# iOS 静态库详解与开发

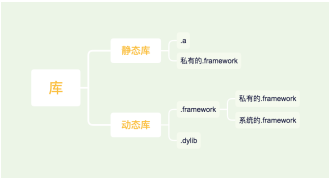
## 一：介绍

### 1. 什么是库?

库是共享程序代码的方式，一般分为静态库和动态库。

静态库：链接时完整地拷贝至可执行文件中，被多次使用就有多份冗余拷贝。

动态库：链接时不复制，程序运行时由系统动态加载到内存，供程序调用，系统只加载一次，多个程序共用，节省内存。




### 2. 两种形式中.framework的区别

如上图所示，静态库的形式包含.a和.framework两种形式。

动态库的形式包含.dylib和.framework，其中.framework包括私有的.framework和系统的.framework。

静态库和动态库都有私有的.framework，但是根本上有所不同，动态库私有的.framework，上架会有机审规则，并且不能动态下载。静态库私有的.framework，类似一个封装的代码块，没有过多的限制。

### 3. 静态库中.a与.framework的区别

.a是一个纯二进制文件，.framework中除了有二进制文件之外还有资源文件。 .a文件不能直接使用，至少还有.h文件配合，.framework文件可以直接使用，因为本身包含了h文件和其他文件

### 4. 静态库的优点

- 实现程序的模块化，将固定的业务模块化成静态库。
- 方便共享代码，即可以和别人分享你的代码库，但别人又看不到你代码的实现。
- 开发第三方sdk的需要，例如两个公司之间业务交流，不可能把源代码都发送给另一个公司，这时候将私密内容打包成静态库，别人只能调用接口，而不能知道其中实现的细节。

公司项目需要开发出一套同时支持微信支付、支付宝支付、银联支付的sdk，既要满足本公司项目需求，还需要提供给友方公司使用。

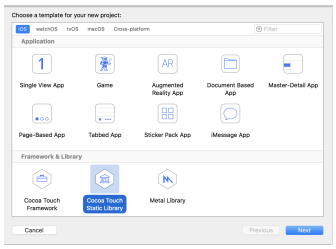
同时集成三家第三方的支付sdk，其中有大量的配置冲突需要解决。这篇文章通过demo和大家介绍一下如何开发自己的静态库.a文件，分享给大家，同时对工作进行总结。

# 二：静态库实现

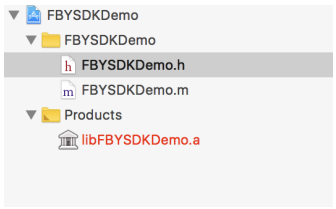
静态库生成demo 静态库使用demo

### 1. 新建一个静态库工程

打开Xcode，点击File\New\Project，选择iOS\Framework & Library\Cocoa Touch Static Library新建一个静态库工程。




将工程命名为FBYSDKDemo，然后将工程保存到一个空目录下。 静态库工程由头文件FBYSDKDemo.h和实现文件FBYSDKDemo.m组成，这些文件将被编译为库本身，如下图：




在开发中，为了让开发的静态库使用起来更方便，只需要让使用者导入一个头文件，便可以访问你所提供的接口，并且通过接口进行数据回调。

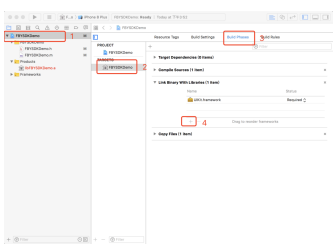
### 2. 导入头文件

导入UIKit的头文件，这是创建一个库所需要的。当你在创建不同的组成类时，你将会为它们添加到这个文件中，确保它们能够被库的使用者获取到。

打开FBYSDKDemo.h，引入头文件

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

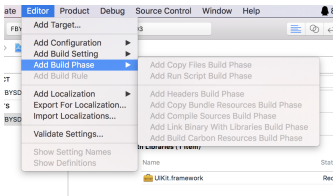
点击Build Phases，展开Link Binary with Libraries这一部分，点击+添加一个新的framework，找到UIKit.framework，点击add添加进来。



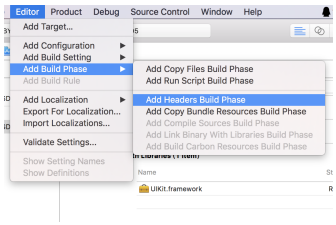
如果不结合头文件，静态库是没有用的，静态库编译一组文件，在这些文件中类和方法都以二进制数据的形式存在。在静态库中类分为两类，一类是公开的public，一类是私有的只能内部访问使用。

接下来，需要在build栏中添加新的phase，来包含所有头文件。 在Xcode的Build Phases界面，选择Editor\Add Build Phase\Add Headers Build Phase。

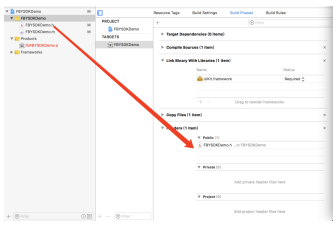
注意：如果发现按上面找到的菜单项是不可点击的，如下图：




点击下方Build Phases界面的白色区域来获取Xcode的应用焦点，然后重新试一下




把FBYSDKDemo.h从项目中拖到Copy Headers下的Public部分。 这里是要保证用户可以使用库中公开的类或者接口。



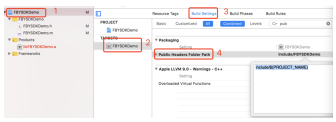
Copy Headers下的Public部分中所添加的类均是对外公开的。 这里有三个分组，Public公开的，Private下的头文件是可以被用户看到的，Project下的文件是私有的，这里建议尽量将文件放在Public和Project下。

