# 12.5 插件开发：iOS端API实现

本节我们接着之前“获取电池电量”插件的示例，来完成iOS端API的实现。以下步骤使用Objective-C，如果您更喜欢Swift，可以直接跳到后面Swift部分。

首先打开Xcode中Flutter应用程序的iOS部分:

1. 启动 Xcode
2. 选择 File > Open…
3. 定位到您 Flutter app目录, 然后选择里面的 iOS文件夹，点击 OK
4. 确保Xcode项目的构建没有错误。
5. 选择 Runner > Runner ，打开AppDelegate.m

接下来，在application didFinishLaunchingWithOptions:方法内部创建一个FlutterMethodChannel，并添加一个处理方法。 确保与在Flutter客户端使用的通道名称相同。

#import <Flutter/Flutter.h>  
  
@implementation AppDelegate  
- (BOOL)application:(UIApplication\*)application didFinishLaunchingWithOptions:(NSDictionary\*)launchOptions {  
 FlutterViewController\* controller = (FlutterViewController\*)self.window.rootViewController;  
  
 FlutterMethodChannel\* batteryChannel = [FlutterMethodChannel  
 methodChannelWithName:@"samples.flutter.io/battery"  
 binaryMessenger:controller];  
  
 [batteryChannel setMethodCallHandler:^(FlutterMethodCall\* call, FlutterResult result) {  
 // TODO  
 }];  
  
 return [super application:application didFinishLaunchingWithOptions:launchOptions];  
}

接下来，我们添加Objective-C代码，使用iOS电池API来获取电池电量，这和原生是相同的。

在AppDelegate类中添加以下新的方法：

- (int)getBatteryLevel {  
 UIDevice\* device = UIDevice.currentDevice;  
 device.batteryMonitoringEnabled = YES;  
 if (device.batteryState == UIDeviceBatteryStateUnknown) {  
 return -1;  
 } else {  
 return (int)(device.batteryLevel \* 100);  
 }  
}

最后，我们完成之前添加的setMethodCallHandler方法。我们需要处理的平台方法名为getBatteryLevel，所以我们在call参数中需要先判断是否为getBatteryLevel。 这个平台方法的实现只需调用我们在前一步中编写的iOS代码，并使用result参数返回成功或错误的响应。如果调用了未定义的API，我们也会通知返回：

[batteryChannel setMethodCallHandler:^(FlutterMethodCall\* call, FlutterResult result) {  
 if ([@"getBatteryLevel" isEqualToString:call.method]) {  
 int batteryLevel = [self getBatteryLevel];  
  
 if (batteryLevel == -1) {  
 result([FlutterError errorWithCode:@"UNAVAILABLE"  
 message:@"电池信息不可用"  
 details:nil]);  
 } else {  
 result(@(batteryLevel));  
 }  
 } else {  
 result(FlutterMethodNotImplemented);  
 }  
}];

现在可以在iOS上运行该应用程序了，如果使用的是iOS模拟器，请注意，它不支持电池API，因此应用程序将显示“电池信息不可用”。

### 使用Swift实现iOS API

以下步骤与上面使用Objective-C相似，首先打开Xcode中Flutter应用程序的iOS部分:

1. 启动 Xcode
2. 选择 File > Open…
3. 定位到您 Flutter app目录, 然后选择里面的 ios文件夹，点击 OK
4. 确保Xcode项目的构建没有错误。
5. 选择 Runner > Runner ，然后打开AppDelegate.swift

接下来，覆盖application方法并创建一个FlutterMethodChannel绑定通道名称samples.flutter.io/battery：

@UIApplicationMain  
@objc class AppDelegate: FlutterAppDelegate {  
 override func application(  
 \_ application: UIApplication,  
 didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {  
 GeneratedPluginRegistrant.register(with: self);  
  
 let controller : FlutterViewController = window?.rootViewController as! FlutterViewController;  
 let batteryChannel = FlutterMethodChannel.init(name: "samples.flutter.io/battery",  
 binaryMessenger: controller);  
 batteryChannel.setMethodCallHandler({  
 (call: FlutterMethodCall, result: FlutterResult) -> Void in  
 // Handle battery messages.  
 });  
  
 return super.application(application, didFinishLaunchingWithOptions: launchOptions);  
 }  
}

接下来，我们添加Swift代码，使用iOS电池API来获取电池电量，这和原生开发是相同的。

将以下新方法添加到AppDelegate.swift底部:

private func receiveBatteryLevel(result: FlutterResult) {  
 let device = UIDevice.current;  
 device.isBatteryMonitoringEnabled = true;  
 if (device.batteryState == UIDeviceBatteryState.unknown) {  
 result(FlutterError.init(code: "UNAVAILABLE",  
 message: "电池信息不可用",  
 details: nil));  
 } else {  
 result(Int(device.batteryLevel \* 100));  
 }  
}

最后，我们完成之前添加的setMethodCallHandler方法。我们需要处理的平台方法名为getBatteryLevel，所以我们在call参数中需要先判断是否为getBatteryLevel。 这个平台方法的实现只需调用我们在前一步中编写的iOS代码，并使用result参数返回成功或错误的响应。如果调用了未定义的API，我们也会通知返回：

batteryChannel.setMethodCallHandler({  
 (call: FlutterMethodCall, result: FlutterResult) -> Void in  
 if ("getBatteryLevel" == call.method) {  
 receiveBatteryLevel(result: result);  
 } else {  
 result(FlutterMethodNotImplemented);  
 }  
});

现在可以在iOS上运行应用程序，如果使用的是iOS模拟器，请注意，它不支持电池API，因此应用程序将显示“电池信息不可用”。