iOS版Facebook SDK

组件包含

- The Facebook Core SDK (includes Analytics)
- The Facebook Login SDK
- The Facebook Sharing SDK

> *Note：可以根据App使用任一组件以节约空间*

集成方式

CocoaPods

```
pod 'FBSDKCoreKit'
pod 'FBSDKLoginKit'
pod 'FBSDKShareKit'
```

```
pod install
```

在工程文件**Info.plist**配置**FacebookAppID**、**FacebookDisplayName**、**CFBundleURLSchemes**

```
<plist version="1.0">
  <dict>
    <key>FacebookAppID</key>
    <string>{your-app-id}</string>
    <key>FacebookDisplayName</key>
    <string>{your-app-name}</string>
    <key>CFBundleURLTypes</key>
    <array>
      <dict>
      <key>CFBundleURLSchemes</key>
      <array>
        <string>fb{your-app-id}</string>
      </array>
      </dict>
    </array>
  </dict>
</plist>
```

入门指南

功能预览

- Facebook Analytics-了解人们如何使用您的产品。
- Facebook Login-使用他们的Facebook凭据对人进行身份验证。
- Share and Send dialogs-允许将应用程序中的内容共享到Facebook。
- App Events-在您的应用程序中记录事件。
- Graph API-读写Graph API。

前期准备

- Facebook 开发者账户
- 您的应用的AppID

步骤 1：设置你的开发环境

将SDK集成到应用程序中

步骤 2：配置工程

使用包含有关您的应用程序数据的XML代码片段配置Info.plist文件。

1. 右键点击**Info.plist**，选择 **Open As ▸ Source Code**.
2. 将以下XML代码段复制并粘贴到文件的主体中（ ... </ dict>）。

```
<key>CFBundleURLTypes</key>
<array>
  <dict>
  <key>CFBundleURLSchemes</key>
  <array>
    <string>fb[APP_ID]</string>
  </array>
  </dict>
</array>
<key>FacebookAppID</key>
<string>[APP_ID]</string>
<key>FacebookDisplayName</key>
<string>[APP_NAME]</string>
```

3. 在 键**CFBundleURLSchemes**的 中，将**[APP_ID]**替换为您的App ID，将**[APP_NAME]** 替换成您的应用名字。
4. 要使用可以执行将应用程序切换到Facebook应用程序的任何Facebook对话框（例如，登录，共享，应用程序邀请等），您应用程序的**Info.plist**还需要包括：...）。

```
<key>LSApplicationQueriesSchemes</key>
<array>
  <string>fbapi</string>
  <string>fbapi20130214</string>
  <string>fbapi20130410</string>
  <string>fbapi20130702</string>
  <string>fbapi20131010</string>
  <string>fbapi20131219</string>
  <string>fbapi20140410</string>
  <string>fbapi20140116</string>
  <string>fbapi20150313</string>
  <string>fbapi20150629</string>
  <string>fbapi20160328</string>
  <string>fbauth</string>
  <string>fb-messenger-share-api</string>
  <string>fbauth2</string>
  <string>fbshareextension</string>
</array>
```

步骤 3：连接应用委托

用以下代码替换**AppDelegate.m**中的代码。 此代码在您的应用启动时初始化SDK，并允许SDK在执行“登录”或“共享”操作时处理来自本地Facebook应用的结果：

```
// Objective-C
//
// AppDelegate.m

@implementation AppDelegate

- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

  [[FBSDKApplicationDelegate sharedInstance]
application:application

didFinishLaunchingWithOptions:launchOptions];
  return YES;
```

```
}


- (BOOL)application:(UIApplication *)application
            openURL:(NSURL *)url
            options:(nonnull
NSDictionary<UIApplicationOpenURLOptionsKey, id> *)options
{
  [[FBSDKApplicationDelegate sharedInstance]
application:application
                                                openURL:url

options:options];
  return YES;
}
```