### 3. 添加配置

添加配置主要是在Build Settings下操作，点击项目名，然后选择FBYSDKDemo静态库目标，选择Build Setting栏，然后搜索public header，双击Public Headers Folder Path，在弹出视图中键入如下内容

```
include/$(PROJECT_NAME)
```

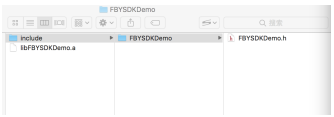
截图如下：



因为你创建好的静态文件供他人使用，最好禁掉无效代码和debug符号，让用户自己选择对自己的项目有利的部分使用。 在搜索框中分别搜索Dead Code Stripping、Strip Debug Symbol During Copy、Strip Style配置如下：

- Dead Code Stripping设置为NO
- Strip Debug Symbol During Copy 全部设置为NO
- Strip Style设置为Non-Global Symbols

到目前为止，项目的构建已经完成，选择目标为Generic iOS Device，按下command + B进行编译，工程导航栏中Product目录下libRWUIControls.a文件将从红色变为黑色，表明现在该文件已经存在了。右键单击libRWUIControls.a，选择Show in Finder，如下图所示：



上图就可以看到对外公开的FBYSDKDemo.h类，其他实现类均以二进制的形式在libFBYSDKDemo.a中。

**4. 功能实现**

这里以实现静态库的开发为主，功能部分实现一个简单的功能demo来举例。

在头文件FBYSDKDemo.h中实现如下代码：

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

typedef void (^FBYSDKCompletion)(NSString *result);

@interface FBYSDKDemo : NSObject

/**
 *
 *  @param urltype       网页类型信息 urltype可为iOS、Android
 *  @param completion    根据urltype获取到相应网页url结果回调
 *
 */
- (void)urlType:(NSString *)urltype withCompletion:(FBYSDKCompletion)completion;

@end
```

其中urltype为网页类型信息 urltype可为iOS、Android，根据urltype获取到相应网页url结果回调completion。

在实现文件FBYSDKDemo.m中代码如下：

```
#import "FBYSDKDemo.h"

@interface FBYSDKDemo ()

@end
@implementation FBYSDKDemo

- (void)urlType:(NSString *)urltype withCompletion:(FBYSDKCompletion)completion{

    if ([urltype isEqualToString:[NSString stringWithFormat:@"iOS"]]) {

        if (completion) {
            completion(@"https://juejin.im/post/5a41c04c6fb9a044fc44fd23");
```

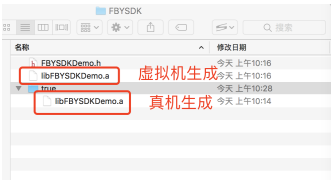
```
        }

    }else if ([urltype isEqualToString:[NSString stringWithFormat:@"Android"]]) {

        if (completion) {
            completion(@"https://juejin.im/post/5a31e6adf265da430c11d41f");
        }

    }

}
@end
```

选择目标为Generic iOS Device，按下command + B进行编译。

**5. 合并静态库**

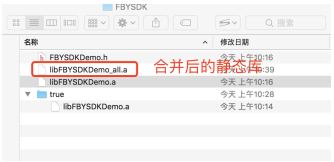
选择目标为Generic iOS Device，编译运行后，右键单击libRWUIControls.a，选择Show in Finder显示的libFBYSDKDemo.a可在真机，如果在虚拟机中运行会报错。

所以还要选择目标为虚拟机(例如iPhone 7)，然后编译运行,右键单击libRWUIControls.a，选择Show in Finder显示的libFBYSDKDemo.a可在虚拟机中运行，如果在真机中运行会报错。




合并方法：打开终端，输入 lipo -create 真机库.a的路径 模拟器库.a的路径 -output 合成库的名字(可以复制模库.a的路径，修改名字).a；回车就可以在模拟库的文件夹中看到新合成的.a文件了，如下图：


合成后静态库文件截图如下：

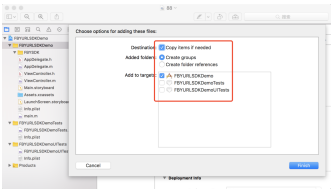



libFBYSDKDemo_all.a文件即为静态库文件，一个通用的静态库.a就做好了。

## 三：静态库使用

**1. 导入静态库**

导入静态库和.h头文件，注意下图选项：



### 2. 调用函数

源码如下：

```
- (void)blogsBtn:(UIButton *)sender {

    if (sender.tag == 6000) {

        [FBYSDKDemo urlType:@"iOS" withCompletion:^(NSString *result) {

            [self contentURL:result];

        }];
    }else if (sender.tag == 6001) {

        [FBYSDKDemo urlType:@"Android" withCompletion:^(NSString *result) {

            [self contentURL:result];
        }];
    }
}
```

result就是通过sdk回调获取到的结果。

### 3. DEMO截图如下


### 4. 本篇文章demo源码：

静态库生成demo

静态库使用demo

希望可以帮助大家，如有问题可加QQ技术交流群: 668562416，如果哪里有什么不对或者不足的地方，还望读者多多提意见或建议

如需转载请联系我，经过授权方可转载，谢谢

本篇已同步到个人博客：FBY展菲

---

欢迎关注我的公众号：网罗开发