Create UIWebView

LiuYi

在 Cocoa 中， 建立自己的 WEB Game
概括了在项目过程中 屏幕控制，本地数据，Cookie 读取，XML 解析，跨域，网络检测，View 和程序的交互 等技术细节，及具体实现

APP WEB GAME 的框架需求列表

1. 在 APP 中嵌入网页
2. 快速展开游戏的界面(本地资源的利用)
3. 保存用户登录的信息，快捷的再次进入游戏
4. 网络状态检测及提示
5. 减少页面切换过程中的白屏
6. 可配置（XML 解析）
7. 固定的方向控制 （仅可以横屏操作）

1. 在 APP 中嵌入网页

通过 UIWebView 可以在 APP 中嵌入网页

2. 快速展开游戏的界面

在 UIWebView 可以打开 APP 包中的 HTML 文件

UIWebView.loadRequest

UIWebView.loadHTMLString

3. 保存用户登录信息

通过保存过去登录过的用户信息，提供再次登录的秒进入口

NSKeyedArchiver 方法可以保存以键值对的方式，保存数据

4. 网络状态检测

从苹果网站的例子中拿到 Reachability 方法

可以用来检测地址是否可以访问，或检测 wifi，3G 是否打开

5. 减少页面切换过程中的白屏

不能完全杜绝，但可以将登录，注册，秒进合并在一个页面中，

通过建立 Script 的 DOM 结点来和服务器做通信

6. 可配置

GOOGLE 开源的库 GData 可以用来作 XML 的 DOM 方式的解析

用 XML 来保存访问的服，及需要存留的长效 Cookie 的键名

7. 固定方向控制

不使用 ipad 的自动方向调整，使得整个 APP 总是固定为横屏显示

`shouldAutorotateToInterfaceOrientation`

目 录

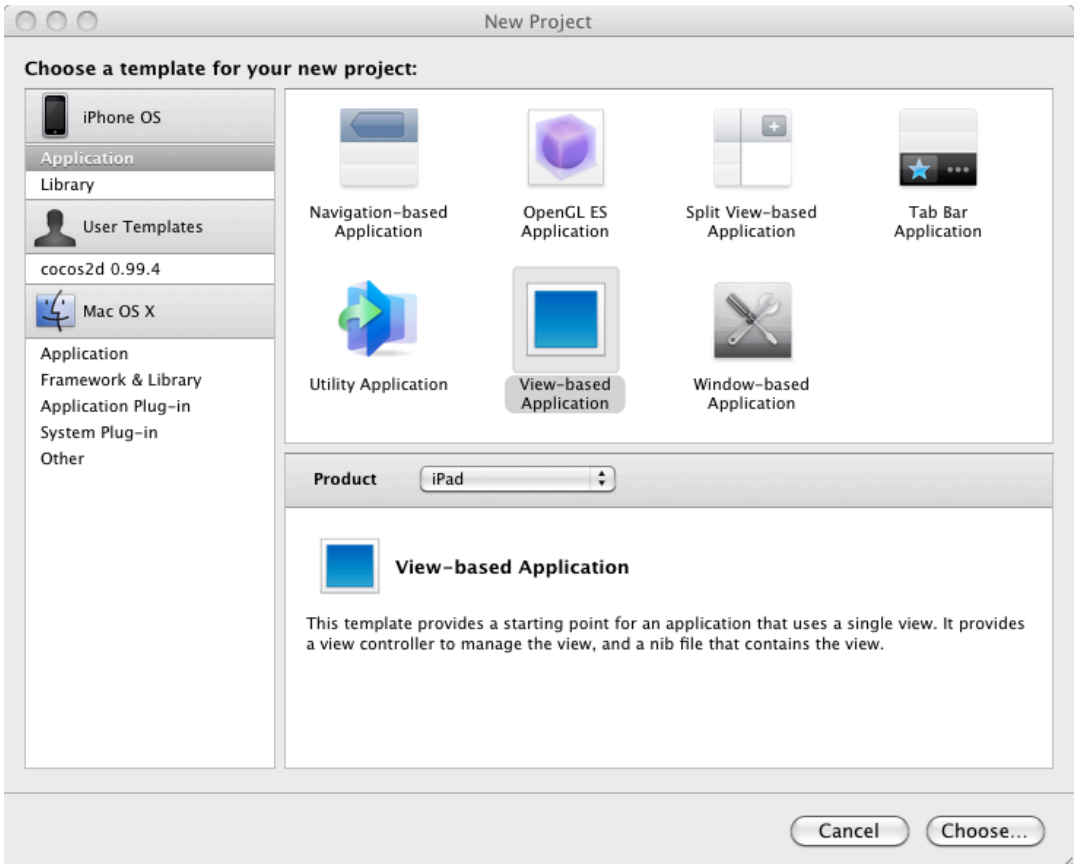
以下以 ipad 开发为例

Xcode 是运行在 Mac OS 下的开发工具
可在苹果的官网注册后免费下载和使用，当前 3.2 版本大约 2.3G
集成的 Object-C 是开发语言
集成的 Cocoa 是一套开发库

在 APP 中创建一个全屏的内置浏览器　　创建一个新 APP 项目

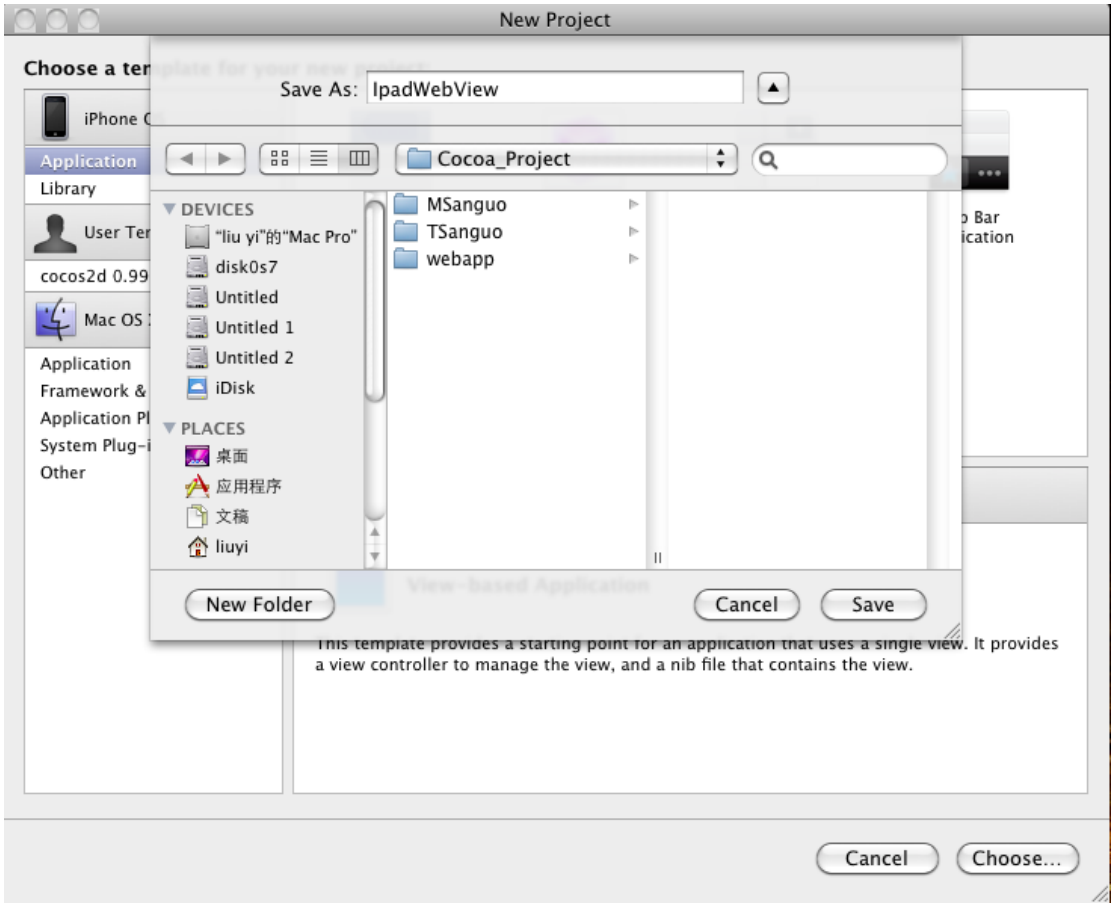
Xcode 运行的首界面 点击 Create a new Xcode project

