# NAVER Cafe SDK 的使用说明

NAVER

# 著作权

# 文件信息

**文件概要**

该文件是说明 NAVER Cafe SDK 适用于应用程序的方法。

**读者**

该文件的读者是想将 NAVER Cafe SDK 用于应用程序的开发者。

**咨询处**

如该文件内容有误或是有与内容相关的疑问，可以用下面的联系方式进行咨询。

联系方式: dl_gamesdkpartner@navercorp.com

**文件版本及介绍**

| 日期 | 介绍 |
| --- | --- |
| 2015/12/23 | 最终发行 |

# 标记规则

### 参考标记

> *参考*
> 记述读者应当参考的内容。

### 注意标记

> **注意**
> 记述读者必须要知道的事项、可能引发系统错误的事项。

### UI 语句，用户输入值标记

- UI 语句： **菜单 > 下级菜单**
- 用户固定输入值：输入 **localhost**。
- 用户可变输入值：http://www.naver.com/{公司名}

### 源代码标记

该文件中源代码用灰底色、黑字体标记。

```
COPYDATASTRUCT st;
st.dwData = PURPLE_OUTBOUND_ENDING;
st.cbData = sizeof(pp);
st.lpData = &pp;
::SendMes(GetTargetHwnd(), WM_COPYDATA, (WPARAM)this->m_hWnd, (LPARAM)&st);
```

# 目录

# 图表和图片目录

## 图表目录



## 图片目录

# 概要

## NAVER Cafe SDK 概要

NAVER Cafe SDK 是与游戏引擎无关，在手游中可以简捷统合 NAVER Cafe 的资源库。使用 NAVER Cafe SDK 时，游戏玩家不离开游戏就可与游戏社区（NAVER Cafe）进行沟通。


**图 1 NAVER Cafe SDK**

## NAVER Cafe SDK 功能及特点

### 主要功能

NAVER Cafe SDK 的主要功能如下

- 支持 iOS 和 Android
- 支持 Unity 4 以上及 Cocos2d-x 2.1 以上游戏引擎
- 提供可以使用 Cafe 功能的画面
- 认证使用 OAuth 2.0 版本的 NAVER ID 进行登录的用户

### 使用环境

#### 操作系统

NAVER Cafe SDK 在如下环境中运作。

- Android: Android 4.2（Jelly Bean）API Level 17 以上
- iOS: iOS 7.0 以上（支持 ARMv7, ARMv7s, ARM64）

#### 登录 NAVER Login

如在 NAVER Cafe SDK 中通过使用“NAVER Login”功能认证用户时，需有客户端 ID 和客户端密码。

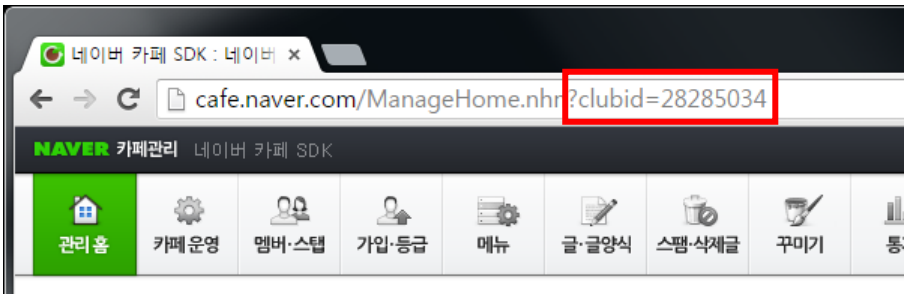
- 客户端 ID：在 NAVER Login 上登录游戏后得到的客户端 ID
- 客户端密码：在 NAVER Login 上登录游戏后得到的客户端密码

在 NAVER Login 上登录应用程序就可以得到客户端 ID 和客户端密码。请参考 NAVER Login Developers 的文件或向负责人咨询(dl_gamesdkpartner@navercorp.com)。

- NAVER Login Developers(韩国语)：https://nid.naver.com/devcenter/main.nhn

### NAVER Cafe ID

如想使用 NAVER Cafe SDK，需有适用的 NAVER Cafe 的 Cafe ID。Cafe ID 是在管理 NAVER Cafe 页面的 URL 中的 **clubid** 参数的值。



# 用于 Android 应用程序的说明

## 开发环境

### 软件要求事项

#### 开发工具

- IDE: Android Studio 或是 Eclipse
- Android Support Library v7

#### 资源库

如想使用 NAVER Cafe SDK，需将下面的资源库添加到项目中一起建立。

**表 1 使用 Android 用 NAVER Cafe SDK 时必需的资源库**

| 资源库 | 下载 URL |
|---|---|
| NAVER Login | • 包含在 NAVER Cafe SDK 资源库文件中（4.1.4 版本）。<br>• 下载 URL: https://static.nid.naver.com/images/web/devcenter/3rdparty_login_library_android_4.1.4.zip |
| NAVER Volleyer | • 包含在 NAVER Cafe SDK 资源库文件中（2.0.1 版本）。<br>• 下载 URL: http://mvnrepository.com/artifact/com.navercorp.volleyextensions/volleyer |
| Volley | • 包含在 NAVER Cafe SDK 资源库文件中（1.0.2 版本）。<br>• 下载 URL: http://mvnrepository.com/artifact/com.mcxiaoke.volley/library |
| Google Gson | • 包含在 NAVER Cafe SDK 资源库文件中（2.3.1 版本）。<br>• 下载 URL: http://mvnrepository.com/artifact/com.google.code.gson/gson |
| Glide | • 包含在 NAVER Cafe SDK 资源库文件中（3.6.1 版本）。<br>• 下载 URL: http://mvnrepository.com/artifact/com.github.bumptech.glide/glide |
| Otto | • 包含在 NAVER Cafe SDK 资源库文件中(1.3.8 版本）。<br>• 下载 URL: http://mvnrepository.com/artifact/com.squareup/otto |

### 资源库构成

Android 用 NAVER Cafe SDK 资源库如下构成。

- lib: NAVER Cafe SDK 资源库和必需资源库
  - cafeSdk-x.x.x.aar: 在 Android Studio 中使用的 NAVER Cafe SDK 资源库文件
  - cafeSdk-x.x.x.zip: 在 Eclipse 中使用的 NAVER Cafe SDK 资源库
  - library: 使用 NAVER Cafe SDK 时必需的资源库文件夹
- sample: 使用 NAVER Cafe SDK 资源库的示例项目文件夹
  - navercafesdk-sample-android-studio: 可以在 Android Studio 中使用的 NAVER Cafe SDK 示例项目文件夹
  - navercafesdk-sample-eclipse-master: 可以在 Eclipse 中使用的 NAVER Cafe SDK 示例项目文件夹

## 设置开发环境

如想使用 NAVER Cafe SDK，根据 IDE 如下设置开发环境。

**Android Studio**

1. 在 Android 项目的 **libs** 文件夹里复制 NAVER Cafe SDK 资源库文件（cafeSdk-x.x.x.aar）。
2. 在 Android 项目的 **libs** 文件夹里复制 NAVER Login 资源库文件（naveroauthlogin-4.x.x.jar）。
3. 在菜单里点击 **File > Project Structure**，并在 **Project Structure** 对话框里点击 **App > Dependencies** 后添加资源库。或如下在 Android 项目的 **build.gladle** 文件中直接添加资源库。

```
compile 'com.android.support:support-v13:23.1.0'
compile 'com.navercorp.volleyextensions:volleyer:2.0.1'
compile 'com.google.code.gson:gson:2.3.1'
compile 'com.github.bumptech.glide:glide:3.6.1'
compile 'com.squareup:otto:1.3.8'
```

**Eclipse**

1. 解除对 NAVER Cafe SDK 资源库文件（cafeSdk-x.x.x.zip）的压缩。
2. 在解除了压缩的 NAVER Cafe SDK 资源库文件夹下面的 **libs** 文件夹中添加构建时必需的资源库文件。
3. 在 Android 项目的 **AndroidManifest.xml** 文件中如下添加在 NAVER Login 中使用的活动和在 NAVER Cafe SDK 中使用的活动。

```
<activity
    android:name="com.naver.glink.android.sdk.ui.VideoPlayActivity"
    android:screenOrientation="sensorLandscape"
    android:theme="@android:style/Theme.Translucent.NoTitleBar.Fullscreen" />
<activity
    android:name="com.nhn.android.naverlogin.ui.OAuthLoginActivity"
    android:screenOrientation="sensorLandscape"
    android:theme="@android:style/Theme.Translucent.NoTitleBar" />
<activity
    android:name="com.nhn.android.naverlogin.ui.OAuthLoginInAppBrowserActivity"
    android:label="OAuth2.0 In-app"
    android:screenOrientation="sensorLandscape" />
```

# 使用 NAVER Cafe SDK

## 设置 AndroidManifest.xml

在 Android 项目的 **AndroidManifest.xml** 文件中添加如下内容。

在 package 属性里输入在 NAVER Login 登录应用程序时在 **Android Intent** 设置的内容。一并设置对调控 NAVER Cafe SDK 的客体 Glink Class 的使用权限。

```
<!—package 要与用 NAVER Login Developers 登录的 Android Intent 相同。-->
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.naver.glink.sample">
<!—对 Glink 需要的实用的权限 -->
   <uses-permission android:name="android.permission.INTERNET"/>
   <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
   <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

## NAVER Cafe SDK 初始化

初始化 NAVER Cafe SDK 的功能如下用 Glink.init()方法实现。

客户端 ID 和客户端密码是在 NAVER Login 上登录应用程序后得到的值。

