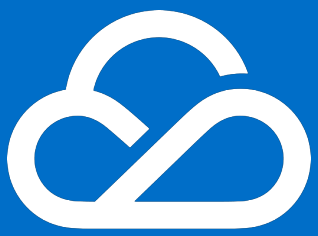
腾讯云手游社交组件

SDK 接入

产品文档



腾讯云

【版权声明】

【商标声明】

【服务声明】

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或模式的承诺或保证。

文档目录

SDK 接入

Android SDK接入指南

开发准备

环境依赖

适用于 JDK1.6 版本以上 。

安装 SDK

以下以 Eclipse 为开发的 IDE 进行范例说明：

1. 创建一个工程，把 open_sdk.jar, mta-sdk.jar 以及 mid-sdk.jar 文件拷贝到 libs（或 lib）目录下。

2. 将 open_sdk.jar, mta-sdk.jar 以及 mid-sdk.jar 加入编译路径中。

   具体的操作步骤为：选中 open_sdk.jar, mta-sdk.jar 以及 mid-sdk.jar，右键菜单中选择【Build Path】>【Add to Build Path】。

3. 配置 AndroidManifest：
   在应用的 AndroidManifest.xml 增加配置的 application 节点下增加以下配置；

   注意：
   不配置将会导致无法调用 API。

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

<application>
 <activity
```

```
 android:name="com.tencent.tauth.AuthActivity"
 android:noHistory="true"
 android:launchMode="singleTask" >
 <intent-filter>
 <action android:name="android.intent.action.VIEW" />
 <category android:name="android.intent.category.DEFAULT" />
 <category android:name="android.intent.category.BROWSABLE" />
 <data android:scheme="tencent你的AppId" />
 </intent-filter>
 </activity>
<application>
```

通过以上步骤，工程就已经配置完成了。接下来就可以在代码里使用 QQ 互联的 SDK 进行开发了。

快速入门

创建实例

Tencent 是 SDK 的功能入口，所有的接口调用都得通过 Tencent 进行调用。因此调用 SDK
首先需要创建一个 Tencent 实例，其代码如下：

```
@Override

public void onCreate(Bundle savedInstanceState){

super.onCreate(savedInstanceState);

setContentView(R.layout.activity_main);

// Tencent类是SDK的主要实现类，开发者可通过Tencent类访问腾讯开放的OpenAPI。

// 其中APP_ID是分配给第三方应用的appid，类型为String。
```

```
mTencent = Tencent.createInstance(APP_ID, this.getApplicationContext());

// 1.4版本:此处需新增参数，传入应用程序的全局context，可通过activity的getApplicationContext方法获取

// 初始化视图

initViews();

}
```

如果你已经添加了 android.permission.INTERNET 和 android.permission.ACCESS_NETWORK_STATE 权限，则无需重复添加。而你的 APPID 则要替换成具体应用的 APPID，例如你的 APPID 是 222222，则 data 标签应该是这样的：

```
<data android:scheme="tencent222222" />
```

实现回调

所有的 SDK 接口调用，都会传入一个回调用以接收 SDK 返回的调用结果。回调的主要接口有两种：

- 实现回调 IUiListener：
  调用 SDK 已经封装好的接口时，例如登录、快速支付登录、应用分享、应用邀请等接口，需传入该回调的实例。IUiListener 的实现示例代码如下：

```
private class BaseUiListener implements IUiListener {

@Override

public void onComplete(JSONObject response) {

mBaseMessageText.setText("onComplete:");
```

```
mMessageText.setText(response.toString());

doComplete(response);

}

protected void doComplete(JSONObject values) {

}

@Override

public void onError(UiError e) {

showResult("onError:", "code:" + e.errorCode + ", msg:"

+ e.errorMessage + ", detail:" + e.errorDetail);

}

@Override

public void onCancel() {

showResult("onCancel", "");

