|  |
| --- |
|  |
| **IOS插件扩展** |

# 开发环境的安装

使用mac mini或者imac一体机、mac 笔记本安装有mac系统的机器都可以进行开发。开发前需要在App Store下载安装Xcode。Xcode是iOS的开发工具，网上有很多有关Xcode的使用教程。

# 扩展插件的开发

## 开发流程

开发流程分三个阶段：

1.调研插件功能；

2.将源代码集成到AppCan基础开发工程；

3.生成插件包。

在这里，我们用一个样例，来讲述插件的开发流程。

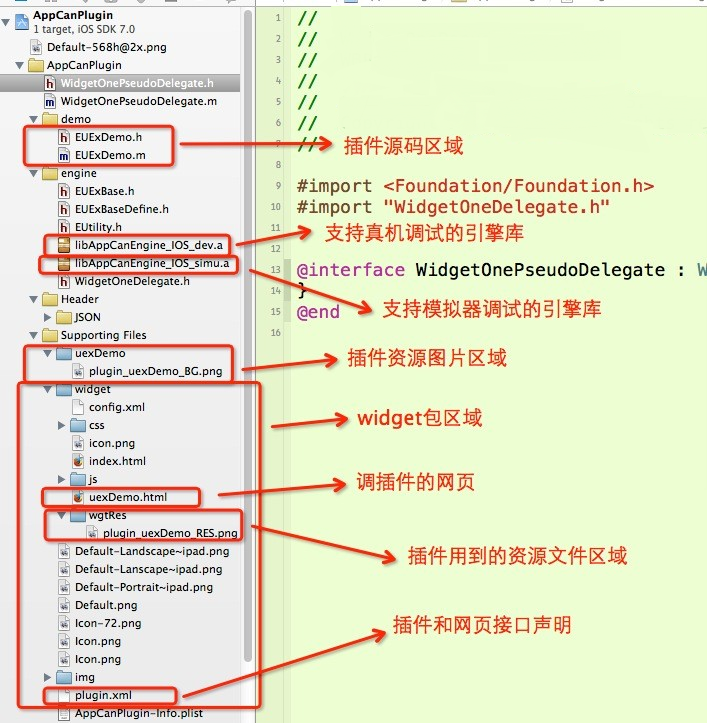
### 调研插件功能

创建一个普通的原生工程，命名Demo（命名一般和插件实现的功能命名一致），开发好要实现的功能，比如实现一个自定义样式的底部弹出框UIActionSheet， 如下图所示。



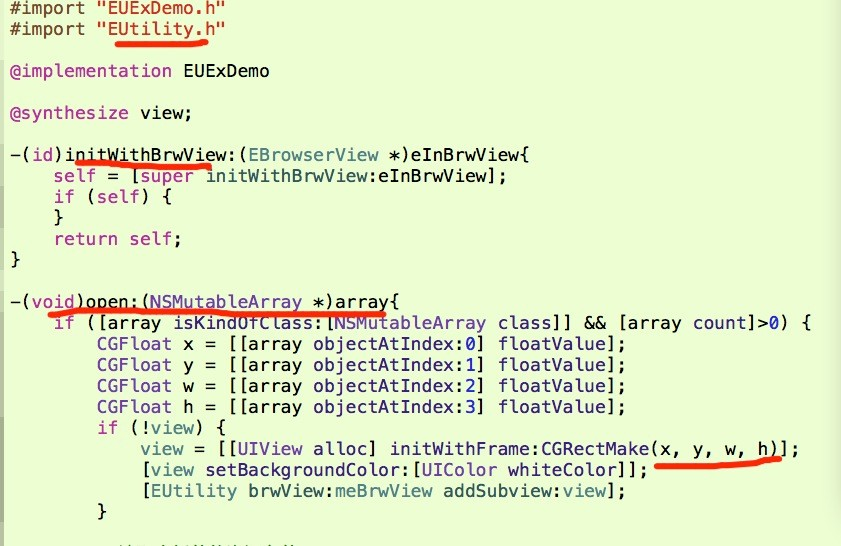
### 将源代码集成到AppCan基础开发工程

用Xcode打开AppCan基础开发工程，根据红色标注，开始插件集成。



1）封装插件功能。右键demo文件夹，创建一个以EUExBase为基类的类EUExDemo(为什么以EUEx命名，请看下边的命名规范)，这个EUExDemo是网页和插件的入口类。

在EUExDemo.m中，开始创建接口函数，比如函数open，传参统一用NSMutableArray实例按顺序获取。例如在工程中添加open,close,share方法(EUExDemo类似于原生开发的AppDelegate，可以直接创建UIView或UIViewController的类，也可以直接初始化视图View)，如下图：



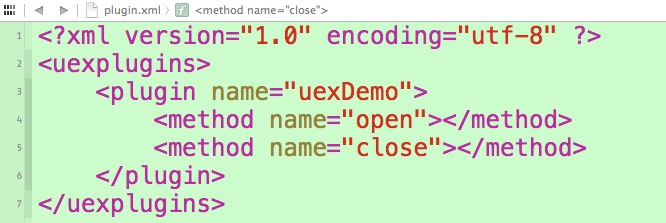
接着在函数中将用到的参数取出来，然后初始化功能的界面、传参，将工程Demo中的功能集成进来，插件就封装好了。

2）在widget文件夹下创建测试.html格式文件，文件名uexDemo.html(uexDemo请看下边命名规范)，在该网页里，有js的部分，也有网页的部分，编辑接口函数open和参数，参数根据需要传递。

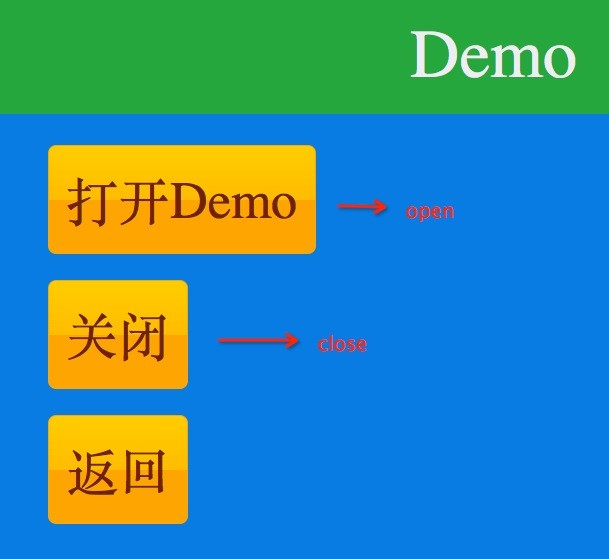


如果单测试一个插件，一般在index.html网页中编辑调用uexDemo.html。

3）在plugin.xml中声明插件和网页的接口，这里是open，如图



然后开始运行整个工程，从网页中打开插件，看功能是否实现，没问题就可以打包。



备注：如果插件中用到一些文件，比如阅读器插件，需要本地的图书资源文件，那就需要将图书文件放在文件夹wgtRes下，在open参数中，传参“res://图书名.pdf”（参考下面如何读取资源图片）。

