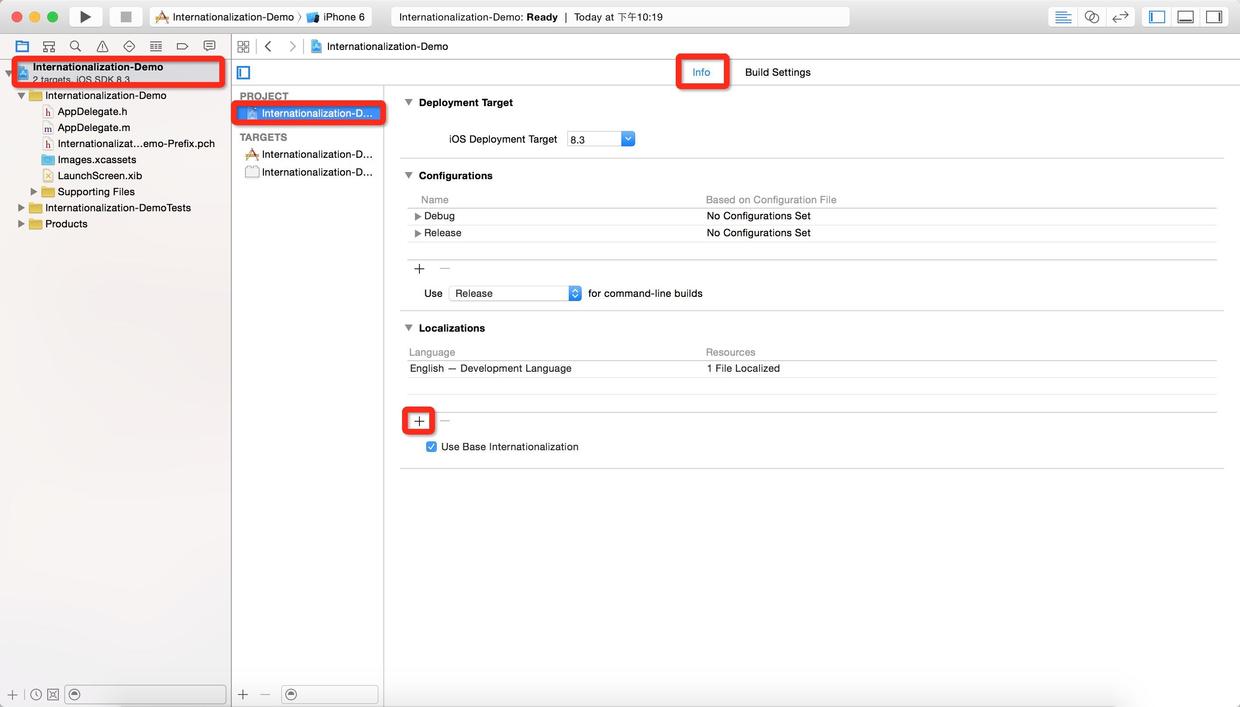
**APP开发到一定的阶段总会有一个质的升华，一些细节也会体现一个应用的品味和气场，当你涉及应用的国际化的时候说明你的APP已经进入一个更好的档次了（此处偷笑）。所谓国际化就是APP支持不同的语言的可阅读可适用性。废话不缩，下面介绍一下ios应用程序国际化。**