}

}
```

- 实现回调 IRequestListener：
  使用 requestAsync、request 等通用方法调用 SDK
  未封装的接口时，例如上传图片、查看相册等接口，需传入该回调的实例。IRequestListener

的实现示例代码如下：

```
private class BaseApiListener implements IRequestListener {

@Override

public void onComplete(final JSONObject response, Object state) {

showResult("IRequestListener.onComplete:", response.toString());

doComplete(response, state);

}

protected void doComplete(JSONObject response, Object state) {

}

@Override

public void onIOException(final IOException e, Object state) {

showResult("IRequestListener.onIOException:", e.getMessage());

}

@Override

public void onMalformedURLException(final MalformedURLException e,

Object state) {

showResult("IRequestListener.onMalformedURLException", e.toString());
```

```
}

@Override

public void onJSONException(final JSONException e, Object state) {

showResult("IRequestListener.onJSONException:", e.getMessage());

}

@Override

public void onConnectTimeoutException(ConnectTimeoutException arg0,

Object arg1) {

// TODO Auto-generated method stub

}

@Override

public void onSocketTimeoutException(SocketTimeoutException arg0,

Object arg1) {

// TODO Auto-generated method stub

}

//1.4版本中IRequestListener 新增两个异常

@Override
```

```
public void onNetworkUnavailableException(NetworkUnavailableException e, Object state){

// 当前网络不可用时触发此异常

}

@Override

public void onHttpStatusException(HttpStatusException e, Object state) {

// http请求返回码非200时触发此异常

}

public void onUnknowException(Exception e, Object state) {

// 出现未知错误时会触发此异常

}

}
```

应用在调用 SDK 提供的接口时，将实现了对应回调接口的实例传入。当 SDK 的接口调用完成后，具体如登录、应用邀请和应用分享调用完成后，会回调传入的接口实例。

> 注意：
> 应用调用 Andriod_SDK 接口时，如果要成功接收到回调，需要在调用接口的 Activity 的 onActivityResult 方法中增加如下代码：

```
@Override
```

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
 mTencent.onActivityResult(requestCode, resultCode, data);
}
```

