# iOS开发------Widget(Today Extension)插件化开发

时间：2016-09-24 12:17:16 阅读：185 评论：0 收藏：0 [点我收藏+]

标签：

iOS10.0发布啦(貌似过去有点时间了吧 - -)，在宏观带给我们使用体验的提升之外，更多的是带给iOS开发者一定的欣喜。

因为我们又要学习新东西来适配10啦。

博文所说的Widget（以下称之为拓展应用）并不是iOS10系统新推出的插件化应用（其实早在iOS8上就已经出现啦，只不过楼主是在iOS10发布之后才算真正的关注它，实在是惭愧呀）。iOS10之前它仅仅是存在于通知那一栏中，至于多隐蔽我就不说了吧。但在iOS10之后获得重生，地位获得了巨大的提升，从这点也不难看出苹果增加了对它的重视。尽管公司的App没有适配Widget，但作为一个“后知后觉”的iOS开发者，注意到了但不研究一下就说不过去了吧?

为了避免真实情况与博文的图不太符合，这里声明一下:楼主用的IDE是最新版的Xcode8.0(没办法，还是迫不及待的进行了升级0.0)，可能会与其他版本的Xcode界面不太一样

博文中的所有代码:https://github.com/RITL/WidgetDemo(如果有用请star支持一下，感谢)

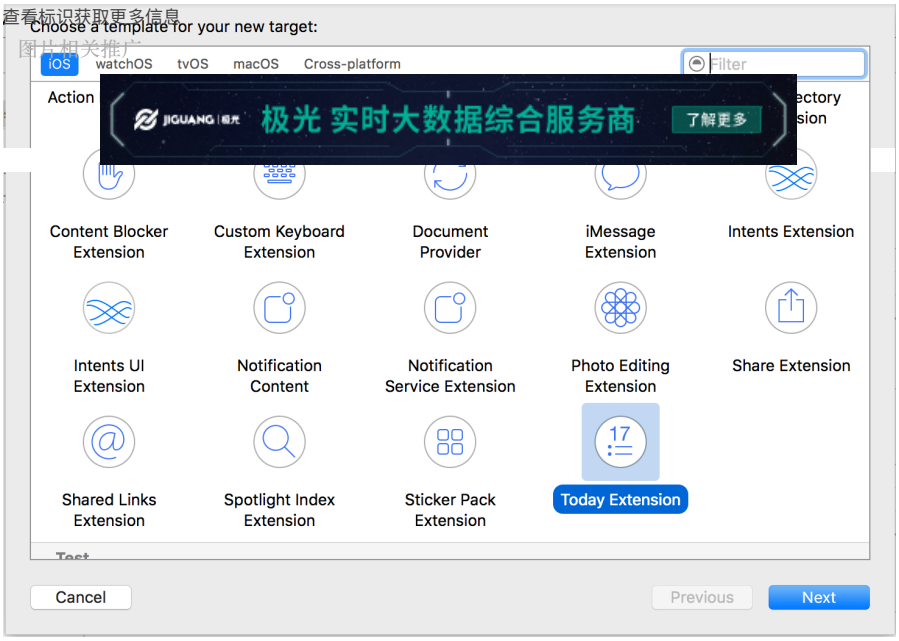
## 预览图

这里附上Widget Demo中完成后的预览图: 这里会稍有不同，如果使用Xcode7及之前版本IDE编译的应用(后面

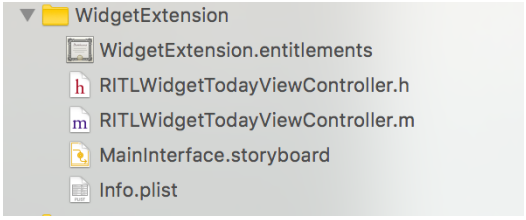
称作宿主应用)，那么找到Widget的方法如图1；如果是Xcode8编译的宿主应用，那么可以直接通过3D Touch唤起Widget，当然通过第一种也是可以的。不过两者本质是一样的。


# 创建Widget Extension

1.首先创建一个新的Target: New->Target，Xcode8 会出现如下界面，选择Today Extension，命名为WidgetExtension:



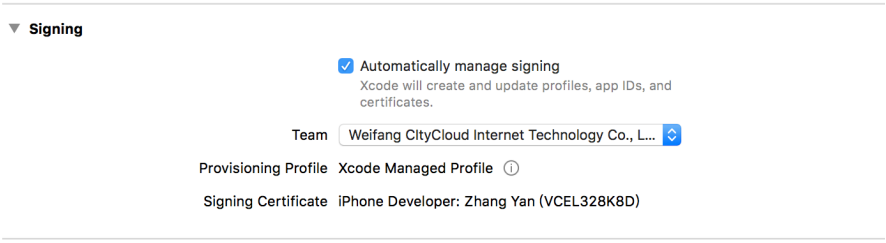

2、创建完毕，则会出现如下文件夹，名字什么的不是问题，一般创建好的名字都为TodayViewController，我只不过是改了改名字而已O(∩_∩)O




3、这里啰嗦一句，虽然作为应用的拓展，但这两个应用是“独立”存在的，你也可以认为这拓展应用与宿主应用是两个完全独立的应用，这也就是说明在开发过程中会出现一些共享的问题，不过共享问题下面博文会有介绍。在此之前，对于拓展应用，我们也是要去开发者申请APP ID以及开发，发布证书的。

由于楼主只是为了学习，用了Xcode8的Automatically manager signing，它的作用是自动生成id以及证书。

作用细说一点就是：如果开发Team没有相应的APP ID，那么Xcode会自动生成APP ID; 如果没有创建相应的证书，那么它会自动创建证书 (当然，正常开发过程中，还是建议手动去创建ID以及配置证书吧)



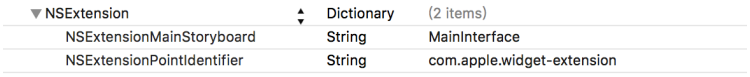

4、证书都配置完毕，运行，添加Widget，就可以看到咱们的项目已经具备了Widget的拓展功能，默认的是MainInterface.storyboard上的内容啦:(我改了改Label上的字，O(∩_∩)O)


# 布局方式interface builder or coding

如果牵扯到UI绘制的方式，这里只需要调整一点东西即可。Demo中楼主选用的是使用storyboard完成快速布局，当然，如果开发者习惯使用代码来完成布局，依旧是可以的。需要对拓展应用的info.plist文件做如下操作:

## 使用interface builder

这个是默认的，如果修改了默认的storyboard，只需要将NSExtensionMainStoryboard的value修改成相应的storyboard名字即可



## 使用coding

首先将NSExtensionMainStoryboard字段删除，添加NSExtensionPrincipalClass字典，value为主控制器的类名即可。

使用这个方法不要忘记在todayViewController的ViewDidLoad中设置preferredContentSize属性调整大小。

| NSExtension | Dictionary | (2 items) |
| --- | --- | --- |
| NSExtensionPrincipalClass | String | RITLWidgetTodayViewController |
| NSExtensionPointIdentifier | String | com.apple.widget-extension |

# 数据共享

很多的时候我们需要Widget与宿主应用共享一些数据，想到数据共享，如果是单一的APP，我们的方法是很多的，比如单例，文件等形式，但由于拓展与宿主应用是两个完全独立的App，并且iOS应用基于沙盒的形式，所以一般的共享数据方法都是实现不了数据共享，这里就需要使用App Groups。

## App Groups

1、首先需要在开发者网站注册一个App Groups


2、在 宿主应用 以及 拓展应用 中将App Groups打开，选中需要共享数据的group


## 两种共享数据的方式

### 使用UserDefaults共享数据

NSUserDefaults大家应该都是非常熟悉的了，通常用法就是

```
//获取UserDefaults的单例对象，完成对应用内相关数据的持久化储存
[NSUserDefaults standardUserDefaults];
```

正像之前所说，由于沙盒机制，拓展应用是不允许访问宿主应用的沙盒路径的，因此上述用法是不对的，需要搭配app group完成实例化UserDefaults，使用UserDefaults类进行数据共享楼主封装为RITL_ShareDataDefaultsManager

通过groups实例化UserDefaults对象的代码如下:

```
//组名
private static let groupIdentifier : String = "group.com.yue.WidgetTest"

/// 获得userDefualt对象
private class func __userDefault() -> UserDefaults
{
    return UserDefaults(suiteName: RITL_ShareDataDefaultsManager.groupIdentifier)!
}
```

