移动开发平台 MobileLine

附录

产品文档

腾讯云

文档目录

附录
iOS TACSocialQQ 快速入门

最近更新时间：2018-08-27 17:24:45

TACSocialQQ 封装了 TencentOpenAPI QQ 互联的 SDK，您可以通过引入 TACSocialQQ 来引入 QQ 互联能力。同时，TACSocialQQ 提供了更方便的改 SDK 集成方式，请不要重复引入 TencentOpenAPI 。

准备工作

在开始使用移动开发平台（MobileLine） TACSocialQQ 之前，您需要：

1. 一个启用了移动开发平台（MobileLine）的应用。
2. 您集成了 TACCore。

将移动开发平台（MobileLine） TACSocialQQ 代码库添加到您的 Xcode 项目中

1. 在您的项目中集成移动开发平台（MobileLine） SDK，并在您的 Podfile 文件中添加移动开发平台（MobileLine）的私有源：

```
source "https://git.cloud.tencent.com/qcloud_u/cocopoads-repo"
source "https://github.com/CocoaPods/Specs"
```

> **注意：**
> 一定要添加 CocoaPods 的原始源，否则会造成部分仓库找不到的问题。

2. 添加 TACSocialQQ 到您的 Podfile，您可以按照以下方法在 Podfile 中纳入一个 Pod：

```
pod 'TACSocialQQ'
```

3. 安装 Pod 并打开 .xcworkspace 文件以便在 Xcode 中查看该项目：

```
$ pod install
$ open your-project.xcworkspace
```

4. 在 UIApplicationDelegate 子类中导入 TACSocialQQ 模块：

Objective-C 代码示例：

```
import <TACSocialQQ/TACSocialQQ.h>
```

Swift 代码示例：

```
import TACSocialQQ
```

5. 配置 TACApplication 共享实例，通常是在 `application:didFinishLaunchingWithOptions:` 方法中配置：

先行配置--引入配置文件

我们使用腾讯云 iOS SDK 统一配置机制。只要您加入 TACSocialQQ 的配置文件，我们会自动化初始化相关的配置和参数。

> **注意：**
> 所有的配置文件（plist）文件都以 `tac_services_configurations` 开始，以扩展名 plist 结束。 我们会加载所有符合正则表达式 `tac_services_configurations*.plist` 的文件，并解析合并。解析顺序为 ASCII 排序，也就是说ASCII排序较后的配置文件的参数将会优先生效。

您需要在您的工程中创建以 `tac_services_configurations` 为前缀的 plist 文件，例如：

```
tac_services_configurations_qq.plist
```

文件的内容为需要配置的参数信息，请注意 QQOptions 的配置路径 （:services:social:qq）：例如配置文件：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>services</key>
<dict>
<key>social</key>
<dict>
<key>qq</key>
<dict>
```

```
<key>appId</key>
<string>请填充您的 AppId</string>
<key>appKey</key>
<string>请填充您的 APPKey</string>
<key>permissions</key>
<array>
<string>get_user_info</string>
<string>get_simple_userinfo</string>
<string>add_t</string>
</array>
</dict>
</dict>
</dict>
</dict>
</plist>
```

目前支持的配置为：

参数Key	参数含义	
appId	QQ 互联中程序的 appID	
appKey	QQ 互联中程序的 appKey	
permissions	进行 QQ 授权的时候，需要申请的权限列表	

程序配置

一般情况下您使用默认配置就可以了，用以下代码使用默认配置启动 Crash 服务。如果您在引入其它模块的时候，调用了该方法，请不要重复调用。

Objective-C 代码示例：

```
[TACApplication configurate];
```

Swift 代码示例：

```
TACApplication.configurate();
```

如果您需要进行自定义的配置，则可以使用以下方法，我们使用了 Objective-C 的语法特性 Category 和一些 Runtime 的技巧保障了，只有在您引入了 TACSocialQQ 模块的时候，才能从 TACApplicaitonOptiosn 里面看到其对应的配置属性，如果你没有引入 TACSocialQQ 模块这些属性就不存在，请不要在没有引入 TACSocialQQ 模块的时候使用这些配置，这将会导致您编译不通过：

Objective-C 代码示例：

```
TACApplicationOptions* options = [TACApplicationOptions defaultApplicationOptions];
// 自定义配置
// options.qqOptions.[Key] = [Value];
[TACApplication configurateWithOptions:options];
```

Swift 代码示例：