iOS SDK 接入指南

开发准备

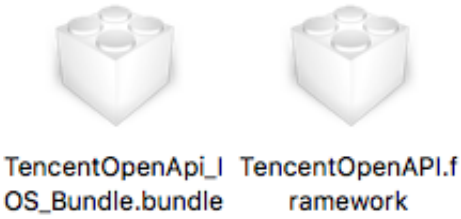
环境依赖

Xcode 集成开发工具

目录结构

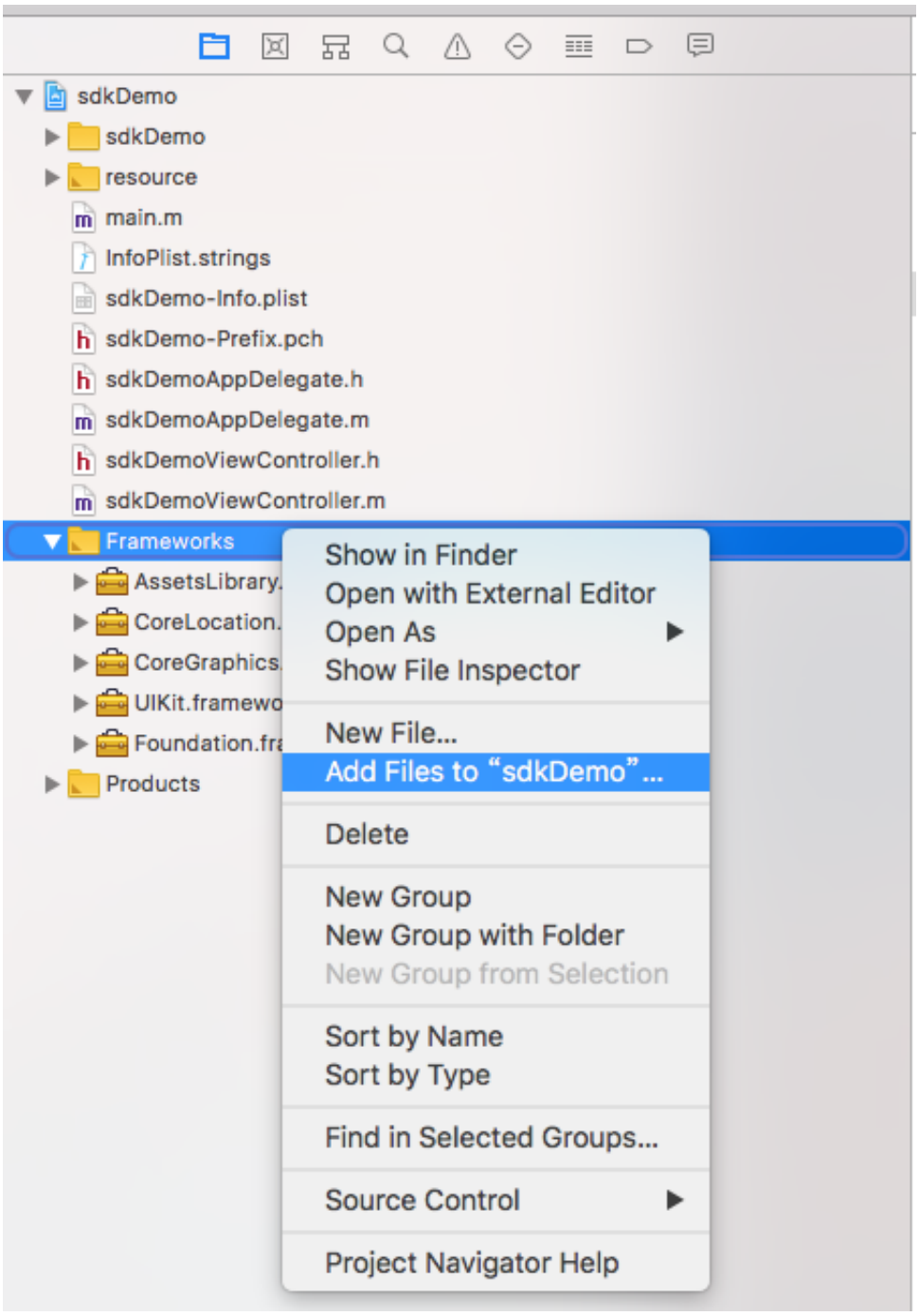
iOS SDK 包中：

