

iOS 平台 SDK 文档

编号: WJOAUTH_IOS_SDK

版本: WJOAUTH _IOS_SDK V1.0.0

修订记录:

时间	文档版本	修订人	备注
2018/05/14	1.0.0	周美丽	初稿

目录

一、SDK 接入设置	3
1、设置工程回调 URL Scheme	3
2、添加 SDK 文件到工程	3
3、在工程中引入静态库之后，需要在编译时添加-objc 编译选项	4
4、添加 Framework 文件到工程	4
5、定义应用 OAuth2.0 认证所需的几个常量	4
6、注册 appkey(clientid)	4
7、重写 AppDelegate 的 handleOpenURL 和 openURL 方法	5
二、应用场景代码示例	5
1、oauth2.0 授权认证	5
2、从第三方应用向 WJOAuthSDK 发送请求	5
3、第三方应用收到 WJOAuthSDK 回调	6


WJOAuthSDK

一、SDK 接入设置

1、设置工程回调 URL Scheme

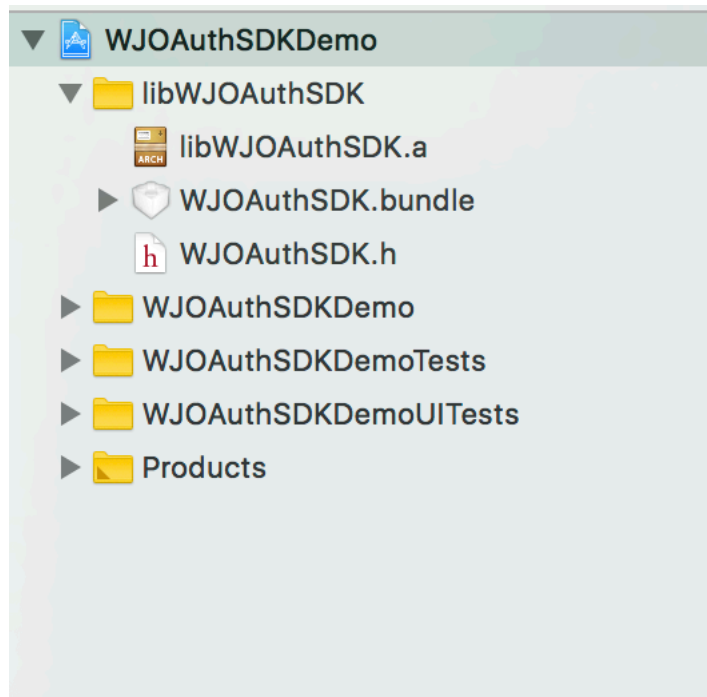
修改 info.plist 文件 URL types 项为自己的授权登录回调地址, ” wjOAuth [你的应用程序的 Appkey] ” , 例如: wjOAuthSampleClientId

▼ URL Types (1)

com.wjOAuth	Identifier <input type="text" value="com.wjOAuth"/>	URL Schemes <input type="text" value="wjOAuthSampleClientId"/>
	Icon <input type="text" value="None"/>	Role <input type="text" value="Editor"/>
▶ Additional url type properties (0)		
+		

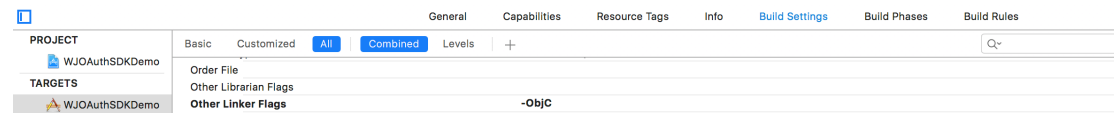
2、添加 SDK 文件到工程

将从 GitHub 上下载的 libWJOAuthSDK 文件夹添加至工程, 其中包含 WJOAuthSDK.h 文件以及 libWJOAuthSDK.a 和 WJOAuthSDK.bundle, 统共 3 个文件。