### 生成插件包

插件是为了实现跨平台统一性和实现一些网页中不能实现的功能，插件最后的展现形式是.zip压缩包，插件又分为两种：一种是用于AppCan SDK移动应用开发系统上，另一种是用于AppCan 3.0 IDE上。

1）如何生成用于AppCan SDK移动应用开发系统的插件包，以及所需文件，请参考下边“如何生成AppCan SDK移动应用开发系统插件包”。

2）如何生成用于AppCan 3.0 IDE的插件包，以及所需文件，请参考下边“如何生成AppCan 3.0 IDE插件包”。

## 命名规范

在AppCan中有统一的命名规范，若我们的类或者方法名称不规范将导致该类或方法无效。

### 类的命名

类的命名前缀为EUEx，即类的命名规范为EUEx+对象英文名称，例如EUExDemo，又例如我们已经封装的下载管理插件EUExDownloaderMgr，音频插件EUExAudio等。值得注意的是，这里的对象英文名称要与plugin.xml中的plugin name的对象英文名称保持一致，否则会调用失败。

### 方法的命名

方法名符合驼峰命名法，例如下载接插件中的创建下载对象“createDownloader”，和下载“download”。值得注意的是，这里的方法名要与plugin.xml中的相应类下的method name保持一致，否则会调用失败。

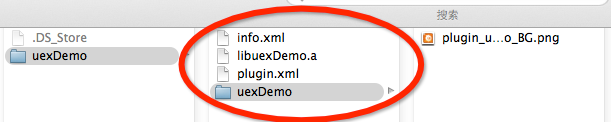
### 库文件的命名

扩展插件最终是以库文件“.a”的形式被引用到AppCan工程中的，也要符合我们的命名规范，即lib+插件名称。其中插件名称要和plugin.xml中的plugin name一致。例如libuexDemo.a，和下载管理插件libuexDownloaderMgr.a等。

## 插件组成

插件是以.zip文件的形式存在的，解压之后如下：

### AppCan SDK移动应用开发系统插件包组成：

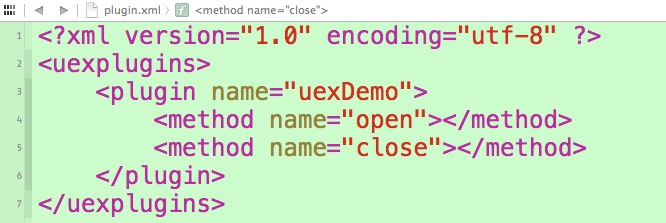


uexDemo插件包含的文件有info.xml（AppCan 3.0新添加文件）、plugin.xml、libuexDemo.a、uexDemo（如果插件使用资源文件就需要有个文件夹，没有就不需要）。

#### plugin.xml文件

plugin.xml是配置js调用方法的xml文件，如果需要开发native plugin，必须加载。

下面以下载管理插件为例：



**plugin name**：为封装的js对象的名称，以uex前缀开头，即uex+对象英文名称。

**method name**: 为对象方法名称，依旧符合驼峰命名法，回掉网页的函数不需要写入到此文件中，但是要写如到插件的API文档中。

在AppCan 3.0中此文件是必须要有的，请开发人员注意。

#### info.xml文件

此文件是AppCan 3.0引入的文件文件内容如下：



uexName：代表插件名字。

version：代表当前当前插件的版本号，以后通过info.xml文件也可以知道当前插件的版本信息。

build：代表当前插件的小版本，内部使用。

info键值：代表当前插件版本修改信息，以后通过info.xml文件也可以知道当前插件此版本修改信息。

在AppCan 3.0中此文件是必须要有的，请开发人员注意。

#### .a静态库文件

如何生成.a文件，请参考下面“如何生成AppCan SDK移动应用开发系统插件包”。

在AppCan 3.0中此文件是必须要有的，请开发人员注意。

#### uexDemo插件资源文件夹

如果插件使用到资源文件就要统一放在此文件夹下，推荐资源文件命名规则：plugin\_插件名称\_资源图片名称;例如：plugin\_uexDemo\_BG；若没有资源文件，可以没有此文件夹。

### AppCan 3.0 IDE插件包组成：



同AppCan SDK移动应用开发系统插件中插件的组成，AppCan 3.0 IDE中没有.a文件，有.dylib文件。

#### plugin.xml文件

同AppCan SDK移动应用开发系统插件中的plugin.xml。

#### info.xml文件

同AppCan SDK移动应用开发系统插件中的info.xml。

#### .dylib动态库文件

如何生成.dylib文件，请参考下面“如何生成AppCan 3.0 IDE 插件包”。

在AppCan 3.0中此文件是必须要有的，请开发人员注意。

#### uexDemo插件资源文件夹

同AppCan SDK移动应用开发系统插件的uexDemo插件资源文件。

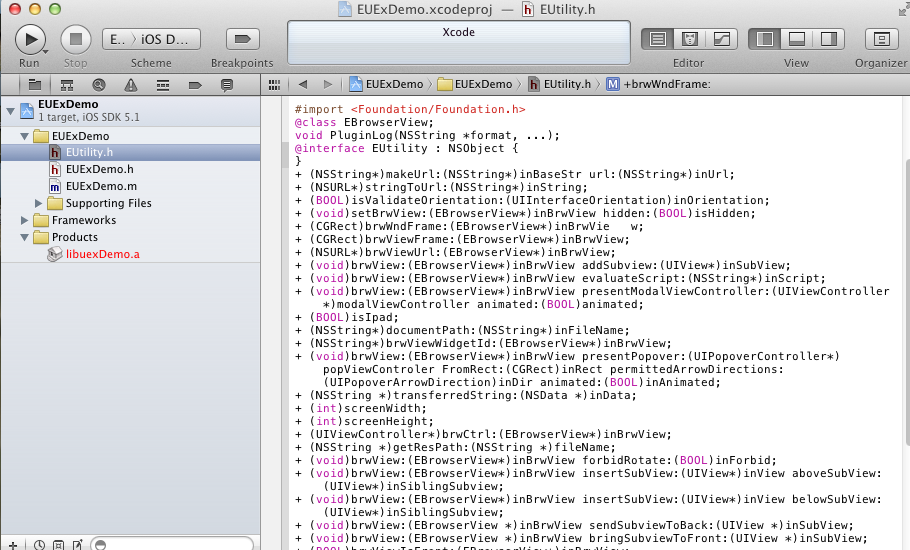
## 配置插件调试工程中使用到的文件

为方便开发者开发，我们封装了一些经常用到的类，例如工具类EUtility、js调用插件的基础类EUExBase.h、提示错误类EUExBaseDefine.h，以及使用JSON时的类。这样我们可以直接引用这些类。