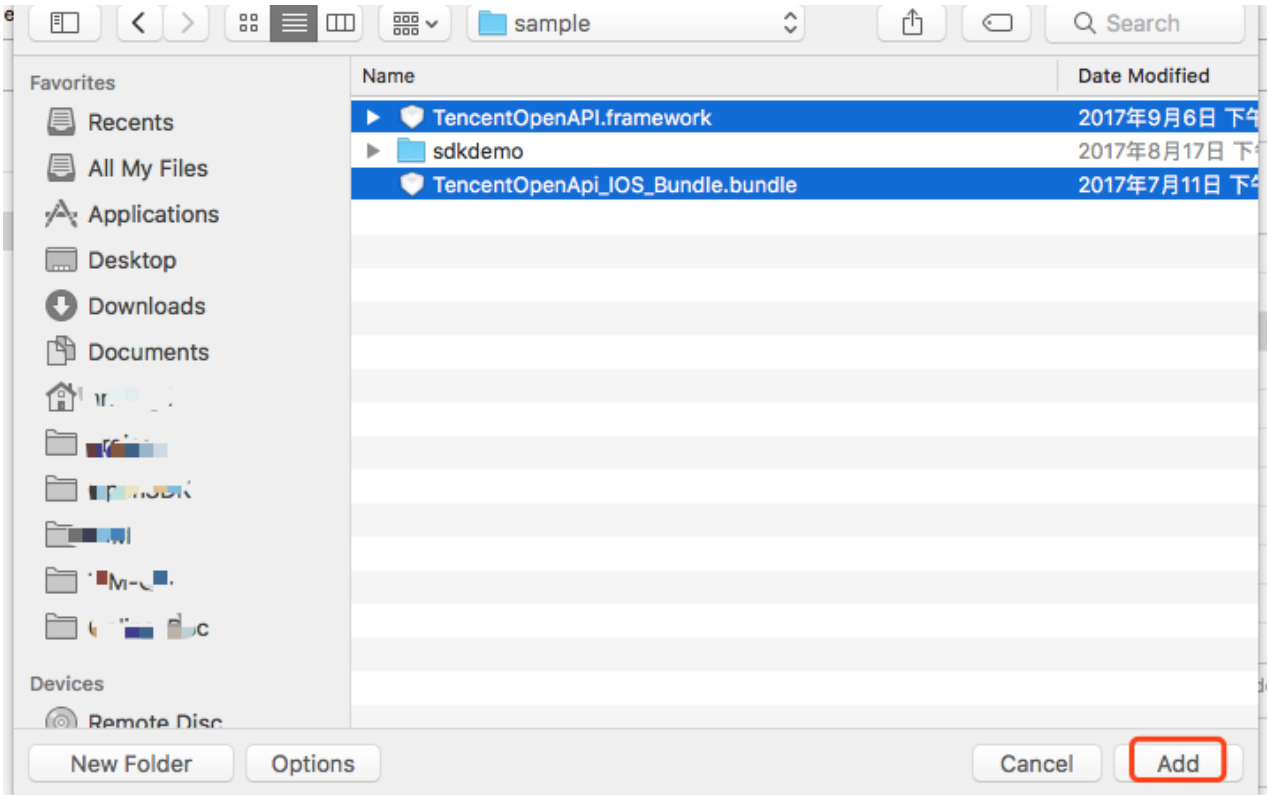
- TencentOpenApi_iOS_Bundle.bundle 打包了 iOS SDK 需要的资源文件。
- TencentOpenAPI.framework 打包了 iOS SDK 的头文件定义和具体实现。



