BMFramework 文档

# BMFramework 起源

BMFramework 原始版本产生于手机 海信广场项目版本,最初解决层与层之间 的简单解耦问题,规范程序员的编程行为，随作者对其不断的改进,至今已发展到 0.1.0 版本。

BMFramework基于 MVC 架构,适用于大规模 App(信息展示、 社交、工具等类型)的快速开发框架,已被海信广场项目采用。

# BMFramework 特点

## 1. 使用简单

## 2. 高度解耦

## 3. 模块化

## 4. 自动生成

## 5. 内建工具

# BMFramework谁应该使用

## 1. 我想快速完成开发

## 2. 我讨厌编写冗长的代码

## 3. 我总是喜欢找点乐子

## 4. 我的项目需求大的吓人,不知从何入手

## 基本知识要求

了解 Objective-C & Cocoa-Touch 了解 MVC 了解 JSON

# BMFramework 架构

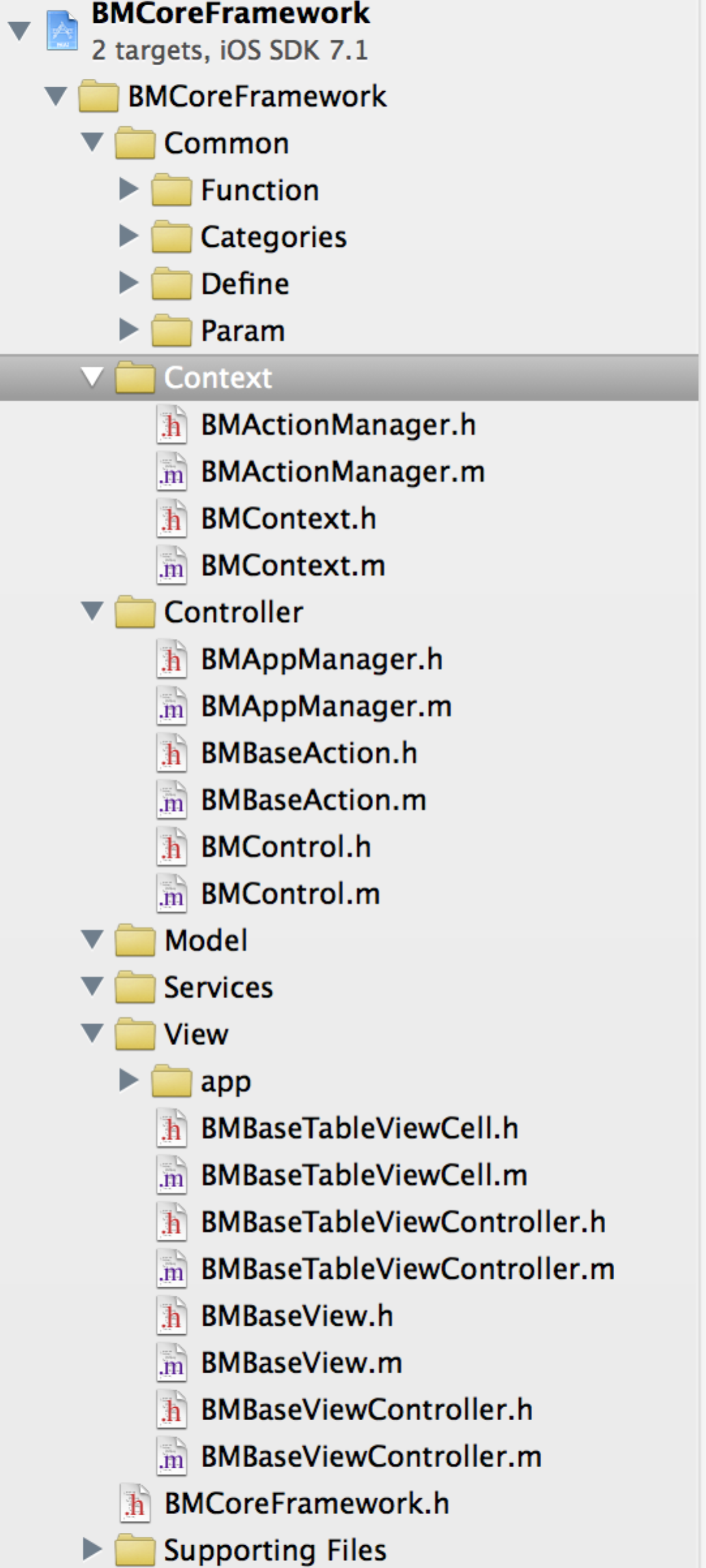
## 整体架构

目前BMFramework 框架分为两个部分。

1. BMCoreFramework 主要框架部分

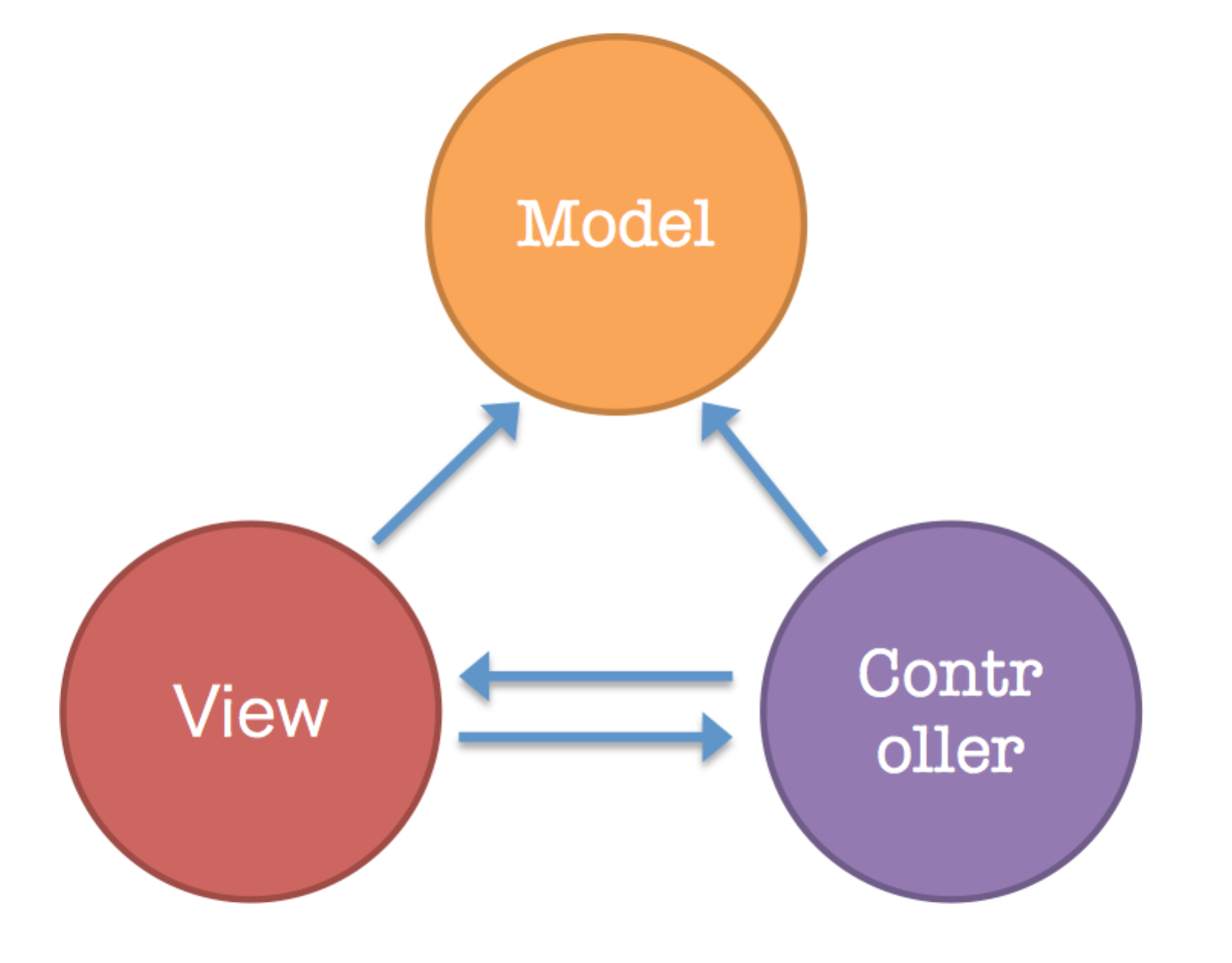
BMCoreFramework 主要是用于搭建程序整体架构，统一定义每个模块，制定每种模块的编程规范。

主要目录结构图如下



## MVC模式

我们把 App 按功 能拆解为 Model-View-Controller 三个部分



按照mvc 模型，归纳代码

1. View 视图

视图是用户看到并与之交互的界面。可继承于BMBaseViewController BMBaseView BMBaseTableViewController BMBaseTableViewCell