3、在工程中引入静态库之后，需要在编译时添加-objc 编译选项

避免静态库中类加载不全造成程序崩溃。方法：程序 Target->Build Settings->Linking 下 Other Linker Flags 项添加-ObjC。



4、添加 Framework 文件到工程

在工程中修改 Other Linker Flags 后，需要修改编译步骤的链接库设置，避免链接阶段由于库的设置错误导致程序崩溃。方法：程序 Target->Build Phases->Link Binary With Libraries 下添加以下 Framework 至工程中。需要添加的 Frameworks 为：SystemConfiguration.framework。

5、定义应用 OAuth2.0 认证所需的几个常量

AppKey: 第三方应用申请的 appkey, 用来身份鉴证、显示来源等; AppRedirectURL: 应用回调页, 在进行 OAuth2.0 登录认证时所用。对于 Mobile 客户端应用来说, 是不存在 Server 的, 故此处的应用回调页地址只要与应用回调页中的 url 地址保持一致就可以了, 如图所示:

```
#define kAppKey          @"3573952935"
#define kRedirectURI     @"http://your_callback_uri"
```

6、注册 appkey(clientid)

程序启动时, 在代码中向 WJOAuthSDK 注册你的 Appkey, 可设置授权界面语言, 默认跟随系统。

```
-(BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    // Override point for customization after application launch.

    [WJOAuthSDK setLanguageType:WJLanguageTypeChinese];

    [WJOAuthSDK registerApp:kAppKey];

    return YES;
}
```

7、重写 AppDelegate 的 handleOpenURL 和 openURL 方法

```
- (BOOL)application:(UIApplication *)app openURL:(NSURL *)url options:(NSDictionary<UIApplicationOpenURLOptionsKey,id> *)options
{
    return [WJOAuthSDK handleOpenURL:url delegate:self];
}

- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url sourceApplication:(NSString *)sourceApplication annotation:(id)annotation
{
    return [WJOAuthSDK handleOpenURL:url delegate:self];
}

- (BOOL)application:(UIApplication *)application handleOpenURL:(NSURL *)url
{
    return [WJOAuthSDK handleOpenURL:url delegate:self];
}
```

二、应用场景代码示例

1、oauth2.0 授权认证

```
{
    WJOAuthRequest *request = [WJOAuthRequest request];
    request.state = @"Verification";
    request.scope = @"user_info";
    request.redirectURI = kRedirectURI;
    [WJOAuthSDK sendRequest:request];
}
```

调用sendRequest的方法后会跳转到WJOAuthSDK。如果当前WJOAuthSDK没有账号,则进入登录界面;如果当前WJOAuthSDK已经有账户,则进入授权登录界面,选择要向第三方授权的账户。当授权完成后会回调给第三方应用程序,第三方实现WJOAuthSDKDelegate 的 didReceiveWJOAuthResponse: responseStatusCode: 方式监听此次请求的 response。

此中 state 内容为用户自定义(可不填写),WJOAuthSDK 回调 Response 中会通过 requestState 包含原 request.state 中的所有数据,用于数据传输过程中校验相关的上下文环境数据;scope 内容参照开放平台权限列表,可不填写。

2、从第三方应用向 WJOAuthSDK 发送请求

代码示例如:

```
{
    WJOAuthRequest *request = [WJOAuthRequest request];
    request.state = @"Verification";
    request.scope = @"user_info";
    request.redirectURI = kRedirectURI;
    [WJOAuthSDK sendRequest:request];
}
```

同上此中 state 内容为用户自定义(可不填写), WJOAuthSDK 回调 Response 中会通过 requestState 包含原 request.state 中的所有数据, 用于数据传输过程中校验相关的上下文环境数据; scope 内容参照开放平台权限列表, 可不填写。