左侧选择 Application，右上选择 View-based Application
Product 我是为 iPad 开发的，所以选择 iPad，然后点击 Choose

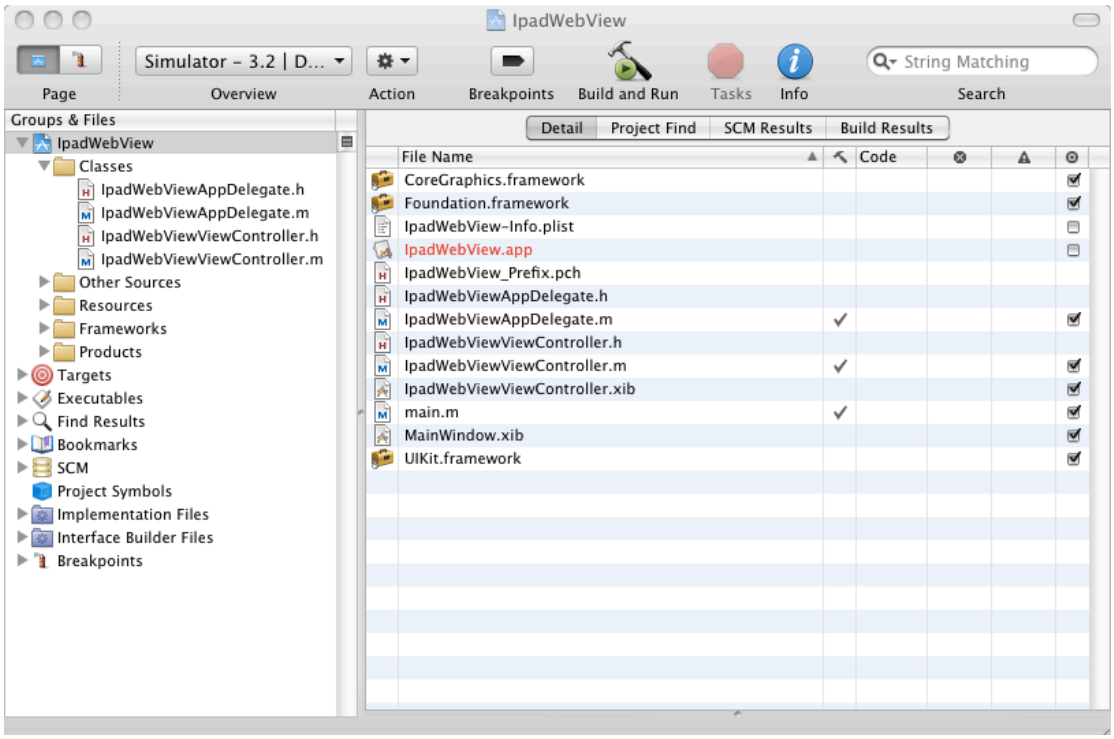


在 APP 中创建一个全屏的内置浏览器　创建一个新 APP 项目

Save AS 后填写一个项目名，然后点 Save



项目建立完成，你能看到新项目的初始的代码文件及结构



在 APP 中创建一个全屏的内置浏览器　创建一个 UIWebView

现在可以点击 Build and Run 应该会打开一个 ipad 的模拟器来，不过新建立的 APP 只是一个白屏

打开 IpadWebViewViewController.h 文件

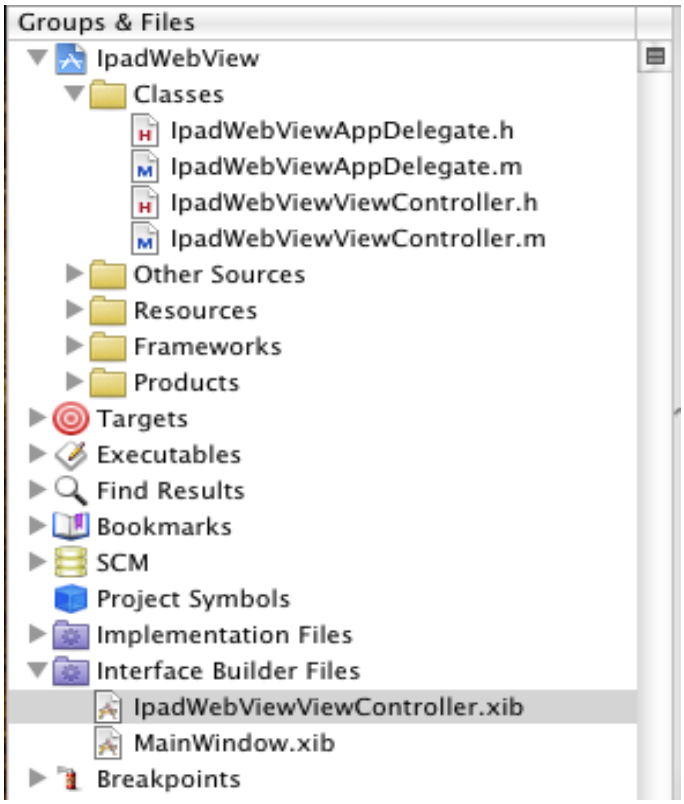
```
UIWebView *MyWebview;
@property (nonatomic,retain) IBOutlet UIWebView *MyWebview;
```

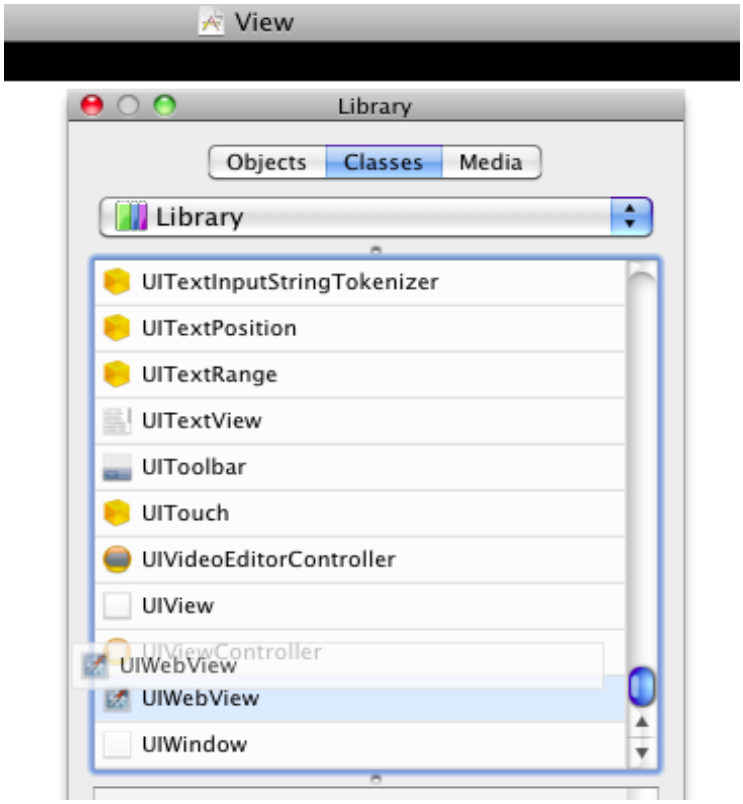
添加上面两行后，整个文件看起来应该是下面这个样子

```
#import <UIKit/UIKit.h>

@interface IpadWebViewViewController : UIViewController {

      UIWebView *MyWebview;
}

@property (nonatomic,retain) IBOutlet UIWebView *MyWebview;

@end
```