2. Model 模型

模型表示数据和规则。

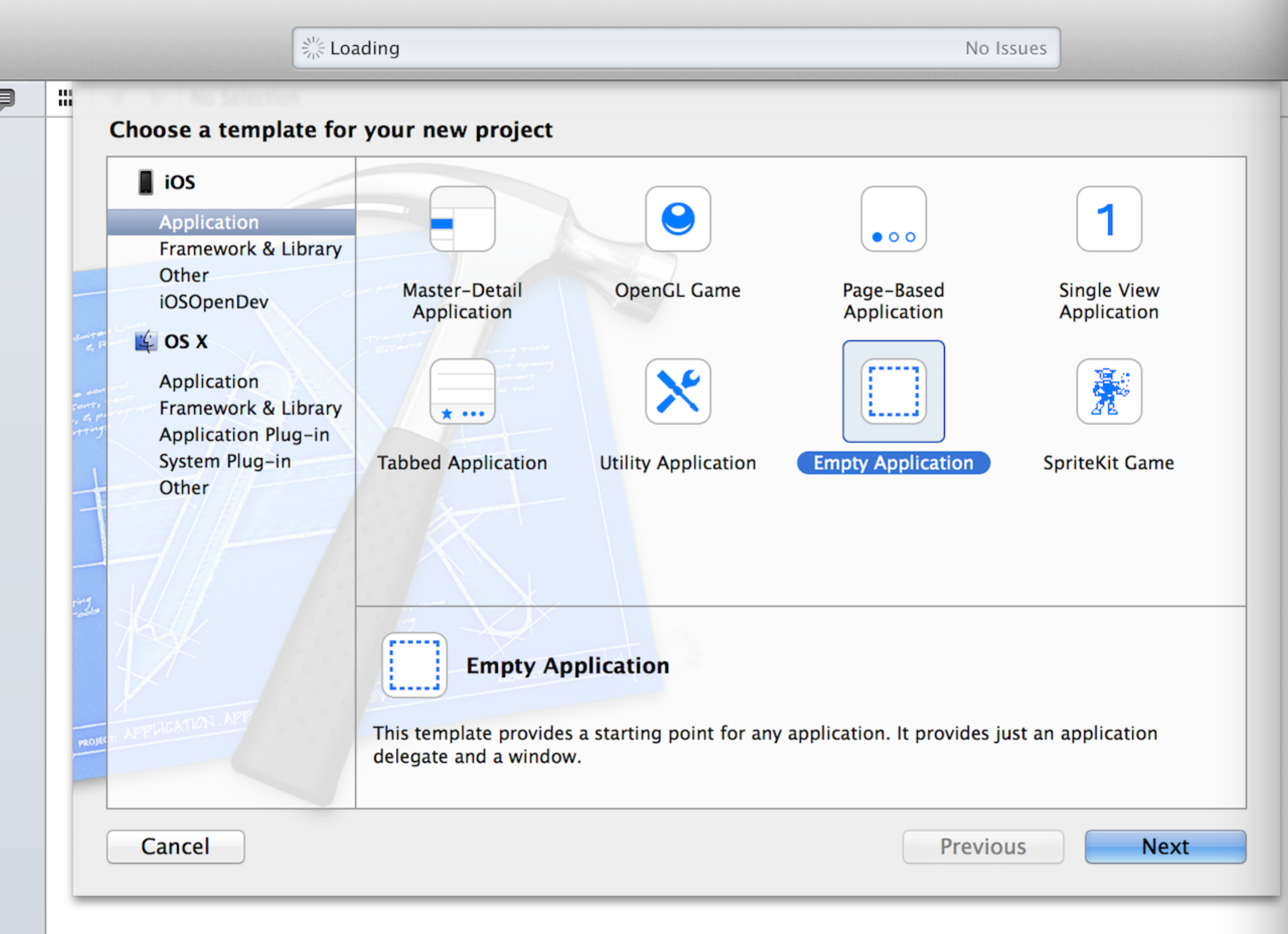
3. Controller 控制器

控制器包括BMControl 和各个用户编写的业务逻辑,通常接受用户的输入,调用网络接口获取数据, 并调用 Model 存储,最后通知 View 显示数据。这里 Controller 必须 继承于 BMBaseAction,更好的一次编写逻辑被多 View 复用。