**一、程序国际化**

**准备工作：**

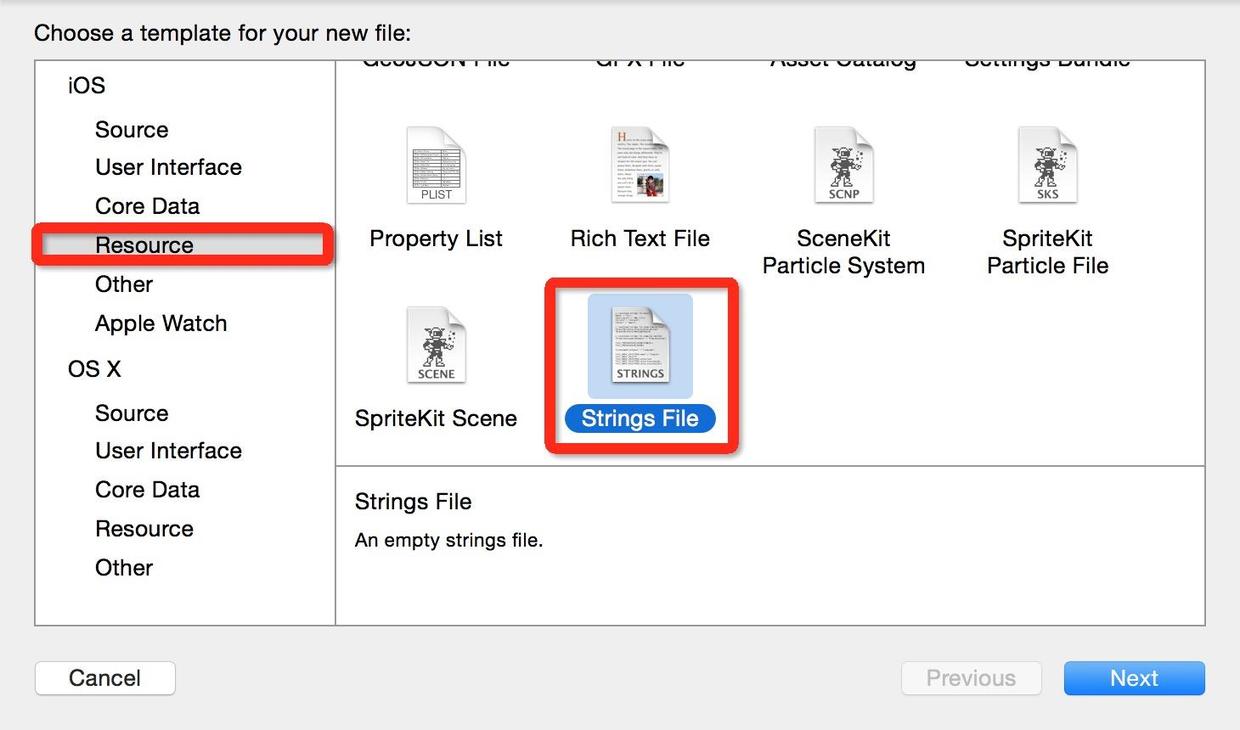
1、首先我们要先创建一个工程，我们给他命名Internationalization-Demo，然后添加语言。