```
let options = TACApplicationOptions.default()
// 自定义配置
// options?.qqOptions.[Key] = [Value];
TACApplication.configurate(with: options);
```

配置 TACSocialQQ 中的配置脚本 (主要为第三方登录模块的配置脚本)

为了配合 TencentOpenApi 的使用，需要 Info.plist 里面注册回调 scheme 和 query scheme。为了方便您快速集成，和减少集成过程中的挫折。我们使用了自动化的技术来执行上报的操作。请确保根据： TACCore 集成指南 中的脚本配置章节正确配置了运行脚本，尤其是构建之前运行脚本。

TACSocialQQ 中的脚本会自动的帮助您完成以下功能：

1. 根据读取您的 tac_services_configurations_qq 中的 appId 信息按照 `tencent[appId]` 的规则增加回调的 scheme。
2. 在 LSApplicationQueriesSchemes 中添加 mqqopensdkapiV2。

6. 使用 TencentOpenApi 的功能。

我们已经为您自动化配置好了 TencentOpenApi 的其他功能，包括 HandleOpenURL 等函数的响应，和在 Info.plist 文件中注册相关的回调和 Scheme 等操作，您不需要重复执行该操作。如果您要使用 TencentOpenApi 的功能，您可以引入头文件：

Objective-C 代码示例：

```
#import <TACSocialQQ/TencentOAuth.h>
```

Swift 代码示例：

```
import TACSocialQQ.TencentOAuth
```

我们其进行了初始化处理，并生成了一个 TencentOAuth 的对象，存储在 TACSociallQQService 中如果您要手动使用其相关的功能，请先配置其响应的 delegate：

Objective-C 代码示例：

```
[[TACSocialQQService defaultService].tencentOAuthDelegate addDelegate:delegate]
```

Swift 代码示例：

```
TACSocialQQService.default().tencentOAuthDelegate.addDelegate(delegate)
```

其中 delegate 为 TencentOAuth 对象的 delegate，这里我们对原始的 delegate 进行了转发。您可以注册多个 delegate，请在不使用的时候移除：

Objective-C 代码示例：

```
[[TACSocialQQService defaultService].tencentOAuthDelegate removeDelegate:delegate]
```

Swift 代码示例：

```
TACSocialQQService.default().tencentOAuthDelegate.removeDelegate(delegate)
```

iOS TACSocialWechat 快速入门

最近更新时间：2018-08-27 17:26:35

TACSocialWechat 封装了 libWeChatSDK 微信开放平台的 SDK，您可以通过引入 TACSocialWechat 来引入微信开放平台能力。同时，TACSocialWechat 提供了更方便的该 SDK 集成方式，请不要重复引入 libWeChatSDK 。

准备工作

在开始使用移动开发平台（MobileLine） TACSocialWechat 之前，您需要：

1. 一个启用了移动开发平台（MobileLine）的应用。
2. 您集成了 TACCore。

将移动开发平台（MobileLine） TACSocialWechat 代码库添加到 Xcode 项目中

1. 在您的项目中集成移动开发平台（MobileLine） SDK，并在 Podfile 文件中添加移动开发平台（MobileLine）的私有源：

```
source "https://git.cloud.tencent.com/qcloud_u/cocopoads-repo"
source "https://github.com/CocoaPods/Specs"
```

> **注意：**
> 一定要添加 CocoaPods 的原始源，否则会造成部分仓库找不到的问题。

2. 添加 TACSocialWechat 到您的 Podfile，您可以按照以下方法在 Podfile 中纳入一个 Pod：

```
pod 'TACSocialWechat'
```

3. 安装 Pod 并打开 .xcworkspace 文件以便在 Xcode 中查看该项目：

```
$ pod install
$ open your-project.xcworkspace
```

4. 在 UIApplicationDelegate 子类中导入 TACSocialWechat 模块：

Objective-C 代码示例：

```
#import <TACSocialWechat/TACSocialWechat.h>
```

Swift 代码示例：

```
import TACSocialWechat
```

5. 配置 TACApplication 共享实例，通常是在 `application:didFinishLaunchingWithOptions:` 方法中配置：

先行配置--引入配置文件

我们使用腾讯云 iOS SDK 统一配置机制,只要您加入 TACSocialWechat 的配置文件，我们会自动化初始化相关的配置和参数。

> **注意：**
> 所有的配置文件（plist）文件都以 `tac_services_configurations` 开始，以扩展名 plist 结束。 我们会加载所有符合正则表达式 `tac_services_configurations*.plist` 的文件，并解析合并。解析顺序为 ASCII 排序，也就是说 ASCII 排序较后的配置文件的参数将会优先生效。

您需要在您的工程中创建以 `tac_services_configurations` 为前缀的 plist 文件，例如：

tac_services_configurations_wechat.plist

文件的内容为需要配置的参数信息，请注意 WechatOptions 的配置路径（:services:social:qq）: 例如配置文件：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>services</key>
<dict>
<key>social</key>
<dict>
<key>wechat</key>
<dict>
<key>appId</key>
```

```
<string>wx256642f480c15e3e</string>
</dict>
</dict>
</dict>
</dict>
</plist>
```