```
/**
 * 用在 NAVER Login Developers 得到的信息初始化 SDK。
 * 调出 Glink 的其他方法前必须先进行初始化。
```

```
 * 开发者中心地址: https://nid.naver.com/devcenter/main.nhn
 */
final int cafeId = 28266581;
final String consumerKey = "客户端 ID";
final String consumerSecret = "客户端密码";
Glink.init(consumerKey, consumerSecret, cafeId);
```

## 开启 NAVER Cafe SDK 画面

NAVER Cafe SDK 画面由 5 个标签构成。开启 NAVER Cafe SDK 时选择以何种选定标签的状态开始。

例如如想以选定主标签页的状态开始，就如下使用 Glink.startHome()方法。

```
// 以主页面开始。
Glink.startHome(activity);
```

除此之外如想以公告标签页、活动标签页、留言板标签页、个人资料标签页开始，就使用下面的方法。

- Glink.startNotice()方法: 以公告标签页开始。
- Glink.startEvent()方法: 以 活动标签页开始。
- Glink.startMenu()方法: 以 留言板标签页开始。
- Glink.startProfile()方法: 以个人资料标签页开始。

## 关闭 NAVER Cafe SDK 画面

关闭 NAVER Cafe SDK 画面的功能用 Glink.stop()方法或 Glink.popBackStack()方法实现。

- Glink.stop()方法：立即关闭画面并结束 NAVER Cafe SDK。
- Glink.popBackStack()方法：用 backStack 一个个关闭累积的画面。关闭所有画面后结束 NAVER Cafe SDK。可以用 backStack 累积的画面为查看文章画面、写文章画面、检索画面。

下面是用 Glink.stop()方法关闭 NAVER Cafe SDK 画面的例子。

```
// 用 stop()方法关闭。
Glink.stop(activity);
```

下面是用 Glink.popBackStack()方法关闭 NAVER Cafe SDK 画面的例子。

```
// 用 popBackStack 关闭。
Glink.popBackStack(activity);
```

## 写文章

游戏玩家在 Cafe 写文章的画面如下用 Glink.startWrite()方法实现。

```
// 填入基本标题，正文，开启写文章画面。
如是 int menuId = 4; // 0，就不选择菜单。
String text = " 填入基本标题,正文，开启写文章画面。";
Glink.startWrite(MainActivity.this, "subject", text);
```

使用 Glink.startImageWrite()方法或是 Glink.startVideoWrite()方法，就可以运行添加图片或视频的状态下执行留言板写文章画面。

## 处理 App Scheme

有用 App Scheme 移动的广告条图像时如下设置 OnClickAppSchemeBannerListener，那么在有触碰活动时就可以实现处理 App Scheme 的功能。

```
// AppScheme touch Listener 设置。
Glink.setOnClickAppSchemeBannerListener(new Glink.OnClickAppSchemeBannerListener() {
        @Override public void onClickAppSchemeBanner(String appScheme) {
          // 在 Cafe 管理中设置的 App Scheme 字符串转交到 NAVER Cafe SDK。
          // 实现处理 App Scheme 的代码。
```

```
        }
    });
```

### 映射用户游戏 ID

映射用户的游戏 ID 和 NAVER ID 的功能如下用 Glink.setGameUserId()方法实现。

```
// 映射用户的游戏 ID 和 NAVER ID。
// 在个人资料画面可以确认映射的游戏 ID。
Glink.setGameUserId(this, "gameUserId", "游戏 ID");
```

### 更改资源图像

包含在 NAVER Cafe SDK 资源库中的资源图像可以如下进行更改。

1. 解除对 NAVER Cafe SDK 资源库文件（.aaa 文件）的压缩。
2. 解除压缩的文件夹里的**/res/drawable-xhdpi** 文件夹中的图片可以更改为所期望的图片。
3. 重新压缩已解除压缩的文件夹成 NAVER Cafe SDK 资源库文件（.aaa 文件）。
4. 使用新压缩的 NAVER Cafe SDK 资源库构建项目。

**注意**

更改图片时图片的大小必须与现有图片相同。

## API 参照

### Glink

调控 NAVER Cafe SDK 的 Class。使用 Glink Class 的方法实现初始化或是开启、结束 NAVER Cafe SDK 的功能。

Glink Class 的方法如下。

- init()
- isShowGlink()
- popBackStack()
- setGameUserId()
- startEvent()
- startImageWrite()
- startHome()
- startMenu()
- startNotice()
- startProfile()
- startVideoWrite()
- startWrite()
- stop()

### init()

**说明**

初始化 NAVER Cafe SDK。

#### 文章版面

```
public static void init(String clientId, String clientSecret, int cafeId);
```

#### 参数

| 参数 | 类型 | 是否必需 | 说明 |
|---|---|---|---|
| clientId | String | Y | 客户端 ID。在 NAVER Login 上登录应用程序后得到的值。 |
| clientSecret | String | Y | 客户端密码。在 NAVER Login 上登录应用程序后得到的值。 |
| cafeId | int | Y | Cafe ID。在 NAVER Cafe 管理页面的 URL 中 **clubid** 参数的值。 |

#### 返回值

无

#### 代码例子

```
// 使用用 NAVER ID 的登录信息和 Cafe ID 初始化 NAVER Cafe SDK。
Glink.init(“abcd”, "aaaa", 33);
```

## isShowGlink()

#### 说明

确认 NAVER Cafe SDK 画面是否打开着。

#### 文章版面

```
public static boolean isShowGlink(Activity activity);
```

#### 参数

| 参数 | 类型 | 是否必需 | 说明 |
|---|---|---|---|
| activity | Activity | Y | 实行方法的活动的 Context 客体 |

#### 返回值

- true: 打开 NAVER Cafe SDK 画面的状态
- false: 关闭 NAVER Cafe SDK 画面的状态

#### 代码例子

```
// 确认 NAVER Cafe SDK 画面是否打开着。
// 如返回值为 true，即为打开状态,如返回值为 false 即为关闭状态。
Glink.isShowGlink(this);
```

## popBackStack()

#### 说明

用backStack一个个关闭积累的画面。关闭所有画面后，就结束NAVER Cafe SDK。可以用backStack积累的画面为 查看文章画面、写文章画面、检索画面。

#### 文章版面

```
public static void popBackStack(Activity activity);
```

#### 参数

| 参数 | 类型 | 是否必需 | 说明 |
| --- | --- | --- | --- |
| activity | Activity | Y | 实行方法的活动的 Context 客体 |

#### 返回值

无

#### 代码例子

```
// 一个个关闭 NAVER Cafe SDK 画面。
Glink.popBackStack (activity);
```

## setGameUserId()

#### 说明

映射用户的游戏 ID 和 Cafe ID。

#### 文章版面

```
public static void setGameUserId(Activity activity, String gameUserId, String fieldName);
```

#### 参数

| 参数 | 类型 | 是否必需 | 说明 |
| --- | --- | --- | --- |
| activity | Activity | Y | 实行方法的活动的 Context 客体 |
| gameUserId | String | N | 用户的游戏 ID |
| fieldName | String | N | 要在个人资料画面标识的 ID（基本值：用户的游戏 ID） |

#### 返回值

无

#### 代码例子

```
// 映射用户的游戏 ID 和 NAVER ID。
// 可以在个人资料画面确认被映射的游戏 ID。
Glink.setGameUserId(this, "gameUserId", "游戏 ID");
```

## startEvent()

#### 说明

以选定活动标签页的状态开启 NAVER Cafe SDK。

#### 文章版面

```
public static void startEvent(Activity activity);
```

#### 参数

| 参数 | 类型 | 是否必需 | 说明 |
| --- | --- | --- | --- |
| activity | Activity | Y | 实行方法的活动的 Context 客体 |

#### 返回值

无

#### 代码例子

```
// 以活动标签页开启 NAVER Cafe SDK。
Glink.startEvent(activity);
```

## startImageWrite()

#### 说明

打开附加了图片的留言板写文章画面。

#### 文章版面

```
public static void startImageWrite(Activity activity, int menuId, String subject, String text, String imagery);
```

#### 参数

| 参数 | 类型 | 是否必需 | 说明 |
|---|---|---|---|
| activity | Activity | Y | 实行方法的活动的 Context 客体 |
| menuId | int | N | 留言板 ID（基本值：-1）。留言板 ID 是 NAVER Cafe 留言板 URL 中 **menuid** 参数的值。 |
| subject | String | N | 文章标题 |
| text | String | N | 文章正文 |
| imageUri | String | N | 图片文件的径路（URI 形式） |

#### 返回值

无

#### 代码例子

```
// 有基本标题，正文，以及添加图片的情况下，开启写文章画面。图片以 URI 形式填入。
如为 int menuId = 4; // 0，就不能选择菜单。
String text = " 有基本标题，正文，以及添加图片的情况下，开启写文章画面。图片以 URI 形式填入。";
String path = "your image uri";
Glink.startImageWrite(MainActivity.this, menuId, "subject", text, path);
```

## startHome()

#### 说明

以选定主标签页的状态打开 NAVER Cafe SDK 画面。

#### 文章版面

```
public static void startHome(Activity activity);
```

#### 参数

| 参数 | 类型 | 是否必需 | 说明 |
|---|---|---|---|
| activity | Activity | Y | 实行方法的活动的 Context 客体 |

#### 返回值

无

#### 代码例子