### AppCan相关文件

#### 工具类 EUtility.h

在插件开发过程中，会用到很多关于工具的方法，例如打印log，便于控制是否显示log；把string 转成url,特别是对于本地url处理等等。EUtility.h类就包含了这些方法，开发者不用重复开发代码，直接引用即可。添加文件如图：



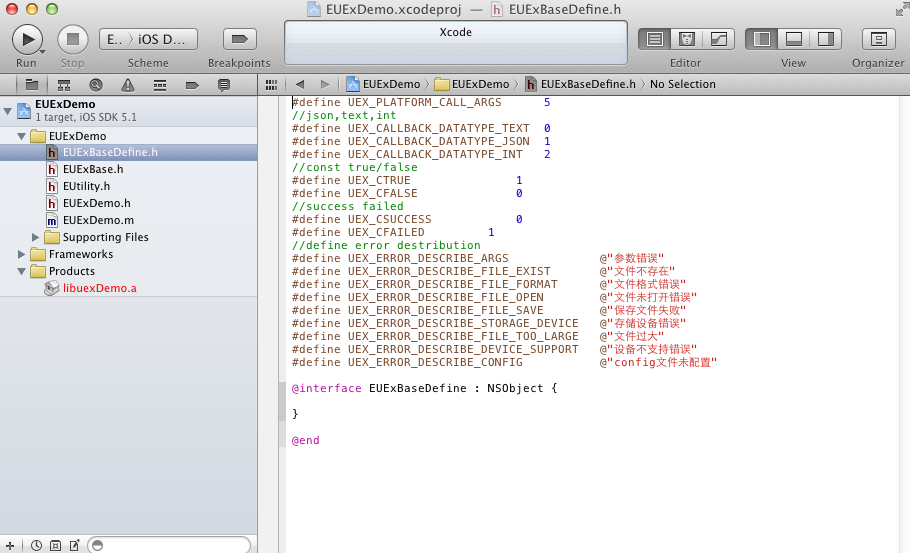
#### 用于js调用基础类EUExBase.h

该类为封装用JS调用扩展native 插件的父类，包括初始化js调用native 层的对象，uiwebview通过js传到网页中的接口等等，此类需要继承。添加文件如图：



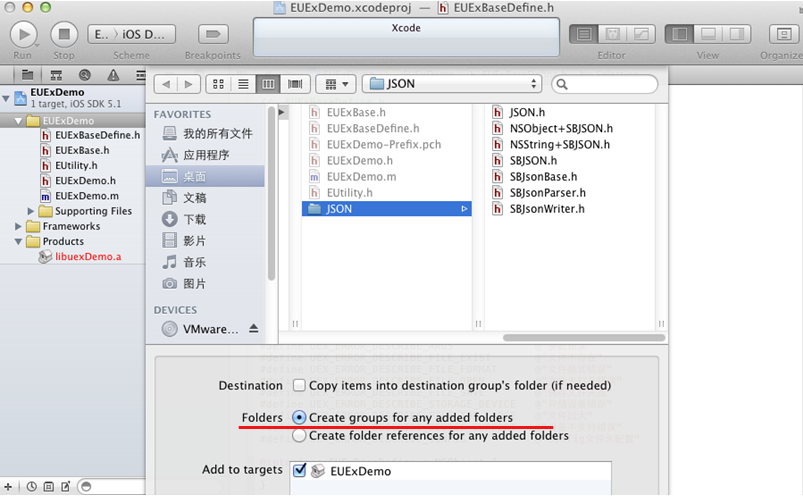
#### 提示错误类EUExBaseDefine.h

在应用的使用过程中，会出现用户操作错误或设备不支持等情况，EUExBaseDefine.h就包含这些特殊情况，方便我们调用以通知上层做相应处理。添加头文件如图：

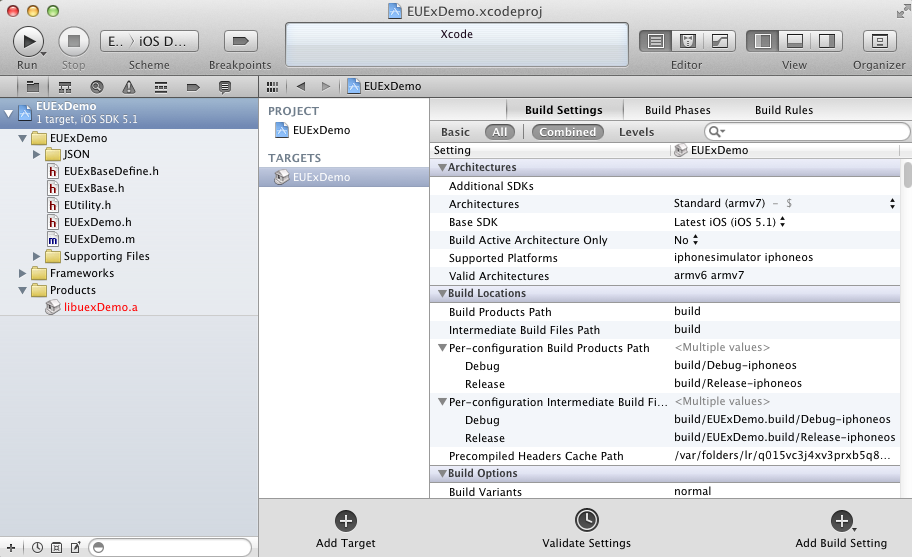


### JSON相关的类

有些插件会用到JSON格式返回数据，这时就要用到有关JSON的类。为便于理解可将这些文件放于同一文件夹中，值得注意的是我们在添加时不能使用实体文件夹，需要将文件置于根目录下，如图：



添加成功后，工程的目录结构变为：



## 代码详细介绍

### 编写代码

编写代码时，其中InitWithBrwView用于初始化EUEx-对象；clean, 是平台封装的调用方法，把一些与当前网页有关的内存等等在切换网页的时候释放掉（请参考demo工程）。工程中其余部分按照插件的开发需求编写，注意命名规范。