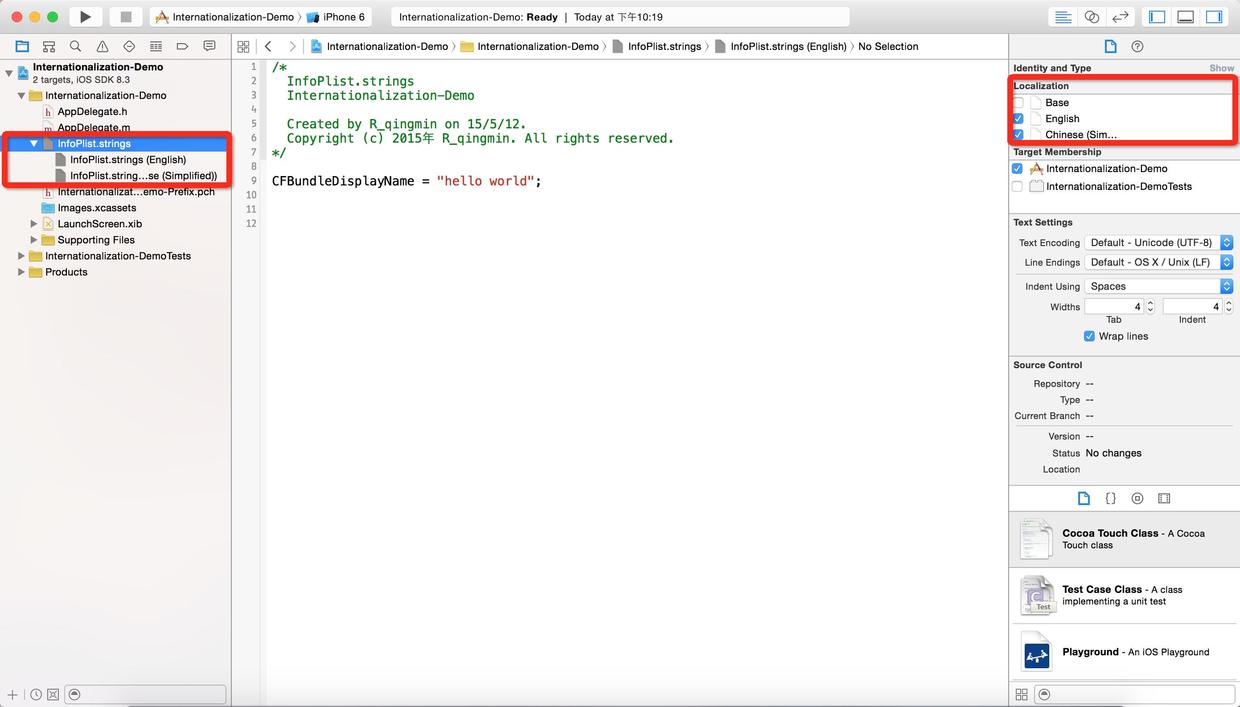
**从代码中分离文本：**

目前，应用展示的所有文本都是以硬编码字符串存在于Main.storyboard 和 ViewController里。为了本地化这些字符串，你需要把它们放在一个单独的文件中。他将会在包中简单地引用这些字符串，而不是在你的方法中进行硬编码。  
Xcode使用带有 .strings 扩展名的文件来存储和检索app中使用的所有字符串，以支持每种语言。根据iOS 设备当前使用的语言，代码中一个简单的方法调用将会查找并返回要求的字符串。

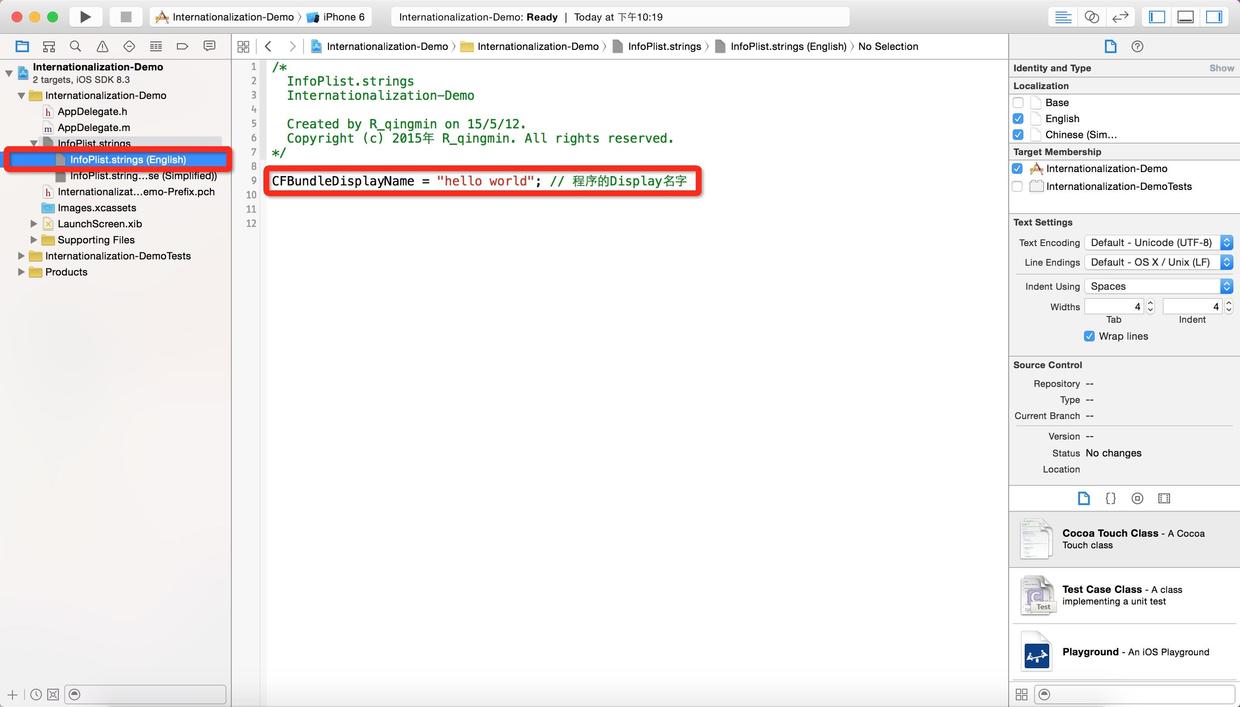
2、创建一个.strings 扩展名的文件  
打开File > New > File，选择Resource中Strings Fils，如图：  
点击下一步，为文件命名为InfoPlist.strings，然后点击save。