```
// 以主标签页打开 NAVER Cafe SDK 画面。
Glink.startHome(activity);
```

## startMenu()

#### 说明

以选定留言板标签页的状态打开 NAVER Cafe SDK 画面。

#### 文章版面

```
public static void startMenu(Activity activity);
```

#### 参数

| 参数 | 类型 | 是否必需 | 说明 |
| --- | --- | --- | --- |
| activity | Activity | Y | 实行方法的活动的 Context 客体 |

#### 返回值

无

#### 代码例子

```
// 以留言板标签页打开 NAVER Cafe SDK 画面。
Glink.startMenu(activity);
```

## startNotice()

#### 说明

以选定公告标签页的状态打开 NAVER Cafe SDK 画面。

#### 文章版面

```
public static void startNotice(Activity activity);
```

#### 参数

| 参数 | 类型 | 是否必需 | 说明 |
| --- | --- | --- | --- |
| activity | Activity | Y | 实行方法的活动的 Context 客体 |

#### 返回值

无

#### 代码例子

```
// 以公告标签页打开 NAVER Cafe SDK 画面。
Glink.startNotice(activity);
```

## startProfile()

### 说明

以选定个人资料标签页的状态打开 NAVER Cafe SDK 画面。

### 文章版面

```
public static void startProfile(Activity activity);
```

### 参数

| 参数 | 类型 | 是否必需 | 说明 |
|---|---|---|---|
| activity | Activity | Y | 实行方法的活动的 Context 客体 |

### 返回值

无

### 代码例子

```
// 用个人资料标签页打开 NAVER Cafe SDK 画面。
Glink.startProfile(activity);
```

## startVideoWrite()

### 说明

以附加视频的状态打开留言板写文章画面。

### 文章版面

```
public static void startVideoWrite(Activity activity, int menuId, String subject, String text, String videoUri);
```

### 参数

| 参数 | 类型 | 是否必需 | 说明 |
|---|---|---|---|
| activity | Activity | Y | 实行方法的活动的 Context 客体 |
| menuId | int | N | 留言板 ID（基本值：-1）。留言板 ID 是 NAVER Cafe 留言板 UR 中 **menuid** 参数的值。 |
| subject | String | N | 文章标题 |
| text | String | N | 文章正文 |
| videoUri | String | N | 视频文件的径路（URI 形式） |

### 返回值

无

### 代码例子

```
// 有基本标题，正文，以及添加录像的情况下，开启写文章画面。 图片以 URI 形式填入。
如是 int menuId = 4; // 0，则不能选择菜单。
String text = " 有基本标题，正文，以及添加录像的情况下，开启写文章画面。 图片以 URI 形式填入。";
String path = "your video uri";
Glink.startVideoWrite(MainActivity.this, menuId, "subject", text, path);
```

## startWrite()

### 说明

打开留言板写文章画面。

### 文章版面

```
public static void startWrite(Activity activity, int menuId, String subject, String text);
```

### 参数

| 参数 | 类型 | 是否必需 | 说明 |
|---|---|---|---|
| activity | Activity | Y | 实行方法的活动的 Context 客体 |
| menuId | int | N | 留言板 ID（基本值：-1）。留言板 ID 是 NAVER Cafe 留言板 UR 中 **menuid** 参数的值。 |
| subject | String | N | 文章标题 |
| text | String | N | 文章正文 |

### 返回值

无

### 代码例子

```
// 填入基本标题，正文开启写文章画面。
如是 int menuId = 4; // 0，则不能选择菜单。
String text = " 填入基本标题，正文开启写文章画面。";
Glink.startWrite(MainActivity.this, "subject", text);
```

## stop()

### 说明

立即关闭 NAVER Cafe SDK 画面，终止 NAVER Cafe SDK。

### 文章版面

```
public static void stop(final Activity activity);
```

### 参数

| 参数 | 类型 | 是否必需 | 说明 |
|---|---|---|---|
| activity | Activity | Y | 实行方法的活动的 Context 客体 |

### 返回值

无

### 代码例子