如果您使用的是iOS 13或更高版本，请在SceneDelegate中添加以下方法：

```
// Objective-C
//
// SceneDelegate.m

#import <FBSDKCoreKit/FBSDKCoreKit.h>

@import FacebookCore;

@interface SceneDelegate ()

@end

@implementation SceneDelegate

- (void)scene:(UIScene *)scene openURLContexts:
(NSSet<UIOpenURLContext *> *)URLContexts
{
  UIOpenURLContext *context = URLContexts.allObjects.firstObject;
  [FBSDKApplicationDelegate.sharedInstance
application:UIApplication.sharedApplication

openURL:context.URL

sourceApplication:context.options.sourceApplication
```

```
 annotation:context.options.annotation];
}
```

步骤 4：在模拟器上编译并运行工程

Event Manager 会显示App发送到Facebook Analytics的事件，如果这是首次使用此代码启动应用，则可能需要等待一段时间才会显示

事件最多可能需要20分钟才能显示

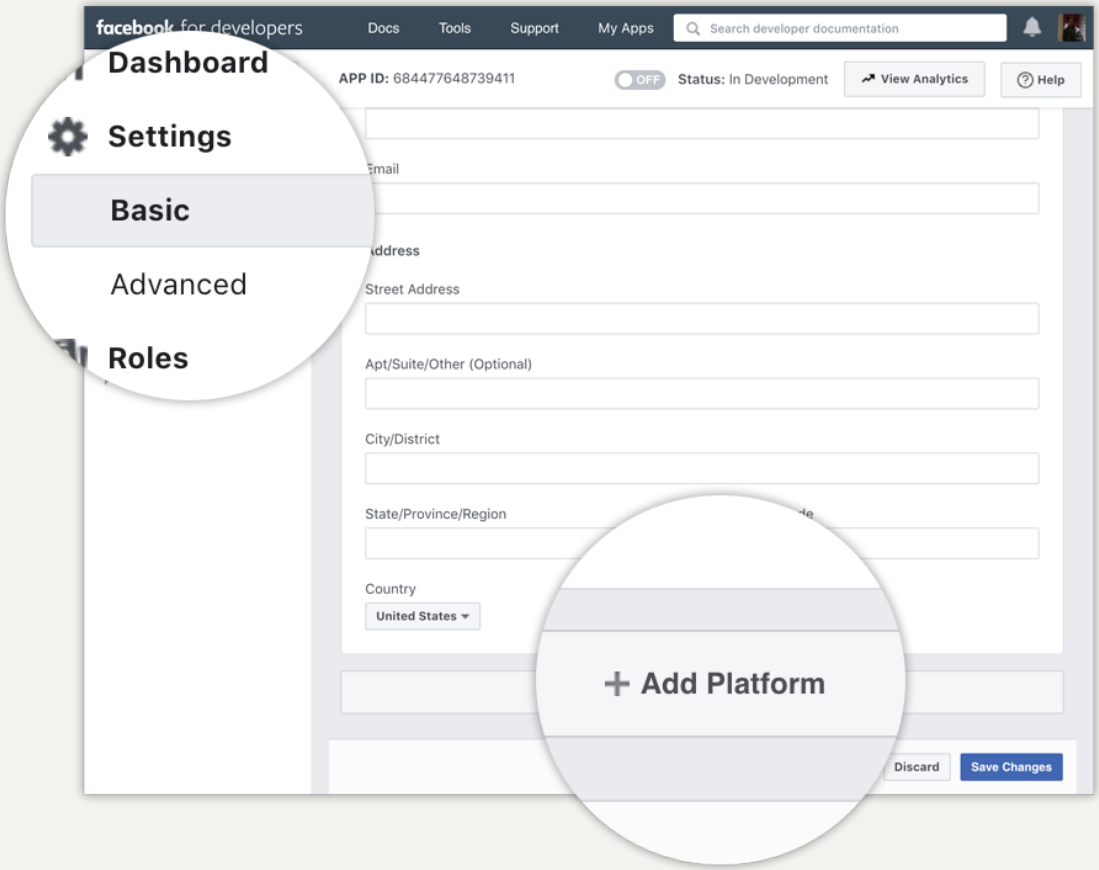
iOS版Facebook App Event

前提

- A **Facebook Developer Account**
- A **Facebook Ad Account**
- A **Facebook app**

