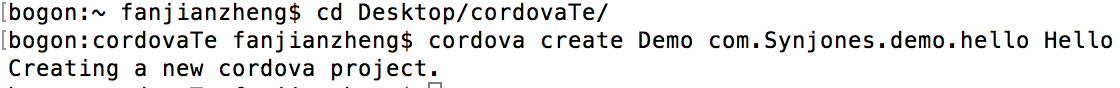
# Cordova使用

1. 创建你的第一个Cordova工程

1.新建项目

打开终端，cd切换目录到工作目录下，输入以下命令，同样，可能需要点时间来完成

cordova create Demo com.Synjones.demo.hello Hello 

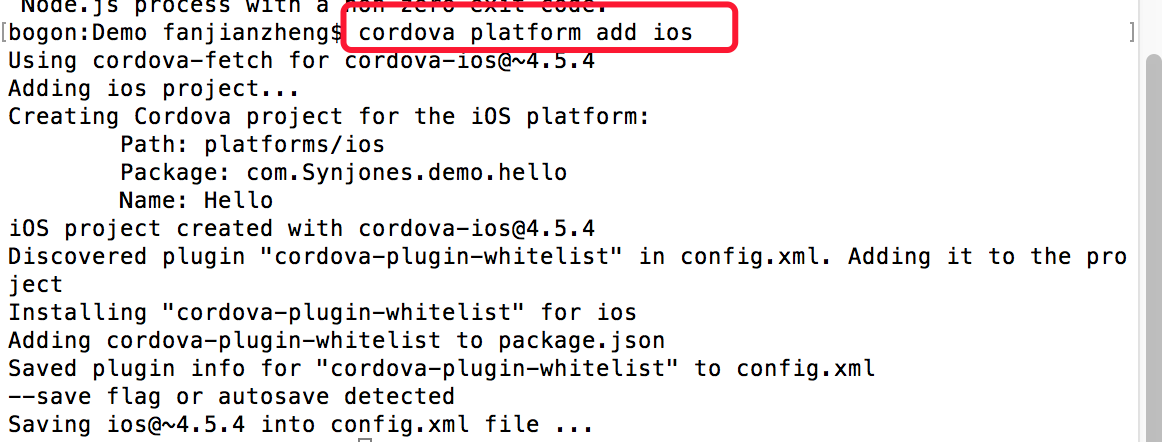
|  |  |  |
| --- | --- | --- |
| **Parameter（参数）** | **Description（描述）** | **Notes（目录）** |
| Demo(参数是必填) | 将为你的项目生成一个Demo目录 | www子目录是应用程序的主页，以及各种资源(css,js,img)，遵循共同的web开发文件命名规范。这些资源将存储在设备上的本地文件系统，而不是远程服务。config.xml文件包含重要的需要生成和分发应用程序的元数据。 |
| com.Synjones.demo.hello (参数可选) | App ID | 如果不填写这个参数，第三个参数就要省略，默认值是 io.cordova.hellocordova，但建议你填写一个适当的值。 |
| Hello(参数可选) | 应用程序的项目名 | 这个参数的默认值是 HelloCordova，但建议你填写一个适当的值。 |

2.添加平台支持

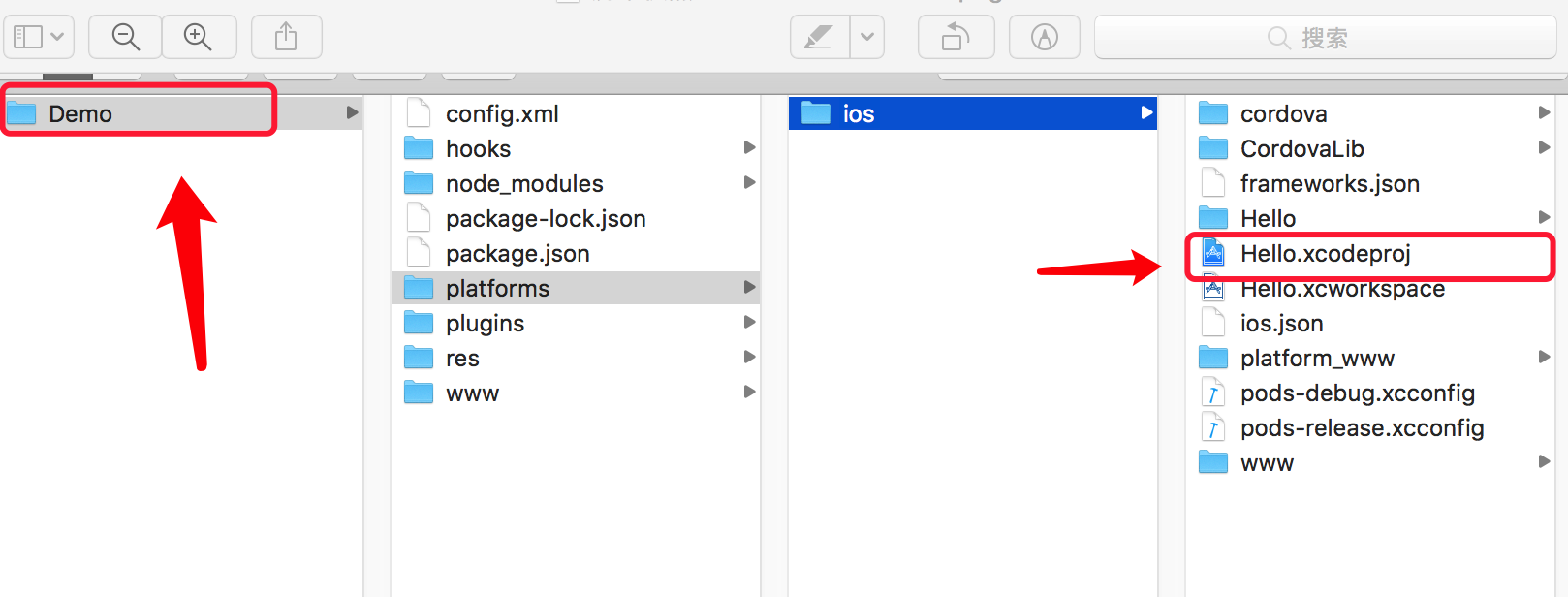
所有后续命令在项目的目录中进行，可在该项目任何子目录中，cd切换到项目目录下:

cd Demo

在构建项目之前，你需要指定一组目标平台。你能够运行这些命令取决于您的机器是否支持每一个SDK，合理是否已安装SDK。在MAC上运行命令：

cordova platform add ios

运行后如下：

如果需要查看Cordova支持平台以及已经添加的平台，终端输入命令：

cordova platforms ls

3.添加插件：

根据项目具体功能需要，可添加插件以简单方便调用原生接口，如需添加插件，可以去[Cordova插件库](https://link.jianshu.com?t=https://cordova.apache.org/plugins/" \t "_blank) 搜索需要的插件：

