# **前言**

前一篇文章讲解了Android原生工程如何集成Flutter项目的具体过程，[Flutter混合开发(一)：Android项目集成Flutter模块详细指南](https://segmentfault.com/a/1190000020939448) ,本篇将带着大家来一起学习原生iOS项目如何集成Flutter。

因为每个版本的集成逻辑都是有差别的，所以这里交代下本篇文章的集成版本:

Flutter （channel dev，v1.10.16）
dart （v2.7.0）
Xcode （v11.2）
cocoapds （1.8.4）

# **创建Flutter module**

假如iOS项目的路径是这样的：flutter/flutter\_hybrid/iOS Project,那么我们需要在iOS Project上一层目录flutter\_hybrid中创建Flutter module。

cd flutter/flutter\_hybrid/

flutter create -t module flutter\_module

输入后控制台打印如下：

$ flutter create -t module flutter\_module

Creating project flutter\_module...

 flutter\_module/test/widget\_test.dart (created)

 flutter\_module/flutter\_module.iml (created)

 flutter\_module/.gitignore (created)

 flutter\_module/.metadata (created)

 flutter\_module/pubspec.yaml (created)

 flutter\_module/README.md (created)

 flutter\_module/lib/main.dart (created)

 flutter\_module/flutter\_module\_android.iml (created)

 flutter\_module/.idea/libraries/Flutter\_for\_Android.xml (created)

 flutter\_module/.idea/libraries/Dart\_SDK.xml (created)

 flutter\_module/.idea/modules.xml (created)

 flutter\_module/.idea/workspace.xml (created)

Running "flutter pub get" **in** flutter\_module... 1.2s

Wrote 12 files.

All done!

Your module **code** is **in** flutter\_module/lib/main.dart.

看到All done就表示我们项目创建好了。整个module目录和原生Flutter基本一样，主要就是Android、iOS的宿主工程和lib目录以及pubspec.yaml文件。

# **添加Flutter module依赖**

为iOS项目添加依赖需要使用CocoaPods，如果你还没有用到CocoaPods，可以参考[https://cocoapods.org/](https://cocoapods.org/%22%20%5Ct%20%22https%3A//segmentfault.com/a/_blank)上面的说明来安装CocoaPods。

如果你的项目之前没有使用过cocoapods，那么需要进行初始化生成podfile文件，进入iOS项目的根目录执行：

pod init

然后打开podfile文件，进行配置：

*# 配置*

flutter\_application\_path = '../flutter\_module/'

load File.join(flutter\_application\_path, '.ios', 'Flutter', 'podhelper.rb')

target 'iOSFlutterHybrid' **do**

 *# Comment the next line if you don't want to use dynamic frameworks*

 use\_frameworks!

 *# Pods for iOSFlutterHybrid*

 *# 配置*

 install\_all\_flutter\_pods(flutter\_application\_path)

 target 'iOSFlutterHybridTests' **do**

 inherit! :search\_paths

 *# Pods for testing*

 **end**

 target 'iOSFlutterHybridUITests' **do**

 *# Pods for testing*

 **end**

配置添加好后在项目根目录运行以下命令进行安装：

pod **install**

控制台输出：

Analyzing dependencies

Downloading dependencies

Installing Flutter (1.0.0)

Installing FlutterPluginRegistrant (0.0.1)

Installing flutter\_module (0.0.1)

Generating Pods project

Integrating client project

[!] Please close any current Xcode sessions and **use** `iOSFlutterHybrid.xcworkspace` **for** this **project** **from** **now** on.

Pod installation **complete**! There **are** 3 dependencies **from** the Podfile **and** 3 total pods installed.

[!] Automatically assigning platform `iOS` **with** **version** `13.2` **on** target `iOSFlutterHybrid` because **no** platform was specified. Please specify a platform **for** this target **in** your Podfile. See `https://guides.cocoapods.org/syntax/podfile.html#platform`.

这里我们看到有三个依赖安装完成了。并提醒我们关闭当前项目，在根目录下面使用iOSFlutterHybrid.xcworkspace来打开运行项目。这里可能很多人在执行命令的时候会发现提示0个依赖完成。这里有可能是你的Xcode版本的问题。因为Flutter要求最低版本是10.2及以上。

当在flutter\_module/pubspec.yaml中添加一个Flutter插件时，需要在flutter\_module目录下运行：

flutter packages **get**

来刷新podhelper.rb脚本中的插件列表，然后在iOS目录下运行：

pod **install**

这样podhelper.rb脚本才能确保添加的插件和Flutter.framework能够添加到iOS项目中。

目前Flutter还不支持Bitcode，所以集成了Flutter的iOS项目需要禁用Bitcode。

在以下路径下找到Bitcode并禁用：

B**uild** Settings->B**uild** Options->Enable Bitcode



flutter以前的版本是需要添加build phase以构建Dart代码，但是最新的版本已经不需要添加了，可以自动构建。

# **调用Flutter module**

Flutter为我们提供了两种调用方式：FlutterViewController和FlutterEngine，FlutterEngine在使用的时候会有一些问题，将在下文进行说明。

##### **FlutterViewController方式：**

我们打开ViewController.m文件，在里面添加一个加载flutter页面的方法并且添加一个按钮看来调用：

**#import "ViewController.h"#import <Flutter/Flutter.h>#import "AppDelegate.h"**

**@interface** **ViewController** ()

**@end**

**@implementation** **ViewController**

- (**void**)viewDidLoad {

 [**super** viewDidLoad];