3、创建完成后，你可以看到工程目录结构文件如下，单击InfoPlist.strings，查看右边的属性，在Localizable栏添加语言。如图

  
4、添加完成后打开对应语言文件，比如  
English的添加：  
CFBundleDisplayName = "hello world";

Chinese的添加：  
CFBundleDisplayName = "世界 你好";

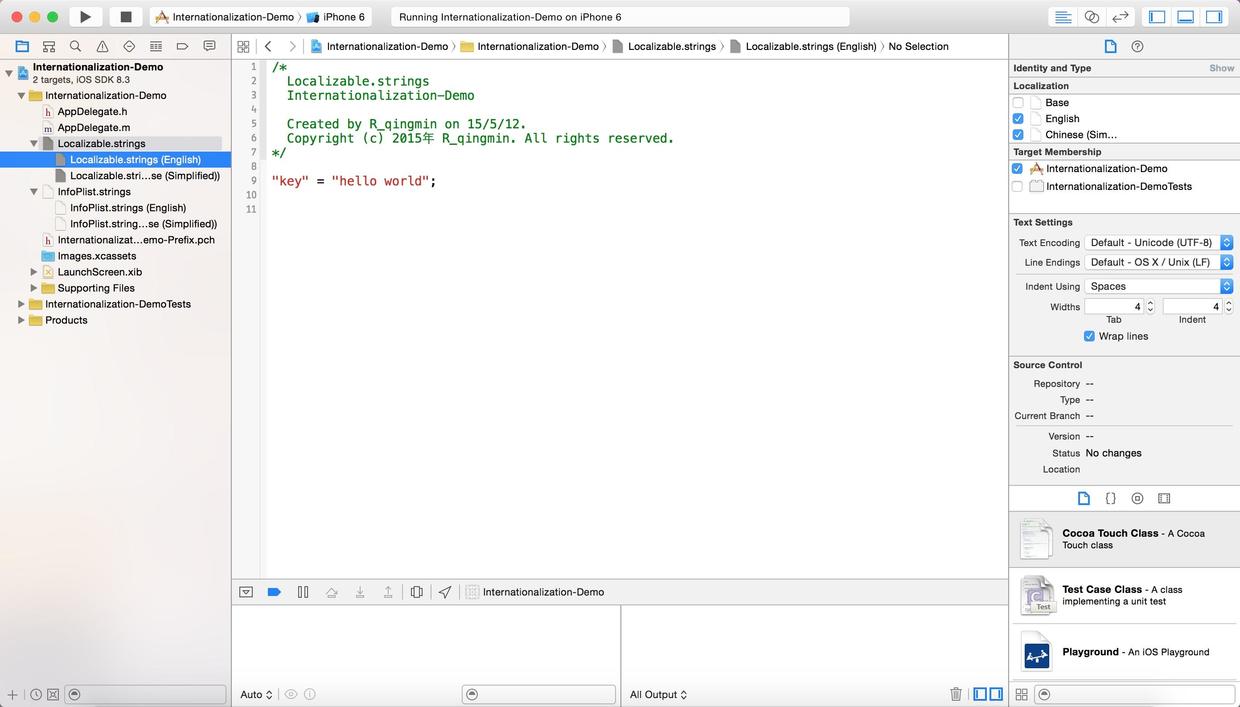


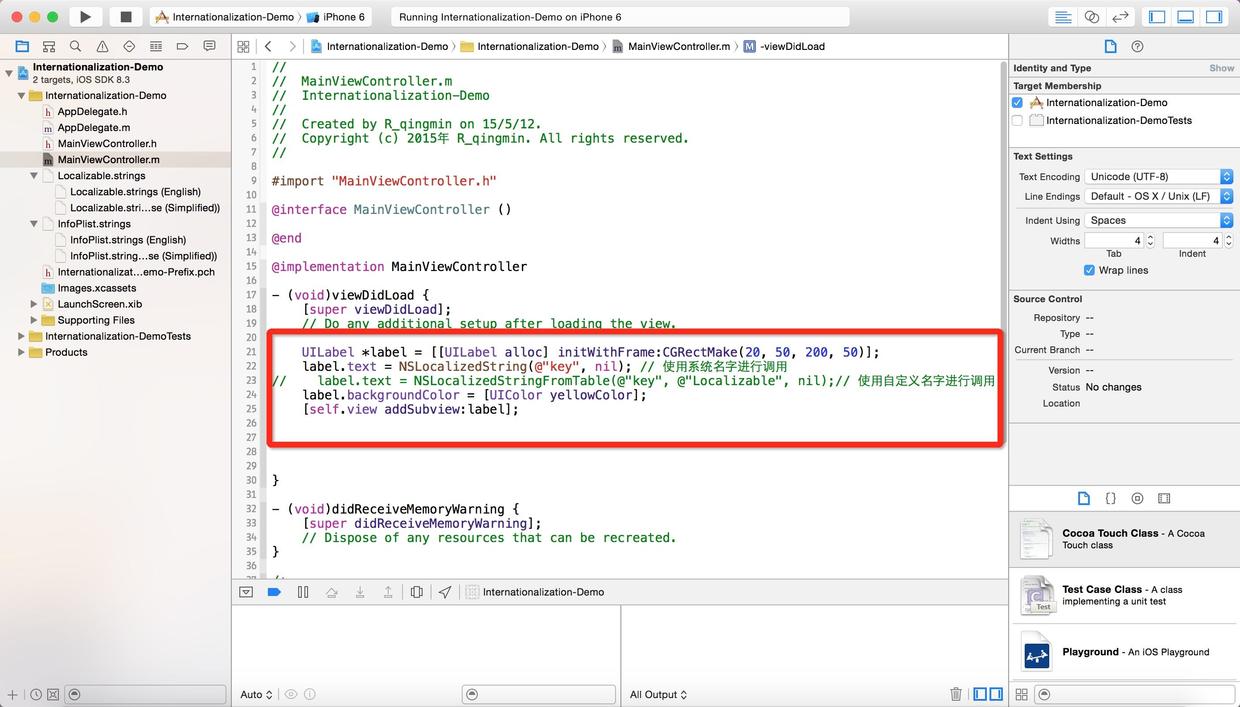
5、运行，如果你的模拟器是中文的，你会看到你的程序名称变成了 世界 你好  
在设置中把语言设置成英文的，你就会看到你的程序名称变成了 hello world

**二、内容国际化**

1、新建一个Localizable.strings 文件 (Localizable.strings 是系统默认名字)  
2、添加语言同上  
3、添加语言内容  