### 如何把视图添加到网页视图上

需要包含#import "EUtility.h"头文件

UIView \*view = [[UIViewalloc] initWithFrame:CGRectMake(0, 0, 320, 416)];

[viewsetBackgroundColor:[UIColorbrownColor]];

[EUtilitybrwView:meBrwViewaddSubview:view];//通过这个方法添加到网页视图上

[viewrelease];

### 如何回掉网页

在.html文件中写入js代码，如下：

uexDemo.CallBack = CallBack;//注册插件的回掉函数

uexDemo.open();//调用插件的open方法

在插件代码中如下：

[self.meBrwViewstringByEvaluatingJavaScriptFromString:@"uexDemo.CallBack();"];

这样就可以回掉网页中的代码，进行网页和native的数据交互。

### 如何读取资源图片

图片文件放在当前插件包下面文件夹名字和插件名字一致

UIImage \*image = [UIImageimageWithContentsOfFile:[[NSBundlemainBundle] pathForResource:@"uexDmeo/plugin\_uexDemo\_BG"ofType:@"png"]];

}

如何读取res：//下面的图片

NSString \*imagePath = @"res://plugin\_uexDemo\_RES.png";

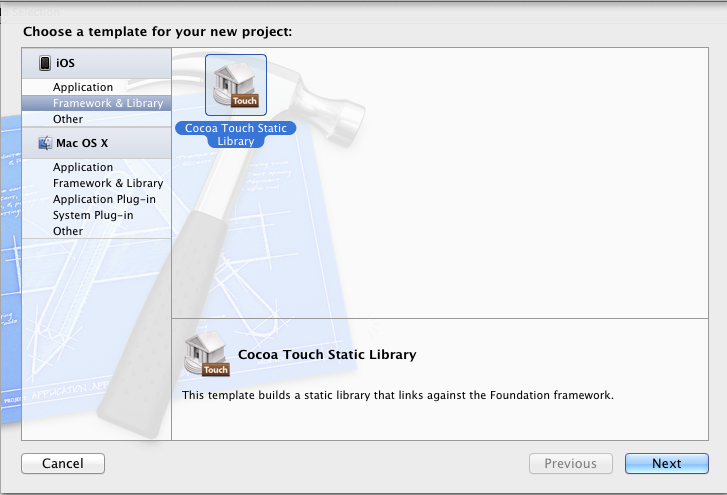
imagePath = [EUtilitygetAbsPath:self.eUExNewsList.meBrwViewpath:imagePath];（引擎方法）

UIImage \*image = [UIImageimageWithContentsOfFile:imagePath];