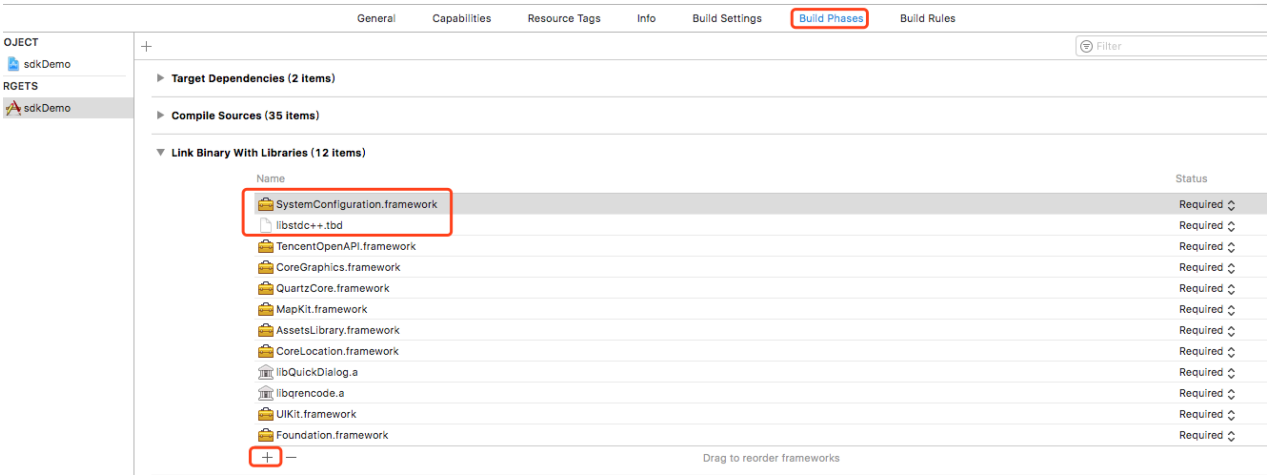
安装SDK

1. 将 iOS SDK 中的 TencentOpenAPI.framework 和 TencentOpenApi_IOS_Bundle.bundle 两个文件拷贝到应用开发的目录下，然后按下图所示添加到工程中：