在终端输入

> cordova plugin add com.phonegap.plugins.barcodescanner

> cordova plugin add org.apache.cordova.file-transfer

> cordova plugin ls

并非所有的插件都是全平台支持的，有些可能只支持安卓，也有的可能同时支持安卓和iOS，所以在多平台开发时，请慎重选择插件。个人建议，可自定义插件类，可参考官网[Cordova自定义插件](https://link.jianshu.com?t=http://cordova.apache.org/docs/en/3.4.0/guide/hybrid/plugins/index.html" \l "Plugin%20Development%20Guide" \t "_blank)。  
4.迭代项目

在工程目录下运行下面的命令来构建项目：

cordova build

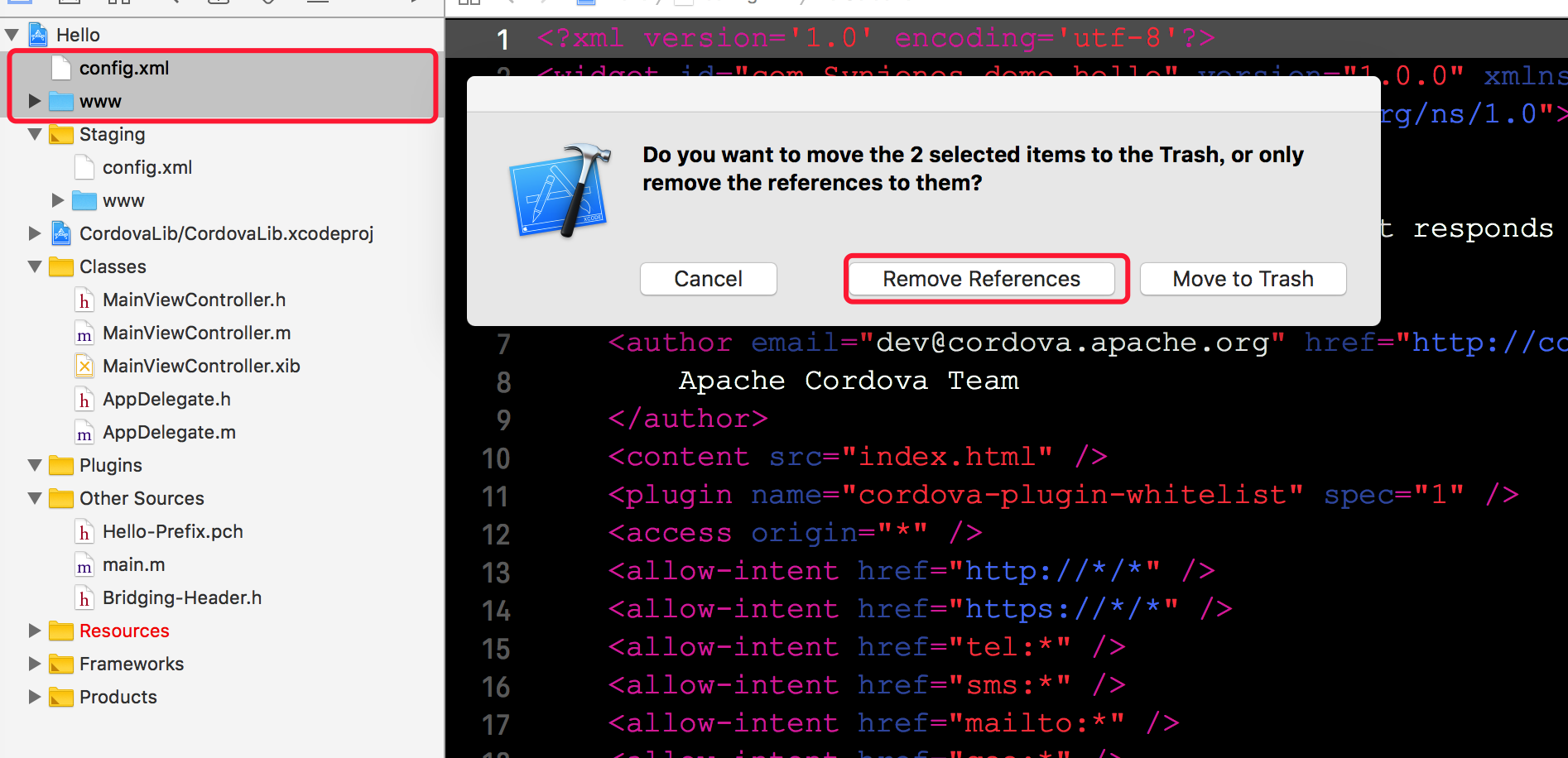
5.运行项目

5.1打开工程

步骤走完第三章，那么，一个简单的Cordova项目就已经搭建完成了，现在，我们运行下我们工程。打开目录下

/Users/fanjianzheng/Desktop/Demo/platforms/ios/Hello.xcworkspace

为了避免引起混淆，建议移除掉下图中两个文件/文件夹引用：



5.2 [Events](https://link.jianshu.com/?t=http://cordova.apache.org/docs/en/latest/cordova/events/events.html" \t "_blank) Cordova生命周期事件

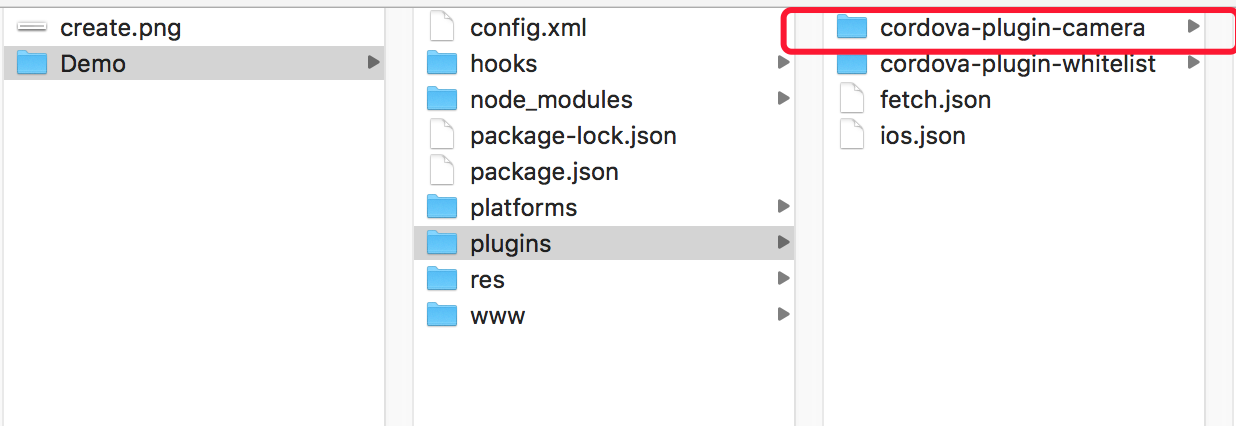
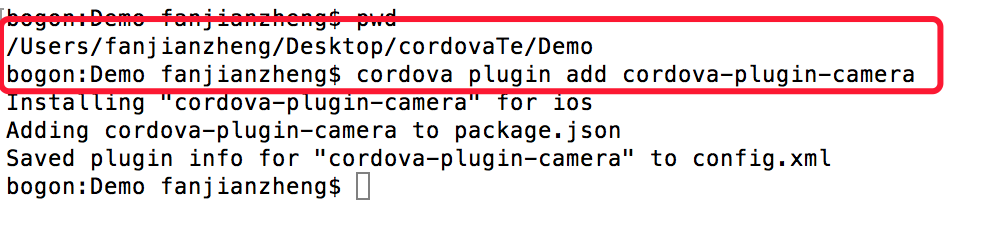
| 生命周期函数 | 释义 |
| --- | --- |
| [deviceready](https://link.jianshu.com/?t=http://cordova.apache.org/docs/en/latest/cordova/events/events.html" \l "deviceready" \t "_blank) | 当Cordova加载完成会触发 |
| [pause](https://link.jianshu.com/?t=http://cordova.apache.org/docs/en/latest/cordova/events/events.html" \l "pause" \t "_blank) | 当应用程序进入到后台时触发 |
| [resume](https://link.jianshu.com/?t=http://cordova.apache.org/docs/en/latest/cordova/events/events.html" \l "resume" \t "_blank) | 应用程序从后台进入到前台会触发 |