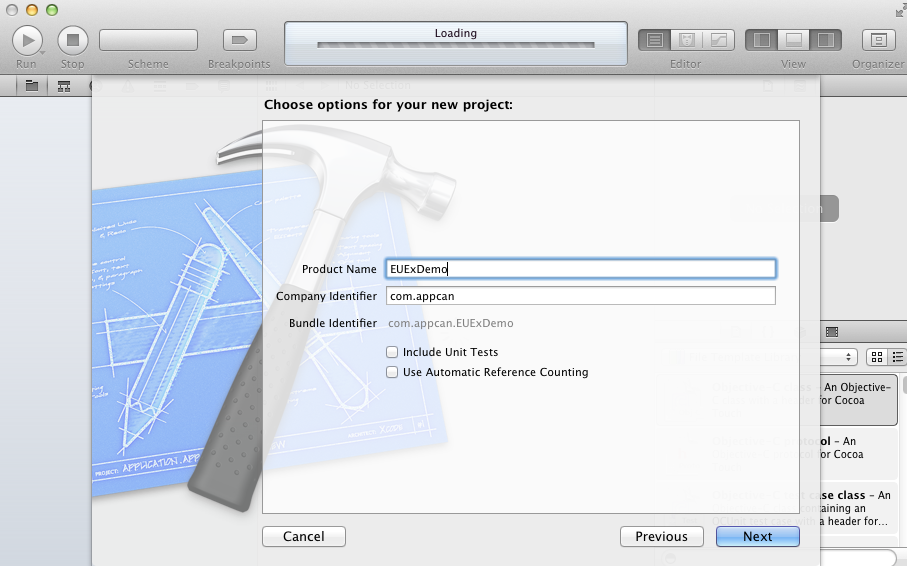
## 如何生成AppCan SDK移动应用开发系统插件包

### 生成静态库工程的步骤

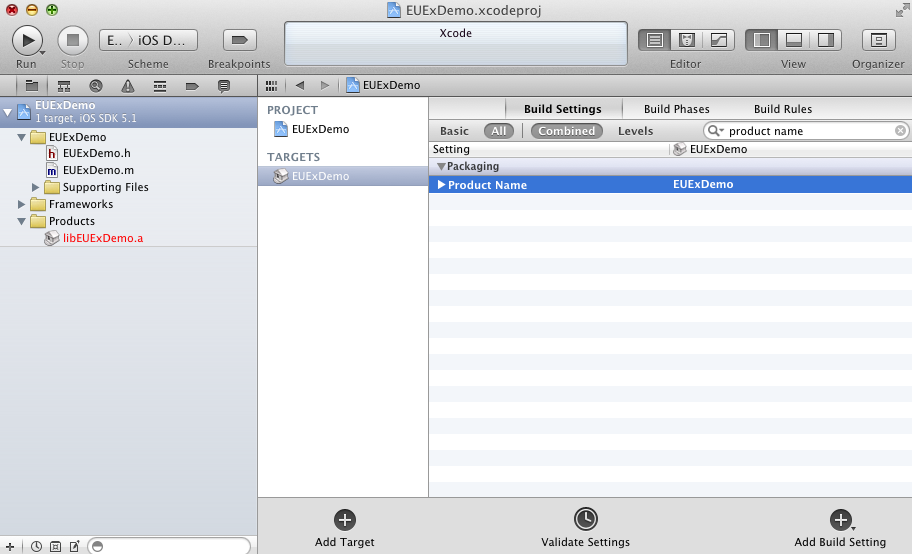
打开xcode创建一个静态库工程，如图：



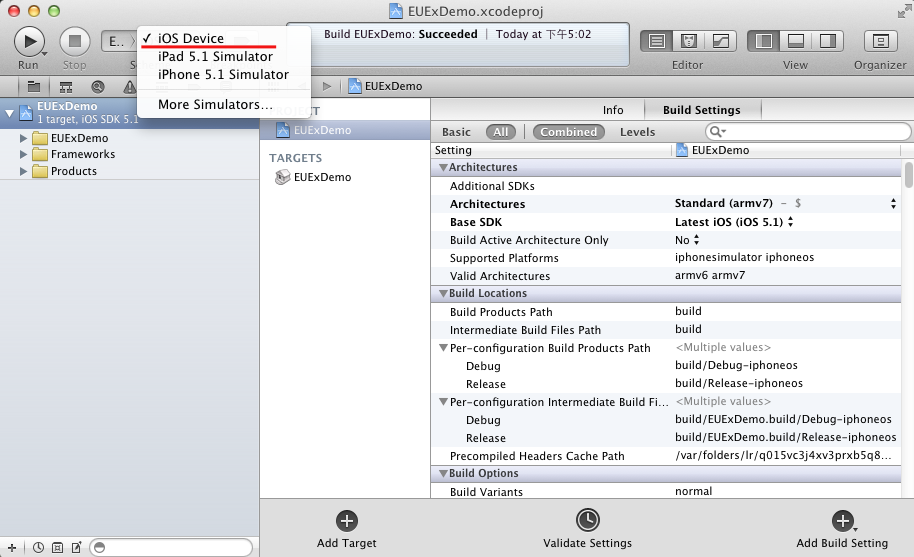
可将工程命名为EUExDemo，填写相关信息，如图：



创建成功，如图：

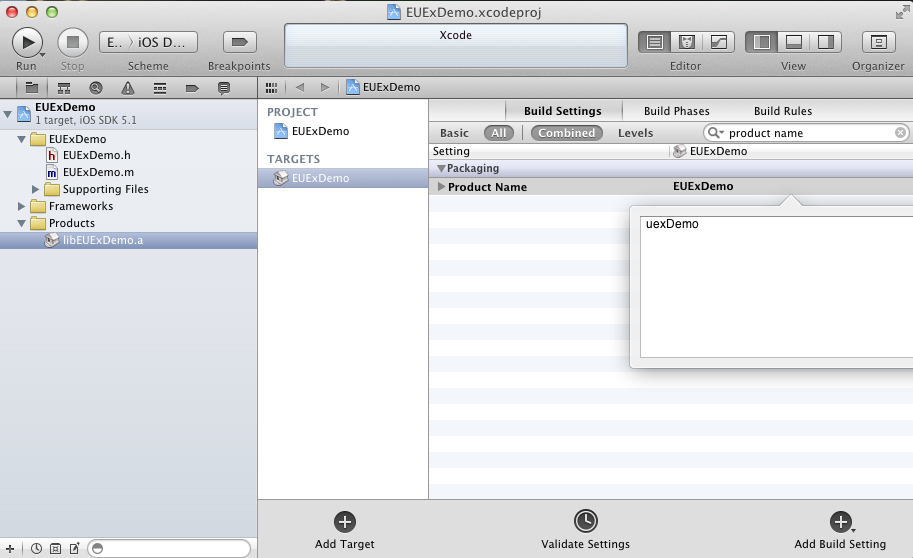


我们可以选择使用模拟器或者真机编译，值得注意的是若选择模拟器方式生成静态库，则只能被模拟器的工程引用，反之亦然。这里我们选择真机。如图：

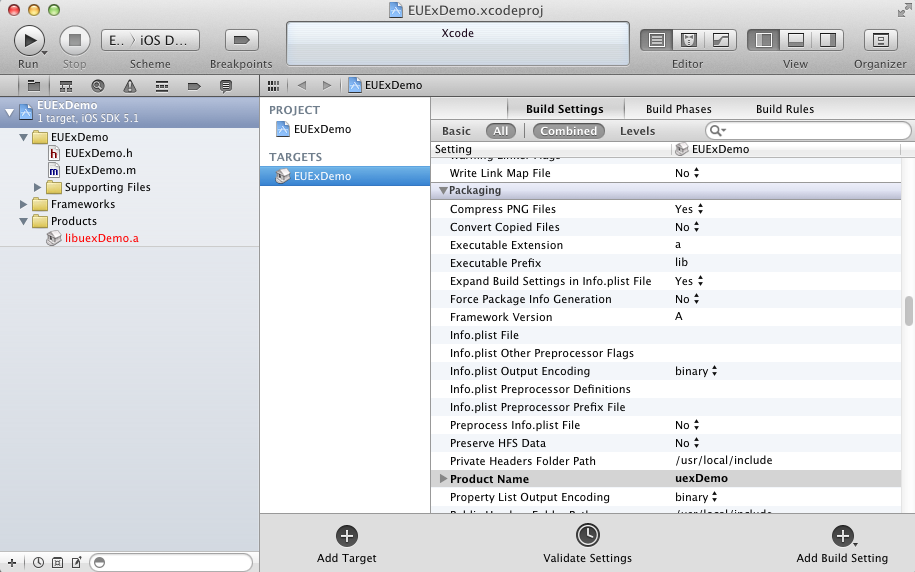


创建工程后，默认的生成文件名称为libEUExDemo.a，这不符合我们的命名规则，应该将其改成libuexDemo.a。

方法是在工程配置中找到“Product Name”，双击内容，将product name的值修改为uexDemo，如图：

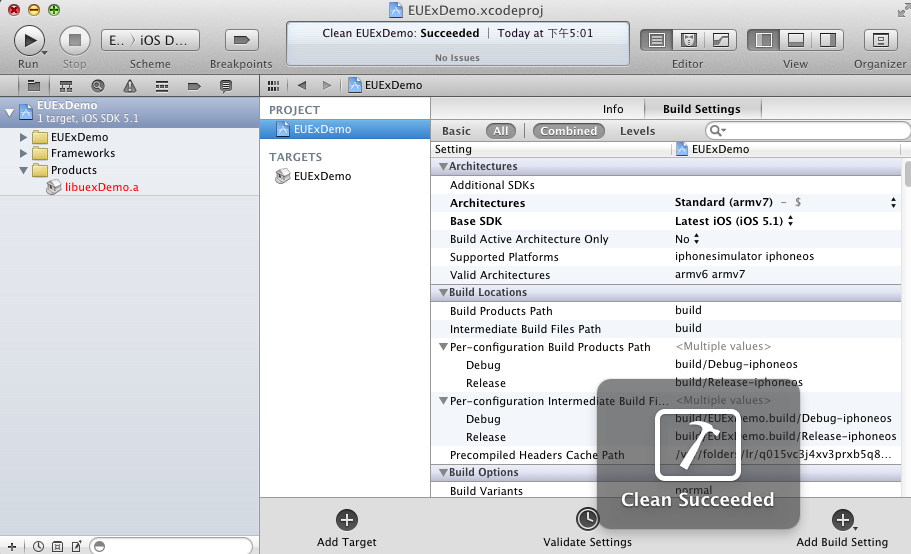


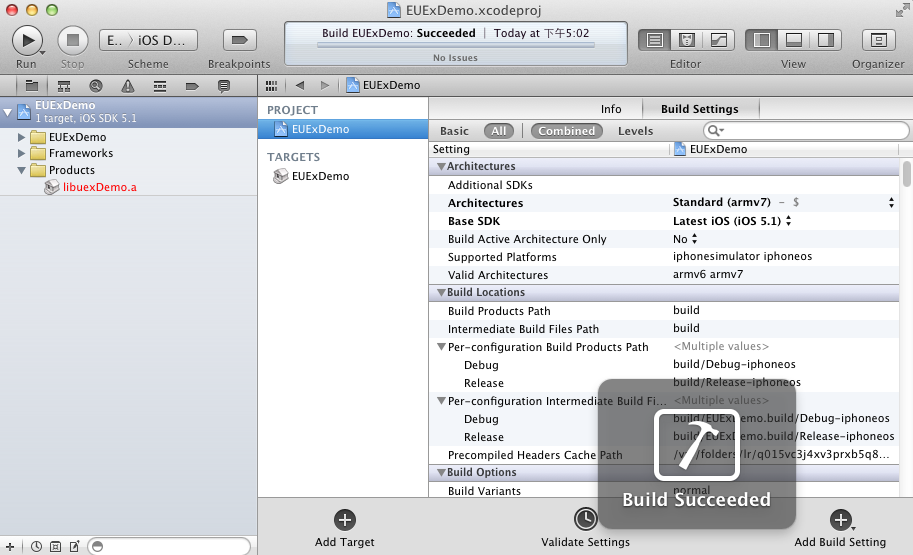
这样生成文件库会自动更名为libuexDemo.a，如图：



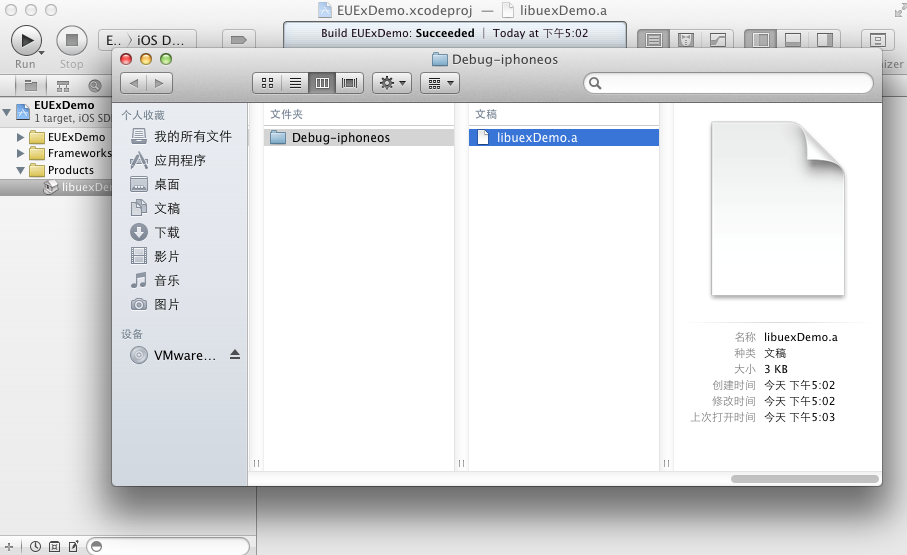