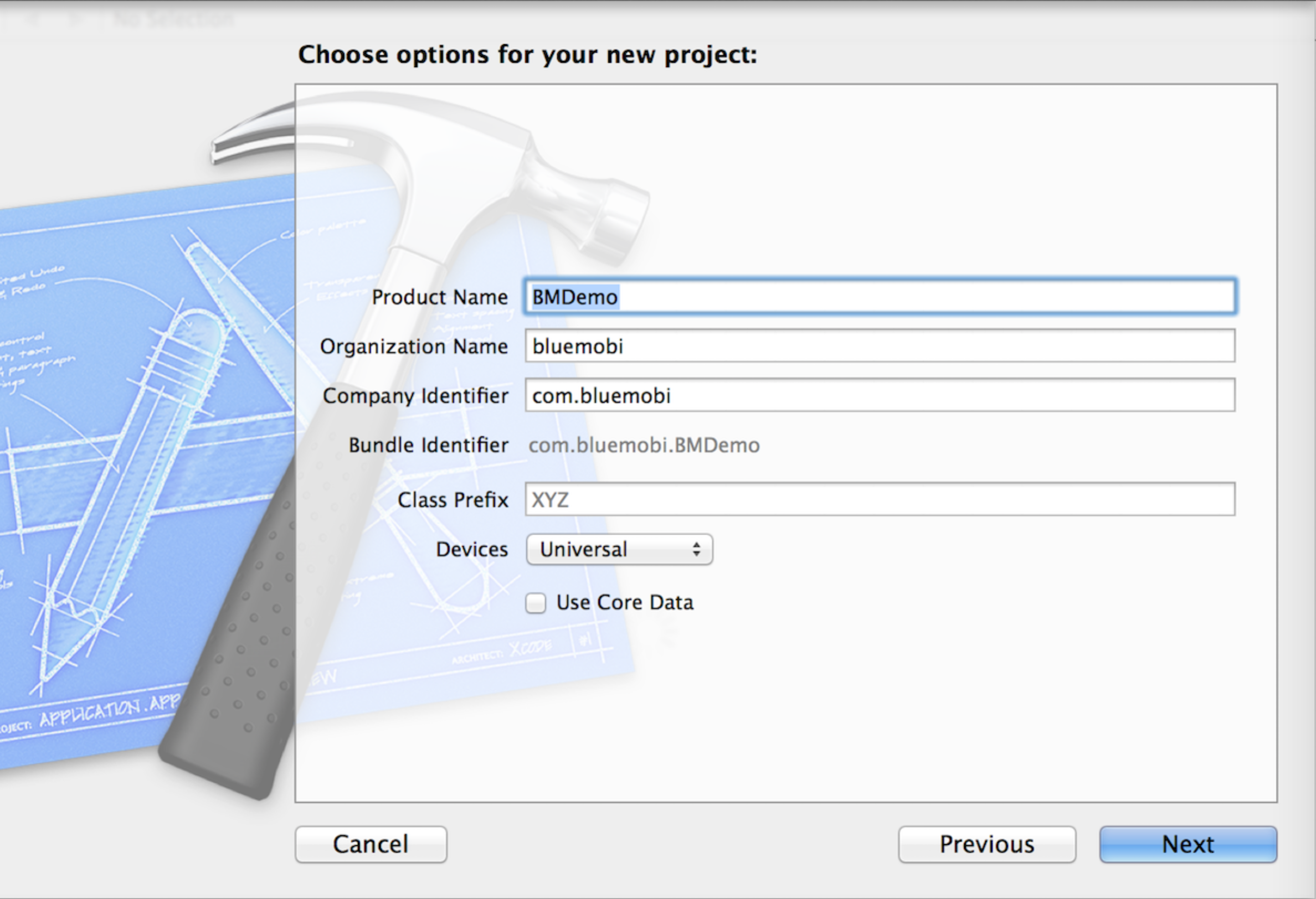
# 使用入门

## 编写第一个程序（使用OC编写）

建立一个空的项目

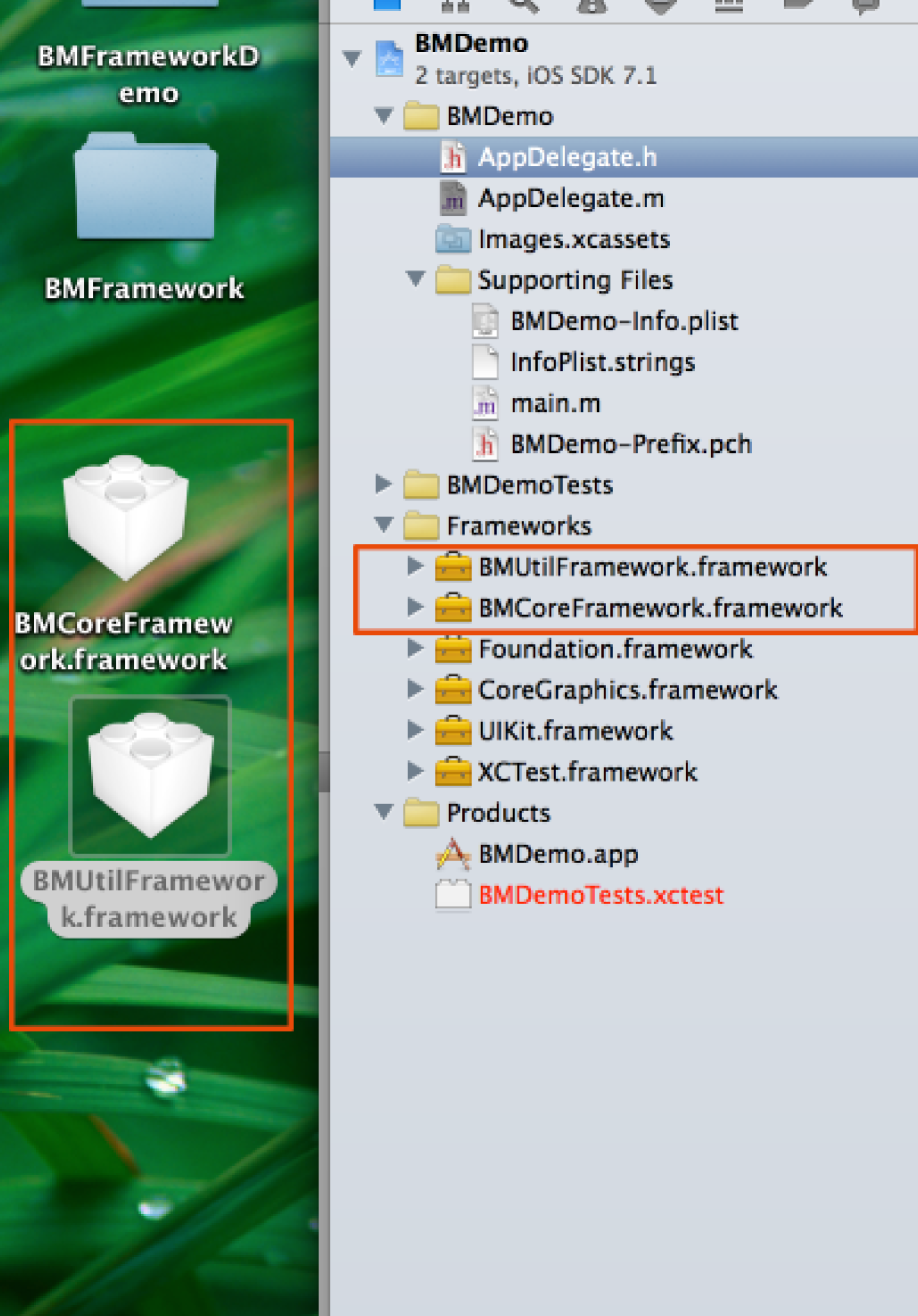


命名为BMDemo



保存。

拖入BMCoreFramework 和 BMUtilFramework 两个包



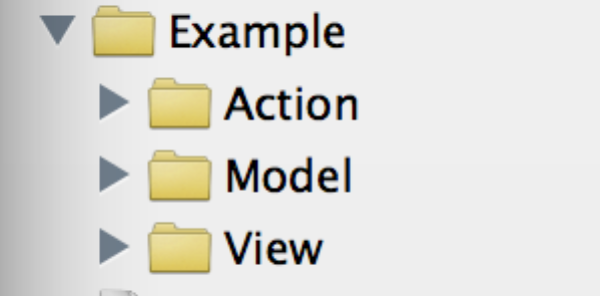
导入

#import <BMCoreFramework/BMCoreFramework.h>

#import <BMUtilFramework/BMUtilFramework.h>