5.3 [Plugin APIs](https://link.jianshu.com/?t=http://cordova.apache.org/docs/en/4.0.0/cordova/plugins/pluginapis.html" \t "_blank) 自定义Plugin方法

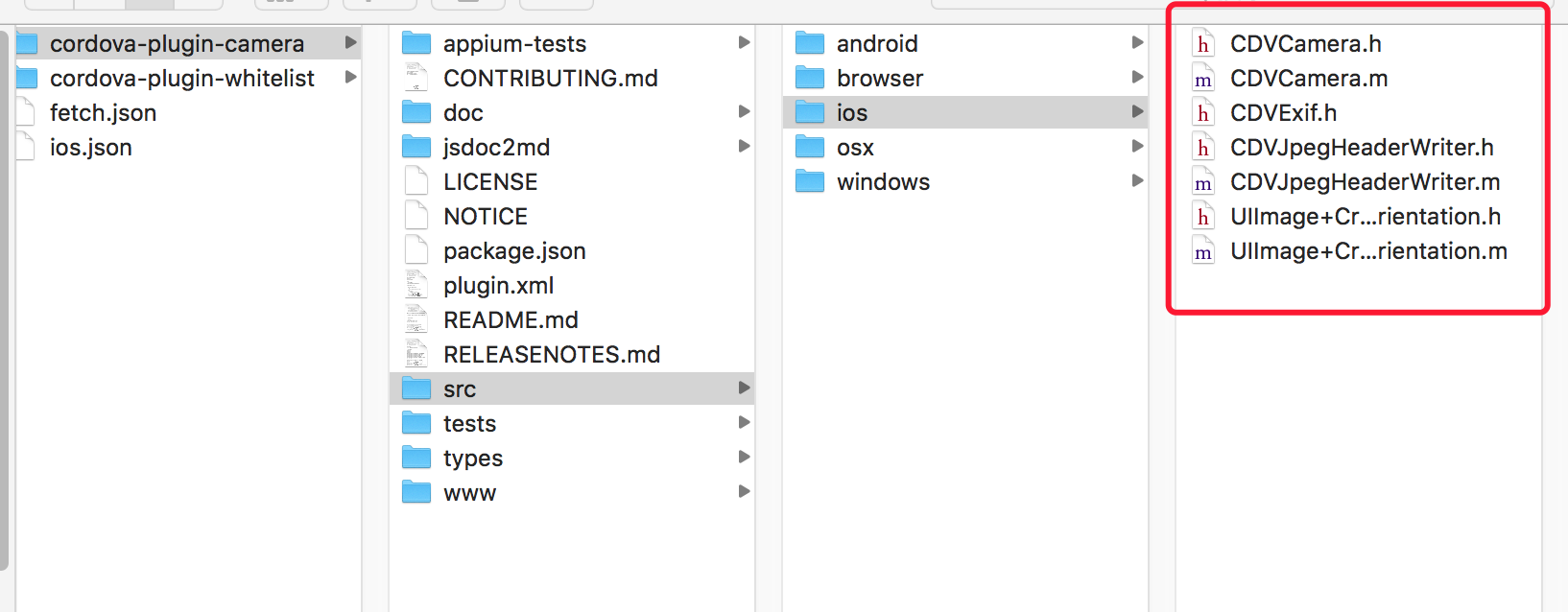
<http://cordova.apache.org/docs/en/4.0.0/cordova/plugins/pluginapis.html>

1. 添加cordova-plugin-camera

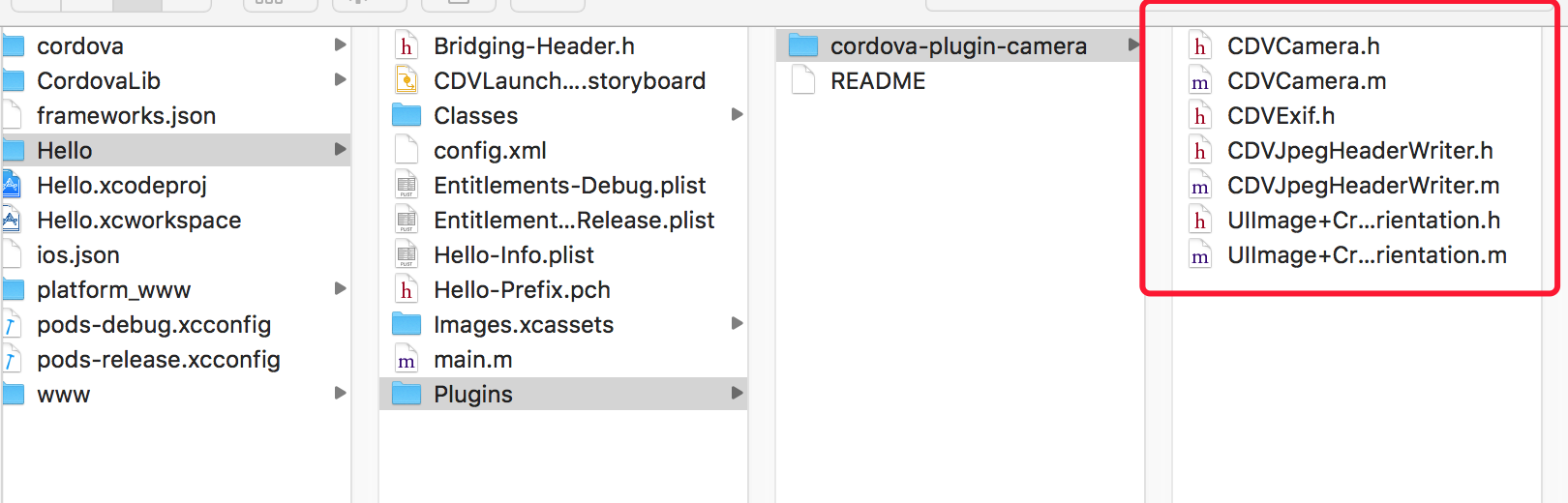
1.切换到iOS项目所在的路径里:



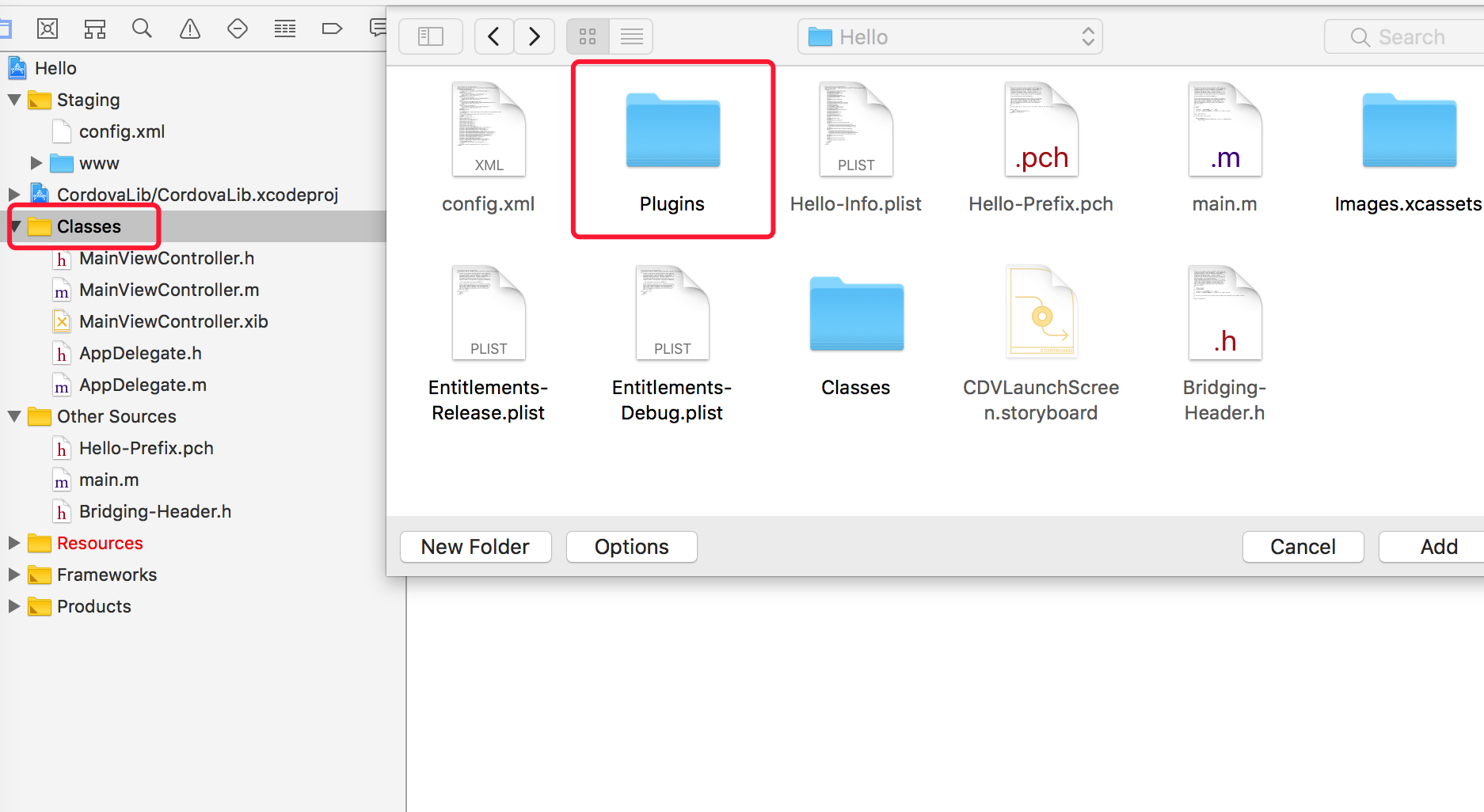
2 找到下面路径中 camera的插件对应的iOS项目的class文件



3 在iOS项目的下图的路径中创建一个Plugins的文件夹，将上图选中的文件copy进来，如下:



4完成上面的步骤，Add Files to ... -> Plugins文件夹:



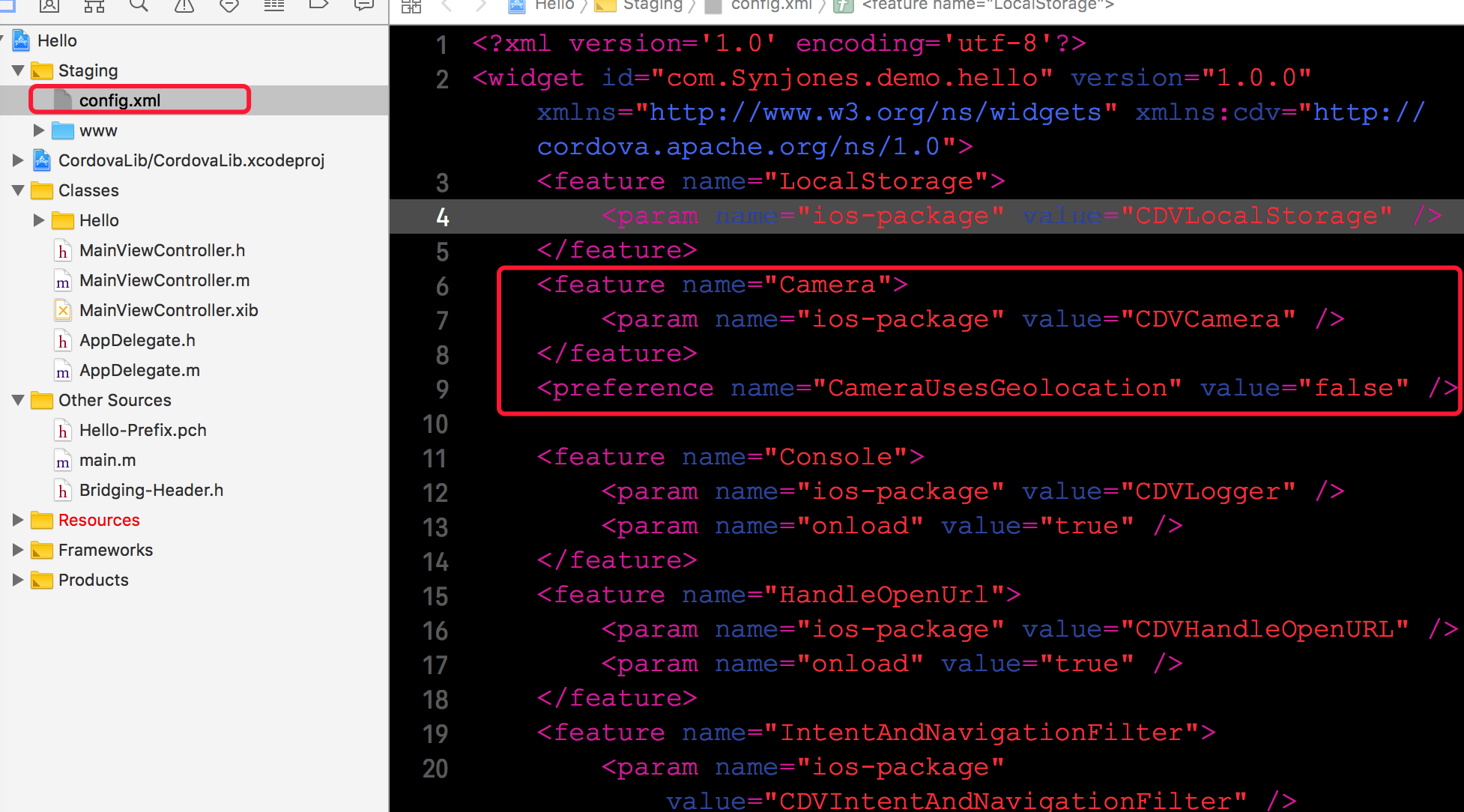
5 在config.xml中添加 camera插件的配置:

<feature name="Camera">

<param name="ios-package" value="CDVCamera" />

</feature>

<preference name="CameraUsesGeolocation" value="false" />



6 编写index.html文件如下:

<!DOCTYPE html>

<html>

<head>

<title>Capture Photo</title>

<script type="text/javascript" charset="utf-8" src="cordova.js"></script>

<script type="text/javascript" charset="utf-8">

var pictureSource;

var destinationType;

document.addEventListener("deviceready",onDeviceReady,false);

function onDeviceReady() {

pictureSource=navigator.camera.PictureSourceType;

destinationType=navigator.camera.DestinationType;

}

function onPhotoDataSuccess(imageData) {

console.log(imageData);

var smallImage = document.getElementById('smallImage');

smallImage.style.display = 'block';

smallImage.src = "data:image/jpeg;base64," + imageData; }

function onPhotoURISuccess(imageURI) {

console.log(imageURI);

var largeImage = document.getElementById('largeImage');

largeImage.style.display = 'block';

largeImage.src = imageURI; }

function capturePhoto() {

navigator.camera.getPicture(onPhotoDataSuccess, onFail, { quality: 50, destinationType: destinationType.DATA\_URL }); }

function capturePhotoEdit() {

navigator.camera.getPicture(onPhotoDataSuccess, onFail, { quality: 20, allowEdit: true, destinationType: destinationType.DATA\_URL }); }

function getPhoto(source) {

navigator.camera.getPicture(onPhotoURISuccess, onFail, { quality: 50, destinationType: destinationType.FILE\_URI, sourceType: source }); }

function onFail(message) { alert('Failed because: ' + message); }

</script>

</head>

<body>

<button onclick="capturePhoto();">Capture Photo</button> <br>

<button onclick="capturePhotoEdit();">Capture Editable Photo</button> <br>

<button onclick="getPhoto(pictureSource.PHOTOLIBRARY);">From Photo Library</button><br>

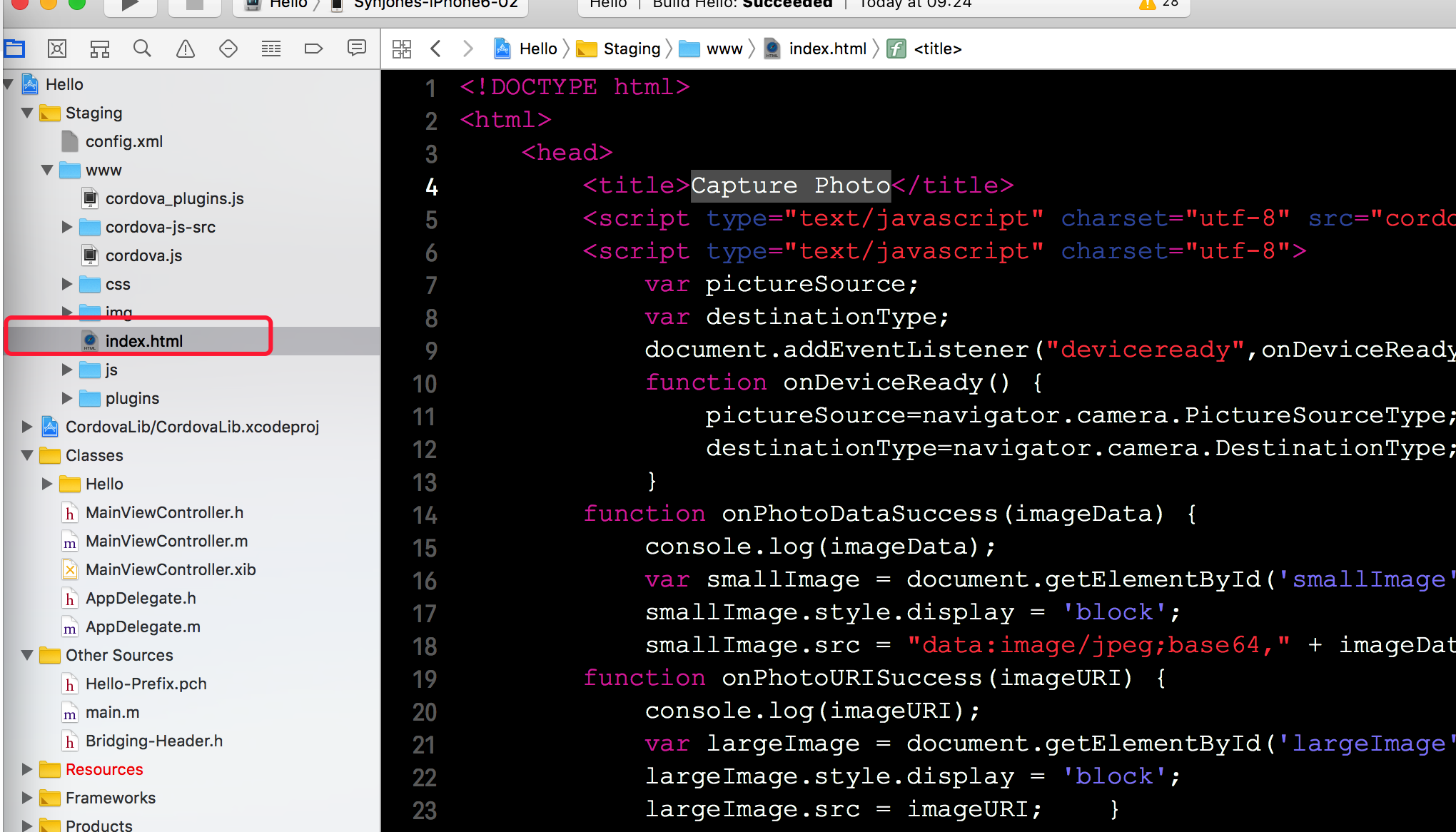
<button onclick="getPhoto(pictureSource.SAVEDPHOTOALBUM);">From Photo Album</button><br>