### 生成静态库

代码编写完成后，可进行clean，build，如图：





编译成功后，在相应目录下生成插件库文件，如图：



### 生成插件包

然后将.a文件放到以该插件命名的文件夹下和info.xml、plugin.xml、uexDemo，压缩打包完成。

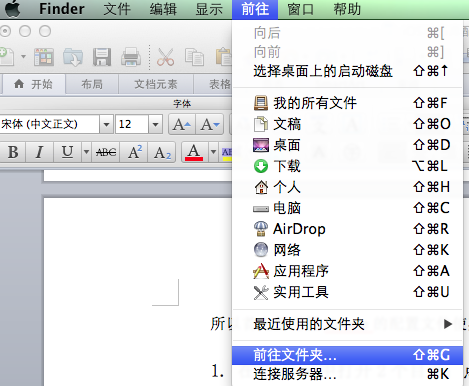


## 如何生成AppCan 3.0 IDE插件包

### 生成动态库的所需配置

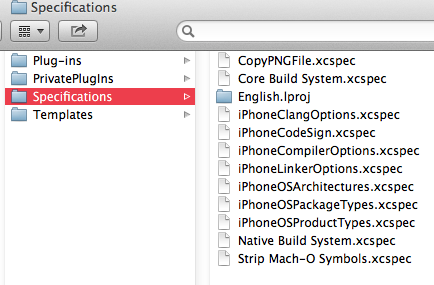
Xcode提供了在iOS工程中创建静态库的功能，在MAC上创建动态库和静态库的功能。但是没有提供在iOS工程中创建动态库的功能；所以首先要配置Xcode的配置文件使其有能够创建 iOS动态库的功能：

1. 在Finder中打开下面的目录：点击Finder 在上面的菜单栏中打开



然后依次找到如下路径的文件夹目录

/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/Library/Xcode/Specifications/



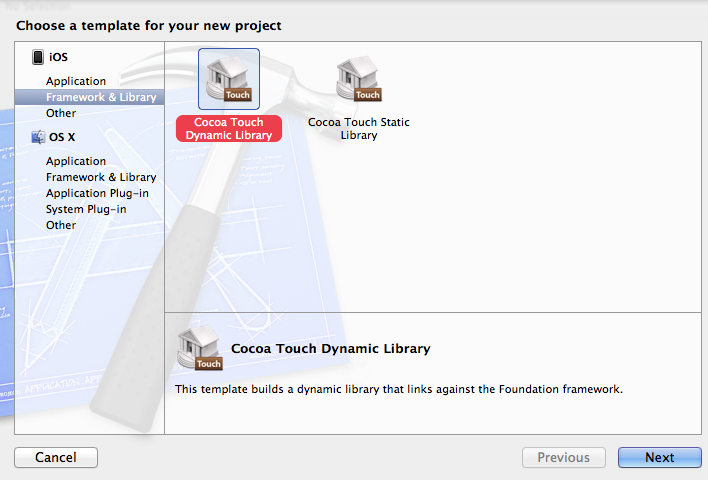
把我们提供的iPhoneOSProductTypes.xcspec 和 iPhoneOSPackageTypes.xcspec 文件直接拷贝到这个文件加下，直接进行替换。