步骤 1：配置App

转到**App Dashboard**，单击**My Apps**，然后创建一个新应用程序（如果尚未安装）。 导航到 **Settings** > **Basic**，以查看具有您的**App ID**，您的**App Secret**的详细信息以及有关您应用程序的其他详细信息。



向下滚动到页面底部，然后单击**Add Platform**。 选择**iOS**，添加您的应用详细信息，然后保存更改。

通过添加以下详细信息来设置广告应用程序：

- **App Domains** - 提供您应用程序的Apple App Store URL。
- **Privacy Policy URL** - -提供隐私权政策网址。 必须公开您的应用。
- **Terms of Service URL** - -提供服务条款URL。
- **Platform** - 滚动到“设置”面板的底部以添加iOS平台。

步骤 2：关联广告账户

要在**Ads Manager**中投放广告并评估安装量，请至少将一个**广告帐户**与您的应用相关联。

在**App Dashboard**(应用列表)中， **Settings > Advanced**。
在授权的广告帐户ID(**Authorized Ad Account IDs**)中，添加您的广告帐户ID。 您可以从**Ads Manager**中获取广告帐户ID。
如果您还有一个企业帐户，请在**Authorized Businesses**中添加您的业务经理ID。 您可以在**Business Manager**的URL中找到您的业务经理ID

步骤 3：添加App Event

在应用中跟踪事件的方式有以下三种：

- 自动记录的事件-应用程序的安装，启动和应用程序内购买均通过Facebook SDK自动记录。
- The Codeless App Events tool-使用此工具可以添加标准事件，而无需向您的应用程序添加代码。
- 手动记录的事件-将代码添加到您的应用中以跟踪标准事件和自定义事件。

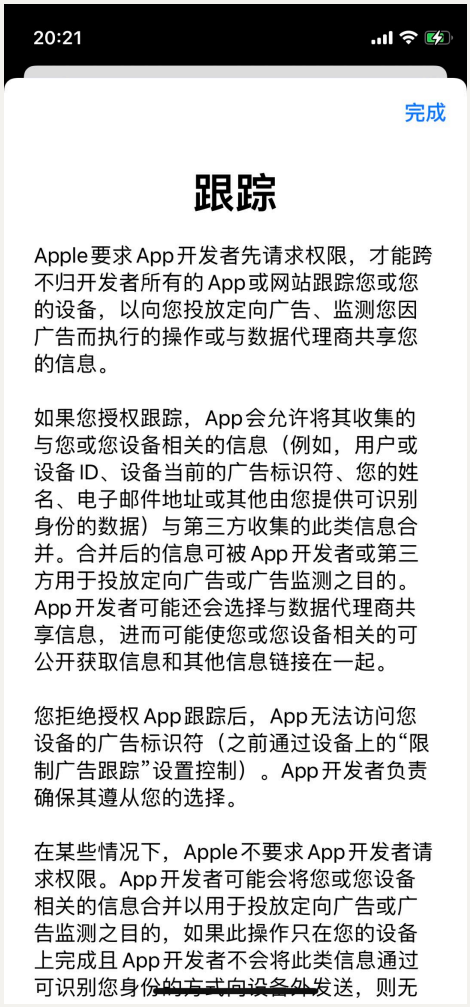
a .自动记录事件

使用Facebook SDK时，除非禁用自动事件记录功能，否则将自动记录并收集应用程序中的某些事件以供Facebook Analytics使用。 这些事件与所有用例（目标，度量和优化）相关。 作为自动应用程序事件记录的一部分，收集了三个关键事件：应用程序安装，应用程序启动和购买。 启用自动日志记录后，广告客户可以禁用这些事件以及其他Facebook内部事件，例如登录印象事件。 但是，如果您禁用了自动日志记录，但仍想记录特定事件，例如安装或购买事件，请在您的应用中手动为这些事件实现日志记录。