<img style="display:none;width:60px;height:60px;" id="smallImage" src="" />

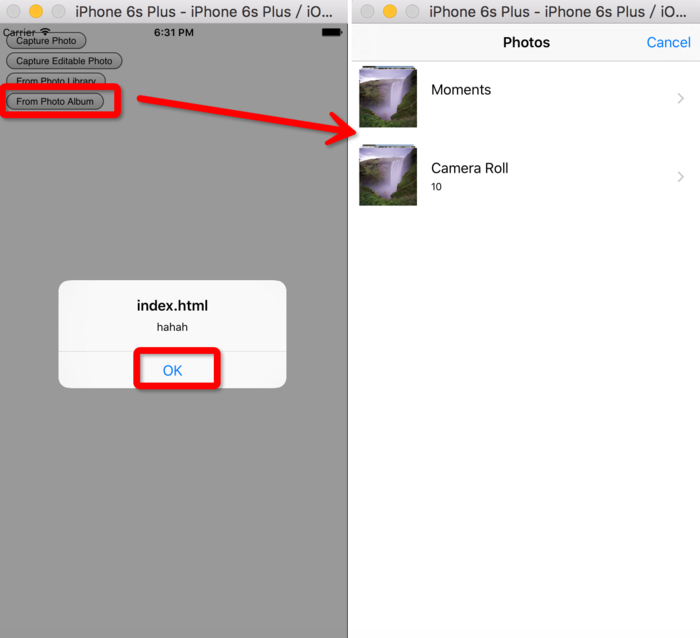
<img style="display:none;" id="largeImage" src="" />

</body>

</html>



7 运行效果



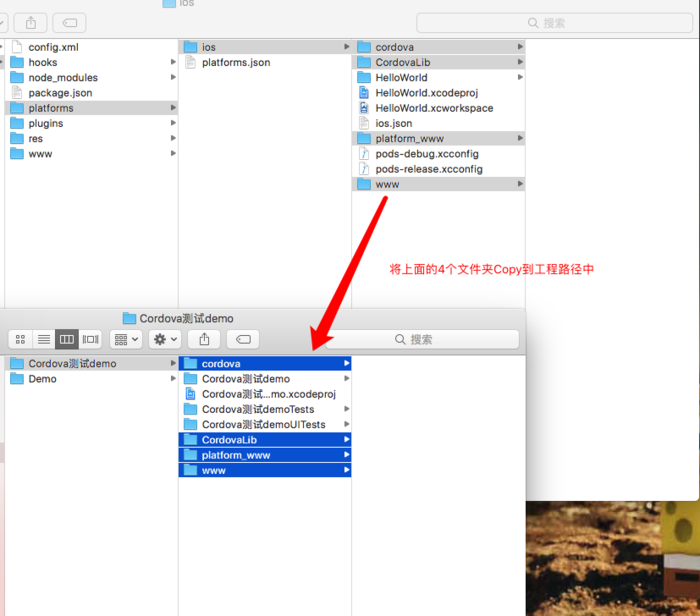
1. 将Cordova本地相关文件copy到已存的项目中

3.1找到相关路径，复制文件到工程目录

首先，上面中创建的Cordova项目的路径中

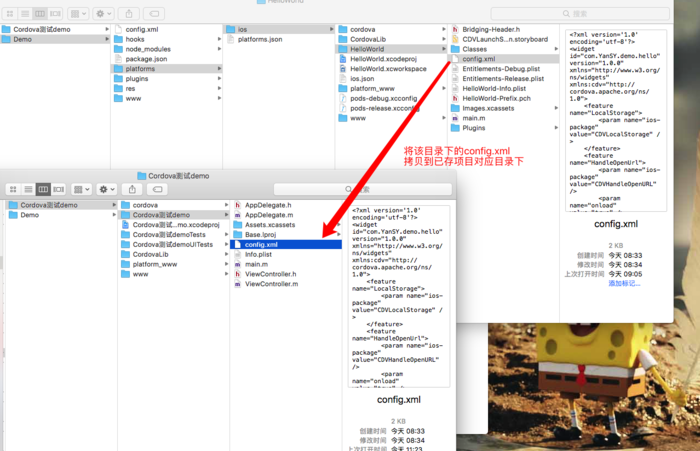
/Users/fanjianzheng/Desktop/Demo/platforms/ios,找到下面需要copy的4个文件夹copy到工程项目的根路径中，

CordovaLib cordova www platform\_www



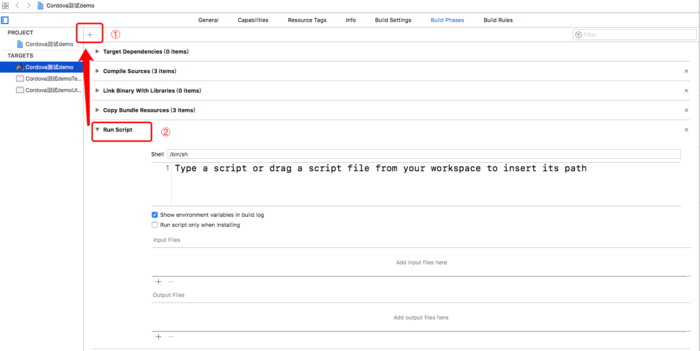
其次，拷贝对应的config.xml文件到对应的工程目录

然后将下面图中所示的config.xml文件copy到需要的iOS项目路径文件夹内，如图：



3.2配置相关参数

3.2.1打开工程，进入工程TARGETS -> Build Phases 下，然后入下图所示，点①中的按键，创建New Run Script Phase,②为创建的Run Script:

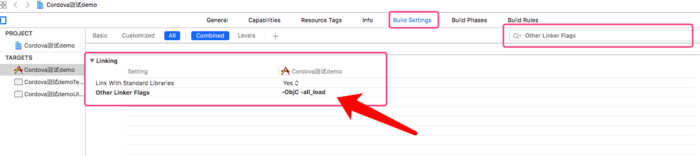


3.2.2将下图示意图中，①原名称Run Script修改为Copy www directory;然后讲②中的选项去掉，最后将下面的代码字段复制到③中：

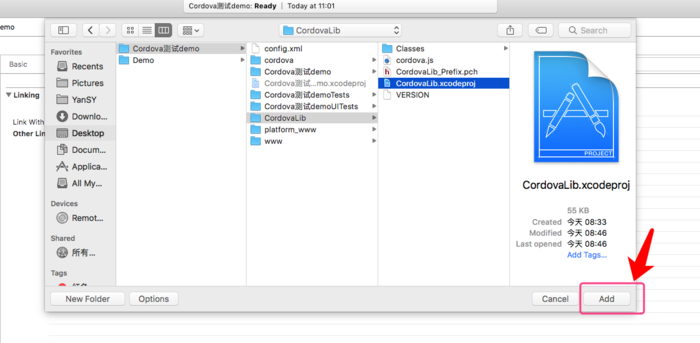
NODEJS\_PATH=/usr/local/bin; NVM\_NODE\_PATH=~/.nvm/versions/node/`nvm version 2>/dev/null`/bin; N\_NODE\_PATH=`find /usr/local/n/versions/node/\* -maxdepth 0 -type d 2>/dev/null | tail -1`/bin; XCODE\_NODE\_PATH=`xcode-select --print-path`/usr/share/xcs/Node/bin; PATH=$NODEJS\_PATH:$NVM\_NODE\_PATH:$N\_NODE\_PATH:$XCODE\_NODE\_PATH:$PATH && node cordova/lib/copy-www-build-step.js