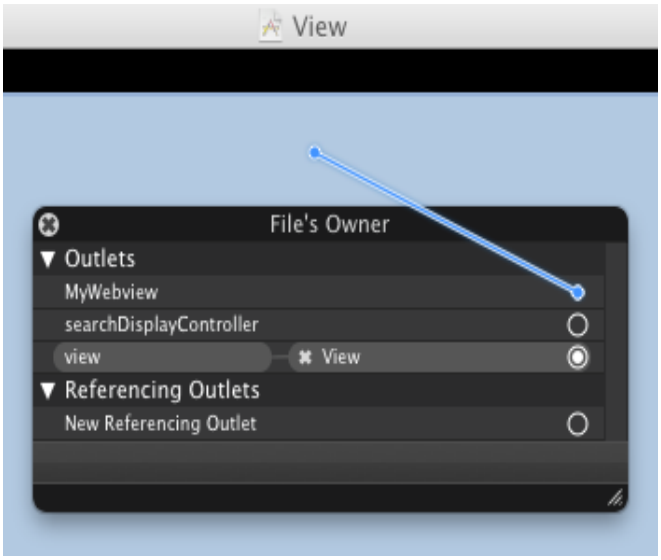
在 APP 中创建一个全屏的内置浏览器　　创建一个 UIWebView

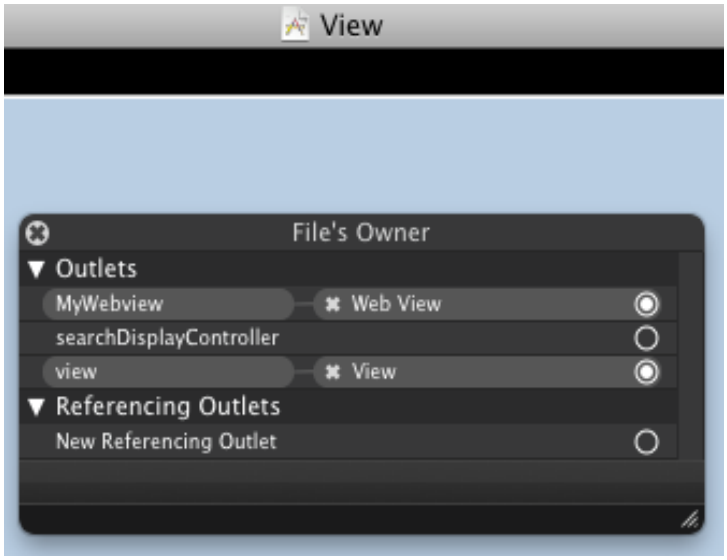
双击左侧 IpadWebViewViewController.xib，弹出 View 编辑器
点击菜单栏 Tools -> Library，弹出 View 列表，选择 Classes
将 UIWebView 拖入到 View 编辑器





将 WebView 指给对象





在 APP 中创建一个全屏的内置浏览器　　例举 UIWebView 的加载

现在，我们的 UIWebView 就创建了，不过，这只是一个空的 UIWebView，
编译并运行他，什么内容也没有，只是一个大白页
所以还需要为这个 UIWebView 添加我们需要的正文内容，
在添加内容前，我们先举例几种 UIWebView 内容的加载方式

大体说来，给 UIWebView 添加内容的方式分两种，
一种是从一个文件加载，类似在浏览器上输入 URL 或本地文件路径
一种是将一段 HTML 原代码，直接赋给 UIWebView

这些代码都是放在 IpadWebViewViewController.m 中

1. 从 URL 上加载内容到 UIWebView

```
#import "IpadWebViewViewController.h"
@implementation IpadWebViewViewController
@synthesize MyWebview;
// 程序主流程
- (void)viewDidLoad {
      // 初始化
      [[UIApplication sharedApplication] statusBarOrientation];
      [super viewDidLoad];
      // 从 URL 中载入一个 html 页面
      NSURL *url = [NSURL URLWithString:@"http://www.google.com"];
      [self.MyWebview loadRequest:[NSURLRequest requestWithURL:url]];
}
```

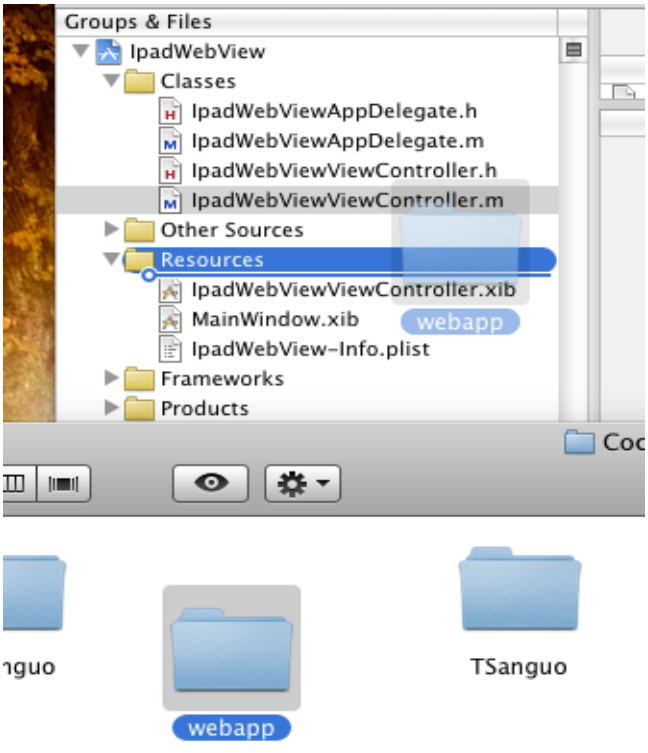
2. 从 APP 包内读取加载一个 HTML 文件

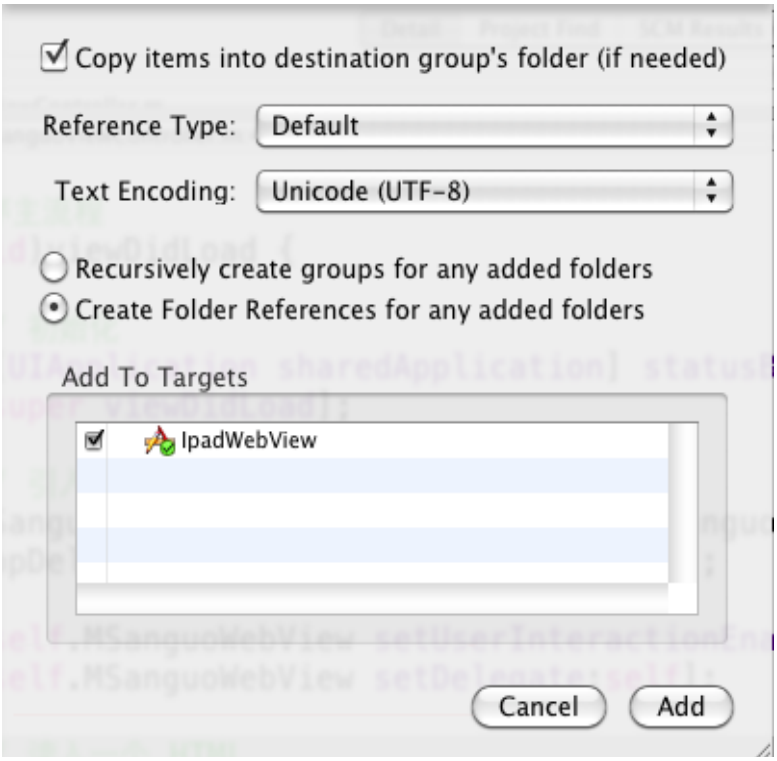
```
#import "IpadWebViewViewController.h"
@implementation IpadWebViewViewController
@synthesize MyWebView;
// 程序主流程
- (void)viewDidLoad {
      // 初始化
      [[UIApplication sharedApplication] statusBarOrientation];
      [super viewDidLoad];
      // 从 APP 包内 载入一个 html 页面
      NSString *htmlPath = [[[NSBundle mainBundle] bundlePath]
stringByAppendingPathComponent:@"webapp/index.html"];
      // NSString *htmlPath = [[[NSBundle mainBundle] resourcePath]
stringByAppendingPathComponent:@"webapp/index.html"];
      [self.MyWebview loadRequest:[NSURLRequest requestWithURL:[NSURL
fileURLWithPath:htmlPath]]];
}
```