```
// 用 stop()方法终止。
Glink.stop(activity);
```

# 用于 iOS 应用程序的说明

## 开发环境

### 软件要求事项

#### 开发工具

- IDE: Xcode 6.0 以上

> **注意**
> ARC（automatic reference counting）适用于资源库。

#### 资源库

如想使用 NAVER Cafe SDK，需将下面的资源库添加到项目中一起建立。

**表 2 使用 iOS 用 NAVER Cafe SDK 必需的资源库**

| 资源库 | 下载 URL |
| --- | --- |
| NAVER Login | • 包含在 NAVER Cafe SDK 资源库文件中（4.0.6 版本）。<br>• 下载 URL: https://static.nid.naver.com/images/web/devcenter/3rdparty_login_library_ios_4.0.6.zip |
| AFNetworking 1.0 以上 | • 包含在 NAVER Cafe SDK 资源库文件中（2.6.1 版本）。<br>• 下载 URL: https://github.com/AFNetworking/AFNetworking |

### 资源库构成

iOS 用 NAVER Cafe SDK 资源库如下构成。

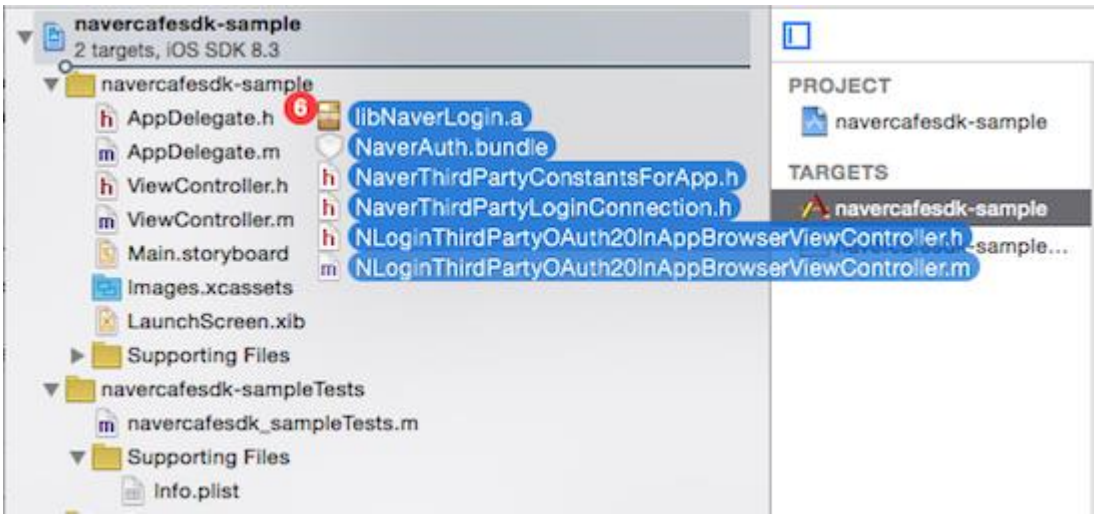
- lib: NAVER Cafe SDK 资源库文件夹
  - NaverCafeSDK.bundle
  - NaverCafeSDK.framework
- sample: 使用 NAVER Cafe SDK 资源库的示例项目文件夹和必需的资源库文件夹
  - external-lib: NAVER Login 资源库和 AFNetworking 资源库文件夹
  - navercafesdk-sample-ios: NAVER Cafe SDK 示例项目文件夹

### 设置开发环境

#### 设置 Xcode

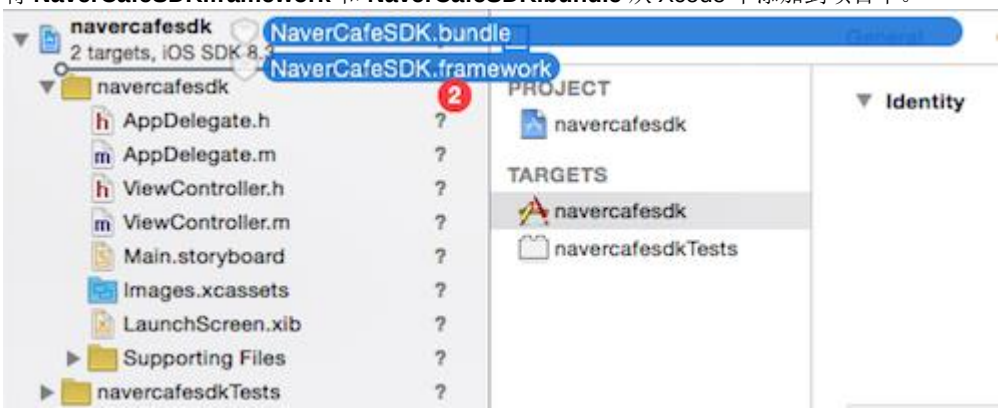
如想使用 NAVER Cafe SDK，就在 Xcode 中如下设置开发环境。

1. 解除对 NAVER Login 资源库的压缩。
2. 将 NAVER ID 登录资源库从 Xcode 添加到项目中。

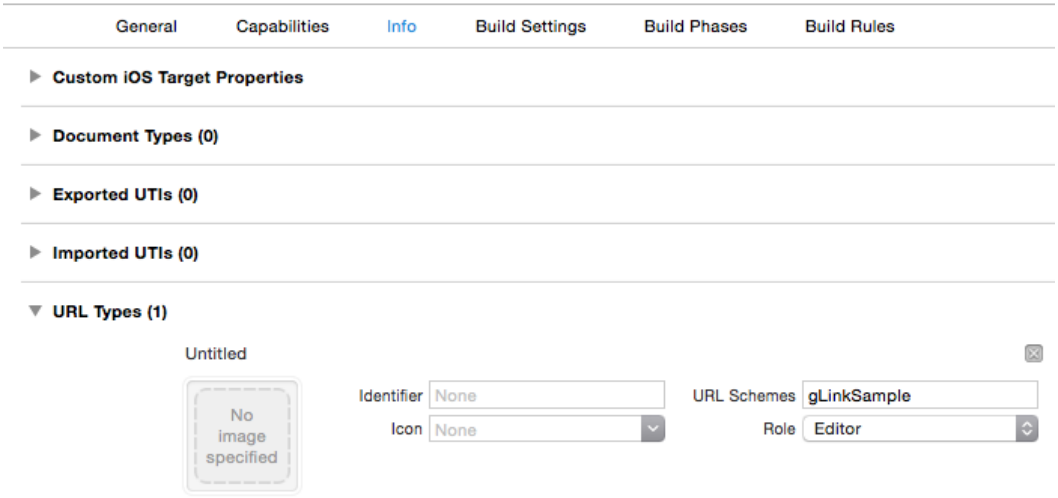



3. 将 **AFNetworking** 资源库从 Xcode 中添加到项目中。
4. 解除对 NAVER Cafe SDK 资源库的压缩。

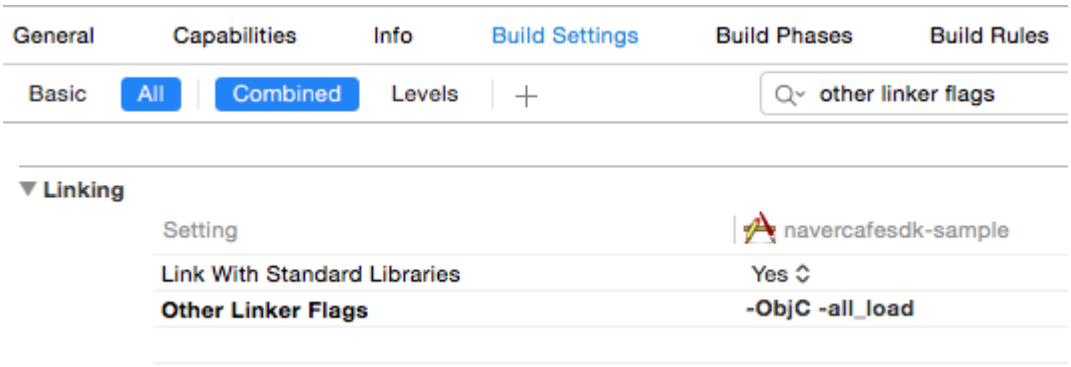
   将 **NaverCafeSDK.framework** 和 **NaverCafeSDK.bundle** 从 Xcode 中添加到项目中。



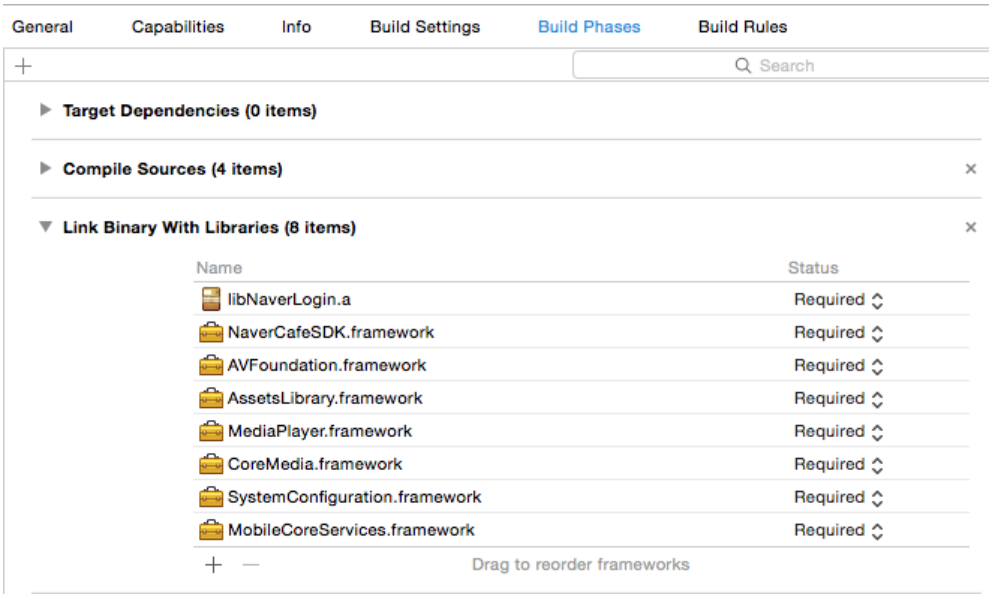

5. 在 NAVER Login 上登录应用程序时输入的 URL Scheme 登录到了项目中。




6. 为了能够使用静态资源库，在 **Build Settings** 的 **Other Linker Flags** 中设置 **-ObjC –all_load** 选项。




7. 在 **Build Phases** 的 **Link Binary With Libraries** 中添加如下资源库。
    - MobileCoreServices.framework
    - SystemConfiguration.framework
    - MediaPlayer.framework
    - AVFoundation.framework
    - CoreMedia.framework



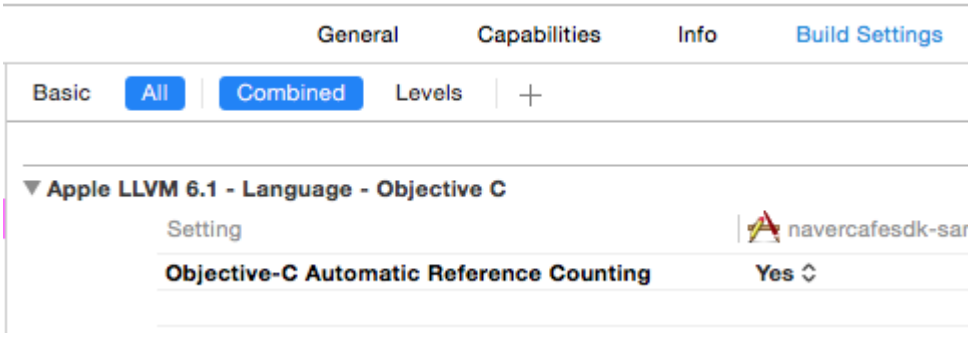

如使用 Cocos2d-x 引擎，要添加如下资源库。

- MobileCoreServices.framework
- SystemConfiguration.framework
- MediaPlayer.framework
- AVFoundation.framework
- CoreMedia.framework
- GameController.framework
- AssetsLibrary.framework
- Security.framework

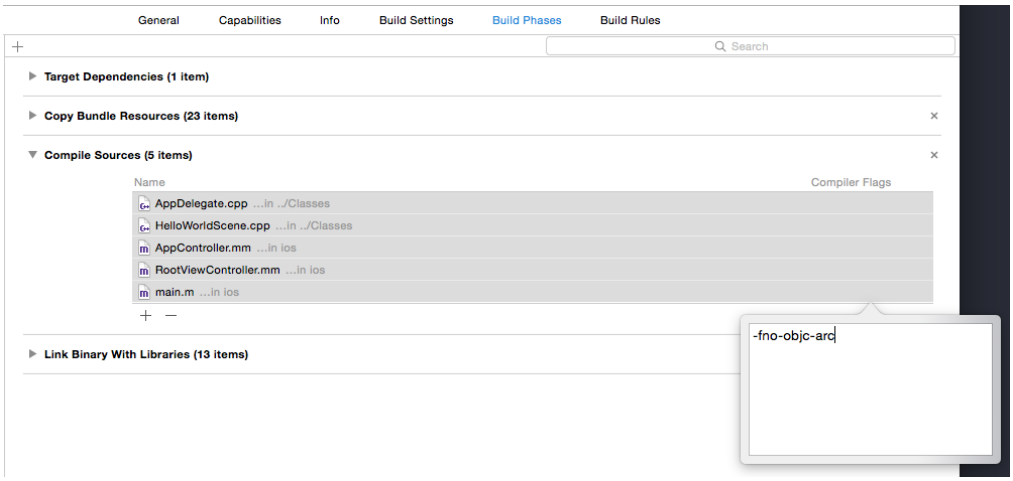
**MRC 环境转换为 ARC 环境**

使用 Cocos2d-x 引擎的项目如果是 MRC（manual reference counting）环境，要用如下方法转换为 ARC（automatic reference counting）环境。

1. 将 **Build Settings** 的 **Objective-C Automatic Reference Counting** 设置为 **YES**。




2. 从 **Build Phases** 的 **Compile Sources** 要编译文件的 **Compile Flags** 设置为**-fno-objc-arc**。




# 使用 NAVER Cafe SDK

## NAVER Cafe SDK 初始化

用如下方式初始化 NAVER Cafe SDK。客户端 ID 和客户端密码是 NAVER Login 应用程序得到的值。

```
//ViewController
#import <NaverCafeSDK/NCSDKManager.h>

- (void)viewDidLoad {
//使用 NAVER Login 的登录信息, Cafe ID 初始化 NAVER Cafe SDK。
    [[NCSDKManager getSharedInstance] setNaverLoginConsumerKey:@"客户端 ID"
                                                naverLoginConsumerSecret:@"客户端密码"
                                                                  cafeId:00000000];