3、第三方应用收到 WJOAuthSDK 回调

```
- (void)didReceiveWJOAuthResponse:(WJBaseResponse *)response responseStatusCode:(WJOAuthSDKResponseStatusCode)responseStatusCode {  
    if (responseStatusCode == WJOAuthSDKResponseStatusCodeSuccess) {  
        WJOAuthSucceededResponse *authorizeResponse = (WJOAuthSucceededResponse *)response;  
        NSString *requestState = authorizeResponse.requestState;  
        NSString *accessToken = authorizeResponse.accessToken;  
        NSString *message = [NSString stringWithFormat:@"response.requestState:%@\nresponse.accessToken:%@", requestState, accessToken];  
        UIAlertController *alertController = [UIAlertController alertControllerWithTitle:@"授权成功" message:message  
                                           preferredStyle:UIAlertControllerStyleAlert];  
        UIAlertAction *actionCancel = [UIAlertAction actionWithTitle:@"确定"  
                                                                    style:UIAlertActionStyleCancel  
                                                                    handler:nil];  
        [alertController addAction:actionCancel];  
        [self.window.rootViewController presentViewController:alertController animated:YES completion:nil];  
    }  
    else if (responseStatusCode == WJOAuthSDKResponseStatusCodeAuthDeny) {  
        WJOAuthFailedResponse *authorizeResponse = (WJOAuthFailedResponse *)response;  
        NSInteger *errorCode = authorizeResponse.errorCode;  
        NSString *errorCodeDescription = authorizeResponse.errorCodeDescription;  
        NSString *message = [NSString stringWithFormat:@"response.errorCode:%@\nresponse.errorCodeDescription:%@", errorCode, errorCodeDescription];  
        UIAlertController *alertController = [UIAlertController alertControllerWithTitle:@"授权失败" message:message  
                                           preferredStyle:UIAlertControllerStyleAlert];  
        UIAlertAction *actionCancel = [UIAlertAction actionWithTitle:@"确定"  
                                                                    style:UIAlertActionStyleCancel  
                                                                    handler:nil];  
        [alertController addAction:actionCancel];  
        [self.window.rootViewController presentViewController:alertController animated:YES completion:nil];  
    }  
    else if (responseStatusCode == WJOAuthSDKResponseStatusCodeUserCancel) {  
        UIAlertController *alertController = [UIAlertController alertControllerWithTitle:@"用户点击取消" message:nil  
                                           preferredStyle:UIAlertControllerStyleAlert];  
        UIAlertAction *actionCancel = [UIAlertAction actionWithTitle:@"确定"  
                                                                    style:UIAlertActionStyleCancel  
                                                                    handler:nil];  
        [alertController addAction:actionCancel];  
        [self.window.rootViewController presentViewController:alertController animated:YES completion:nil];  
    }  
}
```

WJOAuthSDK 回调 responseStatusCode 为

WJOAuthSDKResponseStatusCodeSuccess (授权成功)、

WJOAuthSDKResponseStatusCodeUserCancel (用户取消)、

WJOAuthSDKResponseStatusCodeAuthDeny (授权失败)、的枚举, 可根据对应枚举做相关的数据处理。

当 responseStatusCode 返回为 WJOAuthSDKResponseStatusCodeSuccess 时, 回调的 response 中包含 accessToken、requestState。accessToken 为访问相关信息的认证口令; requestState 包含原 request.state 中的所有数据, 用于数据传输过程中校验相关的上下文环境数据。

当 responseStatusCode 返回为 WJOAuthSDKResponseStatusCodeAuthDeny 时, 回调 Response 中包含 errorCode、errorCodeDescription。授权失败时 WJOAuthSDK 会通过 errorCode 返回相关响应状态码, 通过 errorCodeDescription 返回状态码描述; 具体状态码说明, 请访问开发文档->OAuth2.0->开发相关资源->返回状态码说明。

当 responseStatusCode 返回为 WJOAuthSDKResponseStatusCodeUserCancel 时, 回调 Response 中不包含相关信息。

具体调用示例参见demo