目前支持的配置为：

参数 Key	参数含义
appId	微信开放平台中程序 appId

> **注意：**
> 目前不支持通过配置文件将 appKey 直接配置，因为这是个危险的操作。

程序配置

一般情况下您使用默认配置就可以了，用以下代码使用默认配置启动 Crash 服务。如果您在引入其它模块的时候，调用了该方法，请不要重复调用。

Objective-C 代码示例：

```
[TACApplication configurate];
```

Swift 代码示例：

```
TACApplication.configurate()
```

如果您需要进行自定义的配置，则可以使用以下方法，我们使用了 Objective-C 的语法特性 Category 和一些 Runtime 的技巧保障了，只有在您引入了 TACSocialWechat 模块的时候，才能从 TACApplicaitonOptiosn 里面看到其对应的配置属性，如果你没有引入 TACSocialWechat 模块这些属性就不存在，请不要在没有引入 TACSocialWechat 模块的时候使用这些配置，这将会导致您编译不通过：

> **注意：**
> 如果您希望使用我们提供的在终端获取微信用户信息的能力，您需要在这里配置您的 appKey。请注意这是个危险的操作，我们仍然建议您在自己的后台去获取微信用户的信息。

Objective-C 代码示例：

```
TACApplicationOptions* options = [TACApplicationOptions defaultApplicationOptions];
// 自定义配置
//options.wechatOptions.[Key] = [Value];
options.wechatOptions.appKey = @"您的appKey";
[TACApplication configurateWithOptions:options];
```

Swift 代码示例：

```
let options = TACApplicationOptions.default()
// 自定义配置
// options?.wechatOptions.[Key] = [Value]
TACApplication.configurate(with: options)
```

配置 TACSocialWechat 中的配置脚本 (主要为第三方登录模块的配置脚本)

为了配合 libWeChatSDK 的使用，需要 Info.plist 里面注册回调 scheme 和 query scheme。为了方便您快速集成，和减少集成过程中的挫折。我们使用了自动化的技术来执行上报的操作。请确保根据： TACCore 集成指南 中的脚本配置章节正确配置了运行脚本，尤其是构建之前运行脚本。

TACSocialWechat 中的脚本会自动的帮助您完成以下功能：

1. 根据读取您的 tac_services_configurations_wechat 中的 appId 信息按照 `[appId]` 的规则增加回调的 scheme。
2. 在 LSApplicationQueriesSchemes 中添加 weixin。

6. 使用 libWeChatSDK 的功能：

我们已经为您自动化配置好了 libWeChatSDK 的其他功能，包括 HandleOpenURL 等函数的响应，和在 Info.plist 文件中注册相关的回调和 Scheme 等操作，您不需要重复执行该操作。如果您要使用 libWeChatSDK 的功能，您可以引入头文件：

Objective-C 代码示例：

```
#import <TACSocialWechat/WXApi.h>
```

Swift 代码示例：

```
import TACSocialWechat.WXApi
```

我们其进行了初始化处理，并配置好了其初始化需要的相关程序功能，请先配置其响应的 delegate：

Objective-C 代码示例：

```
[[TACSocialWechatService shareService].delegateProxy addDelegate:delegate];
```

Swift 代码示例：

```
TACSocialWechatService.share().delegateProxy.addDelegate(dlelegate)
```

其中 delegate 为 TencentOAuth 对象的 delegate，这里我们对原始的 delegate 进行了转发。您可以注册多个 delegate，请在不使用的时候移除：

Objective-C 代码示例：

```
[[TACSocialWechatService shareService].delegateProxy removeDelegate:delegate];
```

Swift 代码示例：

```
TACSocialWechatService.share().delegateProxy.removeDelegate(delegate)
```

