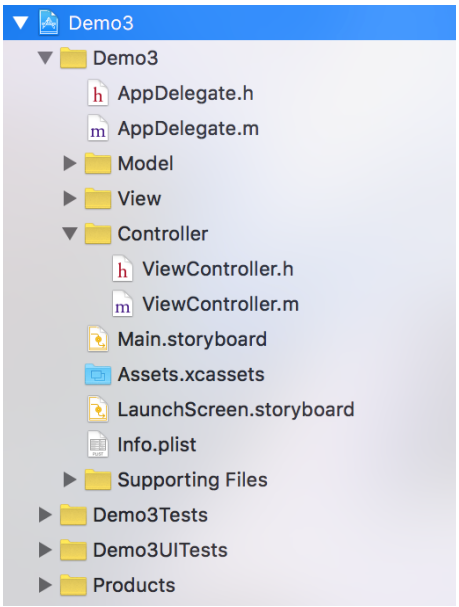
Hello, iOS
3 MVC框架[1]

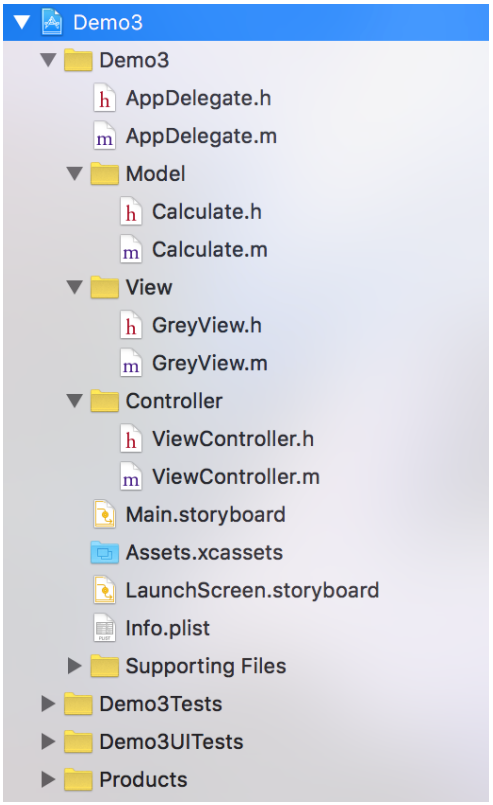
上一次我们简单的介绍了面向对象编程的基础知识。虽然没有给大家写出代码示例(考虑到大部分同学对语法是不熟悉的)，但是这些概念我已经通过和iOS开发相联系的方法做了进一步的解释。

这一节我们将要了解的是著名的MVC框架[2]。关于这个框架的详细概念，你可以参考下方的脚注。但是不少同学不明白，模型、视图和控制器分别是什么，所以我们先简单的为大家介绍一下在iOS开发当中的这三个概念。

我们新建一个工程Demo3，然后在文件区域划新建三个文件夹——“Model”, “View”, “Controller”。这时，我们可以把已经存在的ViewController的两个文件放进Controller文件夹中，此时的文件结构如右图。



利用上节学到的知识，我们可以知道ViewController代表的不仅仅是一个界面，同时还是一个类，对于一个类，Objective-C采用.m和.h两个文件表示。其中，.h是声明部分(declaration)，.m是实现部分(implementation)。这里的ViewController就是我们说的控制器，用户的操作从控制器接收并进行处理。