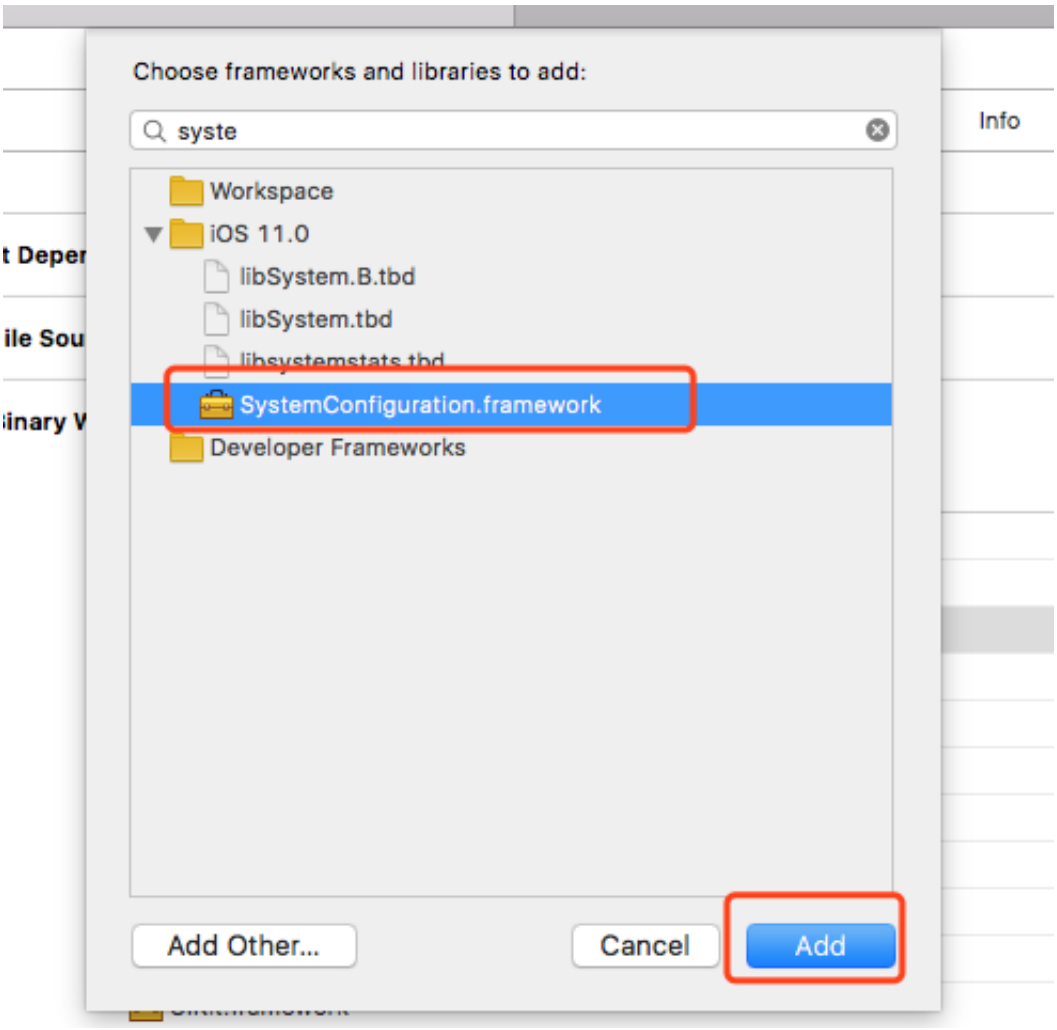

sdkDemo
sdkDemo
resource
main.m
InfoPlist.strings
sdkDemo-Info.plist
sdkDemo-Prefix.pch
sdkDemoAppDelegate.h
sdkDemoAppDelegate.m
sdkDemoViewController.h
sdkDemoViewController.m
Frameworks
AssetsLibrary.
CoreLocation.
CoreGraphics.
UIKit.framewo
Foundation.fra
Products
Show in Finder
Open with External Editor
Open As
Show File Inspector
New File...
Add Files to “sdkDemo”...
Delete
New Group
New Group with Folder
New Group from Selection
Sort by Name
Sort by Type
Find in Selected Groups...
Source Control
Project Navigator Help



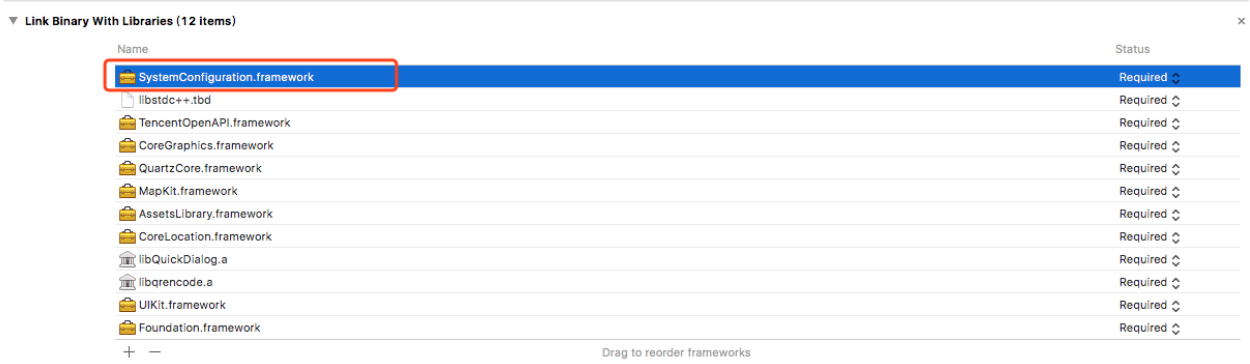
2. 添加 SDK 依赖的系统库文件 SystemConfiguration.framework 和 libstdc++.tbd。然后在 Xcode 中打开工程配置文件，添加依赖库。如图所示：



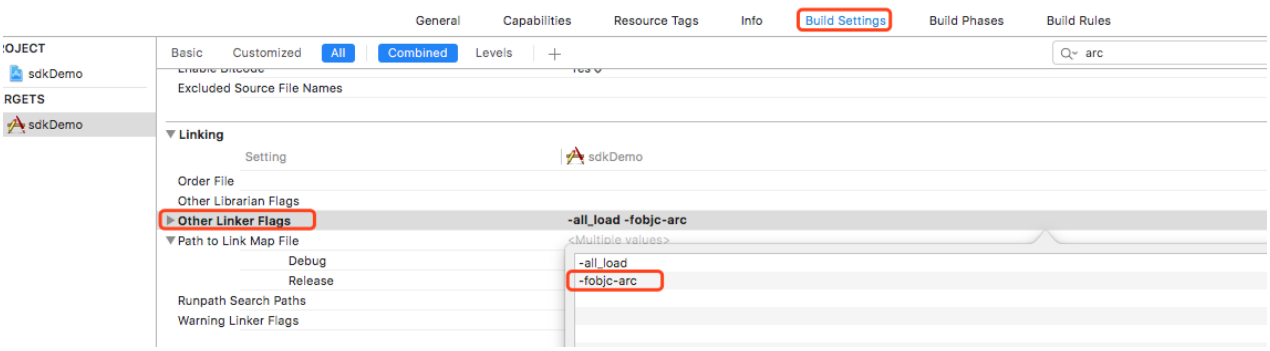
3. 直接在默认库文件中选择后单击【Add】，下图以添加 SystemConfiguration.framework 为例：