存储数据方法如下，至于为什么会有open关键词(与public作用是一样的，只不过开发文档中新的API貌似都改为open了)，因为楼主在Demo中将该文件分离出来了，需要实现”不同命名空间”代码共用，所以Swift默认的Internal作用域就显得权限不足了，至于如何分离下面会提及:

```
//存放数据的键值
private static let defaultKey : String = "com.yue.WidgetTest.value"

/// 保存数据
open class func saveData(_ value : String)
{
    //保存数据
    __userDefault().set(value, forKey: RITL_ShareDataDefaultsManager.defaultKey)
    __userDefault().synchronize()
}
```

获取数据的方法与保存数据很像:

```
/// 获取数据
open class func getData() -> String!
{
    //如果值为nil,表示没有存过值，返回默认的值
    let value = (__userDefault().value(forKey: RITL_ShareDataDefaultsManager.defaultKey))

    __userDefault().synchronize()

    guard value == nil else {
```

```
        return value as! String
    }

    return ""
}
```

因为是通过文件来生成，所以必须要在必要的时候对存储的数据进行删除，如下:

```
/// 清除数据
open class func clearData()
{
    __userDefault().removeSuite(named: RITL_ShareDataDefaultsManager.groupIdentifier)
    __userDefault().synchronize()
}
```

### 使用FileManager共享数据

第二种方法说本质的与第一种是一样的，因为他们都是通过在本地创建文件完成数据的共享，该功能的Demo中封装成了RITL_ShareDataFileManager

与第一种不同的就是，它不但要实例化对象，还需要获得保存数据的路径，如下:

```
//组名
private static let groupIdentifier : String = "group.com.yue.WidgetTest"

//存储的路径
private static let dataSavePathFile : String = "Library/Caches/widgetTest"

/// 获得存储的路径
private class func __fileManagerSavePath() -> URL
{
    //获得当前的组的路径
    var url = FileManager.default.containerURL(forSecurityApplicationGroupIdentifier: RITL_Share
DataFileManager.groupIdentifier)

    //返回拼接完毕的路径
    url?.appendPathComponent(RITL_ShareDataFileManager.dataSavePathFile)

    return url!
}
```

保存数据的方法，因为在Swift中有的方法是throw异常的，所以写法稍有不同，如下:

```
/// 保存数据
```

```
open class func saveData(_ value:String) -> Bool
{
    //进行存储
    do {
        try value.write(to: __fileManagerSavePath(), atomically: true, encoding: String.Encoding.utf8
)

    } catch _ as NSError {//出错

        return false
    }
    return true
}
```

获取数据的方法只是读取存放的数据即可，当然Demo中存的是字符串，方法实现如下:

```
/// 获取数据
open class func getData() -> String
{
    //用于接收数据
    var value : String

    do {//读取数据
        try value = String(contentsOf: __fileManagerSavePath())

    } catch _ as NSError {

        return ""//有误输出空字符串
    }
    return value
}
```

必要时候不要忘记清除数据:

```
/// 清除数据
open class func clearData() -> Bool
{
//其实不太规范，应该先判断是否存在该文件，再进行删除
    do {//开始删除
        try FileManager.default.removeItem(at: __fileManagerSavePath())

    } catch _ as NSError{

        return false
```

```
    }
    return true
}
```