在Finder中打开下面目录的文件夹

/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/Library/Xcode/Templates/Project Templates/Framework & Library/

，把我们提供的Cocoa Touch Dynamic Library.xctemplate文件夹直接拷贝到这个目录下。

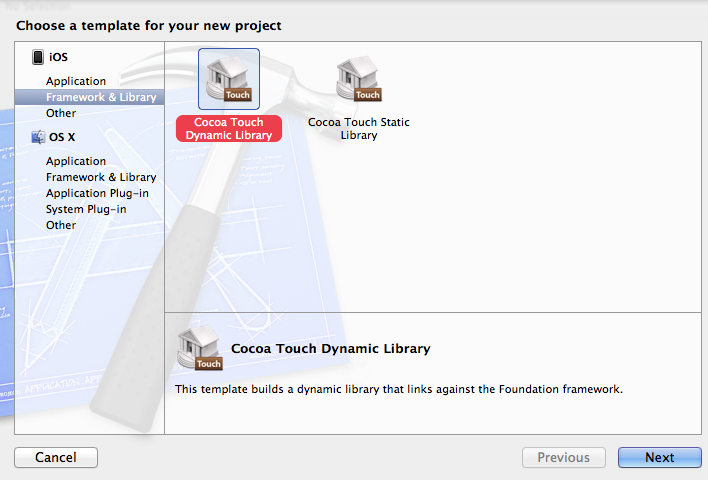
3、然后退出Xcode重启Xcode即可看到能创建iOS动态库了，如图所示：



然后就可以接着创建动态库了，创建动态库的方法和静态库的方法一样：

### 生成动态库工程的步骤

打开Xcode 打开iOS 下的FrameWork&Library 选择Cocoa Touch Dynamic Library，如图：

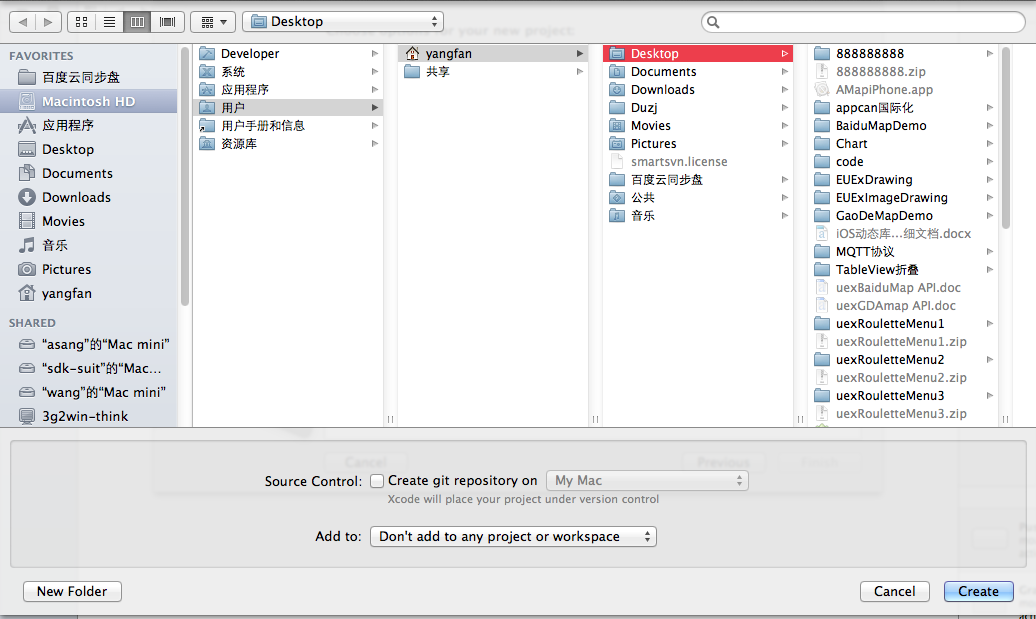


点击next 进入



填写工程名字，点击Next;这里说明一下动态库的命名规则，Product

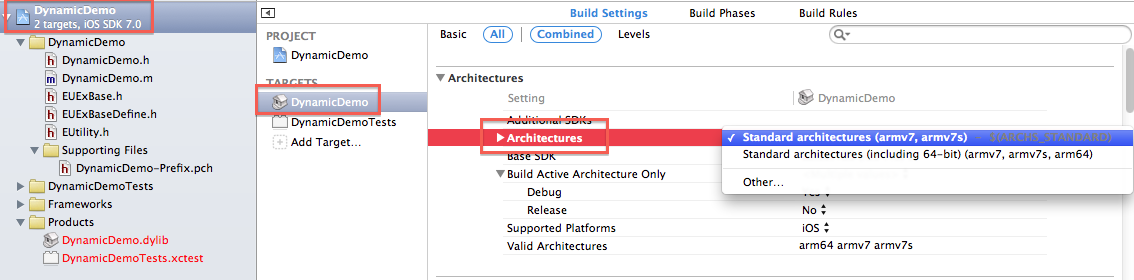
Name要和插件的名称保持一致，一般不需单独修改动态库的名字，默认动态库的名字是和工程名字是一样的。