4. 返回后看到 SystemConfiguration.framework 已经在 Linked Frameworks and Libraries 中出现。如图所示：

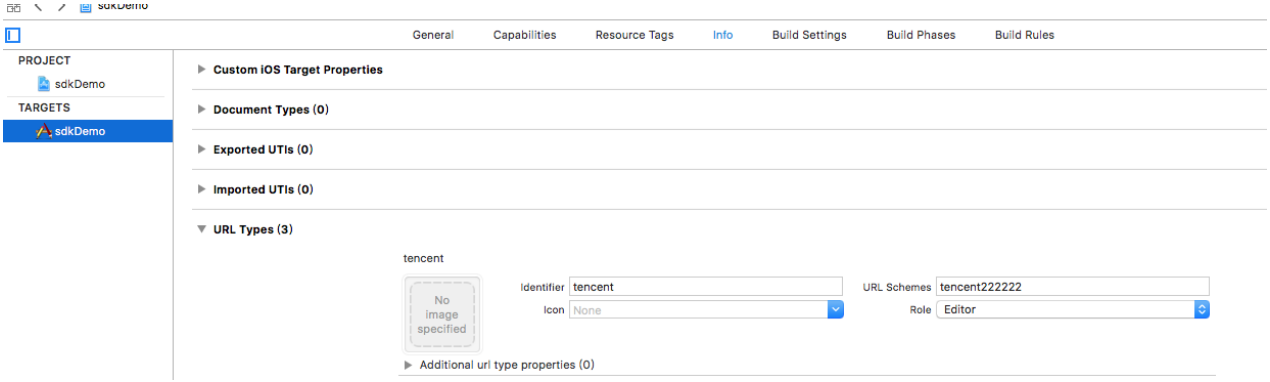


5. 修改必要的工程配置属性：在工程配置中的【Build Settings】一栏中找到 Linking 配置区，给【Other Linker Flags】配置项添加属性值 -fobjc-arc。



修改必要的代码

- 修改工程配置文件:
  在 XCode 中选择你的工程设置项：选中【TARGETS】一栏，在【info】>【URL type】添加一条新的 URL scheme，新的 scheme 是 tencent 和 APPID 的组合。如果您使用的是 XCode3 或者更低的版本，则需要在 plist 文件中添加。Demo 中我们使用的注册 APPID 是 222222。如下图所示：



- 重写 AppDelegate 的 handleOpenURL 和 openURL 方法:
  openURL:

```
-(BOOL)application:(UIApplication *)application openURL:(NSURL *)url
sourceApplication:(NSString *)sourceApplication annotation:(id)annotation{
return [TencentOAuth HandleOpenURL:url];
}
```

  handleOpenURL:

```
-(BOOL)application:(UIApplication *)application handleOpenURL:(NSURL *)url{
return [TencentOAuth HandleOpenURL:url];
```

}

- 在代码中实现 TencentSessionDelegate 协议中的方法:
  具体协议可以参照 TencentOpenAPI.framework /Headers 中的 TencentOAuth.h 文件。
- 初始化 iOS SDK API 数据对象 TencentOAuth：
  1. 创建 TencentOAuth 并初始化其 appid，demo 为 222222。delegate 为实现 TencentSessionDelegate 的对象：

     ```
     //这里delegate不能为空
     _tencentOAuth = [[TencentOAuth alloc] initWithAppId:@"222222", andDelegate:self];
     ```

