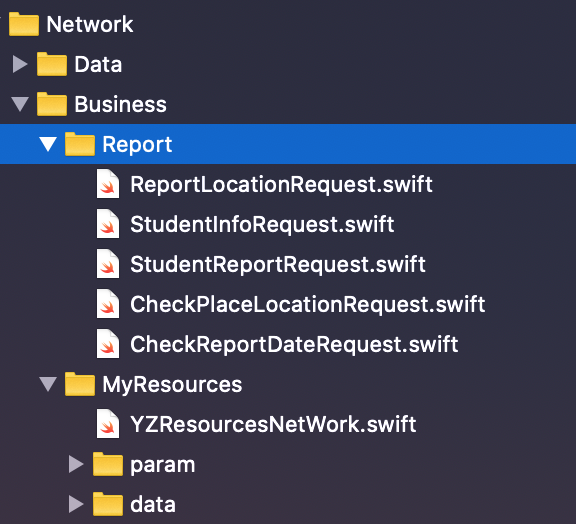
View 文件夹是用来存放一些自定义封装的控件

Model 文件夹并不是所有功能模块都有，是用来存放一些不需要网络请求的数据模型

其他需要网络的数据模型放在Network中



**网络请求模型以陕西交通的扫脸报道与我的资源为例**

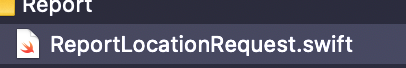


在我的资源中，网络请求模型依然使用之前的模式，param 与 data分开



但这样的分类不利于查找模型，所以后期将param 与 data 放在了同一个文件

例如：





文件取名为Request 在内部创建两个类 param 与 model（或data）方便在查找param时同样能找到model

项目中的Tools 文件



阿里播放器主要用于直播及时通讯方面