}
```

## 设置 NAVER Login

如下设置 AppDelegate，如用 NAVER ID 完成登录就在 NAVER Cafe SDK 设置登录信息。

```
//AppDelegate
#import <NaverCafeSDK/NCSDKLoginManager.h>

- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url sourceApplication:(NSString *)sourceApplication
annotation:(id)annotation {
//在 NAVER Login 的客体上设置登录信息。
    return [[NCSDKLoginManager getSharedInstance] finishNaverLoginWithURL:url];
```

```
}
```

### 开启 NAVER Cafe SDK 画面

如想开启 NAVER Cafe SDK 画面，就要设置 NAVER Cafe SDK View Controller 实行 presentMainViewController 方法。

```
#import <NaverCafeSDK/NCSDKManager.h>

//运行 Cafe 主页。
[[NCSDKManager getSharedInstance] setParentViewController:self];
[[NCSDKManager getSharedInstance] presentMainViewController];
```

如替代 presentMainViewController 方法使用 presentMainViewControllerWithTabIndex 方法，就可以以 NAVER Cafe SDK 画面的特定标签页状态开启 NAVER Cafe SDK 画面。

如使用 presentMainViewControllerWithArticleId 方法，可以以选定的留言板文章打开的状态开启 NAVER Cafe SDK 画面。

### 写文章

如打开在 Cafe 留言板写文章的画面，就要如下使用 presentArticlePostViewControllerWithMenuId 方法。

```
//运行写文章画面。
[[NCSDKManager getSharedInstance] setParentViewController:self];
[[NCSDKManager getSharedInstance] presentArticlePostViewControllerWithMenuId:0 subject:@"文章标题" content:@"文章内容" filePath:@"document"];
```

## API 参照

### NCSDKManager

调控 NAVER Cafe SDK 的 Class。使用 NCSDKManager Class 的方法实现初始化或是开启、终止 NAVER Cafe SDK 的功能。运行 NAVER Cafe SDK 的最高级 ViewController parentViewController 具有 Class。

NCSDKManager Class 的方法如下。

- dismissViewController
- dismissTopViewController
- getSharedInstance
- navercafeRootViewController
- resetSharedInstance
- presentArticlePostViewControllerWithMenuId
- presentArticlePostViewControllerWithType
- presentMainViewController
- presentMainViewControllerWithArticleId
- presentMainViewControllerWithTabIndex
- presentViewController
- setGameUserId
- setNaverLoginClientId

### @property (nonatomic, weak) id parentViewController

**说明**

运行 NAVER Cafe SDK 的 ViewController。运行 NAVER Cafe SDK 的父类。

#### 文章版面

```
@property (nonatomic, weak) id parentViewController;
```

#### 参数

无

#### 返回值

UIViewController 客体

#### 代码例子

```
[[NCSDKManager getSharedInstance] setParentViewController:self]
```

## (void)dismissViewController

#### 说明

删除在运行 NAVER Cafe SDK 的 ViewController 上面被运行的其他 ViewController。

#### 文章版面

```
- (void)dismissViewController:(id)viewController;
```

#### 参数

| 参数 | 类型 | 是否必需 | 说明 |
| --- | --- | --- | --- |
| viewController | id | Y | 在 NAVER Cafe SDK ViewController 上面运行的 ViewController |

#### 返回值

无

#### 代码例子

```
[[NCSDKManager getSharedInstance] dismissViewController:self];
```

## (void)dismissTopViewController

#### 说明

删除在运行 NAVER Cafe SDK 的 ViewController 上面运行的其他 ViewController 中位于最上面的 ViewController。

#### 文章版面

```
- (void)dismissTopViewController;
```

#### 参数

无

#### 返回值

无

#### 代码例子

```
[NCSDKManager getSharedInstance] dismissTopViewController];
```

## (NCSDKManager *)getSharedInstance

**说明**

获取 NAVER Cafe SDK Instance（单例 Instance）。

**文章版面**

```
+ (NCSDKManager *)getSharedInstance;
```

**参数**

无

**返回值**

NCSDKManager 客体

**代码例子**

```
[NCSDKManager getSharedInstance]
```

## (id)navercafeRootViewController

**说明**

获取 NAVER Cafe SDK 最上端的 ViewController 客体。

**文章版面**

```
- (id)navercafeRootViewController;
```

**参数**

无

**返回值**

UIViewController 客体

**代码例子**

```
[[NCSDKManager getSharedInstance] navercafeRootViewController]
```

## (void)resetSharedInstance

**说明**

删除 NCSDKManager 客体。

**文章版面**

```
+ (void)resetSharedInstance;
```

**参数**

无

**返回值**

无

#### 代码例子

```
[NCSDKManager resetSharedInstance]
```

## (void)presentArticlePostViewControllerWithMenuId

#### 说明

打开留言板写文章画面。

#### 文章版面

```
- (void)presentArticlePostViewControllerWithMenuId:(NSInteger)menuId
                                            subject:(NSString *)subject
                                            content:(NSString *)content;
```

#### 参数

| 参数 | 类型 | 是否必需 | 说明 |
| --- | --- | --- | --- |
| menuId | NSInteger | Y | 留言板 ID（基本值：0）。留言板 ID 是 NAVER Cafe 留言板 URL 中 **menuid** 参数的值。 |
| subject | NSString | N | 文章标题 |
| content | NSString | N | 文章正文 |

#### 返回值

无

#### 代码例子

```
[[NCSDKManager getSharedInstance] presentArticlePostViewControllerWithMenuId:1
                                                                      subject:@"标题"
                                                                      content:@"内容"];
```

## (void)presentArticlePostViewControllerWithType

#### 说明

附加文件的状态下打开留言板写文章画面。

#### 文章版面

```
- (void)presentArticlePostViewControllerWithType:(GLArticlePostType)type
                                          menuId:(NSInteger)menuId
                                         subject:(NSString *)subject
                                         content:(NSString *)content
                                        filePath:(NSString *)filePath;
```

#### 参数

| 参数 | 类型 | 是否必需 | 说明 |
| --- | --- | --- | --- |
| type | GLArticlePostType | Y | 附加文件类型<br>• 1: 图片文件<br>• 2: 视频文件 |

| 参数 | 类型 | 是否必需 | 说明 |
| --- | --- | --- | --- |
| menuId | NSInteger | Y | 留言板 ID（基本值：0）。留言板 ID 是 NAVER Cafe 留言板 URL 中 **menuid** 参数的值。 |
| subject | NSString | N | 文章标题 |
| content | NSString | N | 文章正文 |
| filePath | NSString | Y | 附加文件径路 |

#### 返回值

无

#### 代码例子

```
 [[NCSDKManager getSharedInstance] presentArticlePostViewControllerWithType:kGLArticlePostTypeVideo
                                                                      menuId:1
                                                                     subject:@"标题"
                                                                     content:@"内容"

filePath:@"private/var/mobile/Applications/0D1657F9-EACF-4D64-BC8A-4E01EB4FF247/tmp/trim.2CC623C7-78C3-4597-BA75-9BA12BFEF333.MOV"];
```

## (void)presentMainViewController

#### 说明

打开 NAVER Cafe SDK 画面。

#### 文章版面

```
- (void)presentMainViewController;
```

#### 参数

无

#### 返回值

无

#### 代码例子

```
[[NCSDKManager getSharedInstance] presentMainViewController];
```

## (void)presentMainViewControllerWithArticleId

#### 说明

以打开留言板文章的状态打开 NAVER Cafe SDK 画面。

#### 文章版面

```
- (void)presentMainViewControllerWithArticleId:(NSUInteger)articleId;
```

#### 参数

| 参数 | 类型 | 是否必需 | 说明 |
| --- | --- | --- | --- |
| articleId | NSUInteger | Y | 留言 ID（基本值：0） |

#### 返回值

无

#### 代码例子

```
[[NCSDKManager getSharedInstance] presentMainViewControllerWithArticleId:10];
```

## (void)presentMainViewControllerWithTabIndex

#### 说明

以选定制定标签页的状态打开 NAVER Cafe SDK 画面。

#### 文章版面

```
- (void)presentMainViewControllerWithTabIndex:(NSUInteger)tabIndex;
```

#### 参数

| 参数 | 类型 | 是否必需 | 说明 |
| --- | --- | --- | --- |
| tabIndex | NSUInteger | Y | 要选的标签页索引值（基本值：0）<br>• 0: 主标签页<br>• 1: 公告标签页<br>• 2: 活动标签页<br>• 3: 统合留言板标签页<br>• 4: 个人资料标签页 |

#### 返回值

无

#### 代码例子

```
[[NCSDKManager getSharedInstance] presentMainViewControllerWithTabIndex:1];
```

## (void)presentViewController

#### 说明

在 NAVER Cafe SDK 最上端的 ViewController 中运行其他 ViewController。

#### 文章版面

```
- (void)presentViewController:(id)viewController;
```

#### 参数

| 参数 | 类型 | 是否必需 | 说明 |
| --- | --- | --- | --- |
| viewController | id | Y | ViewController 客体的 ID |

#### 返回值

无

#### 代码例子

```
[[NCSDKManager getSharedInstance] presentViewController:self];
```

## (void)setGameUserId

### 说明

关联用户的游戏 ID 和 NAVER ID，在个人资料画面标识出 ID。

### 文章版面