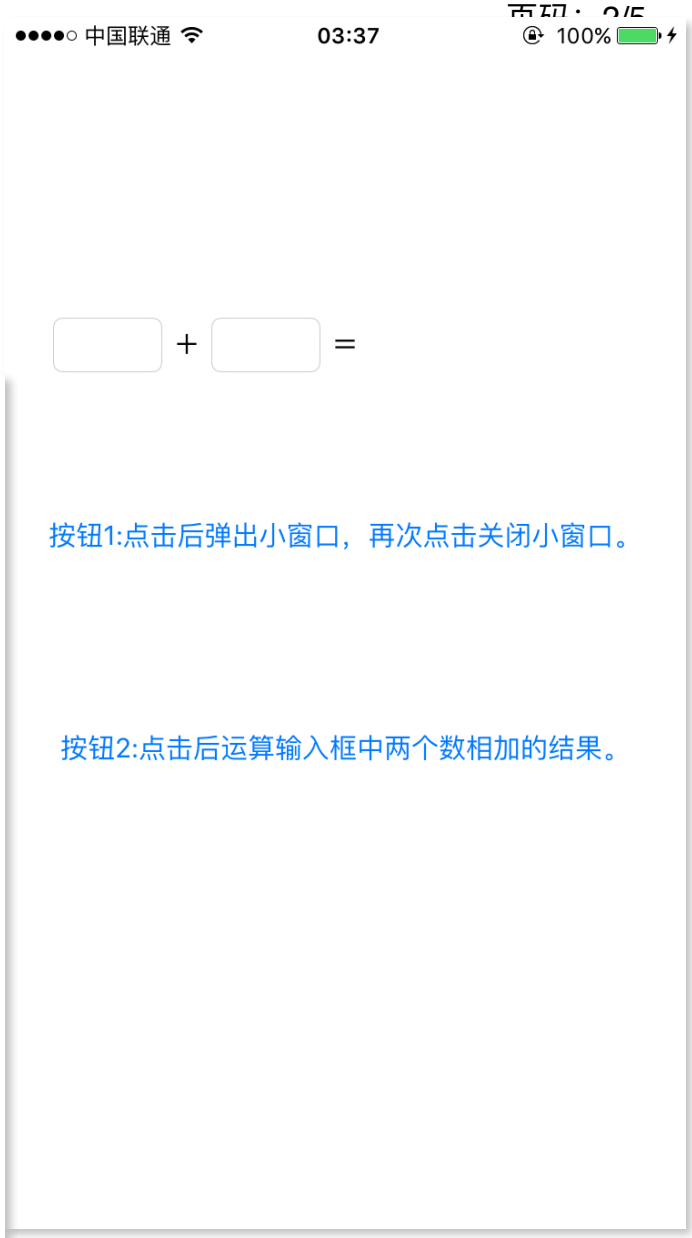
为了完整的表现MVC三个模块，我们可以写一个界面，点击按钮1弹出一个小窗口，点击按钮2可以对输入框的两个数做加法运算。我不具体描述实现的细节，但是关于概念的关键部分我会重点提及。

---

[1] 2015年9月1日，基于OS X 10.11 beta7， iOS 9 beta 7， Xcode 7 beta 5.

[2] 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，将业务逻辑聚集到一个部件里面，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。MVC被独特的发展起来用于映射传统的输入、处理和输出功能在一个逻辑的图形化用户界面的结构中。

成品的文件结构如上图。我在模型文件夹里创建了Calculate类，也就是计算类，View中我们创建了一个GreyView类。下面我们结合着实际的操作来展现这三个模块如何相辅相成地发挥作用。



在点击了按钮1之后，屏幕下方会出现一个灰色的方块[3]。

灰色的方块其实是可以继续定制功能的，所有的功能都可以在GreyView相关的文件里去写，我在这里只是抛砖引玉。

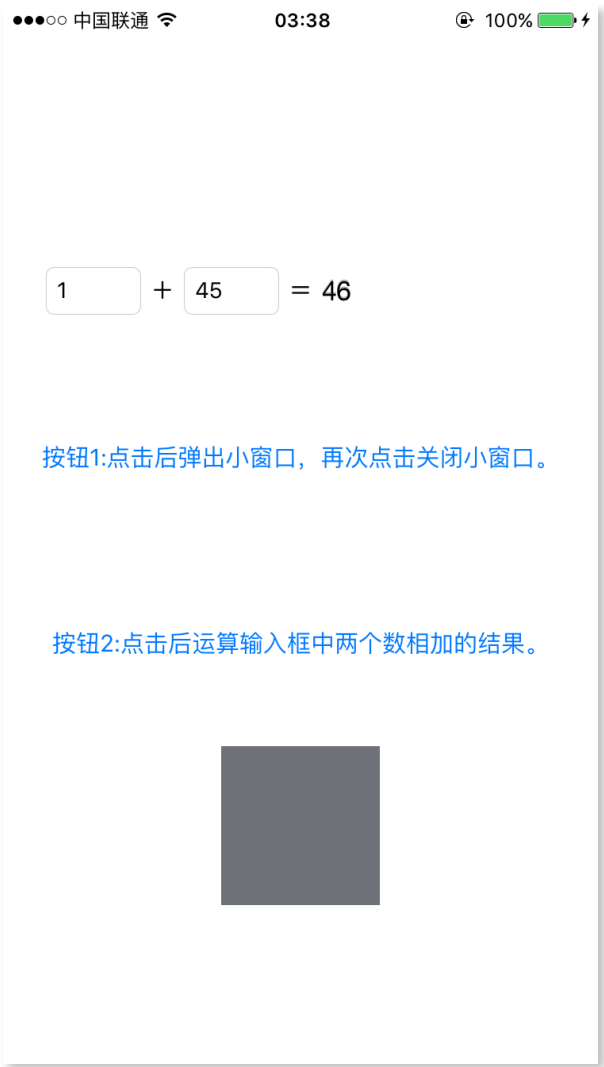
下面是关于计算加法的部分，我在Calculate这个文件里其实只写了一个简单的加法方法，如下图。