3. 直接在 UIWebView 中写入一段 HTML 代码
htmlPath 似乎是一个必须的参数，
而 htmlString 是可以可以直接写成
NSString *htmlString = @"<html ….″ ；
不是必须从一个文件来获取 HTML 串的

```
#import "IpadWebViewViewController.h"
@implementation IpadWebViewViewController
@synthesize MyWebview;
// 程序主流程
- (void)viewDidLoad {
        // 初始化
        [[UIApplication sharedApplication] statusBarOrientation];
        [super viewDidLoad];
        // 读入一个 HTML
       NSString *htmlPath = [[[NSBundle mainBundle] bundlePath]
stringByAppendingPathComponent:@"webapp/loader.html"];
       NSString *htmlString = [NSString stringWithContentsOfFile: htmlPath
encoding:NSUTF8StringEncoding error:NULL];
        [self.MyWebview loadHTMLString:htmlString baseURL:[NSURL
fileURLWithPath:htmlPath]];
}
```

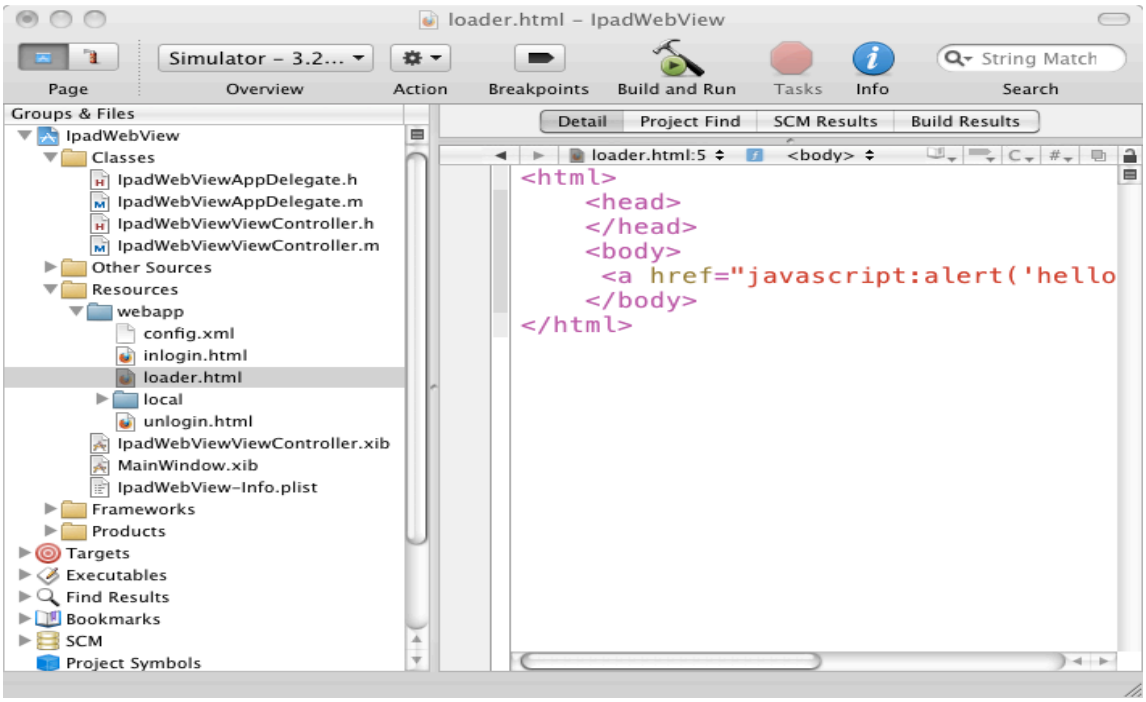
将资源存放在 APP 包内，非常简单，只要将文件或文件夹拖入就行，以下两步操作就实现
注意第二张图的选择，不是默认给的，要将资源文件复制进包内
并且复制进来的是文件目录，而不是项目的一个组





在 APP 中创建一个全屏的内置浏览器　　例举 UIWebView 的加载

文件夹拖进来后，左边 webapp 应该是蓝色的文件夹，右边，我们小小编辑一下，loader.html



编译运行

在 APP 中创建一个全屏的内置浏览器　　控制屏幕的翻转，固定为横屏

Ipad 上，横屏和竖屏对页面的显示，是有放大缩小效果的，一是可能出现图片的变形，特别是地图，二是一般网页的浏览习惯是横长竖短，所以我们需要加入以下调整，使得屏幕固定为横屏
编辑 Project / Resources / IpadWebView-info.plist 替换或加入下面的代码
此文件是你的 APP 的配置文件，它还可以配置你的启动的图片，LOGO 等

```
	<key>UISupportedInterfaceOrientations</key>
	<string>UIInterfaceOrientationLandscapeRight</string>
	<key>UIInterfaceOrientation</key>
	<string>UIInterfaceOrientationLandscapeRight</string>
```

编辑 Project / Classes / IpadWebViewViewController.m

```
// 仅仅允许横屏显示,如果要四个面都能旋转,return YES; 即可
-
(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interface
Orientation {
	// return YES;
	return ((interfaceOrientation == UIInterfaceOrientationLandscapeRight)
|| (interfaceOrientation == UIInterfaceOrientationLandscapeLeft)) ? YES : NO;
}
```