在Localization.strings 中，按照"key" ＝ "value"的格式；然后使用时用NSLocalizedString(@"key", @"")读取内容; 如果不是用系统默认名字那么使用  
Localization.strings English 文件添加  
"key" ＝ "hello world"

Localization.strings Chinese 文件添加  
"key" ＝ "世界 你好"





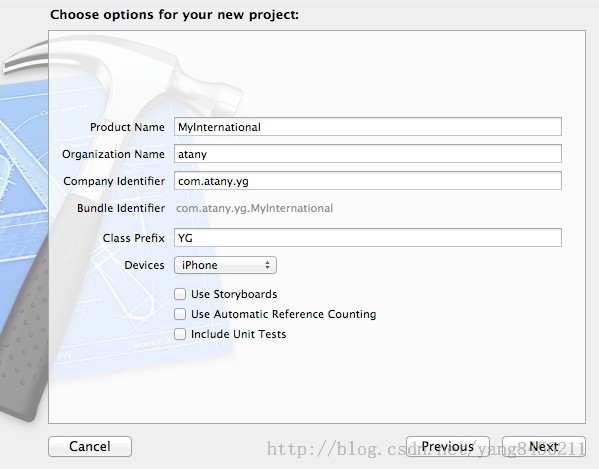
续：☺

至此ios应用的国际化就介绍完了。

=======================别开玩笑了，对于想象空间如股市般捉摸不定的产品们怎么会满足如此垃圾的体验。优秀的产品们当然会想到不要跟随系统语言而实现国际化要APP自己控制，置于是后台控制还是APP客户端开关控制那就无所谓了。接下来我们正式进入ios国际化的重点------------实现APP应用内控制国际化功能。