---

[3] 其实，这个灰色的方块是UIView类的实例，也就是说，灰色的方块就是我在View组里放的GreyView。

```objc
#import "Calculate.h"

@implementation Calculate

+ (NSInteger)add:(NSInteger)a With:(NSInteger)b {
    return a + b;
}

@end

```

当我们点按第二个按钮的时候，读取两个输入框中的数然后相加即可[4]。



左边的图就是输入了两个数之后点击按钮2的结果。

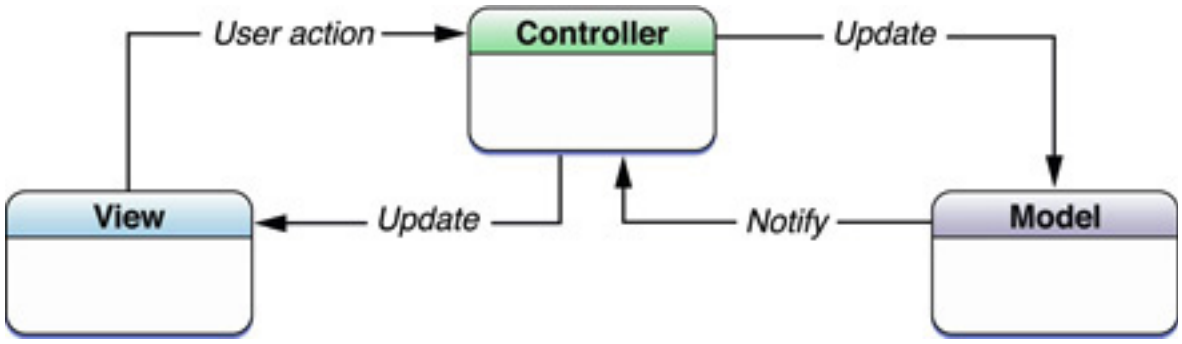
[4] 有两点说明：NSInteger和NSString比较类似，都是基本库中的类型，和C语言中的int比较相似，但是NSInteger本身也是一个类；输入框中的文字是NSString，所以不能直接带到方法里使用，具体的使用方法可以参考我之后给出的代码。

模型的存在实际上是为了减少控制器模块的代码量，把大量重复的业务逻辑放到统一的地方管理。因此，模型中可以处理业务逻辑、基础数据类型、常用的代码工具等等。

下面可以参考我在控制器当中的代码，由于这个项目比较小，MVC的分离显得不那么明显，但是结构是可以看清楚的[5]。

```
 9  #import "ViewController.h"
10  #import "Calculate.h"
11  #import "GreyView.h"
12
13  @interface ViewController ()
14
15  @property (weak, nonatomic) IBOutlet GreyView *greyView;
16  @property (weak, nonatomic) IBOutlet UITextField *textField1;
17  @property (weak, nonatomic) IBOutlet UITextField *textField2;
18  @property (weak, nonatomic) IBOutlet UILabel *showLabel;
19  - (IBAction)button1DidTouched:(id)sender;
20  - (IBAction)button2DidTouched:(id)sender;
21
22  @end
23
24  @implementation ViewController
25
26  - (void)viewDidLoad {
27      [super viewDidLoad];
28      // Do any additional setup after loading the view, typically from a nib.
29  }
30
31  - (void)didReceiveMemoryWarning {
32      [super didReceiveMemoryWarning];
33      // Dispose of any resources that can be recreated.
34  }
35
36  - (IBAction)button1DidTouched:(id)sender {
37      if (self.greyView.isHidden == true) {
38          self.greyView.hidden = false;
39      } else {
40          self.greyView.hidden = true;
41      }
42  }
43
44  - (IBAction)button2DidTouched:(id)sender {
45      self.showLabel.text = [NSString stringWithFormat:@"%ld", (long)[Calculate add:[self.textField1.text
            integerValue] With:[self.textField2.text integerValue]]];
46  }
47  @end
48
```

看过程序样例后，我们拉回到概念上。下面这张图片完美地展现了这三个模块的交互过程，可以看到，控制器充当了数据层和视图层之间的沟通桥梁，这便是控制器的最大作用。同时，iOS的开发中对于MVC的概念体现的应该是最深刻的，所以对这种框架一定要有比较深入的了解。



[5] 可以注意到第15行，灰色方块的类型是GreyView，第45行调用了Calculate类的add方法。

本节我们着重介绍了MVC框架并且展示了一个样例程序，下一次我们会介绍iOS开发当中的代码基础，荷枪实弹的较量就要开始了！

第3节 EOF[6]

[6] 关于本节的内容，你还可以参考苹果官方文档。网址: https://developer.apple.com/library/mac/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html