```
- (void)setGameUserId:(NSString *)gameUserId fieldName:(NSString *)fieldName;
```

### 参数

| 参数 | 类型 | 是否必需 | 说明 |
|---|---|---|---|
| gameUserId | NSString | Y | 用户的游戏 ID |
| fieldName | NSString | N | 要在个人资料画面标识的 ID（基本值: 游戏 ID） |

### 返回值

无

### 代码例子

```
[[NCSDKManager getSharedInstance] setGameUserId:@"abc3251235" fieldName:@"游戏 ID"];
```

## (void)setNaverLoginClientId

### 说明

设置 NAVER Login 的客体。

### 文章版面

```
- (void)setNaverLoginClientId:(NSString *)naverLoginClientId
        naverLoginClientSecret:(NSString *)naverLoginClientSecret
                      cafeId:(NSInteger)cafeId;
```

### 参数

无

### 返回值

无

### 代码例子

```
[[NCSDKManager getSharedInstance] setNaverLoginConsumerKey:@"Consumer ID"
                                    naverLoginConsumerSecret:@"Secret ID"
                                                      cafeId:00000000];
```

## NCSDKLoginManager

在 NAVER Cafe SDK 中调控 NAVER Login 功能的 Class。

NCSDKLoginManager Class 的方法如下。

- accessToken
- accessTokenExpireTime
- finishNaverLoginWithURL

- getSharedInstance
- isLoginWithFinish
- isValidAccessTokenExpireTimeNow
- loginWithFinish
- logout
- refreshAccessToken
- refreshAccessTokenWithFinish
- requestDeleteToken
- setIsInAppOauthEnable
- setIsNaverAppOauthEnable

## @property (nonatomic, weak) UIViewController *rootViewController

**说明**

运行登录 UIView 的 ViewController

**文章版面**

```
@property (nonatomic, weak) UIViewController *rootViewController;
```

**参数**

无

**返回值**

UIViewController 客体

**代码例子**

```
[[NCSDKLoginManager getSharedInstance] setRootViewController:self];
```

## (NSString *)accessToken

**说明**

NAVER 登录认证后回收在 NAVER SERVER 中获取的访问令牌（access token）。

**文章版面**

```
- (NSString *)accessToken;
```

**参数**

无

**返回值**

访问令牌

**代码例子**

```
[[NCSDKLoginManager getSharedInstance] accessToken];
```

## (NSString *)accessTokenExpireTime

**说明**

确认访问令牌（access token）的终结时间。

#### 文章版面

```
- (NSString *)accessTokenExpireTime;
```

#### 参数

无

#### 返回值

访问令牌的终结时间

#### 代码例子

```
[[NCSDKLoginManager getSharedInstance] accessTokenExpireTime];
```

## (BOOL)finishNaverLoginWithURL

#### 说明

NAVER Login 完成后运行 AppDelegate。

#### 文章版面

```
- (BOOL)finishNaverLoginWithURL:(NSURL *)url;
```

#### 参数

| 参数 | 类型 | 是否必需 | 说明 |
| --- | --- | --- | --- |
| url | NSURL | Y | 完成 NAVER Login 后用 AppDelegate 可回拨的 URL Scheme |

#### 返回值

- true: 登录成功
- false: 登录失败

#### 代码例子

```
 - (BOOL)application:(UIApplication *)application openURL:(NSURL *)url sourceApplication:(NSString *)sourceApplication
annotation:(id)annotation {
    return [[NCSDKLoginManager getSharedInstance] finishNaverLoginWithURL:url];

}
```

## (NCSDKLoginManager *)getSharedInstance

#### 说明

NAVER Login 获取 Instance（单例 Instance）。

#### 文章版面

```
+ (NCSDKLoginManager *)getSharedInstance;
```

#### 参数

无

#### 返回值

NCSDKLoginManager 客体

#### 代码例子

```
[NCSDKLoginManager getSharedInstance]
```

## (void)isLoginWithFinish

#### 说明

确认 NAVER 登录认证后在 NAVER SERVER 中获取的访问令牌（access token）是否有效。

#### 文章版面

```
- (void)isLoginWithFinish:(void (^)(BOOL successACToken))finish;
```

#### 参数

无

#### 返回值

- true: 有效的访问令牌。
- false: 访问令牌过期，是无效的访问令牌。

#### 代码例子

```
[[NCSDKLoginManager getSharedInstance] isLoginWithFinish:^(BOOL successACToken) {
        if (successACToken) {

        }
}];
```

## (BOOL)isValidAccessTokenExpireTimeNow

#### 说明

确认是否有访问令牌（access token）以及有效时间剩余多少。但是，在 SERVER 中有效期已过的情况下不得确认。

#### 文章版面

```
- (BOOL)isValidAccessTokenExpireTimeNow;
```

#### 参数

无

#### 返回值

- true: 有效的访问令牌。
- false: 访问令牌过期，是无效的访问令牌。

#### 代码例子

```
[[NCSDKLoginManager getSharedInstance] isValidAccessTokenExpireTimeNow];
```

## (void)loginWithFinish

#### 说明

开始 NAVER Login 进行认证过程，并获取退回的访问令牌（access token）。

#### 文章版面

```
- (void)loginWithFinish:(void (^)(BOOL successACToken))finish;
```

#### 参数

无

#### 返回值

- true: 有效的访问令牌。
- false: 访问令牌过期，是无效的访问令牌。

#### 代码例子

```
[[NCSDKLoginManager getSharedInstance] isLoginWithFinish:^(BOOL successACToken) {
        if (successACToken) {

        }
}];
```

## (void)logout

#### 说明

删除保存在客户端的访问令牌（access token）和刷新令牌（refresh token）退出。

#### 文章版面

```
- (void)logout;//delete local accesstoken
```

#### 参数

无

#### 返回值

无

#### 代码例子

```
[[NCSDKLoginManager getSharedInstance] logout];
```

## (void)refreshAccessToken

#### 说明

在 NAVER 登录认证后，用在 NAVER SERVER 中获取的刷新令牌（refresh token）更新访问令牌（access token）。

#### 文章版面

```
- (void)refreshAccessToken;
```

#### 参数

无

#### 返回值

无

#### 代码例子

```
[[NCSDKLoginManager getSharedInstance]refreshAccessToken];
```

## (void)refreshAccessTokenWithFinish

#### 说明

NAVER 登录认证后在 NAVER SERVER 获取退回的刷新令牌（refresh token）。

#### 文章版面

```
- (void)refreshAccessTokenWithFinish:(void (^)(BOOL successACToken))finish;
```

#### 参数

无

#### 返回值

- true: 访问令牌更新成功
- false: 访问令牌更新失败

#### 代码例子

```
 [[NCSDKLoginManager getSharedInstance] refreshAccessTokenWithFinish:^(BOOL successACToken) {
            if (successACToken) {

            }
    }];
```

## (void)requestDeleteToken

#### 说明

删除保存在客户端和 SERVER 中的访问令牌（access token）和刷新令牌（referesh token）。

#### 文章版面

```
- (void)requestDeleteToken;//delete server authorization
```

#### 参数

无

#### 返回值

无

#### 代码例子

```
[[NCSDKLoginManager getSharedInstance] requestDeleteToken];
```

## (void)setIsInAppOauthEnable

#### 说明

运行 IN APP 浏览器，进行登录步骤。

#### 文章版面

```
- (void)setIsInAppOauthEnable:(BOOL)enable;
```

**参数**

| 参数 | 类型 | 是否必需 | 说明 |
| --- | --- | --- | --- |
| enable | BOOL | Y | 是否运行 IN APP 浏览器 |

**返回值**

无

**代码例子**

```
[[NCSDKLoginManager getSharedInstance] setIsInAppOauthEnable:YES];
```

## (void)setIsNaverAppOauthEnable

**说明**

运行 NAVER APP 后，进行登录步骤。

**文章版面**

```
- (void)setIsNaverAppOauthEnable:(BOOL)enable;
```

**参数**

| 参数 | 类型 | 是否必需 | 说明 |
| --- | --- | --- | --- |
| enable | BOOL | Y | 是否运行 NAVER APP |

**返回值**

无

**代码例子**