3.3.3

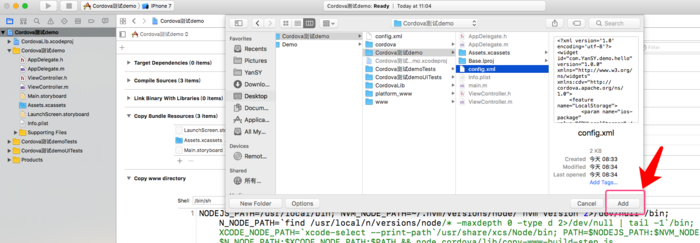
3.2.3在工程 Build Settings -> Other Linker Flags 中添加-ObjC -all\_load



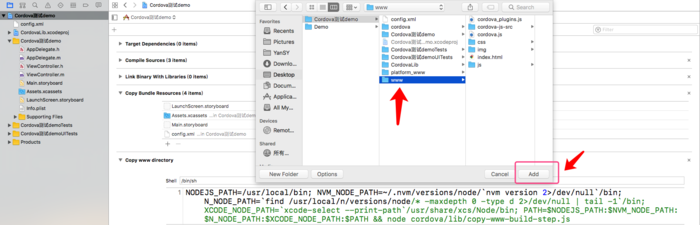
3.2.4Add Files to ... -> CordovaLib.xcodeproj



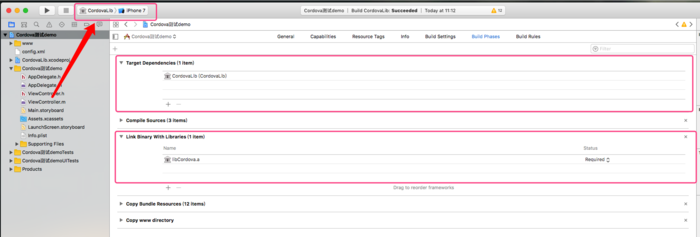
3.2.5 Add Files to ... -> config.xml



3.2.6 Add Files to ... -> www文件夹



3.2.7 Schemes按照下图配置，然后编译一次程序（command+R），最后导入下面两项  
Build Phases -> Target Dependencies -> CordovaLib  
Build Phases -> Link Binary With Libraries -> libCordova.a



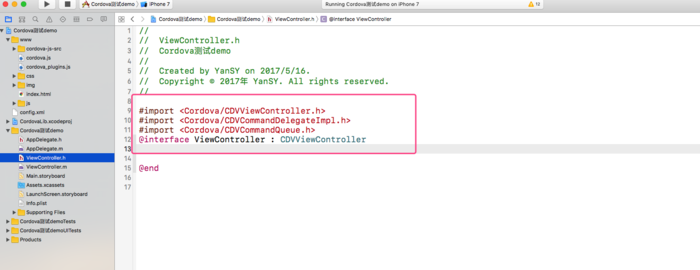
3.2.8 Schemes按照下图配置，将"ViewController.h" 文件改为:

#import <Cordova/CDVViewController.h>

#import <Cordova/CDVCommandDelegateImpl.h>

#import <Cordova/CDVCommandQueue.h> @interface ViewController : CDVViewController

@end

  
至此，Cordova嵌入已存的开发项目就已经完成了，运行程序就可看到工程中wwww文件目录下，index.html文件中的网页信息了。只需要将该文件内容，改为公司需要的网页内容即可。具体交互以及自定义插件，将在后面文章中介绍。

3.3 自定义插件

3.3.1 在config.xml文件中加入下面代码：

<feature name="YourPluginName">

<param name="ios-package" value="YanSYPlugin" />

<param name="onload" value="true" />

</feature>



### 3.1 创建一个YanSYPlugin类，实现下面的方法

下面介绍一个简单的web端调用原生手机相机以及选择相册图片并返回web显示的自定义插件调用：

* （1）YanSYPlugin.h 的实现

#import <Cordova/CDVPlugin.h> @interface YanSYPlugin : CDVPlugin<UIImagePickerControllerDelegate, UINavigationControllerDelegate> {

UIImagePickerController \*\_imagePickerController; // 定义相机控件类 } // 本类插件方法 - 当在web调用时 - (void)myMethod:(CDVInvokedUrlCommand\*)command;

@end

* （2）YanSYPlugin.m 的实现

#import "YanSYPlugin.h"

//#import <GTMBase64.h> // base64 转码

typedef void(^imgBlock)(NSString \* data);

@interface YanSYPlugin()

@property(nonatomic,strong) imgBlock MyBlock;

@end

@implementation YanSYPlugin

- (void)myMethod:(CDVInvokedUrlCommand\*)command{

/\*

NSString\* \_callbackId; 回调时的id

NSString\* \_className; 类名

NSString\* \_methodName; 方法名

NSArray\* \_arguments; 参数

\*/

[self.commandDelegate runInBackground:^{

NSString\* myarg = [command.arguments objectAtIndex:0];

if (myarg !=nil) {

[self switchMethodWithName:myarg andCommand:(CDVInvokedUrlCommand\*)command];

}

else{

return ;

}

}];

}

// 根据方法选择判断调用具体内容

- (void)switchMethodWithName:(NSString \*)name andCommand:(CDVInvokedUrlCommand\*)command{

\_imagePickerController = [[UIImagePickerController alloc] init];

\_imagePickerController.delegate = self;

\_imagePickerController.modalTransitionStyle = UIModalTransitionStyleFlipHorizontal;

\_imagePickerController.allowsEditing = YES;

\_\_block CDVPluginResult\* pluginResult = nil;

NSString \* standbyJSStr = [NSString stringWithFormat:@"%@",@"data:image/png;base64,"];

if ([name isEqualToString:@"调用相机"]) {

[self selectImageFromCameraWithBlock:^(NSString \*data) {

// 因当前转码格式为iOS原生转码，web端无法解析 ，所以百度了一波图片转码成功后复用，若转码格式正确 可直接上传字符串

// NSString \*jsStr = [NSString stringWithFormat:@"data:image/png;base64,%@",data];

// 使用备用字符串上传

pluginResult = [CDVPluginResult resultWithStatus:CDVCommandStatus\_OK messageAsString:standbyJSStr];

[self.commandDelegate sendPluginResult:pluginResult callbackId:command.callbackId];

}];

}

else if ([name isEqualToString:@"查看相册"]){

[self selectImageFromAlbumWithBlock:^(NSString \*data) {

// 因当前转码格式为iOS原生转码，web端无法解析 ，所以百度了一波图片转码成功后复用，若转码格式正确 可直接上传字符串

// NSString \*jsStr = [NSString stringWithFormat:@"data:image/png;base64,%@",data];

// 使用备用字符串上传

pluginResult = [CDVPluginResult resultWithStatus:CDVCommandStatus\_OK messageAsString:standbyJSStr];

[self.commandDelegate sendPluginResult:pluginResult callbackId:command.callbackId];

}];

}

else{

pluginResult = [CDVPluginResult resultWithStatus:CDVCommandStatus\_ERROR messageAsString:@"You Are Error"];

[self.commandDelegate sendPluginResult:pluginResult callbackId:command.callbackId];

}

}

#pragma mark 从摄像头获取图片或视频

- (void)selectImageFromCameraWithBlock:(imgBlock)block{

self.MyBlock = block;

\_imagePickerController.sourceType = UIImagePickerControllerSourceTypeCamera;