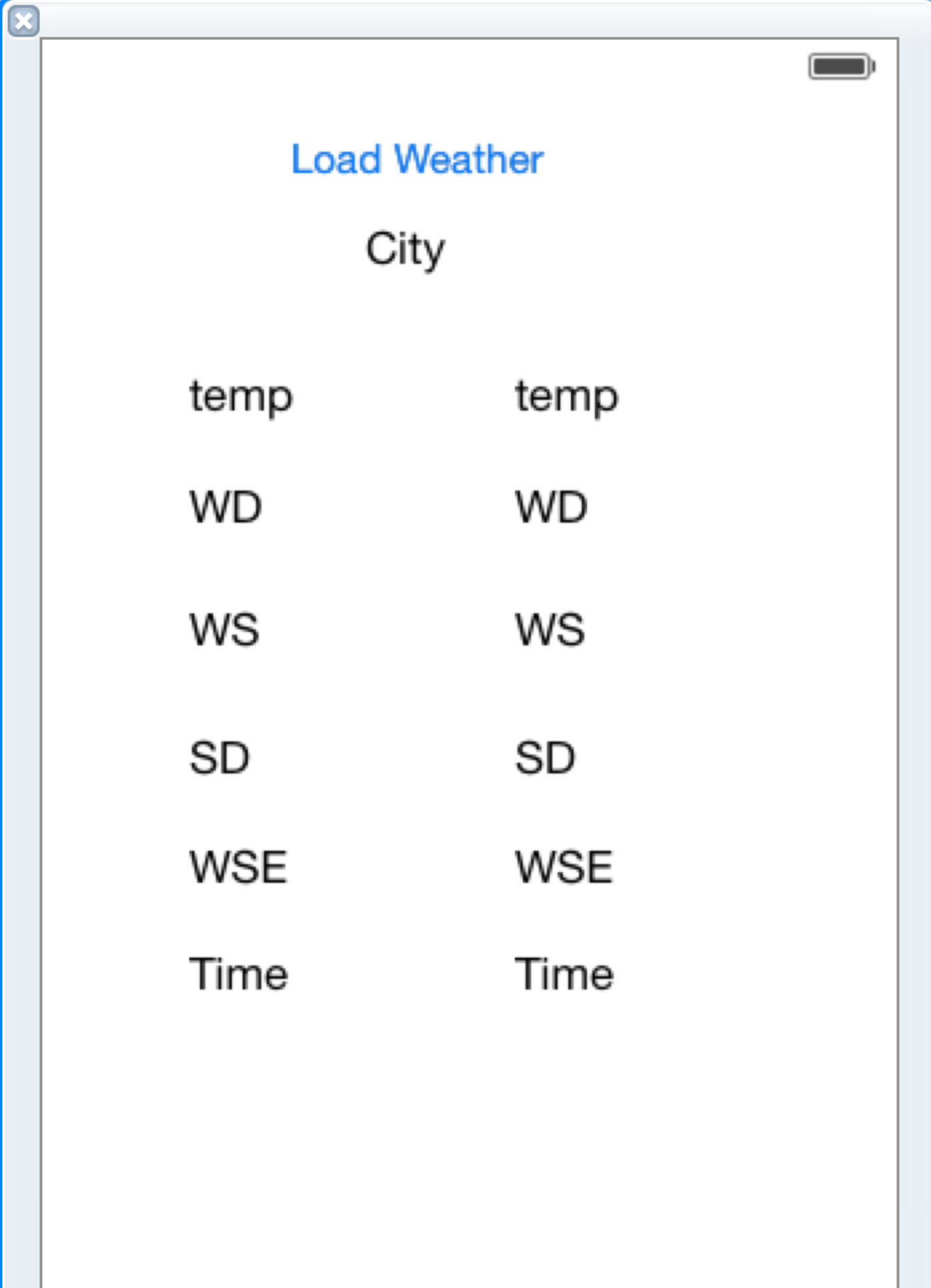


建立目录 Example



建立第一个ExampleViewController 继承BMBaseViewController

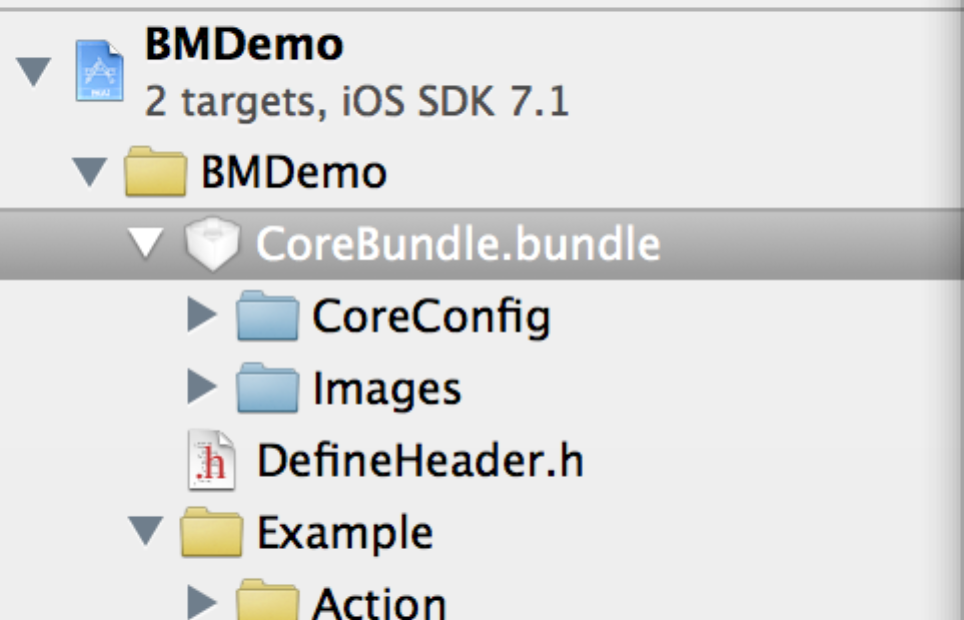
在xib 上拖入几个UILabel 和 UIButton



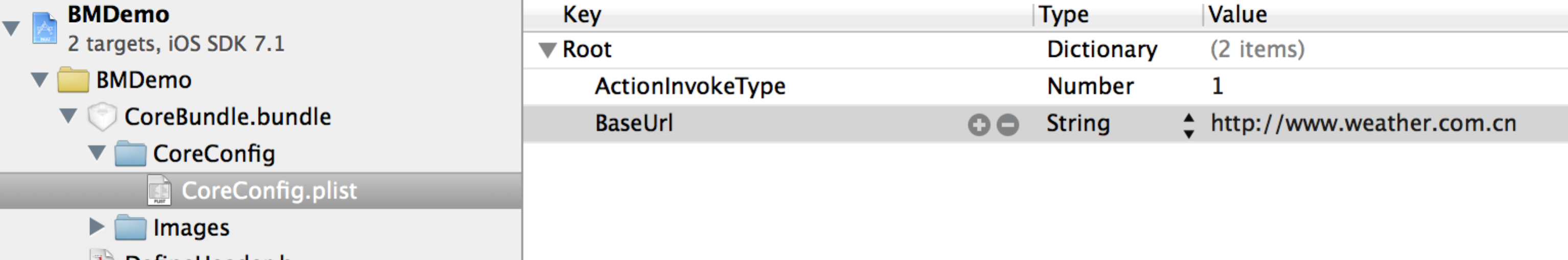
并链接相应的事件。

配置网络访问服务器

拖入BMFramework 文件夹中的CoreBundle.bundle



修改配置文件的url 修改为自己的服务器地址



编写第一个业务逻辑

创建一个ExampleAction 类 继承自 BMBaseAction

编写Action中的获取天气方法

/\*\*

\* 根据参数获取城市天气信息

\*

\* @param baseParam 传入传出参数，所有Action里面的接口方法必须使用BMBaseParam

\*/

-(void)getWeather:(BMBaseParam\*)baseParam

{

//通过baseParam中的String参数获取城市代码

NSString \*strCityCode = baseParam.paramString;