 UIButton \*button = [UIButton buttonWithType:UIButtonTypeCustom];

 [button addTarget:**self** action:**@selector**(handleButtonAction) forControlEvents:UIControlEventTouchUpInside];

 [button setTitle:@"加载Flutter" forState:UIControlStateNormal];

 [button setBackgroundColor:[UIColor blueColor]];

 button.frame = CGRectMake(100, 100, 160, 60);

 [**self**.view addSubview:button];

}

- (**void**)handleButtonAction{

 FlutterViewController \*flutterViewController =[FlutterViewController new];

 *//设置路由参数*

 [flutterViewController setInitialRoute:@"route2"];

 [**self** presentViewController:flutterViewController animated:false completion:nil];

}

**@end**

当我们运行项目点击加载Flutetr按钮时，将会调用Flutter页面。和Android项目集成一样，这里的setInitialRoute可以设置一个json数组来传递需要交互的参数。并在Flutter中使用window.defaultRouteName来获取传递的参数。

##### **FlutterEngine方式：**

我们需要在AppDelegate中对FlutterEngine进行初始化。打开AppDelegate.h文件：

**#import <UIKit/UIKit.h>#import <Flutter/Flutter.h>**

**@interface** **AppDelegate** : **FlutterAppDelegate@property** (**nonatomic**,**strong**) FlutterEngine \*flutterEngine;**@end**

在打开AppDelegate.m文件：

*// 如果你需要用到Flutter插件时***#import <FlutterPluginRegistrant/GeneratedPluginRegistrant.h> #include "AppDelegate.h"**

**@implementation** **AppDelegate**

- (BOOL)application:(UIApplication \*)application

 didFinishLaunchingWithOptions:(NSDictionary \*)launchOptions {

 **self**.flutterEngine = [[FlutterEngine alloc] initWithName:@"io.flutter" project:nil];

 [**self**.flutterEngine runWithEntrypoint:nil];

 [GeneratedPluginRegistrant registerWithRegistry:**self**.flutterEngine]; *//如果你需要用到Flutter插件时*

 **return** [**super** application:application didFinishLaunchingWithOptions:launchOptions];

}**@end**

然后在ViewController.m文件定义的handleButtonAction中调用：

- (**void**)handleButtonAction{

 FlutterEngine \*flutterEngine = [(AppDelegate \*)[[UIApplication sharedApplication] delegate] flutterEngine];

 FlutterViewController \*flutterViewController = [[FlutterViewController alloc] initWithEngine:flutterEngine nibName:nil bundle:nil];

 [flutterViewController setInitialRoute:@"route2"];

 [self presentViewController:flutterViewController animated:false completion:nil];

}

当我们运行项目点击加载Flutter按钮时，将会调用Flutter页面。前文讲到使用FlutterEngine会有问题，就是我们setInitialRoute传递的参数在flutter中永远获取到的都是 “/” ，这个是Fltter SDK的一个Bug，所以如果必须依赖setInitialRoute，还是使用FlutterViewController的形式来加载Flutter模块。

# **热重启/重新加载**

大家在写纯Flutter应用的时候，知道是有热重启/重新加载功能的，但是在做混合开发的过程中，你会发现热重启/重新加载功能失效了。那么如何在混合开发中开启热重启/重新加载功能呢？

* 首先接入我们的设备或者模拟器
* 将我们的App关闭，退出后台，在terminal中运行 flutter attach命令

$ flutter attach

Waiting **for** a connection **from** Flutter **on** Android SDK built **for** x86...

复制代码此时就在等待设备的连接。这里要注意的是，如果电脑连接了多台设备需要使用 -d 命令来指定一台设备，参数为设备的id。

flutter attach -d '你的设备id'

然后启动我们的应用会看到控制台输出：

Done.

Syncing files **to** device Android SDK built **for** x86... 1,393ms

**To** hot reload changes **while** running, press "r". **To** hot restart (and rebuild state), press "R".

An Observatory debugger and profiler on Android SDK built **for** x86 is available at: http:*//127.0.0.1:59354/zRsDBfpesrk=/***For** a more detailed help message, press "h". **To** detach, press "d"; **to** quit, press "q".

这样就表示我们连接成功了。在输出的日志中也告诉了我们如何使用热重启/重新加载功能。

在Terminal中输入以下命令​：​

r : 热加载；

R : 热重启；

h : 获取帮助；

d : 断开连接；

q : 退出；

这里的的 d 和 q 的命令都有退出调试，区别在于 d 命令只是单纯的断开而 q 命令会将应用退到后台。

# **调试Dart代码**

同样在混合开发过程中我们如何调试dart代码呢？

* 关闭我们的应用
* 点击Android Studio工具栏上的Flutter Attach按钮(需要安装Flutter与Dart插件)



* 启动我们的应用

接下来就可以像调试普通Flutter项目一样来调试混合开发模式下的Dart代码了。

# **总结**

本文主要是讲解了iOS集成Flutter项目的步骤，其中也遇到了一些问题，由于我的Xcode版本较低，在集成的过程中iOS项目的依赖一直失败。最后才发现是Xcode的版本问题。这里花费了很多时间去排查问题。