EVENT	DETAILES
应用安装	新用户首次激活应用程序或应用程序首次在特定设备上启动。
应用启动	当有人启动您的应用程序时，Facebook SDK会初始化并记录事件。 但是，如果第二个应用程序启动事件在第一个应用程序启动事件之后的60秒内发生，则不会记录第二个应用程序启动事件。
应用内购买	由Apple App Store或Google Play处理的购买完成后。 如果您使用其他付款平台，则需要手动添加购买事件代码。
Facebook SDK Crash Report	如果您的应用程序由于Facebook SDK而崩溃，则当您重新启动应用程序时，将生成崩溃报告并发送给Facebook。 该报告不包含用户数据，可帮助Facebook确保SDK的质量和稳定性。 要选择不记录此事件，请禁用自动记录的事件。

注意：这些说明适用于SDK的4.0版或更高版本。 如果您仍在使用3.X版，请在“应用程序事件”调用中省略“ SDK”。 例如，您将使用FBAppEvents而不是FBSDKAppEvents。

允许App请求跟踪权限



从iOS 14开始，您需要设置用户同意标志以共享用户同意状态。 如果用户同意，则调用FBSDKSettings类的setAdvertiserTrackingEnabled方法，并将其设置为iOS（对于iOS）或true（对于Swift）。 如果用户不允许跟踪，则将iOS的setAdvertiserTrackingEnabled设置为NO，或者将Swift设置为false。 该方法将返回一个布尔值，以指示是否成功设置了用户同意标志。

```
// Set the flag to YES Or NO if a user allows advertiser tracking
[FBAdSettings setAdvertiserTrackingEnabled:YES];
```

禁用/启用自动记录的事件

要禁用自动事件日志记录，请在Xcode中以代码形式打开应用程序的.plist并将以下XML添加到属性字典中：

```
<key>FacebookAutoLogAppEventsEnabled</key>
<false/>
```

在某些情况下，您希望延迟自动记录事件的收集，例如获得用户同意或履行法律义务，而不是禁用它。 在这种情况下，在最终用户提供同意之后，调用FBSDKSettings类的setAutoLogAppEventsEnabled方法以重新启用自动记录。

```
[FBSDKSettings setAutoLogAppEventsEnabled:YES];
```

要出于任何原因再次暂停收集，请将iOS的setAutoLogAppEventsEnabled方法设置为NO，或者将Swift设置为false。

```
[FBSDKSettings setAutoLogAppEventsEnabled:NO];
```

还可以使用应用仪表板禁用自动应用内购买事件日志记录。 转到 **Basic>Settings**下的iOSApp，然后将开关切换到“否”。

禁用自动SDK初始化

要禁用SDK的自动初始化，请在Xcode中以代码的形式打开应用程序的.plist，并将以下XML添加到属性字典中：

```
<key>FacebookAutoInitEnabled</key>
<false/>
```

在某些情况下，您希望延迟自动SDK初始化，例如获得用户同意或履行法律义务，而不是禁用它。 在这种情况下，请调用ApplicationDelegate类的initializeSDK方法。 最终用户同意后，通过将Swift的AutoInitEnabled设置为true或将Objective-C的YES设置为true，为将来的日志记录启用SDK自动初始化。

```
[FBSDKSettings setAutoInitEnabled: YES ];
[ApplicationDelegate initializeSDK:nil];
```

禁止收集广告商ID

要禁用**advertiser-id**的收集，请在Xcode中以代码的形式打开应用程序的**Info.plist**，并将以下XML添加到属性字典中：

```
<key>FacebookAdvertiserIDCollectionEnabled</key>
<false/>
```

在某些情况下，您希望延迟收集Advertiser_id，例如获得用户同意或履行法律义务，而不是禁用它。 在这种情况下，请在最终用户表示同意后，调用FBSDKSettings类的setAdvertiserIDCollectionEnabled方法并将其设置为iOS（对于iOS）或true（对于Swift）。

```
[FBSDKSettings setAdvertiserIDCollectionEnabled:@YES/@NO];
```

b .The Codeless App Events tool(无代码应用程序事件工具)