**1、新建工程**

      新建Single View Application工程，取名为MyInternational。



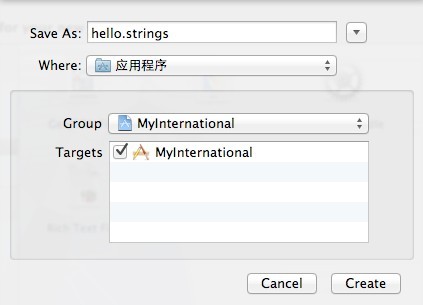
**2、添加多语言文件**

      我们选择使用自己的语言文件，而不是系统的localizable.string，因为在项目开发使用svn时，多人操作同一文件可能会造成冲突。

在Resource中添加新的Strings File：



      取名为hello（记得Targets要勾选项目）

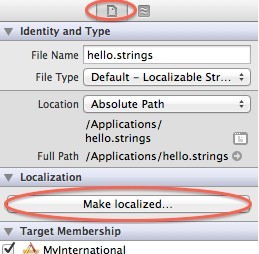


      新建完成之后会生成hello.string文件，把它拖进Supporting Files里。

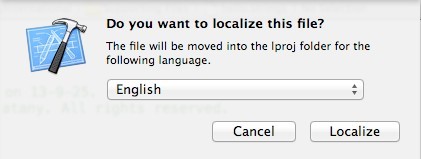
**3、配置hello.string国际化**

      首先配置刚才的hello.string文件。

      点击hello.strings文件，点击圈中的Make localized（本地化）。

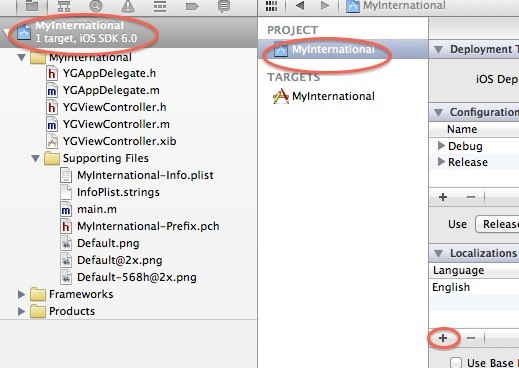


      只有English，没关系点击确定。

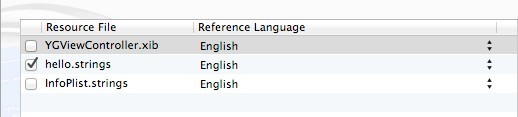


**4、配置项目国际化**

      网络上很多配置方法不靠谱，因为新版的Xcode位置“+”号的位置换了，这里教大家正确的方法，点击项目—>选择Project—>选择Info—>点击“+”号。



      点击“+”号后，添加中文，当然也可以添加其他的语言，弹出对话框，infoPlist是配置系统的国际化文件，可以配置图标之类的多语言环境，不属于本文范围，nib文件看需要配，这里选择不配。



      配置完成之后，再看hello.string便分成了两个文件。



      完成上面的四步便完成了项目内的多语言环境配置，下面来看看代码怎么写。

**5、添加需要的字符串**

      分别在hello.strings（English）与hello.strings（Chinese）里面添加对应语言的字符串。

      字符串的格式为“key”=“value”;键值对（别忘记分号！）

      接下来配置项目需要的字符串，一个用于lable的显示，一个用于button的显示。

    hello.strings（English）

    "invite" ="hello atany,come to join in us!";

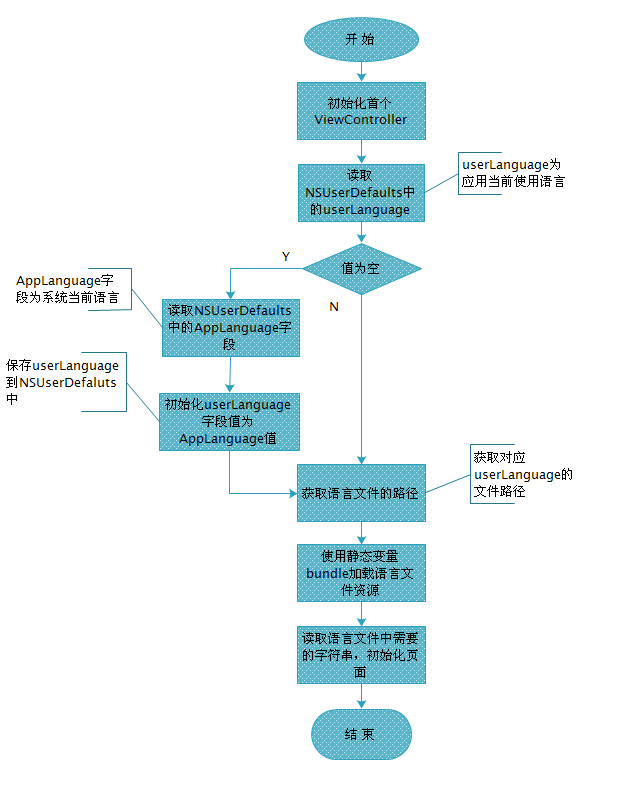
   "buttonInfo" ="change Language";

    hello.strings（Chinese）

    "invite" ="你好 atany,快来加入我们吧！";