Android SDK 1.1+ 集成方式

最近更新时间：2018-07-10 15:42:29

准备工作

您首先需要一个 Android 工程，这个工程可以是您现有的工程，也可以是您新建的一个空的工程。

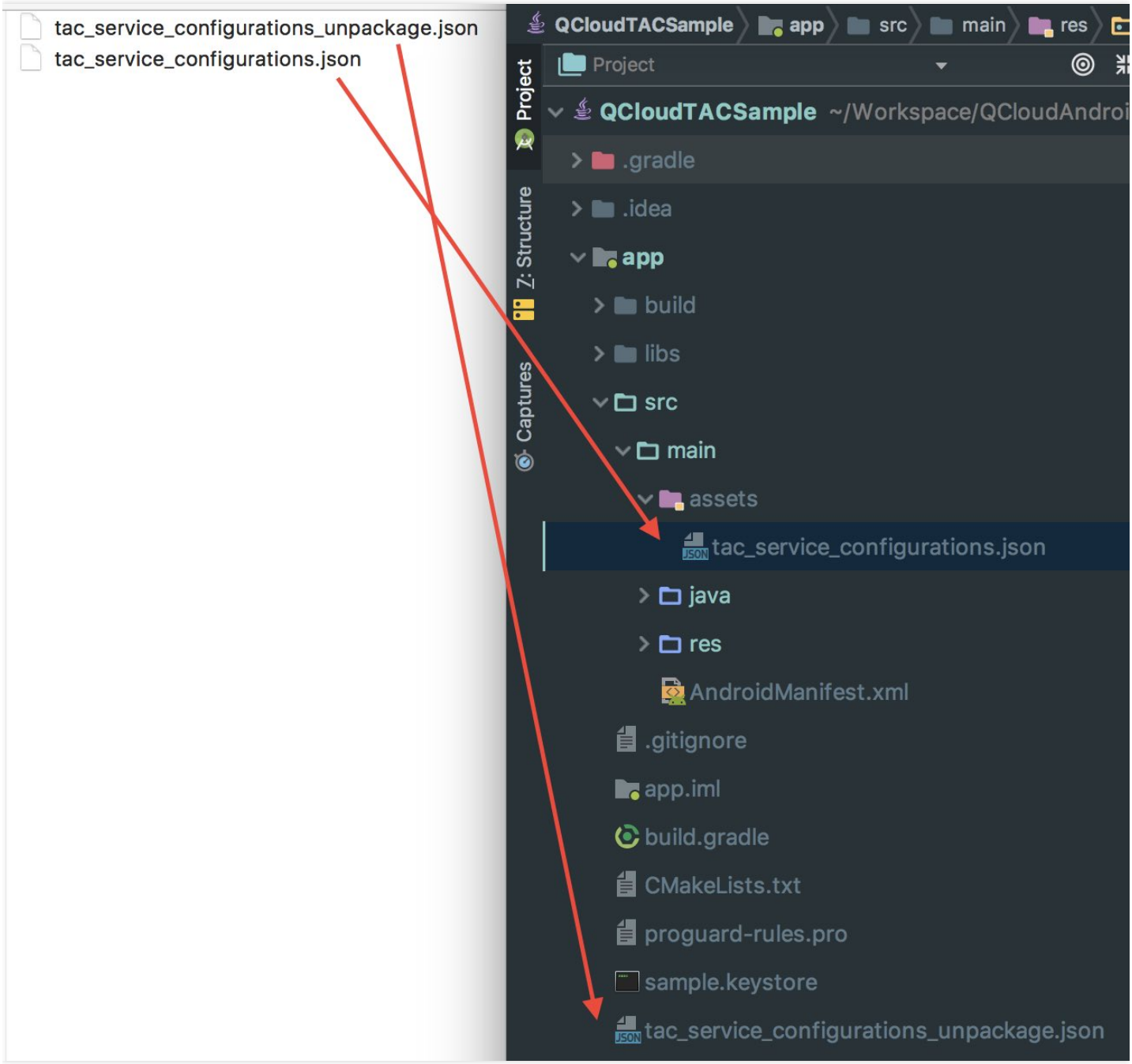
第一步：创建项目和应用（已完成请跳过）

在使用我们的服务前，您必须先在 MobileLine 控制台上 创建项目和应用。

第二步：添加配置文件

在您创建好的应用上单击【下载配置】按钮来下载该应用的配置文件的压缩包：

解压该压缩包，您会得到 tac_service_configurations.json 和 tac_service_configurations_unpackage.json 两个文件，请您如图所示添加到您自己的工程中去。



注意：
请您按照图示来添加配置文件，`tac_service_configurations_unpackage.json` 文件中包含了敏感信息，请不要打包到 apk 文件中，MobileLine SDK 也会对此进行检查，防止由于您误打包造成的敏感信息泄露。

集成 SDK

使用 jcenter 作为仓库来源

在工程根目录下的 build.gradle 使用 jcenter 作为远程仓库：

```
buildscript {
repositories {
jcenter()
}
}

allprojects {
repositories {
jcenter()
}
}
```

添加 Android SDK 库依赖

在您的应用级 build.gradle（通常是 app/build.gradle）添加所需要的 SDK 的依赖，例如核心库：

```
dependencies {
//增加这行
compile 'com.tencent.tac:tac-core:1.2.+'
}
```

然后，单击 IDE 的 gradle 同步按钮，会自动将依赖包同步到本地。