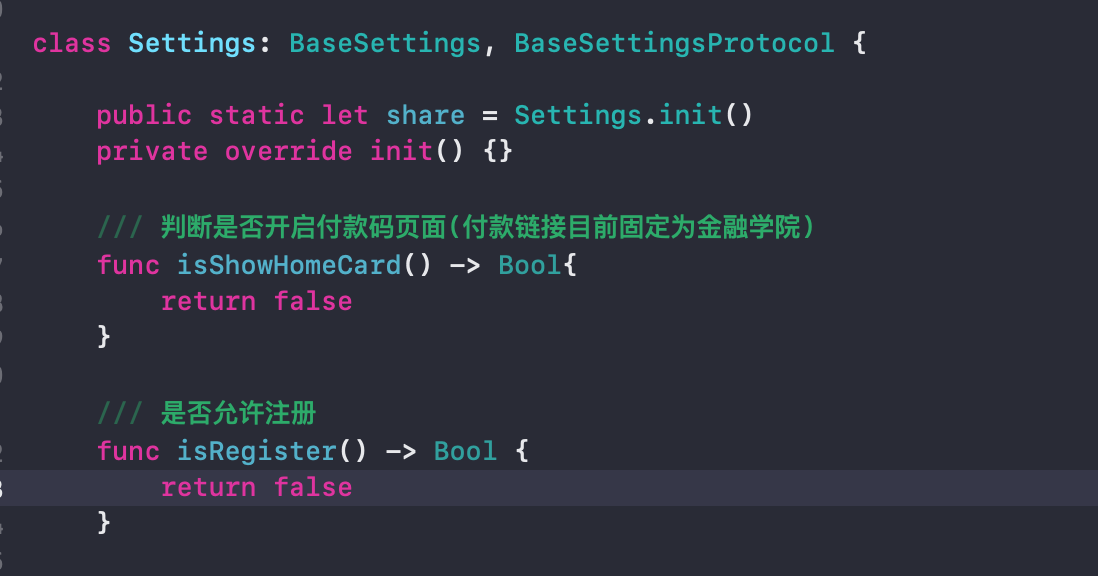
FDMTools

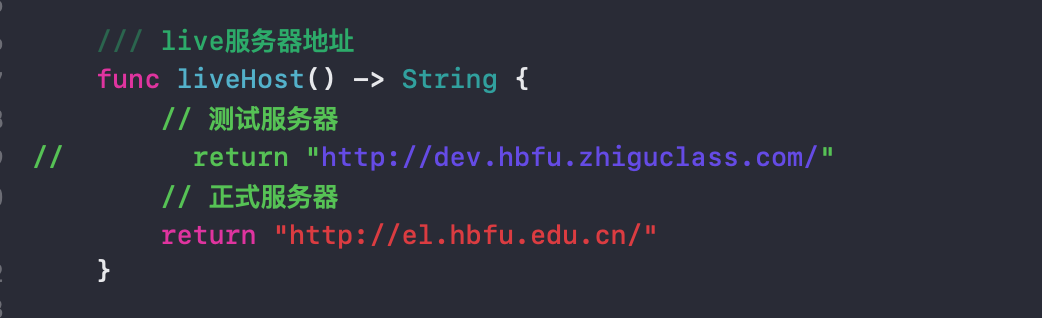


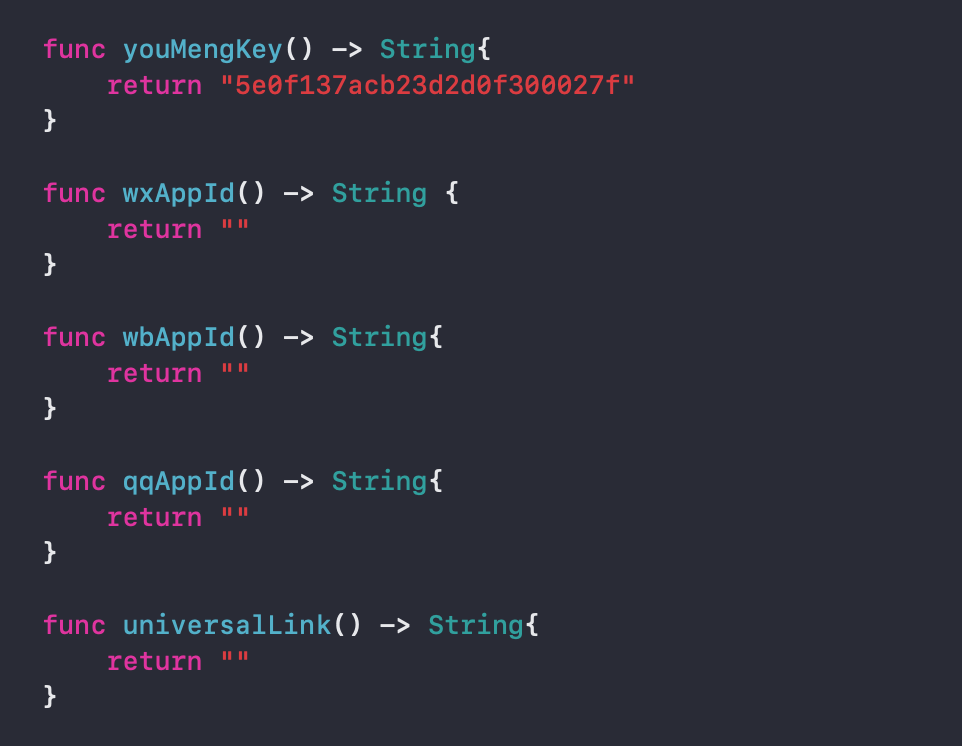
项目中常用的是 FDMQuick 封装了一些常用的功能例如对视图，字符串等操作。以及打印信息

该工具并不完善，大部分使用的是 FDMQuic.shared 来调用功能，在教务系统与国元公考项目中，这些工具被更换为 view，string，UIimage 等的扩展。

Settings文件新增内容







在新版本中，苹果要求支持微信登录的App同样也需要支持苹果登录，

在该项目中使用了隐藏的方法实现。

在App审核期间，需要到应用管理中进行设置。



将 oa\_to\_examine属性设置为1，当审核结束后改为0。

如果没有该属性需自行添加。