//根据城市编号拼接url

NSString\* strurl=[NSString stringWithFormat:@"%@%@.html",DEF\_API\_GETWEATHER,strCityCode];

//提交Get请求

[[AFAppDotNetAPIClient sharedClient] GET:strurl parameters:nil success:^(NSURLSessionDataTask \*task, id responseObject)

{

//获取json数据，并解析json数据，然后通过block传入调用者

NSLog(@"%@",responseObject);

NSError \*eor=nil;

WeatherResultModel \*weatherresultmodel

= [[WeatherResultModel alloc]initWithDictionary:responseObject error:&eor];

if (baseParam.withresultobjectblock) {

baseParam.withresultobjectblock(0, nil,weatherresultmodel);

}

} failure:^(NSURLSessionDataTask \*task, NSError \*error) {

if (baseParam.withresultobjectblock) {

//解析网络错误

NSString \*strError = [parseerror error];

baseParam.withresultobjectblock(error.code,strError,nil);

}

}];

}

附baseParam传入传出参数，所有Action里面的接口方法必须使用BMBaseParam

#define DEF\_API\_GETWEATHER @"data/sk/" 这个宏表示api的地址

ViewController 中如何调用Action

-(void)loadWeather

{

BMBaseParam\* baseparam=[BMBaseParam new];

baseparam.paramString=self.cityCodeStr;

\_\_weak ExampleViewController \*weakself=self;

baseparam.withresultobjectblock=^(int intError,NSString\* strMsg,id obj)

{

if (intError == 0)

{

//提交成功个

WeatherResultModel \*weatherresultmodel = (WeatherResultModel\*)obj;

if (weatherresultmodel) {

//数据填充

[weakself refreshExampleVC:weatherresultmodel];

}

}

else

{

NSLog(@"获取数据失败");

}

};

NSMutableDictionary\* dicParam=[NSMutableDictionary createParamDic];

//这里传入需要调用的类名和方法名

[dicParam setActionID:DEF\_ACTIONID\_EAMPLE strcmd:DEF\_CMD\_GETWEATHER];

[dicParam setParam:baseparam];

[SharedControl excute:dicParam];

progressTest = [MBProgressHUD showHUDAddedTo:self.view animated:YES];

progressTest.labelText = @"加载中...";

progressTest.mode = MBProgressHUDModeIndeterminate;//可以显示不同风格的进度

[progressTest show:YES];

}

其中 两个宏的定义如下

//---所有命令的调用

//这个是需要调用的方法名

#define DEF\_CMD\_GETWEATHER @"getWeather:"

//需要调用的类名

#define DEF\_ACTIONID\_EAMPLE @"ExampleAction"

编写AppDelegate 代码，修改AppDelegate 继承父类为 BMUIApplication

注释 (BOOL)application:(UIApplication \*)application didFinishLaunchingWithOptions:(NSDictionary \*)launchOptions

方法。

加入load方法

- (void)load

{

self.window.rootViewController = [[ExampleViewController alloc]initWithCityCode:@"101200101"];