找一个位置存放工程，然后点击Create，进入到工程里面：

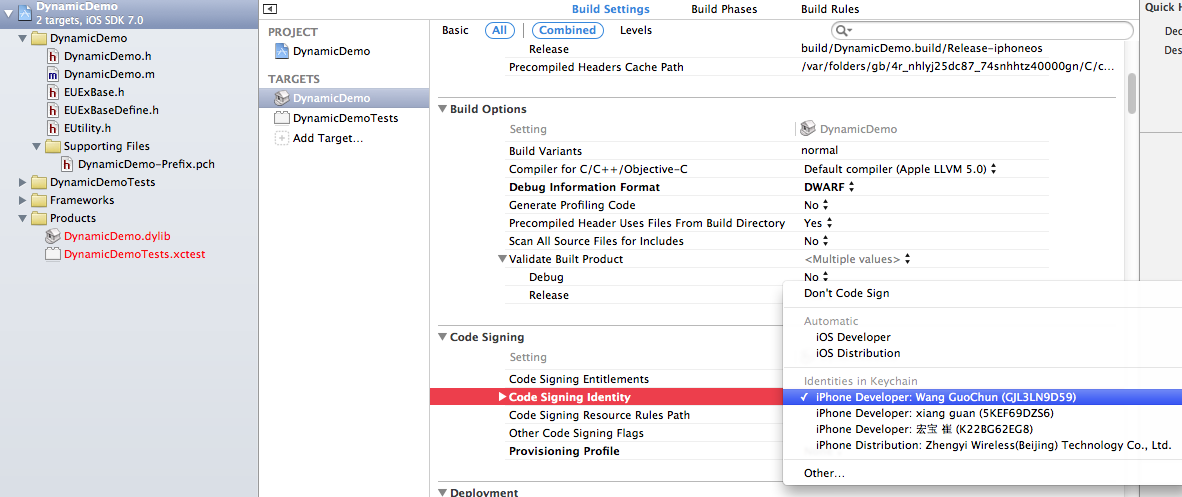
然后把静态库使用的所有文件引入到工程中，接下来配置工程文件：

点击工程TARGETS里面的Build Setting 下的Architectures，选择自己需要的ARM

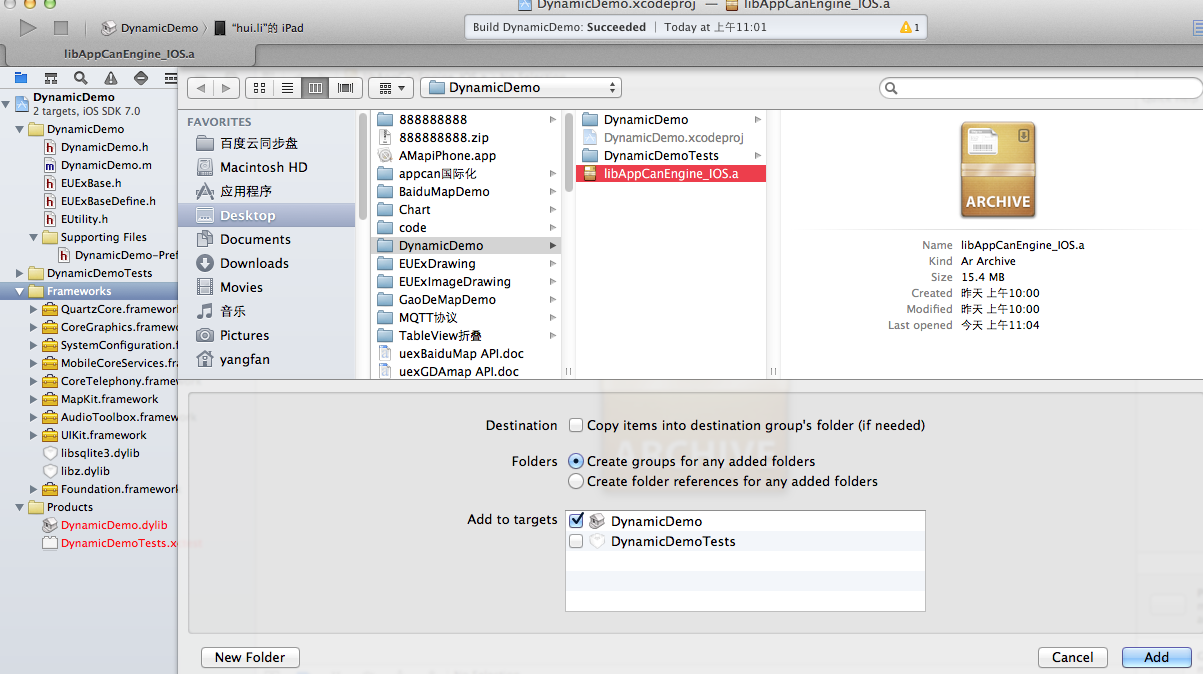


然后选AppCan平台提供的证书，如下图

说明，生成动态库一定要使用证书，且和最后使用这个动态库的工程的证书要一样。

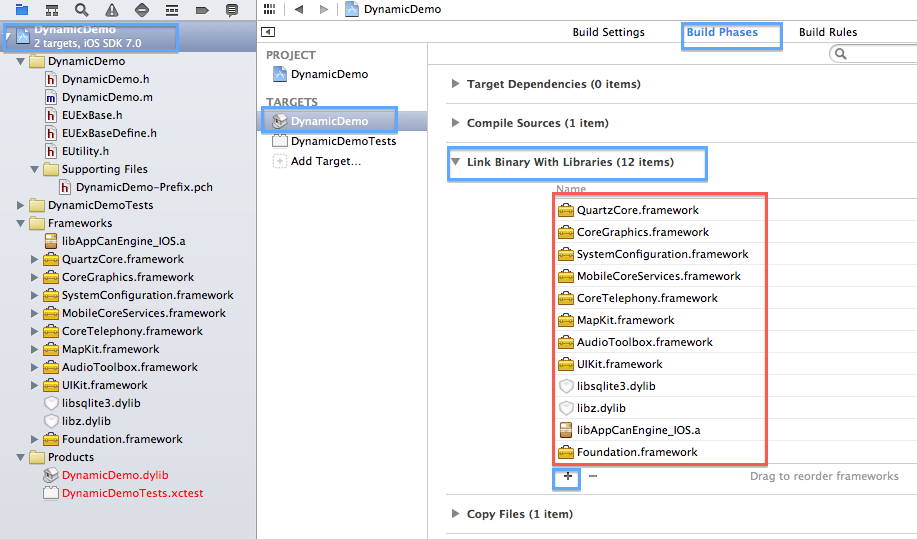


点击工程下的FrameWork 文件夹右键 选择 add File to XXXX,找到引擎的静态库文件选择 ，并点击add 添加进来。如下图



接着要添加一些支持引擎和自己代码的系统FrameWork库，如下图点击“+”号，添加支撑库，如下图，在Demo中需要添加的动态库如下图红色标注的

说明：当用户自己开发动态库时，加入引擎文件后，运行时报错，一般是少添加支撑库，可以找到错误提示，若还是不知道添加什么库可以上网把错误信息输入 ，自己查找添加响应的库。



### 生成动态库

配置完成后，点击运行，成功后，在Products 文件夹下 看到自己的动态库，点击右键，show in finder（参照如何生成静态库），在文件中找到自己编译完成的动态库：

### 生成插件包

然后将.dylib文件放到以该插件命名的文件夹下和info.xml、plugin.xml、uexDemo，压缩打包完成。



# 验证插件

在线打包，或者是本地打包，把生成的插件，压缩成.zip格式的文件，然后和API文档一起上传到打包服务器上，配合网页代码进行打包测试。