要求

- Facebook iOS Full SDK v4.34 or higher
- Facebook iOS Core SDK v4.38 or higher
- Facebook iOS Core SDK v4.34-4.37 and the Marketing Kit

请访问“ iOS应用程序事件入门指南”以安装最新版本的iOS SDK，《升级指南》以升级到最新版本的SDK，或者，如果仅安装了Core SDK v4.34-4.37 ，通过将以下内容添加到Podfile中来添加Marketing Kit。

```
pod 'FBSDKMarketingKit'
```

实现无代码应用程序事件日志功能

通过打开应用程序的.plist作为Xcode中的代码来打开无代码调试事件日志记录，并将以下XML添加到属性字典中：

```
<key>FacebookCodelessDebugLogEnabled</key>
<true/>
```

添加App事件

转到Events Manager以无代码方式添加您要跟踪的应用程序事件。

1. 在**Event Manger**中，单击**Add Data Source**，然后在下拉菜单中选择**App Events**。
2. 单击**User our codeless event setup tool**，然后选择要向其中添加事件的应用。
3. 通过单击**Start Setup**选择平台。
4. 如果这是您第一次访问无代码流程，您将看到一个小教程。 教程结束后，在移动设备上打开您的应用程序的新会话。
5. 摇动手机，直到出现应用程序的版本。
6. 单击任何元素以添加应用程序事件。 导航到应用程序的不同页面，以选择整个应用程序中的元素。
7. 在弹出菜单中单击**Save**，或单击“取消”以不添加事件。
8. 添加所有事件后，单击**Review and Finish**。
9. 单击**Test Events**或保存并退出。

*注意：事件最多可能需要30分钟才会显示在**Event Manger**中。*

验证您的集成

转到**App Ads Helper**。

- 选择一个应用程序，然后单击**Submit**。
- 转到底部，然后选择**App Events Tester**。

- 如果您的应用正在发送**fb_codeless_debug**事件，则这些事件将在表中列出。

FB_CODELESS_DEBUG

Time Logged	Platform	App Version	Event Parameters
08/09/18 13:52:06	iOS	1.0	_activity_name: UITabBarController

c. 手动记录事件

代码生成器

使用代码生成器获取标准事件(standard events)或自定义事件以及参数(parameters)的代码。*详情见官网*

为标准或自定义事件生成代码

- 选择所需事件代码类型的标签，**Standard Event** 或 **Custom Event**。
- 在**Event Name**中，输入事件的名称。 如果要向自定义事件添加参数，
- 请单击**Add Event Paramater**。
- 单击**Get Code**。
- 在窗口中，选择一种语言以获取代码以复制并粘贴到您的应用中。

示例：点击事件

```
/**
 * For more details, please take a look at:
 *
developers.facebook.com/docs/reference/ios/current/class/FBSDKApp
Events
 */
- (void)logAdClickEvent:(NSString *)adType {
    NSDictionary *params =
    @{FBSDKAppEventParameterNameAdType : adType};
    [FBSDKAppEvents
     logEvent:FBSDKAppEventNameAdClick
     parameters:params];
}
```

要记录自定义事件，只需将事件名称作为**NSString**传递即可：

```
[FBSDKAppEvents logEvent:@"battledAnOrc"];
```

d . 测试事件

App Ads Helper允许您在应用程序中测试应用程序事件，以确保您的应用程序将事件发送到Facebook。

1. 打开**App Ads Helper**
2. 在**Select an App**中，选择您的应用程序，然后选择**Submit**。
3. 滚动到底部，然后选择**Test Event**。
4. 启动应用并发送事件，该事件将会显示在页面上。

iOS版Facebook Login

前期工作

已经配置好基本信息

1. 添加登录功能到代码中

```
#import "ViewController.h"
#import <FBSDKCoreKit/FBSDKCoreKit.h>
#import <FBSDKLoginKit/FBSDKLoginKit.h>
@interface ViewController ()
@end

@implementation ViewController
- (void)viewDidLoad {
    [super viewDidLoad];
    FBSDKLoginButton *loginButton = [[FBSDKLoginButton alloc]
init];
    // Optional: Place the button in the center of your view.
    loginButton.center = self.view.center;
    [self.view addSubview:loginButton];
}
```