  2. 初始化 redirectURI（这里需要填写注册 APP 时填写的域名。默认可以不用填写。建议不用填写。demo 可以在这里注册：[demo 注册地址]

     ```
     _tencentOAuth.redirectURI = @"www.qq.com";
     ```

  3. 设置应用需要用户授权的 API 列表。 (建议如果授权过多的话，可能会造成用户不愿意授权。这里最好只授权应用需要用户赋予的授权。)：

     ```
     _permissions = [[NSArray arrayWithObjects:@"get_user_info", @"get_simple_userinfo", @"add_t", nil] retain];
     ```

快速入门

登录时，调用 TencetnOAuth 对象的 authorize 方法：

```
[_tencentOAuth authorize:_permissions inSafari:NO]；
```

登录完成后，会调用 TencentSessionDelegate 中关于登录的协议方法。

[_tencentOAuth authorize:_permissions inSafari:NO]；

登录成功：

```
@protocol TencentSessionDelegate <NSObject>
- (void)tencentDidLogin
{
  _labelTitle.text = @"登录完成";

  if (_tencentOAuth.accessToken && 0 != [_tencentOAuth.accessToken length])
{
  // 记录登录用户的OpenID、Token以及过期时间
  _labelAccessToken.text = _tencentOAuth.accessToken;
  }
  else
  {
  _labelAccessToken.text = @"登录不成功 没有获取accesstoken";
  }
}
```

非网络错误导致登录失败：

```
@protocol TencentSessionDelegate <NSObject>
-(void)tencentDidNotLogin:(BOOL)cancelled
{
if (cancelled)
{
_labelTitle.text = @"用户取消登录";
  }
  else
```

```
{
  _labelTitle.text = @"登录失败";
  }
}
```

网络错误导致登录失败：

```
@protocol TencentSessionDelegate <NSObject>
-(void)tencentDidNotNetWork
{
  _labelTitle.text=@"无网络连接，请设置网络";
}
```

登录成功后，即可获取到 access token 和 openid。accessToken 和 openid 保存在 TencentOAuth 对象中。可以通过相应的属性方法直接获得。

```
[_tencentOAuth accessToken] ;
[_tencentOAuth openId] ;
```

注意：

- 由于登录是异步过程，这里可能会由于用户的行为导致整个登录的的流程无法正常走完，即有可能由于用户行为导致登录完成后不会有任何登录回调被调用。开发者在使用 SDK 进行开发的时候需要考虑到这点，防止由于一直在同步等待登录的回调而造成应用的卡死，建议在登录的时候将这个实现做成一个异步过程。
- 获取到的 access token 具有三个月有效期，过期后提示用户重新登录授权。
- 第三方应用可存储 access token 信息，以便后续调用 OpenAPI
  访问和修改用户信息时使用。如果需要保存授权信息，需要保存登录完成后返回的 accessToken，openid 和 expirationDate
  三个数据，下次登录的时候直接将这三个数据是设置到 TencentOAuth 对象中即可。
  获得：

  ```
  [_tencentOAuth accessToken] ;
  ```

```
[_tencentOAuth openId] ;
[_tencentOAuth expirationDate] ;
```

设置：

```
[_tencentOAuth setAccessToken:accessToken] ;
[_tencentOAuth setOpenId:openId] ;
[_tencentOAuth setExpirationDate:expirationDate]
```

- 建议应用在用户登录后，即调用 getUserInfo
  接口获得该用户的头像、昵称并显示在界面上，使用户体验统一。
- iOS SDK 支持应用跳转到手机 QQ 进行登录，给用户提供更加安全、快捷的体验
  。如果用户没有安装手机 QQ，且开发者具有 webview
  权限，则显示登录页；如果开发者没有 webview 权限，SDK
  版本高于2.9，则显示登录页，SDK 低于2.9，则显示下载页。由于跳转至下载页在当前苹果
  APP 审核有被拒风险，所以，希望开发者尽快升级是用最新版 SDK。