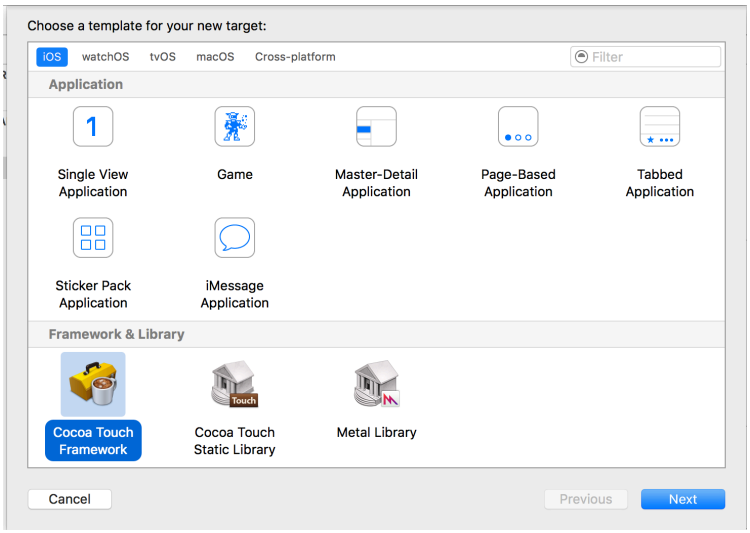
# 代码共享

这里为什么会有代码共享呢，如果上面两个存储的类写在了宿主应用目录下，那么宿主应用使用是没有问题的，but，这个时候拓展应用是获取不到这两个类的，当然，如果每个应用里都写一套不就可以了，虽然这样也能解决问题，但我很难用完美解决问题来形容他，因为这样不仅会出现命名，不好维护等众多问题，严重的时候还会带来很多问题，话不多说，如何共享代码呢?

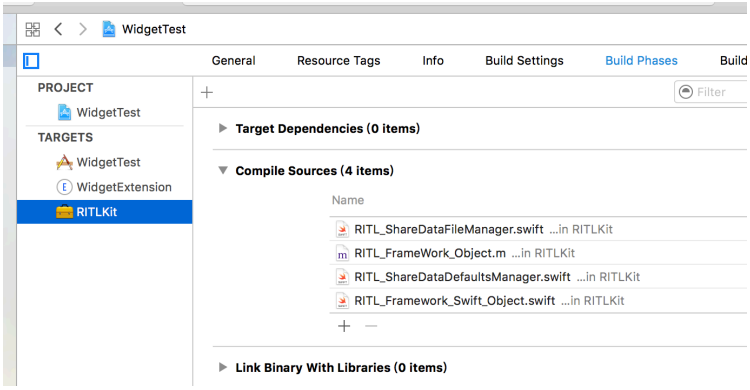
## 使用Framework

这个问题在iOS8之后能够完美的用framework来解决，(如果有人问iOS7怎么办？请面壁3秒钟，Widget不是iOS8才对我们开放的么0.0)

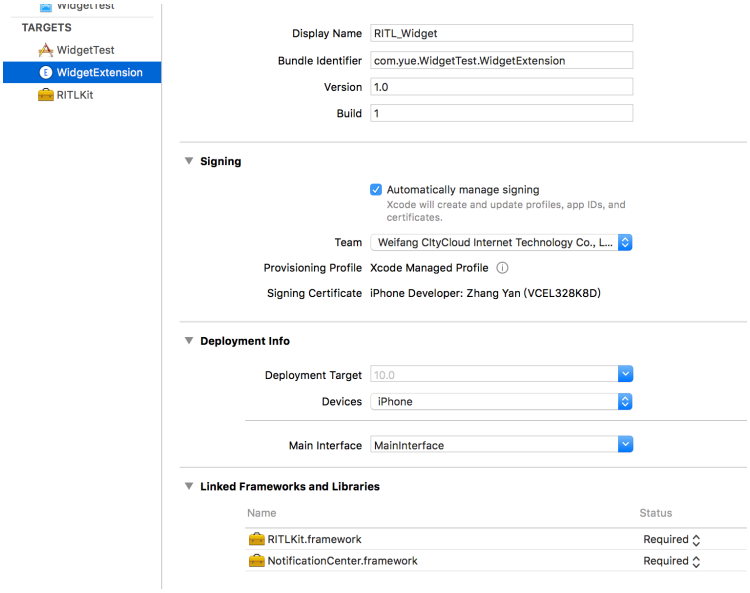
1、与创建拓展一样，New->Target，选择Cocoa Touch Framework来创建framework，Demo中命名随便了一点，起名为RITLKit