此时，您应该能运行应用并使用Facebook“登录”按钮登录。

2.检查当前登录状态

- 应用一次只能允许一位用户登录。我们会以 `[FBSDKAccessToken currentAccessToken]` 代表登录应用的每位用户。
- `FBSDKLoginManager` 将为您设置此口令，且在设置`currentAccessToken` 时，还会自动将口令写入 Keychain 缓存。
- `FBSDKAccessToken` 包含 `userID`，您可以使用此编号识别用户。

您应该更新视图控件，在加载时便检查当前口令。这有助于在用户已经向您的应用授予权限后，避免不必要地重复显示登录流程：

```
// Objective-C
- (void)viewDidLoad {
  [super viewDidLoad];
  if ([FBSDKAccessToken currentAccessToken]) {
    // User is logged in, do work such as go to next view
controller.
  }
}
```

3.请求更多权限

使用 Facebook 登录时，您的应用可以请求一些用户数据的访问权限。

Facebook“登录”按钮的读取权限

要请求额外的读取权限，请设置 `FBSDKLoginButton` 对象的 `readPermissions` 属性。

```
// Objective-C // public_profile、email基本权限，不需要审核即可用
// 将下列代码添加到 viewDidLoad 方法中：loginButton.readPermissions =
@[@"public_profile", @"email"];
```

系统会提示用户向您的应用授予所请求的权限。请注意，部分权限必须通过登录审核。如需详细了解权限，请参阅管理权限。

4.进阶

我们使用登录管理工具类(`FBSDKLoginManager`)和自定义按钮(`UIButton`)调用“登录”对话框

```
// Add this to the header of your file
#import "ViewController.h"
#import <FBSDKCoreKit/FBSDKCoreKit.h>
#import <FBSDKLoginKit/FBSDKLoginKit.h>

// Add this to the body
@implementation ViewController

- (void)viewDidLoad {
  [super viewDidLoad];

  // Add a custom login button to your app
  UIButton *myLoginButton=[UIButton
buttonWithType:UIButtonTypeCustom];
  myLoginButton.backgroundColor=[UIColor darkGrayColor];
  myLoginButton.frame=CGRectMake(0,0,180,40);
  myLoginButton.center = self.view.center;
  [myLoginButton setTitle: @"My Login Button" forState:
UIControlStateNormal];

  // Handle clicks on the button
  [myLoginButton
    addTarget:self
    action:@selector(loginButtonClicked)
forControlEvents:UIControlEventTouchUpInside];

  // Add the button to the view
  [self.view addSubview:myLoginButton];
}

// Once the button is clicked, show the login dialog
-(void)loginButtonClicked
{
  FBSDKLoginManager *login = [[FBSDKLoginManager alloc] init];
  [login
    logInWithReadPermissions: @[@"public_profile"]
          fromViewController:self
                     handler:^(FBSDKLoginManagerLoginResult
*result, NSError *error) {
    if (error) {
      NSLog(@"Process error");
    } else if (result.isCancelled) {
```

```
      NSLog(@"Cancelled");
    } else {
      NSLog(@"Logged in");
    }
  }];
}

@end
```

使用 Interface Builder

要使用 `Interface Builder` 添加"登录"按钮图形：

1. 向布局添加 `View` 对象。根据需要调整其尺寸。
2. 在 `Identity inspector` 中，将 `Class` 属性更改为 `FBSDKLoginButton`。

现在，您的布局将如下所示：

然后，您可以更新在应用启用时运行的代码。此代码将在视图展示之前加载 `FBSDKLoginButton`

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
// Override point for customization after application launch.
[FBSDKLoginButton class];
...
return YES;
}
```

访问口令

通知

您可以使用 `NSNotificationCenter` 中的 `FBSDKAccessTokenDidChangeNotification` 来追踪 `currentAccessToken` 变化。这样可方便您响应用户登录状态的变化。