```
[[NCSDKLoginManager getSharedInstance] setIsNaverAppOauthEnable:YES];
```

# 用于 Unity 应用程序的说明

## 开发环境

### 软件要求事项

#### 开发工具

- 游戏引擎：Unity 4 以上

### 资源库构成

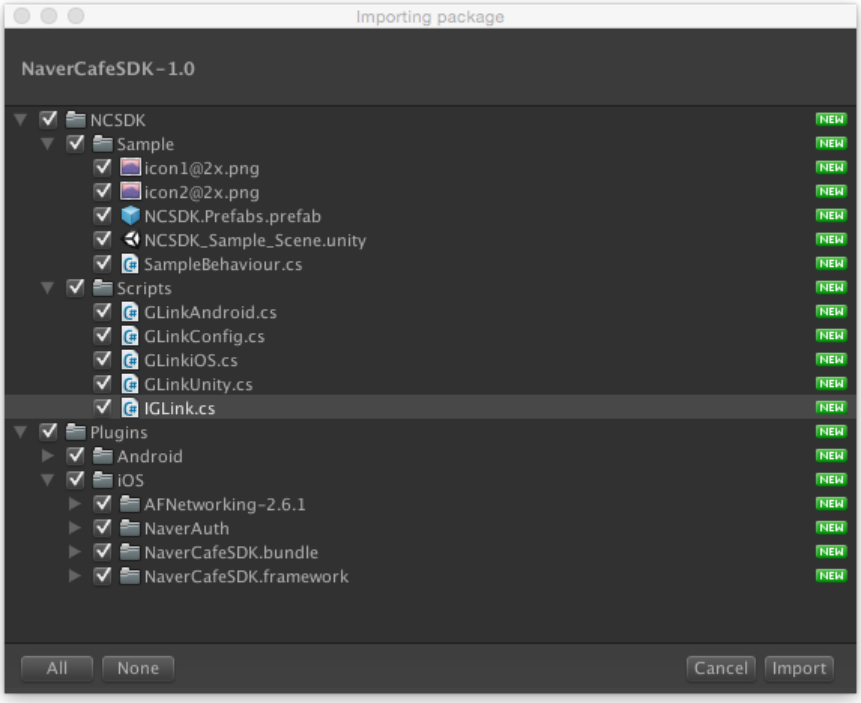
- NaverCafeSDK-1.0.unitypackage:Unity 用 NAVER Cafe SDK 资源库

### 设置开发环境

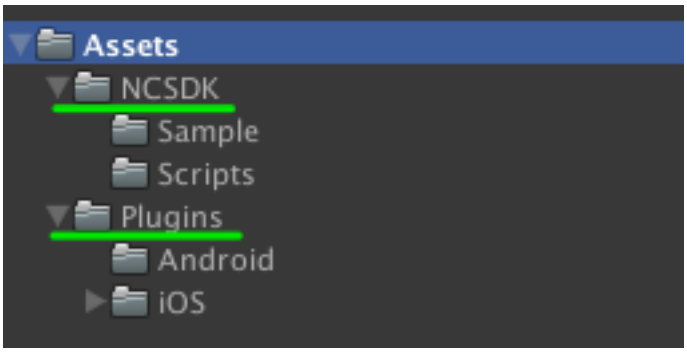
如想使用 NAVER Cafe SDK，如下设置开发环境。

#### 设置 Unity

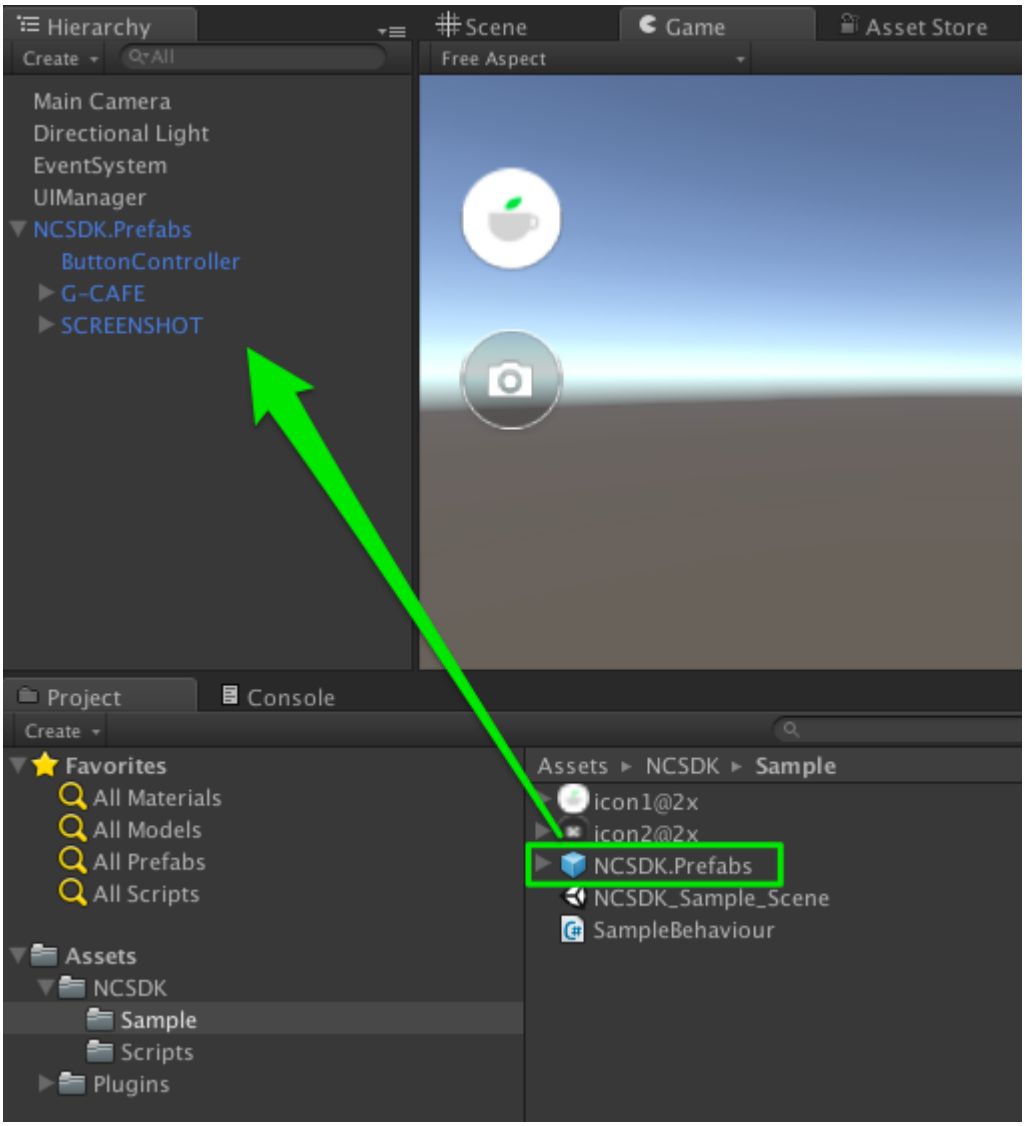
1. 点击 Unity 菜单中的 **Asset > Import package** 后，在 **Custom Package** 对话框中选择下载的 **NaverCafeSDK-1.0.unityPackage** 文件。




2. 在 **Custom Package** 对话框中点击 **Import**，就会生成 **NCSDK** 文件夹和 **Plugins** 文件夹。




3. 把在 **NCSDK** 文件夹的 **Sample** 文件夹中想要的 **NCSDK.Prefabs** 拉入其中。在 NAVER Cafe SDK 中生成默认提供的按钮。




4. 在 **NCSDK** 文件夹里 **Scripts** 文件夹的 **GLinkConfig** 文件中输入 Cafe ID、客户端 ID 和客户端密码。

```
//NCSDK/Scripts/GLinkConfig
static class GLinkConfig
{
	public const string NaverLoginConsumerKey =
		"Consumer ID";
```

```
    public const string NaverLoginConsumerSecret =
        "Secret ID";