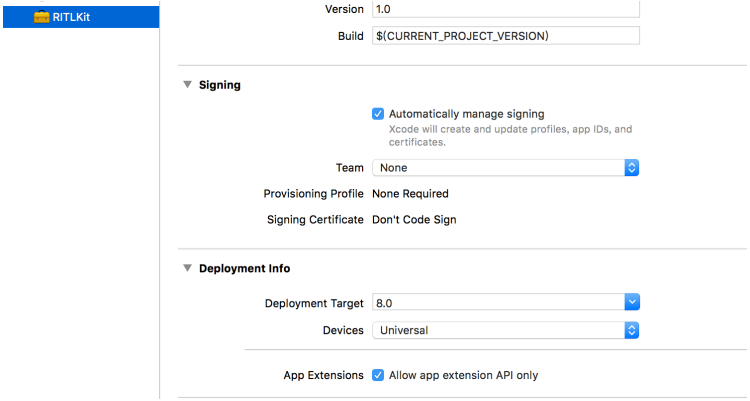

2、将需要共享的代码从源项目的编译源中删除，添加到RITLKit中




3、将创建的framework都要链接到 宿主项目 以及 拓展应用 的Linked Frameworks and Libraries中，不要忘了，都要添加，不然可能会出现找不到文件的问题




4、这里提示一下，如果上面的步骤完成，但是在拓展中还是提示找不到文件，那么还需要做一个步骤，就是将我们的framework添加到拓展应用中Allow app extension API only选中即可，将如下:



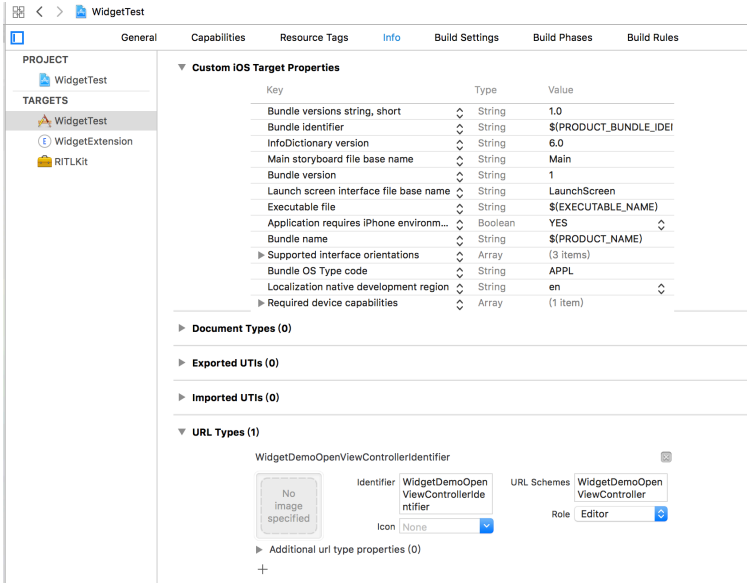

5、以上步骤完毕之后，应该就可以在拓展以及宿主应用中实现代码共用了，但还有一点:

如果是ObjC项目，导入Objc的文件，只需使用#import"XX.h"导入即可，但是如果framework中含有Swift文件，使用#import "Project-Swift.h"是导入不进去项目的，可以使用@import RILTKit; 对创建的framework编译的文件进行导入，就可以使用Swift文件了，这件事在Demo中也已经实现。

# Extension与宿主应用交互

通过点击Widget上的按钮来打开宿主应用并实现响应操作也是一种重要的交互手段，如何实现呢?

1、首先我们需要在宿主应用的Target->Info->URL Types中添加url Schemes




2、通过Widget来打开宿主应用，Demo中点击Widget中的按钮跳转至不同的界面，通过Widget打开宿主应用的操作如下:

```
/// 打开我的App
- (void)openMyApplication:(NSString *)title
{
    NSURL * url = [NSURL URLWithString:[NSString stringWithFormat:@"WidgetDemoOpenView
Controller://%@",title]];

   [self.extensionContext openURL:url completionHandler:^(BOOL success) {}];
}
```

3、宿主App通过AppDelegate中的响应openUrl的代理方法，接收信息并发出通知来响应全局:

```
///
-(BOOL)application:(UIApplication *)app openURL:(NSURL *)url options:(NSDictionary<UIApplic
ationOpenURLOptionsKey,id> *)options
{
    if ([url.scheme isEqualToString:@"WidgetDemoOpenViewController"])
    {
        NSLog(@"host = %@",url.host);

        //发送通知
        [[NSNotificationCenter defaultCenter] postNotificationName:@"ExtenicationNotification" obje
ct:url.host];
    }
    return false;
}
```