```
[[NSNotificationCenter defaultCenter]
addObserverForName:FBSDKAccessTokenDidChangeNotification
                                                    object:nil
                                                     queue:
[NSOperationQueue mainQueue]
                                                usingBlock:
 ^(NSNotification *notification) {
   if (notification.userInfo[FBSDKAccessTokenDidChangeUserID]) {
     // Handle user change
   }
 }];
```

iOS SDK 可以随着时间更新 `currentAccessToken`，例如：当 SDK 刷新使用更长有效日期的口令时。所以，您应该检查通知中的 `userInfo` 字典是否包含 `FBSDKAccessTokenDidChangeUserID`，从而确定用户是否已更改。

个人主页

FBSDKProfile 包含公开的个人主页信息，可方便您使用用户的姓名和头像打造个性化应用体验。您可以使用 loadCurrentProfileWithCompletion: 加载已登录用户的个人主页。

```
[FBSDKProfile loadCurrentProfileWithCompletion:
 ^(FBSDKProfile *profile, NSError *error) {
   if (profile) {
     NSLog(@"Hello, %@!", profile.firstName);
   }
 }];
```

您可以将 enableUpdatesOnAccessTokenChange 属性设置为 true，让个人主页自动加载到 [FBSDKProfile currentProfile]。这样还可方便您检查 FBSDKProfileDidChangeNotification，以便响应个人主页变化：

```
[FBSDKProfile enableUpdatesOnAccessTokenChange:YES];
[[NSNotificationCenter defaultCenter]
addObserverForName:FBSDKProfileDidChangeNotification
                                                    object:nil
                                                     queue:
[NSOperationQueue mainQueue]
                                                usingBlock:
 ^(NSNotification *notification) {
   if ([FBSDKProfile currentProfile]) {
     // Update for new user profile
   }
 }];
```

显示头像

FBSDKProfilePictureView 类是显示用户的 Facebook 头像的简单方法。实际操作时，只需设置 profileID 属性。您可以将此属性设置为 [[FBSDKAccessToken currentAccessToken] userID] 值，从而显示当前登录用户的头像。

```
FBSDKProfilePictureView *profilePictureView =
[[FBSDKProfilePictureView alloc] init];
profilePictureView.frame = CGRectMake(0,0,100,100);
profilePictureView.profileID = [[FBSDKAccessToken
currentAccessToken] userID];
[self.view addSubview:profilePictureView];
```

iOS版Facebook Sharing

前提条件

- 将 iOS 版 Facebook SDK 添加到您的移动开发环境中
- 配置并链接您的 Facebook 应用编号
- 将应用编号、显示名称和需要获取照片访问权限的理由添加到应用的 `.plist` 文件中。请注意，该理由必须能让人轻松看懂。
- 将 `FBSDKShareKit.framework` 链接到您的项目。

内容建模

每种内容类型都有一个您可以用于表示该内容类型并且符合 `<FBSDKSharingContent>` 的界面。对内容建模后，将符合 `<FBSDKSharing>` 的分享界面（例如 `FBSDKShareDialog`）添加到应用。

链接

```
FBSDKShareLinkContent *content = [[FBSDKShareLinkContent alloc]
init];
content.contentURL = [NSURL
URLWithString:@"https://developers.facebook.com"];
```

注意：如果您的应用分享的是 iTunes 或 Google Play 商店的链接，我们不会发布您在分享中指定的任何图片或说明，而会发布通过网络爬虫直接从应用商店抓取的一些应用信息，其中可能不包括图像。要预览转至 iTunes 或 Google Play 的链接分享，请在分享调试器中输入您的网址。

照片

用户可以使用分享对话框或自定义界面，通过您的应用将照片分享到 Facebook：

- 照片大小必须小于 12MB
- 用户需要安装版本 7.0 或以上的原生 iOS 版 Facebook 应用