现在不论你 ipad 竖放，还是横放，app 总固定为横屏了

在 APP 中创建一个全屏的内置浏览器　　将屏幕固定为全屏

编辑 Project / Resources / IpadWebView-info.plist 替换或加入下面的代码

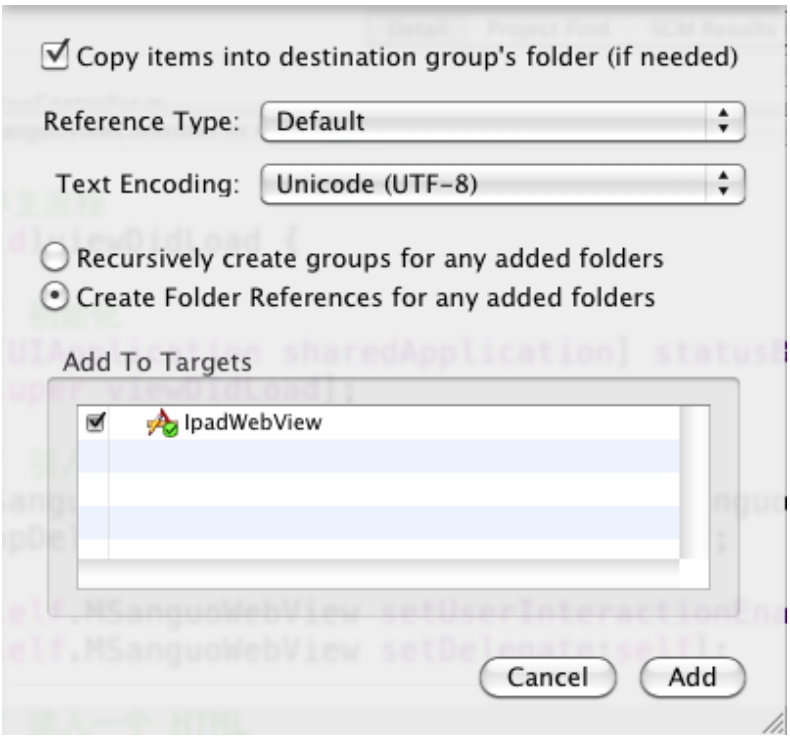
```
    <key>UIStatusBarHidden</key>
    <true/>
```

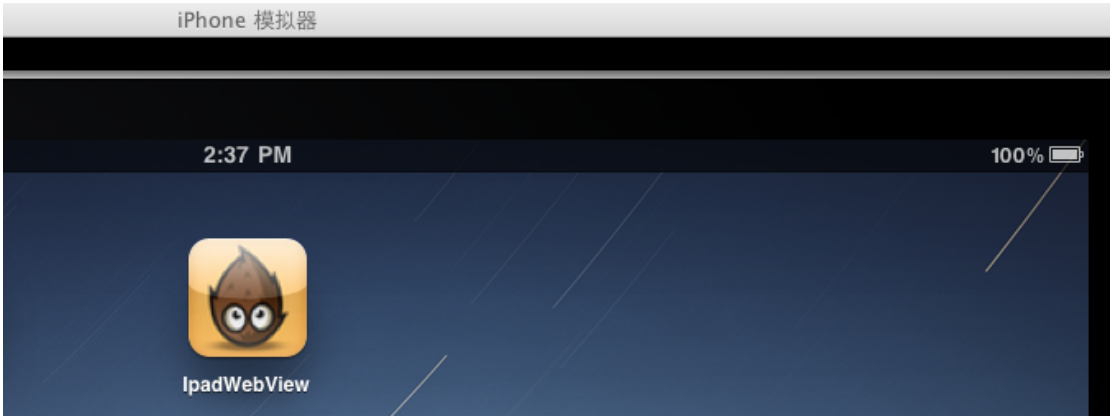
全屏时，没有状态栏，没有时间，电量的显示

添加 LOGO 和 启动图片

默认的，只要在你的项目的目录下，放两个文件 icon.png 和 Default.png 即可
icon.png 是 APP 的 LOG
Default.png 是启动的画面，瞬间即逝，
启动画面会被其他事件阻塞(如启动时检测网络) 而停在启动画面上一会

这两个文件的加入方式，和前面 webapp 目录拖入的选择是一样的，拖入图片到 Project / Resources 下，资源加入方式的选择，和 LOGO 显示如下图





应该可以做到更好

OK

我们现在有一个 UIWebView 了，而且你随着深入的了解，你会发现，这个壳几乎是万能的，它只用编译一次，如果你想修改 HTML 内容，通过 zip 工具来解包，用你喜欢的文本处理工具来修改 HTML，再重新压缩成 zip 文件，即可完成包的修改。这在后面会详述。

但是这个 UIWebView 你会发现有些问题，使得它不那么像一个你想象中的 APP，如：
它可能处在一个没有网络的环境中，却在傻傻的等待服务响应。

- 或是打开一个远端网络时，在数据没有加载前，长时间显示一个白屏（浏览器的个性）。
- 或是当下次打开同一个网址时，你却发现，你还要再登录一次，它一点不智能的存下 Cookie。
- 然后，你希望做一些简单的配置来控制你的 APP，却不想每次调整都要去代码中修改，你最希望的是有个 XML 文件来帮你。
- 而且，你还希望保存一些长效数据，在下次再进入 APP 时，使得不用重头开始，比如等级。
- 最后你还想为当次做一些运行中的状态标记，以便处理一些特殊事件，比如当前打开的页面，是用来登录，还是注册，还是得存 COOKIE 了

噢，还有一点很重要，你一定希望程序和页面有一些交互，比如你能在程序中执行一个 JS 方法，而且是你的自定义的 JS 方法。

好得，上面的需求很多，看起来有些难度，但实际 Cocoa 已经想到了解决方法，你可以轻松应付，然后做得更有价值和趣味
但在解决上面的问题前，我们需要在代码中添加这么一段代码，它的作用是在你的 APP 退出时，将 UIWebView 的内存回收。

```
- (void)viewDidUnload {
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
    self.MyWebview = nil;
}

- (void)dealloc {
    [MyWebview release];
    [super dealloc];
}
```

应该做到更好　优化 View 的加载过程

在 APP 启动时，我希望去检查一下网络，比如和 Google.com 做一次链接。
如果这个链接正常，那好，APP 可以继续往下工作，但因为这个检测不是一个异步的运行过程，(我现在还不了解异步启动一个进程的方法) 在等待网络相应过程中，很容易将 APP 卡在启动动画中，而不是停在 Loading 的页面。这个有点像死机的感觉，所以，我希望网络检测不是放在 APP 启动过程中，而是 Loading 页面加载完之后。

先介绍两 Cocoa 里预置的方法

- `- (void)viewDidLoad`

  是 APP 在启动时运行的第一个方法

- `- (void)webViewDidFinishLoad:(UIWebView *)webView`

  是 UIWebView 每次加载完 HTML 时调用一此方法

1. 在 View 加载完时，加载 UIWebView 并显示出 Loader 页面

```
- (void)viewDidLoad {
	// 初始化
	[[UIApplication sharedApplication] statusBarOrientation];
	[super viewDidLoad];
	// 读入一个 HTML
	NSString *htmlPath = [[[NSBundle mainBundle] bundlePath]
stringByAppendingPathComponent:@"webapp/loader.html"];
	NSString *htmlString = [NSString stringWithContentsOfFile: htmlPath
encoding:NSUTF8StringEncoding error:NULL];
	[self.MyWebview loadHTMLString:htmlString baseURL:[NSURL
fileURLWithPath:htmlPath]];
}
```

2. 当 loader 页面加载完后，开始作网络检，具体网络检查的代码后面会描述

```
- (void)webViewDidFinishLoad:(UIWebView *)webView {
	// 检查网络
	BOOL netConnect = [self CheckNetworkStatus];
	// 如果网络通畅
	if (netConnect == YES) {
		// 开始登录游戏
	} else {
		// 网络链接失败,操作显示
	}
}
```

应该做到更好　用全局变量保存当前实例的状态

```
- (void)webViewDidFinishLoad:(UIWebView *)webView
```

有一个你不喜欢的地方，他总是在切换页面的过程中被调用
比如前面代码里的网络检查，如果你就这么写进 APP 的话，那么每一次页面的切换，都会使得 APP 做一次网络检查，你肯定不喜欢这样，你希望他只在第一次加载完页面时检查网络。
我不知道在这里有无更好的方法，但目前，我用了一个全局的变量，在运行过程中将状态标记起来，比如：
- 现在是第一次加载页面，那么我要去检查网络
- 现在是登录的页面，我要判断用户是否想记住密码
- 现在登录成功了，我想将 COOKIE 保存下来
在不同的过程中，都会调用这个函数，我可以依靠全局的变量来标记，从而告诉这个方法已经做过什么了，现在应该做什么了