4、宿主应用中响应通知的控制器接收通知即可，比如Demo中是主页进行跳转:

```
//添加获得拓展打开基础应用的通知
[[NSNotificationCenter defaultCenter] addObserverForName:@"ExtenicationNotification" object:n
il queue:nil usingBlock:^(NSNotification * _Nonnull note) {

    //获得类型
    NSString * type = note.object;

    [weakSelf presentTextController:type];

}];
```

# NCWidgetProviding协议

如何仔细看，其实Widget的控制器与其他的控制器是没有区别的，只不过它履行了一个叫做”NCWidgetProviding”的协议。协议方法不多，在iOS10中新增了一个，废弃了一个，如下:

```
// 这个就不用多说了吧，没有很难得单词哦0.0
typedef NS_ENUM(NSUInteger, NCUpdateResult) {
    NCUpdateResultNewData,
    NCUpdateResultNoData,
    NCUpdateResultFailed
} NS_ENUM_AVAILABLE_IOS(8_0);



/* 该方法是用来告知Widget控制器是否需要更新的一个协议方法 */
- (void)widgetPerformUpdateWithCompletionHandler:(void (^)(NCUpdateResult result))completion
Handler;
```

比如Demo中为了避免重复刷新做了如下操作:

```
- (void)widgetPerformUpdateWithCompletionHandler:(void (^)(NCUpdateResult))completionHandl
er {
    // Perform any setup necessary in order to update the view.

    // If an error is encountered, use NCUpdateResultFailed
    // If there‘s no update required, use NCUpdateResultNoData
    // If there‘s an update, use NCUpdateResultNewData
    //获得数据
    NSString * newValue = [RITL_ShareDataDefaultsManager getData];

    if ([newValue isEqualToString:self.textLabel.text])//表明没有更新
    {
        completionHandler(NCUpdateResultNoData);
```

```
    }

    else//需要刷新
    {
        completionHandler(NCUpdateResultNewData);
    }
}
```

```
// iOS10 版本之后将不会再被唤起
// 用来设置Widget控制器边框间距的方法，如果出现偏差，可以调整此方法的返回值进行操作
- (UIEdgeInsets)widgetMarginInsetsForProposedMarginInsets:(UIEdgeInsets)defaultMarginInsets
NS_DEPRECATED_IOS(8_0, 10_0, "This method will not be called on widgets linked against iOS v
ersions 10.0 and later.");


// iOS10 新增的方法
// 用来设置Widget是展开还是折叠状态的方法，可以设置相关的preferredContentSizes属性修改大小
- (void)widgetActiveDisplayModeDidChange:(NCWidgetDisplayMode)activeDisplayMode withMax
imumSize:(CGSize)maxSize NS_AVAILABLE_IOS(10_0);
```

# 保存数据的时机

这个看具体的需求，比如Demo中就是选择在宿主应用将要失去Active状态的时候进行数据的保存，实现如下:

```
    //获得失去前台的监听
    [[NSNotificationCenter defaultCenter] addObserverForName:UIApplicationWillResignActiveN
otification object:nil queue:nil usingBlock:^(NSNotification * _Nonnull note) {//进行数据的保存

        //保存当前的数据
#ifdef RITL_ShareDataType_UserDefaults
        //第一种保存数据
        [RITL_ShareDataDefaultsManager saveData:weakSelf.mainTextField.text];

#else
        //第二种保存数据
        [RITL_ShareDataFileManager saveData:weakSelf.mainTextField.text];
#endif

    }];
```

# Widget无法展开折叠问题

2016-09-24补充

之前丢了一点，也有小伙伴们问，就是说按照上面的方式来开发插件，不能折叠的问题.

解决方案:

```
//在TodayViewController的ViewDidLoad里面需要设置最大展示的类型
#ifdef __IPHONE_10_0 //因为是iOS10才有的，还请记得适配
    //如果需要折叠
    self.extensionContext.widgetLargestAvailableDisplayMode = NCWidgetDisplayModeExpanded
;
#endif
```

更多的欢迎下载Github代码一起钻研~3Q

感谢一下博文对我的帮助，感谢

iOS开发之widget实现

WWDC 2014 Session笔记 - iOS 通知中心扩展制作入门

iOS开发------Widget(Today Extension)插件化开发

标签:

登录后才能评论! 登录

**友情链接**

兰亭集智 国之画 百度统计 站长统计 阿里云