//相机类型（拍照、录像...）字符串需要做相应的类型转换

//视频上传质量

//UIImagePickerControllerQualityTypeHigh高清

//UIImagePickerControllerQualityTypeMedium中等质量

//UIImagePickerControllerQualityTypeLow低质量

//UIImagePickerControllerQualityType640x480

\_imagePickerController.videoQuality = UIImagePickerControllerQualityTypeHigh;

//设置摄像头模式（拍照，录制视频）为录像模式

\_imagePickerController.cameraCaptureMode = UIImagePickerControllerCameraCaptureModePhoto;

[self.viewController presentViewController:\_imagePickerController animated:YES completion:nil];

}

#pragma mark 从相册获取图片或视频

- (void)selectImageFromAlbumWithBlock:(imgBlock)block{

self.MyBlock = block;

\_imagePickerController.sourceType = UIImagePickerControllerSourceTypePhotoLibrary;

[self.viewController presentViewController:\_imagePickerController animated:YES completion:nil];

}

#pragma mark UIImagePickerControllerDelegate

//该代理方法仅适用于只选取图片时

- (void)imagePickerController:(UIImagePickerController \*)picker didFinishPickingImage:(UIImage \*)image editingInfo:(nullable NSDictionary<NSString \*,id> \*)editingInfo {

NSString \*encodedImageStr = [self imageProcessing:image];

self.MyBlock(encodedImageStr);

}

//适用获取所有媒体资源，只需判断资源类型

- (void)imagePickerController:(UIImagePickerController \*)picker didFinishPickingMediaWithInfo:(NSDictionary<NSString \*,id> \*)info{

NSString \*encodedImageStr = [self imageProcessing:info[UIImagePickerControllerEditedImage]];

self.MyBlock(encodedImageStr);

[self.viewController dismissViewControllerAnimated:YES completion:nil];

}

// 图像处理 若未下载base64者 可不用查看

- (NSString \*)imageProcessing:(UIImage \*)proImage{

// 判断传过来照片的大小进行裁剪

CGFloat width = proImage.size.width;

CGFloat height = proImage.size.height;

CGSize size;

if (width>height) {

size = CGSizeMake(800, 450);

}

else{

size = CGSizeMake(450, 800);

}

// 创建一个bitmap的context

// 并把它设置成为当前正在使用的context

UIGraphicsBeginImageContext(size);

// 绘制改变大小的图片

[proImage drawInRect:CGRectMake(0,0, size.width, size.height)];

// 从当前context中创建一个改变大小后的图片

UIImage \* tailoringImage =UIGraphicsGetImageFromCurrentImageContext();

// 使当前的context出堆栈

UIGraphicsEndImageContext();

// 判断图片大小进行压缩

NSData \*imageData = UIImageJPEGRepresentation(tailoringImage,1.0);

NSLog(@"imagedata == %lud,size.width = %f==%f",(unsigned long)imageData.length,tailoringImage.size.width,tailoringImage.size.height);

if (imageData.length>100\*1024) {

if (imageData.length>1024\*1024) { //1M以及以上

imageData=UIImageJPEGRepresentation(tailoringImage, 0.1);

}else if (imageData.length>512\*1024) {//0.5M-1M

imageData=UIImageJPEGRepresentation(tailoringImage, 0.3);

}else if (imageData.length>200\*1024) {//0.25M-0.5M

imageData=UIImageJPEGRepresentation(tailoringImage, 0.7);

}

}

NSLog(@"imagedata == %lud,size.width = %f==%f",(unsigned long)imageData.length,tailoringImage.size.width,tailoringImage.size.height);

// 最后进行 base64 转码

// 注：iOS端转base64 的方法可能web无法简析为图片，可从pod下载第三方库 GTMBase64 进行转码 ，这里不再赘述

// NSString \* encodedImageStr = [GTMBase64 stringByEncodingData:imageData];

// 原生转码方法

NSString \* encodedImageStr = [imageData base64EncodedStringWithOptions:NSDataBase64Encoding64CharacterLineLength];

return encodedImageStr;

}

* （3）编写index.html文件如下

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title></title>

<script type="text/javascript" src="cordova.js"></script>

<script type="text/javascript" src="cordova\_plugins.js"></script>

<script type="text/javascript" src="jquery.min.js"></script>

<script type="text/javascript">

document.addEventListener("deviceready", yourCallbackFunction, false);

function cameraBTClick(){

Cordova.exec(successFunction, failFunction, "YanSYPlugin", "myMethod", ["调用相机"]);

}

function PhotoAlbumBTClick(){

Cordova.exec(failFunction, failFunction, "YanSYPlugin", "myMethod", ["查看相册"]);

}

function failFunction(error){

alert("error");

document.getElementById("returnValue").value = img;

}

function aaaa(){

document.getElementById("2").innerHTML = "<img src='data:image/png;base64,"

}

function successFunction(img){

document.getElementById("returnValue").value = img;

document.getElementById("2").innerHTML = "![]("+img+")"

}

</script>

</head>

<body>

<p>点击下面按钮调用相机拍照</p>

<button onclick="cameraBTClick()">调用相机</button>

</body>

<body>

<p>点击下面按钮选择相册照片</p>

<button onclick="PhotoAlbumBTClick()">查看相册</button>

<button onclick="aaaa()">测试</button>

<h1>这是回调结果展示区</h1>

<textarea id ="returnValue" type="value" rows="5" cols="40">

</textarea>

</body>

<body>/Users/YanSY/Desktop/代码运行示意11.gif

<p>

<img id="1" src=“”/>

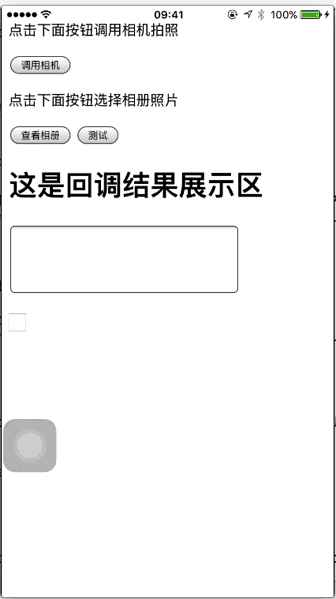
</p>

<p id="2" > </p>

</body>

</html>

将以上代码，按步骤写入程序中，可测试调用原生相机，注意，需要在plist文件中添加隐私权限。运行效果示例如下，因为转码格式为iOS原生格式转码，与web不兼容，所以上传回调时并未用选择照片或者拍摄照片上传，如果需要，请在工程中下载第三方库 GTMBase64 进行转码回调。



三方插件的使用：Camera插件):

https://www.jianshu.com/p/1e3d0c915dbc

Cordova常用插件:<https://blog.csdn.net/github_39500961/article/details/76270299>

参考文档:<https://www.jianshu.com/p/b24465b80014>

Xcode以后出现上述问题，可以运行app，但是在打包Product-Archive时会报“Cordova/CDVViewController file not found”   
解决办法：   
在Build Setting->Search Paths->Header Search Paths中加入   
“$(OBJROOT)/UninstalledProducts/$(PLATFORM\_NAME)/include”   
如下图所示：