这里我用了 appDelegate 对象，这个对象是内建的，在 APP 启动时产生，在 APP 退出时销毁

1. 在 IpadWebViewAppDelegate.h 中加入下面的代码（加在 interface 外）

```
// 记录当次运行实例中 WebView 当前动作
@property (nonatomic, retain) NSString *WebviewAction;
```

2. 在 IpadWebViewAppDelegate.m 文件的前面加入下面的代码

```
// 记录当次运行实例中 WebView 当前动作
@synthesize WebviewAction;
```

3. 在 IpadWebViewViewController.m 文件的 - (void)viewDidLoad 方法中加入下面的代码

```
// 引入全局变量
IpadWebViewAppDelegate *appDelegate = (IpadWebViewAppDelegate
*)[[UIApplication sharedApplication] delegate];
appDelegate.WebviewAction = @"loading";
```

你在 IpadWebViewViewController.m 任意一个地方，都可以通过

```
IpadWebViewAppDelegate *appDelegate = (IpadWebViewAppDelegate
*)[[UIApplication sharedApplication] delegate];
```

来获得这个全局的对象，然后可以对 appDelegate.WebviewAction 进行读写
所有的读写操作都是全局的
现在我们可以在 WebView 加载完页面时作个小小的判断，然后作不同的处理了

应该做到更好　用全局变量保存当前实例的状态

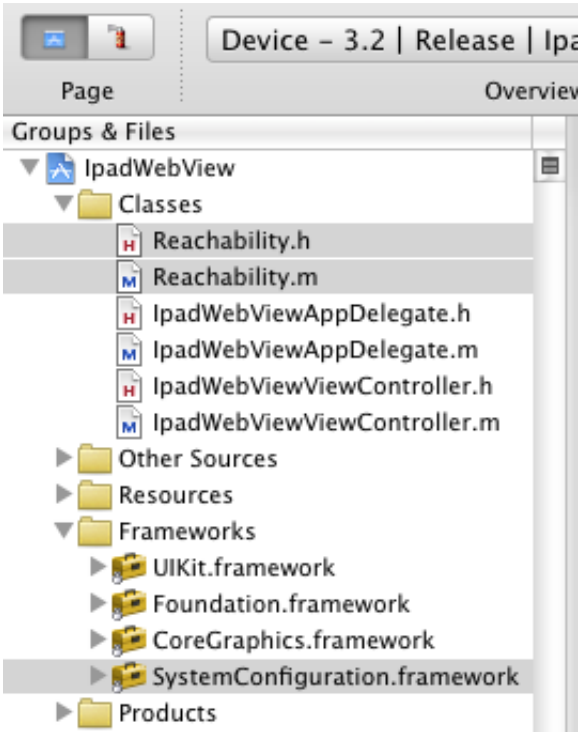
4. 在 IpadWebViewViewController.m 文件的 webViewDidFinishLoad 方法中加入下面的代码

```
// 页面加载完时执行
- (void)webViewDidFinishLoad:(UIWebView *)webView {
	// 引入全局变量
	IpadWebViewAppDelegate *appDelegate = (IpadWebViewAppDelegate *)[[UIApplication
sharedApplication] delegate];
	// 如果是第一次加载
	if (appDelegate.WebviewAction == @"loading") {
		// 效验网络链接
		BOOL netConnect = [self CheckNetworkStatus];
		// 如果网络通畅
		if (netConnect == YES) {
			// 标记状态改变
			appDelegate.WebviewAction = @"login";
			// 开始进入游戏，代码略
		}
		else {
			// 网络链接失败,操作显示，代码略
		}
	}
	else if (appDelegate.WebviewAction == @"login")
	{
		// 登录，保存 COOKIE，标记状态改变
		// 略去业务代码
		appDelegate.WebviewAction = @"savecookie";
	}
}
```

还有一个好的方法来存储你的当前实例的全局变量
再建立一个类，专用来存放你的全局的变量，这样可以分得更清楚
特别是数据相对较多，较复杂的情况
用专用的类来存储你的全局数据，是更好的办法

应该做到更好　网络状况检查

在开发 WEB 等网络应用时，如果网络不能联通，最好能提醒应用者保持网络的连通，如果不处理网络的环境，据说，Apple 的审查是不能通过的。
Apple 的例程 Reachability 中介绍了取得/ 检测网络状态的方法，在我们的 APP 中仅需要将 Reachability.h 和 Reachability.m 拷贝到工程中，将 SystemConfiguration.framework 添加到工程
见下图



Reachability 定义了三种网络状态

```
typedef enum {
      NotReachable = 0,
      ReachableViaWiFi,
      ReachableViaWWAN
} NetworkStatus;
```

NotReachable　无连接

ReachableViaWiFi　使用 WiFi 网络

ReachableWWAN　使用 3G / GPRS 网络

1. 在 IpadWebViewViewController.m 文件中引入 Reachability

```
/**
 * 引入 Reachability 用来检测网络
 */
#import "Reachability.h"
```

应该做到更好　网络状况检查

2. 检测某站点的连接状况

```
Reachability *r = [Reachability reachabilityWithHostName:@"www.mydomain.com"];

switch ([r currentReachabilityStatus]) {
	case NotReachable:
		// 没有网络
		break;
	case ReachableViaWiFi:
		// 使用 WiFi 网络
		break;
	case ReachableViaWWAN:
		// 使用 3G / GPRS 网络
		break;
}
```

我只检查站点能否连接，放在一个方法中

```
// 网络检查
- (BOOL) CheckNetworkStatus {

	NSString *hostName = @"www.mydomain.com";

	Reachability *r = [Reachability reachabilityWithHostName:hostName];
	if ([r currentReachabilityStatus]==NotReachable)
	{
		return NO;
	}
	else {
		return YES;
	}
}
```

方法需要在头文件 IpadWebViewViewController.h 中声明

```
- (BOOL) CheckNetworkStatus;
```

参考网址：

http://www.yifeiyang.net/iphone-web-development-skills-of-the-article-4-make-sure-the-network-environment-3gwifi/

http://developer.apple.com/library/ios/#samplecode/Reachability/Introduction/Intro.html

应该做到更好　用一个 XML 文件来配置 APP

在实际开发中，检测网络的地址，可能常常发生变化，我并不喜欢将这个配置放在代码中，每次更换，都去编译一次，那么我可以用一个 XML 的资源文件来配置这个地址，程序会从这个 XML 中获得用来检测的网络地址，下面就小论下 XML 的解析，之所以小论，是因为 XML 的解析方法很多，这里只说下 用 Gdata 的库来解析 XML，这个项目的网址如下