```
- (void)imagePickerController:(UIImagePickerController
*)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info
{
  UIImage *image =
info[UIImagePickerControllerOriginalImage];

  FBSDKSharePhoto *photo = [[FBSDKSharePhoto alloc]
init];
  photo.image = image;
  photo.userGenerated = YES;
  FBSDKSharePhotoContent *content =
[[FBSDKSharePhotoContent alloc] init];
  content.photos = @[photo];
  ...
}
```

视频

应用用户可通过分享对话框或您专属的自定义界面将视频分享到 Facebook：

- 视频大小必须小于 50MB。
- 分享内容的用户应安装版本 26.0 或以上的 iOS 版 Facebook 客户端。

```
 - (void)imagePickerController:(UIImagePickerController
*)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info
{
  FBSDKShareVideo *video = [[FBSDKShareVideo alloc]
init];
  if (@available(iOS 11, *)) {
    video.videoAsset = [info
objectForKey:UIImagePickerControllerPHAsset];
  } else {
    video.videoURL = [info
objectForKey:UIImagePickerControllerReferenceURL];
  }
  FBSDKShareVideoContent *content =
[[FBSDKShareVideoContent alloc] init];
  content.video = video;
  ...
}
```

多媒体

应用用户可通过分享对话框从应用向 Facebook 分享同时包含照片和视频的内容。请注意以下事项：

- 用户使用的 iOS 版本至少应为 7.0。
- 分享内容的用户应安装版本 52.0 或以上的 iOS 版 Facebook 客户端。
- 照片大小必须小于 12MB，视频大小必须小于 50MB。
- 用户最多可以分享 1 个视频加 29 张照片，或最多分享 30 张照片。

```
FBSDKSharePhoto *photo = [FBSDKSharePhoto photoWith...
FBSDKShareVideo *video = [FBSDKShareVideo videoWith...
FBSDKShareMediaContent *content = [FBSDKShareMediaContent
new];
content.media = @[photo, video];
}
```

分享方法

通过构建模型处理内容后，您可以触发“分享”或“消息”对话框。

“分享”按钮

实施“分享”按钮后，用户可将内容分享到自己的 Facebook 时间线、好友的时间线或小组中。“分享”按钮将调用分享对话框。要在视图中添加“分享”按钮，请将下列代码片段添加到您的视图：

```
FBSDKShareButton *button = [[FBSDKShareButton alloc] init];
button.shareContent = content;
[self.view addSubview:button];
```

“发送”按钮

用户可以使用“发送”按钮，以私密方式向好友和使用 Facebook Messenger 的联系人发送照片、视频和链接。“发送”按钮将调用消息对话框。要在视图中添加“发送”按钮，请将下列代码片段添加到您的视图：

```
FBSDKSendButton *button = [[FBSDKSendButton alloc] init];
button.shareContent = content;
[self.view addSubview:button];
```

如果未安装 Messenger 应用，“发送”按钮将隐藏。请确保此按钮隐藏时，您的应用布局是适当的。要检查“发送”按钮能否在当前设备上显示，可使用 `FBSDKSendButton` 属性 `isHidden`：

```
if (button.isHidden) {
  NSLog(@"Is hidden");
} else {
  [self.view addSubview:button];
}
```

分享对话框

要使用 Facebook 构建的分享体验，您需要如以上内容建模部分中所述定义您的内容，然后调用分享对话框。例如，要使用分享对话框分享链接：

```
FBSDKShareLinkContent *content = [[FBSDKShareLinkContent alloc]
init];
content.contentURL = [NSURL
URLWithString:@"http://developers.facebook.com"];
[FBSDKShareDialog showFromViewController:self
                             withContent:content
                                delegate:nil];
```

消息对话框

消息对话框会切换到原生 iOS 版 Messenger 应用，并在发布帖子后将控制权交还给您的应用。

```
[FBSDKMessageDialog showWithContent:content delegate:nil];
```

注意：iPad 当前不支持消息对话框。