}

最后调用效果

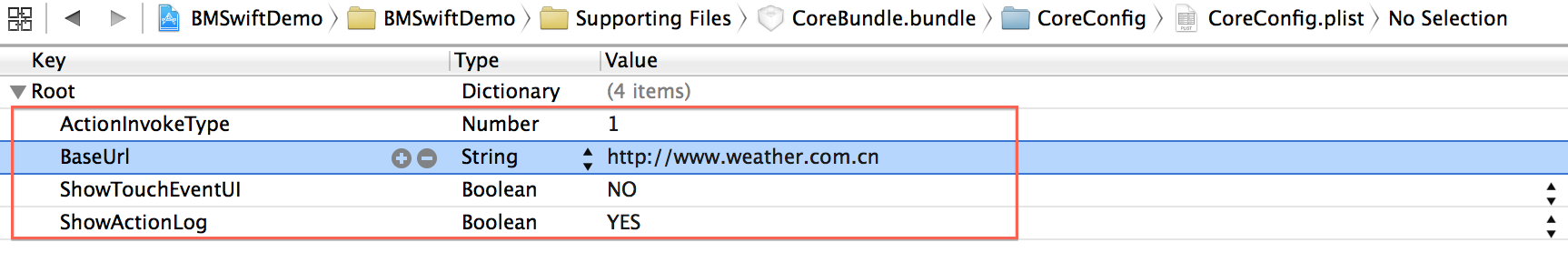


## 使用Swift编写

建立一个swift项目

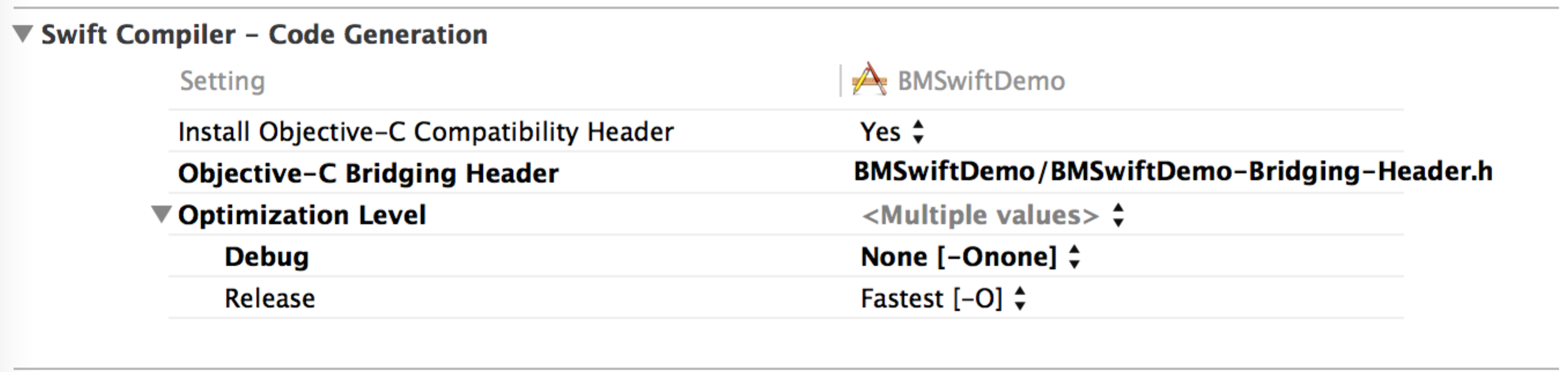


拖入BMUtilFramework.framework BMCoreFramework.framework 两个库和CoreBundle.bundle



修改CoreBundle.bundle里面的配置，设置为自己的项目的服务器地址等。

建立BMSwiftDemo-Bridging-Header.h文件，加入到桥接文件配置



#import <BMCoreFramework/BMCoreFramework.h>

#import <BMUtilFramework/BMUtilFramework.h>

建立第一个业务逻辑

建立文件ExampleAction.swift ,编写getWeather函数

//获取天气函数

public func getWeather(baseParam:BMBaseParam)

{

let strCityCode:String = baseParam.paramString;

let strurl: String = "\(api\_getweather)"+"\(strCityCode)"+".html"

let any:AnyObject?=nil;

//调用服务器api

baseParam.paramObject = AFAppDotNetAPIClient.sharedClient() .GET(strurl, parameters: nil, success: { (task:NSURLSessionDataTask!, responseObject:

AnyObject!) -> Void in

println("\(responseObject)")

let dic:NSDictionary = responseObject as NSDictionary

let weatherbaseclass:WeatherBaseClass =

WeatherBaseClass.modelObjectWithDictionary(dic)

baseParam.withresultobjectblock(0,"1",weatherbaseclass)

}, failure: { (task:NSURLSessionDataTask!, error:NSError!) -> Void in

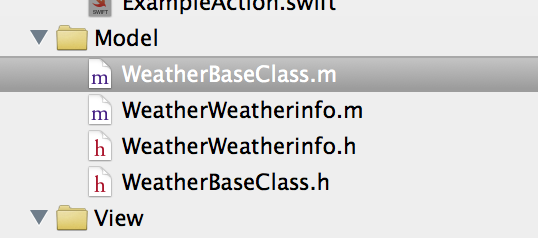
baseParam.withresultobjectblock(0,"1",any)

println("\(error)")

})

}

通过json Accelerator生成文件。



编写界面文件WeatherVC.swift

界面调用业务逻辑代码如下

func loadWeather()

{

if (\_task != nil)

{

\_task?.cancel()

}

//实例化一个传入传出参数

var baseparam:BMBaseParam=BMBaseParam()

//参数

baseparam.paramString="101200101"

baseparam.withresultobjectblock = {(intError:Int32,strMsg:String?,obj:AnyObject?) in

println("come back!!!!")

self.weatherbaseclass = obj as? WeatherBaseClass

self.tableView.reloadData()

};

let dicParam:NSMutableDictionary = NSMutableDictionary.createParamDic();

dicParam.setActionID("ExampleAction", strcmd: "getWeather:");

dicParam.setParam(baseparam);

//.excute(dicParam);

BMControl.sharedInstance().excute(dicParam)

if(progressTest==nil)

{

progressTest = MBProgressHUD.showHUDAddedTo(self.view,animated:true)

}

progressTest!.labelText = "加载中...";

progressTest!.mode = MBProgressHUDModeIndeterminate;//可以显示不同风格的进度；

\_task = baseparam.paramObject as? NSURLSessionTask;

progressTest?.setAnimatingWithStateOfTask(\_task!)

}

效果如下图