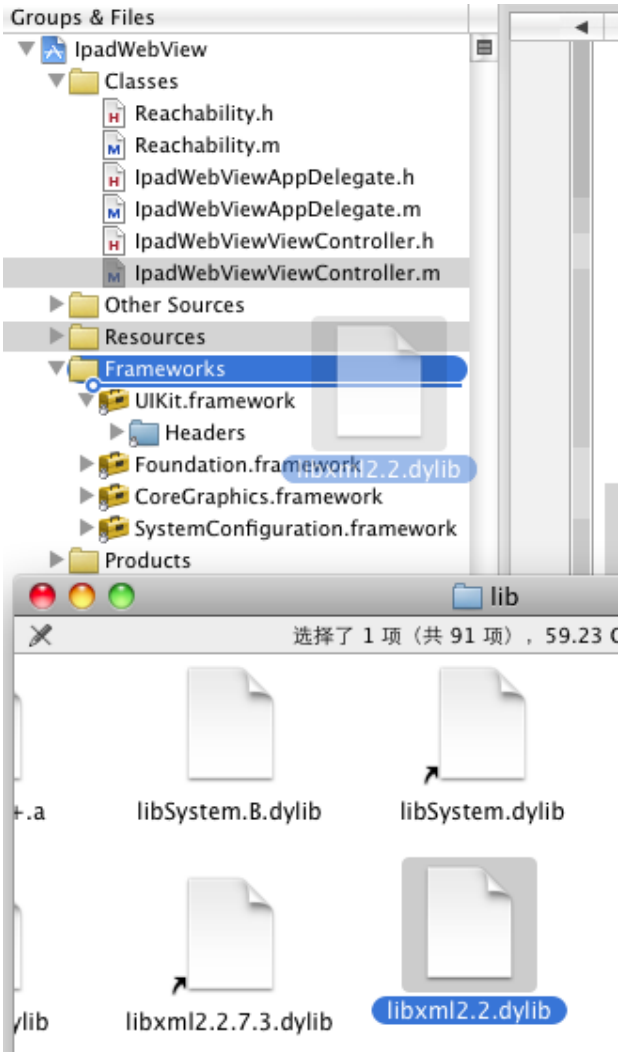
http://code.google.com/p/gdata-objectivec-client/

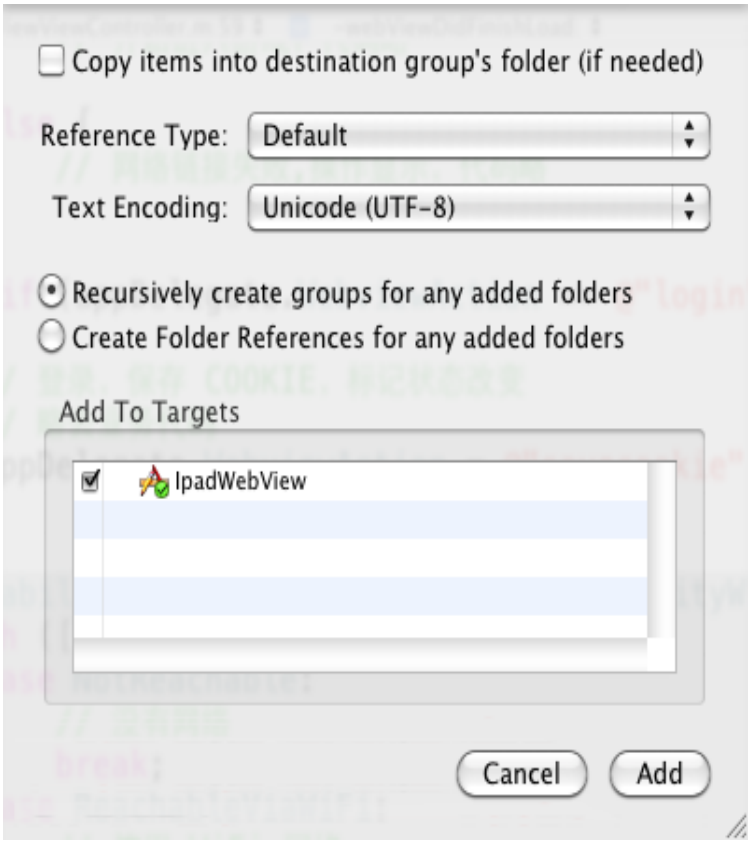
更多的 XML 解释的介绍，可以参考下面这个网址，作者写得更为详尽

http://www.yifeiyang.net/iphone-web-development-techniques-of-the-chapter-1-parsing-xml/

Google Data APIs 是第三方类库，但还是使用了 libxml2

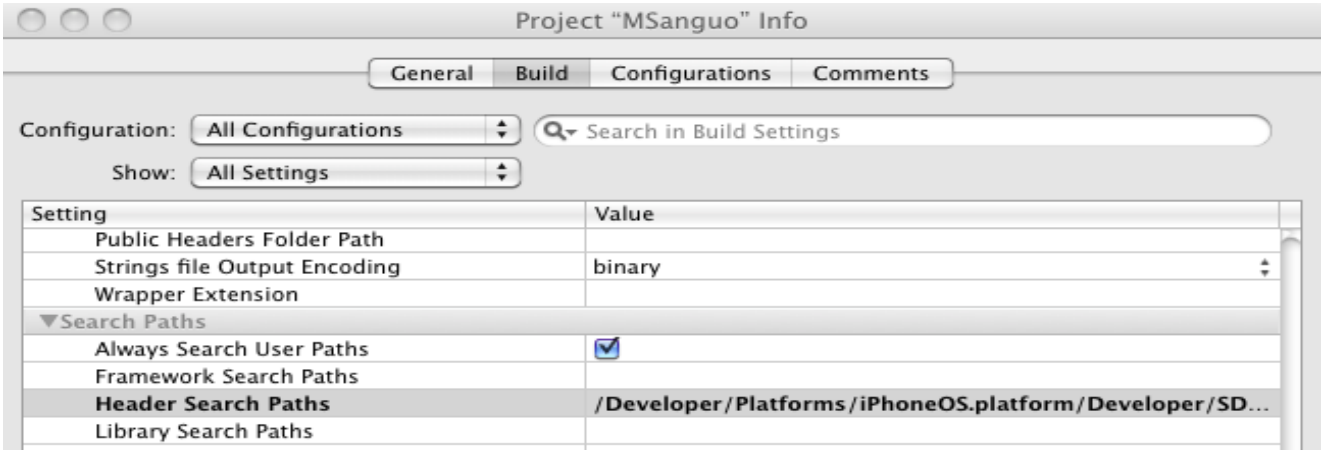
1. 需要将 libxml2 的库添加到项目中，此文件路径
   /Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS{ver}.sdk/usr/lib/libxml2.2.dylib





应该做到更好　用一个 XML 文件来配置 APP

2. 需要将 libxml2 的头文件，添加到编译环境中
/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS{ver}.sdk/usr/include