    "buttonInfo" ="切换语言";

**6、看看流程**

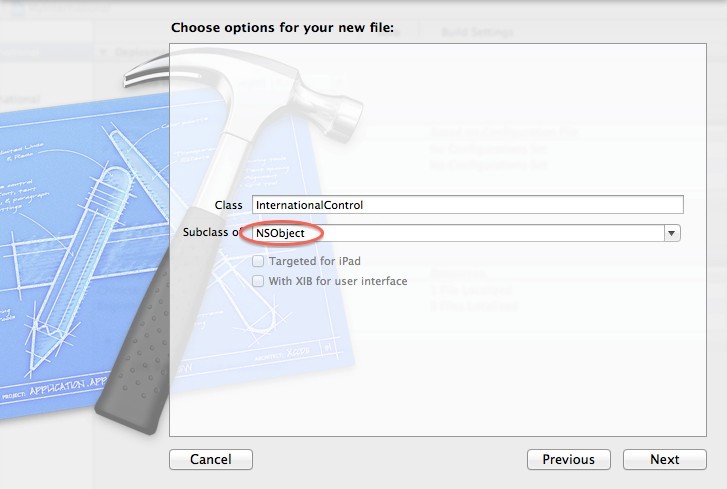


【实现思路:】

      在初始化首个Controller的时候，进行应用语言的加载。首次加载因为userLanguage（应用内语言）不存在，则使用系统当前的语言，保存到userlanguage，下次直接读。读出来之后，获取到对应文件路径，得到文件的索引，储存到工具类InternationalControl的静态变量bundle中。

其他地方需要用字符串时，用工具类获得bundle即可读取相应文件中的字符串。

**7、新建工具类InternationalControl**



**InternationalControl.h**

1. #import <Foundation/Foundation.h>
3. #import <UIKit/UIKit.h>
5. @interface InternationalControl : NSObject
7. +(NSBundle \*)bundle;//获取当前资源文件
9. +(**void**)initUserLanguage;//初始化语言文件
11. +(NSString \*)userLanguage;//获取应用当前语言
13. +(**void**)setUserlanguage:(NSString \*)language;//设置当前语言
15. @end

**InternationalControl.m**

1）创建静态变量bundle，以及获取方法bundle（注：此处不要使用getBundle）。

1. **static** NSBundle \*bundle = nil;
3. + ( NSBundle \* )bundle{
5. **return** bundle;
7. }

2）初始化方法：

 userLanguage储存在NSUserDefaults中，首次加载时要检测是否存在，如果不存在的话读AppleLanguages，并赋值给userLanguage。

1. +(**void**)initUserLanguage{
3. NSUserDefaults \*def = [NSUserDefaults standardUserDefaults];
5. NSString \*string = [def valueForKey:@"userLanguage"];
7. **if**(string.length == 0){
9. //获取系统当前语言版本(中文zh-Hans,英文en)
11. NSArray\* languages = [def objectForKey:@"AppleLanguages"];
13. <pre **class**="cpp" name="code">        NSString \*current = [languages objectAtIndex:0];
15. string = current;
17. [def setValue:current forKey:@"userLanguage"];
19. [def synchronize];//持久化，不加的话不会保存
20. }
22. //获取文件路径
23. NSString \*path = [[NSBundle mainBundle] pathForResource:string ofType:@"lproj"];
25. bundle = [NSBundle bundleWithPath:path];//生成bundle
26. }
27. </pre>