3. 在 IpadWebViewViewController.m 中添加方法，从指定文件和节点上读取数据
这个方法里，主体代码是如何从 XML 中获取数据

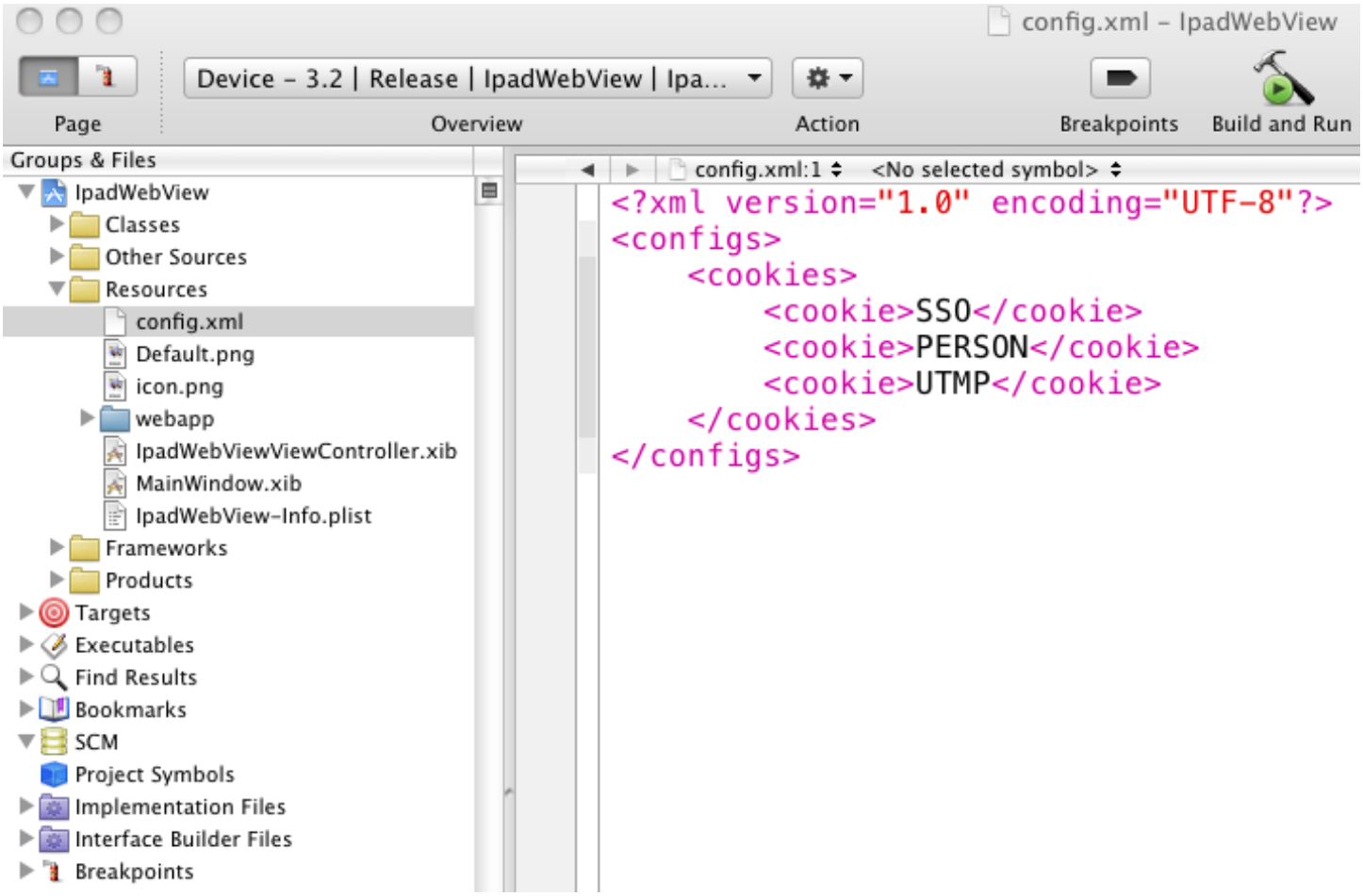
```
// 从 XML 中读取配置信息
// 使用 GData Client 库来解释  XML
- (NSArray *)GetConfigFromXml:(NSString *)xmlFilePath xmlNodePath:(NSString
*)xmlNodePath {

       NSString *xmlPath = [[[NSBundle mainBundle] bundlePath]
stringByAppendingPathComponent:xmlFilePath];
       NSString *xmlString = [NSString stringWithContentsOfFile: xmlPath
encoding:NSUTF8StringEncoding error:NULL];

    NSError *error;
    GDataXMLDocument *document = [[GDataXMLDocument alloc] initWithXMLString:xmlString
options:0 error:&error];
    GDataXMLElement  *rootNode = [document rootElement];

    // get cookie key by xpath
    NSArray *nodeList = [rootNode nodesForXPath:xmlNodePath error:&error];
    return nodeList;
}
```

比如我要在下面的 XML 中获取 /configs/cookies 下的所有 cookie 的内容

```
<?xml version="1.0" encoding="UTF-8"?>
<configs>
     <cookies>
              <cookie>SSO</cookie>
              <cookie>PERSON</cookie>
              <cookie>UTMP</cookie>
     </cookies>
</configs>
```

应该做到更好　用一个 XML 文件来配置 APP

下图是 XML 所在项目的位置



4. 读取 XML

```
// 读入配置文件
NSArray *cookiesNodes = [self GetConfigFromXml:@"config.xml"
                                       xmlNodePath:@"//configs/cookies/cookie"];

// 循环操作
for (int i=0; i<[cookiesNodes count]; i++) {
       // 取值
       NSString *cookieName  = [[cookiesNodes objectAtIndex:i] stringValue];
       // 在 GDB 窗口输出 LOG
       NSLog(@"%@", cookieName);
}
```

GetConfigFromXml 方法需要在 IpadWebViewViewController.h 中声明

关于 XML 的学习就到此了，更多的 XML 的解析方式，可以百度，但最好去 Google

应该做到更好　Cookie 的读取

这里就简单写段代码了，它很容易懂，你应该将它放在 webViewDidFinishLoad 开始后执行

```
NSArray *nCookies = [[NSHTTPCookieStorage sharedHTTPCookieStorage] cookies];
NSHTTPCookie *cookie;
for (id c in nCookies)
{
      if ([c isKindOfClass:[NSHTTPCookie class]])
      {
           cookie=(NSHTTPCookie *)c;
           NSLog(@"%@: %@", cookie.name, cookie.value);
      }
}
```

应该做到更好　存储长效数据

我要读取 COOKIE，为了将 COOKIE 保存下来，在下次进入 APP 的时候，再取出来使用，我写的下面两个方法，可以存取 NSMutableDictionary 类型的数据，这种类型的数据，可以理解为键值对，而且可以很方便的通过下面两行代码读取和修改一个健的值

读取

```
NSString *cookieValue = [cookieData valueForKey:cookieName];
```

设置 / 增加

```
NSMutableDictionary *jsonCookie = [[NSMutableDictionary alloc] init];
[jsonCookie setValue:cookieValue forKey:cookieName];
```

```
// 保存数据到本地
- (BOOL) SetLocalData:(NSString *)dataFile dataObject:(NSMutableDictionary *)dataObject
{
      // 设置路径,并保存
      NSString *savePath = [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES) objectAtIndex:0];
      NSString *saveFile = [savePath stringByAppendingPathComponent:dataFile];
      [NSKeyedArchiver archiveRootObject:dataObject toFile:saveFile];
      return YES;
}

// 读取本地保存的数据
- (NSMutableDictionary *) GetLocalData:(NSString *)dataFile {
      // 按文件名来读取数据
      NSString *savePath = [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES) objectAtIndex:0];
      NSString *saveFile = [savePath stringByAppendingPathComponent:dataFile];
      return [NSKeyedUnarchiver unarchiveObjectWithFile: saveFile];
}
```

补充　NSLog 小释

可能你曾经有过Windows编程的经验，通常在你调试程序的时候，如果希望看到某个中间结果，你会习惯使用MessageBox来显示中间的结果。

有了Cocoa的NSLog，你在写Cocoa程序的时候，就可以无需每次都调用NSAlert来观察中间结果了。
NSLog定义在NSObjCRuntime.h中，如下所示：
void NSLog(NSString *format, ...);

基本上，NSLog很像printf，同样会在console中输出显示结果。不同的是，传递进去的格式化字符是NSString的对象，而不是chat *这种字符串指针。

NSLog可以如下面的方法使用：
NSLog (@"this is a test");
NSLog (@"string is :%@", string);
NSLog (@"x=%d, y=%d", 10, 20);
但是下面的写法是不行的：
int i = 12345;
NSLog(@"%@", i);
原因是， %@需要显示对象，而int i明显不是一个对象，要想正确显示，要写成：
int i = 12345;
NSLog(@"%d", i);
NSLog的格式如下所示：

- %@　对象
- %d,　%i 整数
- %u　无符整形
- %f　浮点/双字
- %x,　%X 二进制整数
- %o　八进制整数
- %zu　size_t
- %p　指针
- %e　浮点/双字　（科学计算）
- %g　浮点/双字
- %s　C 字符串
- %.*s　Pascal字符串
- %c　字符
- %C　unichar
- %lld　64位长整数（long long）
- %llu　无符64位长整数
- %Lf　64 位双字