3）获得当前语言方法

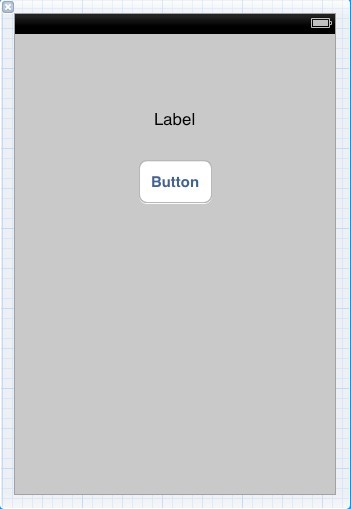
1. +(NSString \*)userLanguage{
3. NSUserDefaults \*def = [NSUserDefaults standardUserDefaults];
5. NSString \*language = [def valueForKey:@"userLanguage"];
7. **return** language;
8. }

4）设置语言方法

1. +(**void**)setUserlanguage:(NSString \*)language{
3. NSUserDefaults \*def = [NSUserDefaults standardUserDefaults];
5. //1.第一步改变bundle的值
6. NSString \*path = [[NSBundle mainBundle] pathForResource:language ofType:@"lproj" ];
8. bundle = [NSBundle bundleWithPath:path];
10. //2.持久化
11. [def setValue:language forKey:@"userLanguage"];
13. [def synchronize];
14. }

**8、拖nib，配置点击事件**

一个button，一个label，button用于切换语言，label用于显示信息。



**9、配置加载的第一个Controller，这里是YGViewController**

**YGViewController.h**

1. #import <UIKit/UIKit.h>
3. @interface YGViewController : UIViewController
5. @property (retain, nonatomic) IBOutlet UILabel \*inviteLabel;//label
7. - (IBAction)changeLanguage:(id)sender;//点击事件
9. @property (retain, nonatomic) IBOutlet UIButton \*btChange;//button
11. @end

**YGViewController.m**

1）加载：

1. - (**void**)viewDidLoad
2. {
3. //注册通知，用于接收改变语言的通知
4. [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(changeLanguage) name:@"changeLanguage" object:nil];
6. [InternationalControl initUserLanguage];//初始化应用语言
8. NSBundle \*bundle = [InternationalControl bundle];
10. NSString \*inviteMsg = [bundle localizedStringForKey:@"invite" value:nil table:@"hello"];
12. NSString \*buttonInfo = [bundle localizedStringForKey:@"buttonInfo" value:nil table:@"hello"];//table为hello.string的文件名
14. [\_btChange setTitle:buttonInfo forState:UIControlStateNormal];
16. \_inviteLabel.text = inviteMsg;
18. [super viewDidLoad];
19. }

2）点击修改语言方法

1. - (IBAction)changeLanguage:(id)sender {
3. NSString \*lan = [InternationalControl userLanguage];
5. **if**([lan isEqualToString:@"en"]){//判断当前的语言，进行改变
7. [InternationalControl setUserlanguage:@"zh-Hans"];
9. }**else**{
11. [InternationalControl setUserlanguage:@"en"];
12. }
14. //改变完成之后发送通知，告诉其他页面修改完成，提示刷新界面
15. [[NSNotificationCenter defaultCenter] postNotificationName:@"changeLanguage" object:nil];
16. }

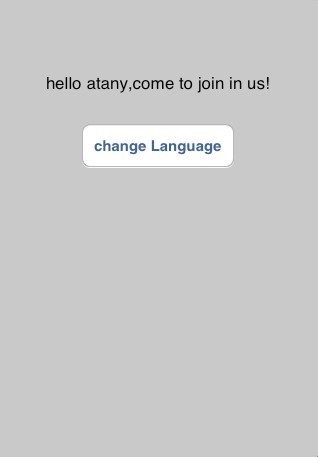
3）接收到通知执行方法，刷新界面

1. -(**void**)changeLanguage{
3. [\_btChange setTitle:[[InternationalControl bundle] localizedStringForKey:@"buttonInfo" value:nil table:@"hello"] forState:UIControlStateNormal];
5. \_inviteLabel.text =[[InternationalControl bundle] localizedStringForKey:@"invite" value:nil table:@"hello"];
6. }

如果有多个界面的话，需要每个界面都配置接收通知，用于修改界面。这样做的确很坑，可以接收到通知初始化应用这样就可以在线切换了。

**10、运行界面**





可以看到点击切换语言之后，语言切换成功。

退出程序再次进入，保留了上次选择的语言。☺

下面提供个比较好的工程供大家学习<https://github.com/githhhh/Test_Local_Two>。

是不是很坑直接看工程那来的多块，小编是想让你看完整个教程在看工程这样效果会比较好。小编真的是为了大家好O(∩\_∩)O~。