    public const int CafeId =
        00000000;
}
```

### 设置 Android

使用 Unity 5 引擎时，如下添加 NAVER Cafe SDK 资源库。

- 将 NAVER Cafe SDK 资源库文件（cafeSdk-x.x.x.aar 文件）和必需资源库文件添加到 **Assets/Plugins/Android** 文件夹。

使用 Unity 4 引擎时，在 Eclipse 中需要如下的添加设置。

1. 解除获取的 NAVER Cafe SDK 资源库文件夹中 cafeSdk-x.x.x.zip 文件的压缩。
2. 在解除了压缩的 NAVER Cafe SDK 资源库文件夹下的 **libs** 文件夹中添加构建时必需的资源库文件。
3. 在 Eclipse 中解除压缩的文件夹导入到项目。
4. 在项目设置对话框中点击 **Android**，选择 **Is Library**。
5. 在 **AndroidManifest.xml** 文件中如下添加在 NAVER Cafe SDK 中使用的活动。

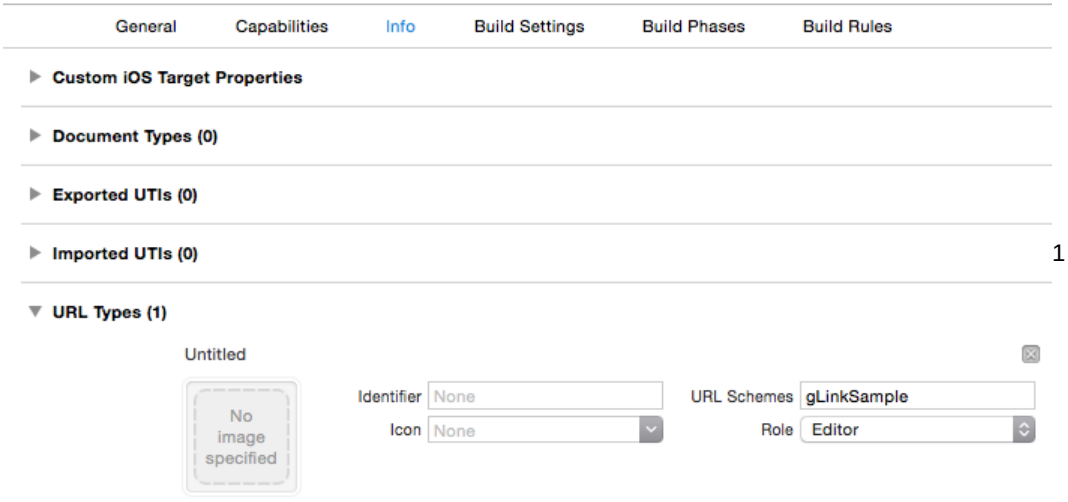
```
<activity
          android:name="com.nhn.android.naverlogin.ui.OAuthLoginActivity"
           android:screenOrientation="sensorLandscape"
           android:theme="@android:style/Theme.Translucent.NoTitleBar"/>
<activity
           android:name="com.nhn.android.naverlogin.ui.OAuthLoginInAppBrowserActivity"
           android:screenOrientation="sensorLandscape"
           android:label="OAuth2.0 In-app"/>
<activity
           android:name=".ui.VideoPlayActivity"
           android:screenOrientation="sensorLandscape"
           android:theme="@android:style/Theme.Translucent.NoTitleBar.Fullscreen"/>
```

6. 构建相应的项目。
7. 将用 Eclipse 构建的文件添加到 Unity 项目的 **Assets/Plugins/Android** 文件夹中。
8. 构建 Unity 项目。

### 设置 iOS

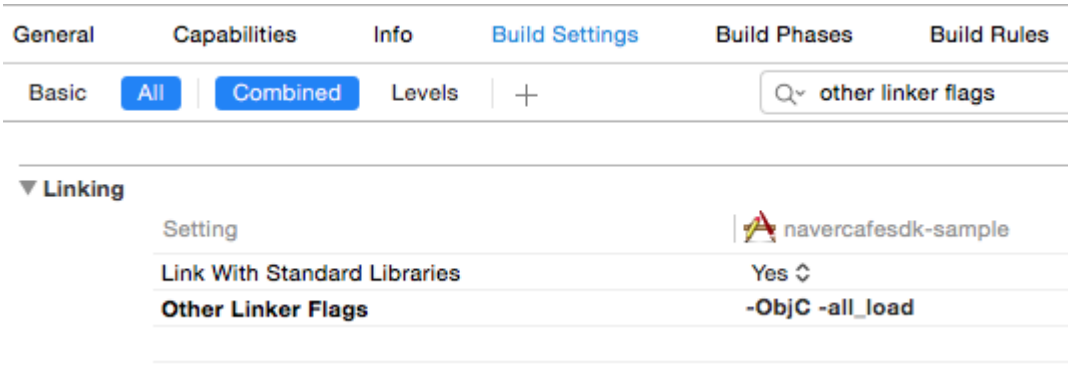
开发 iOS 应用程序时如下设置 Xcode。

1. 在 NAVER Login 上登录应用程序时，将输入的 URL Scheme 登录到 Xcode 项目中。

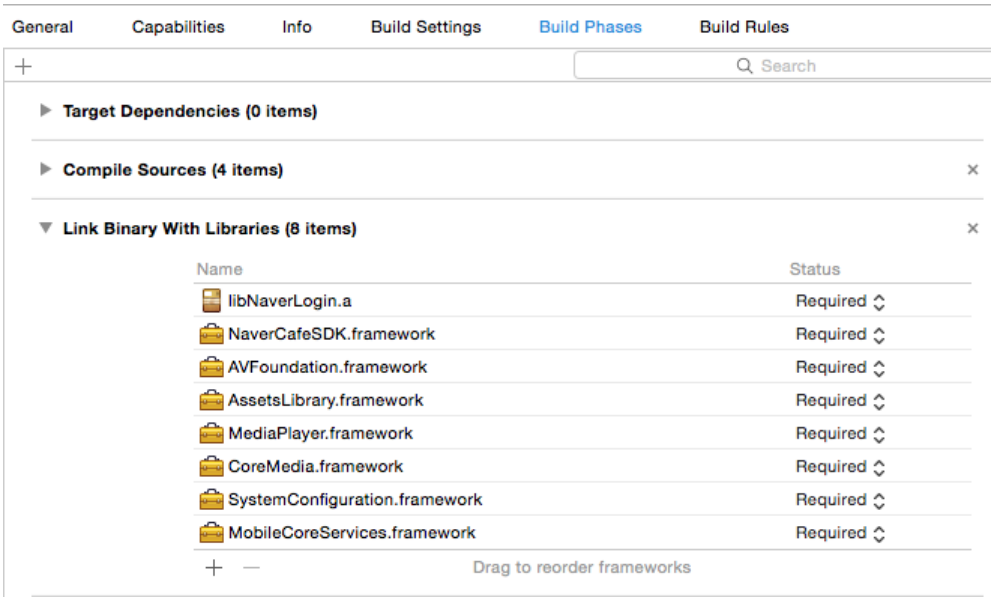



1

2. 为了能够使用静态资源库，在 **Build Settings** 的 **Other Linker Flags** 中设置**-ObjC –all_load** 选项。




3. 在 **Build Phases** 的 **Link Binary With Libraries** 中添加下面的资源库。
   - MobileCoreServices.framework
   - SystemConfiguration.framework
   - MediaPlayer.framework
   - AVFoundation.framework
   - CoreMedia.framework
   - Security.framework
   - AssetsLibrary.framework




4. 如下设置 App Delegate，使 NAVER Login 完成后 App Delegate 被呼出。

```
//AppDelegate
# import    <NaverCafeSDK/NCSDKLoginManager.h>

- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url sourceApplication:(NSString
*)sourceApplication annotation:(id)annotation {
        return    [[NCSDKLoginManager getSharedInstance] finishNaverLoginWithURL:url];

}
```

# 使用 NAVER Cafe SDK

## 运行

如想开启 NAVER Cafe SDK 画面，要如下呼出 GlinkUnity.executeMain()方法。

```
// Cafe 主页
GlinkUnity.executeMain ();
```

游戏玩家在 Cafe 写文章的画面如下用 GlinkUnity.executeArticlePostWithImgae()方法实现。

```
//截屏
GlinkUnity.executeArticlePostWithImgae(menuId, "留言标题", "留言内容", image path);
```

如使用 GlinkUnity .executeArticlePostWithImage()方法或是 GlinkUnity .executeArticlePostWithVideo()方法，可以在附加图片或是视频的状态下打开留言板写文章画面。

## 映射用户的游戏 ID 和 NAVER ID

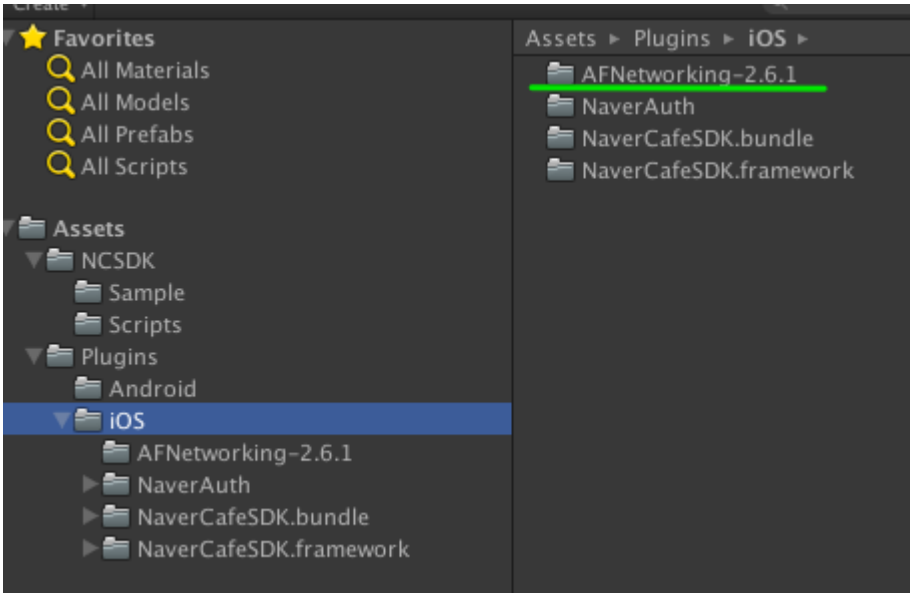
NAVER Cafe SDK 提供映射用户的游戏 ID 和 NAVER ID 的功能。如在 PC 上接触 Cafe 管理者页面，可以确认以影射的游戏 ID 和 NAVER ID 的目录。

下面是在 iOS 应用程序中实现游戏 ID 和 NAVER ID 映射功能的例子。

```
/*
 为了联动 NAVER ID 和游戏 ID，而联动用户的游戏 ID。
 fieldName 只是要在简历方面标识的字符串(基本值: 游戏 ID)
 */
- (void)setGameUserId:(NSString *)gameUserId fieldName:(NSString *)fieldName;
```

## 替换 AFNetworking

在 **NaverCafeSDK.unitypackage** 中默认包含用于互联网通信的资源库 AFNetworking 2.6.1。根据项目的状况也可删除 AFNetworking 资源库或是更改为其他版本进行构建。但是，NAVER Cafe SDK 只支持 AFNetworking 1.x 以上版本。



# API 参照

## GLinkUnity

调控 NAVER Cafe SDK 的 Class。使用 GLinkUnity Class 的方法实现开启或打开写文章画面的功能。

GLinkUnity Class 的方法如下。

- executeArticlePost()
- executeArticlePostWithImage()
- executeArticlePostWithVideo()
- executeMain()

## executeArticlePost()

### 说明

打开留言板写文章画面。

### 文章版面

```
public static void executeArticlePost(int menuId, string subject, string content) {
        sharedInstance().executeArticlePost (menuId, subject, content);
    }
```

### 参数

| 参数 | 类型 | 是否必需 | 说明 |
|---|---|---|---|
| menuId | int | Y | 留言板 ID（基本值: -1）。留言板 ID 是 NAVER Cafe 留言板 URL 中 **menuid** 参数的值。 |
| subject | string | Y | 文章标题 |
| content | string | Y | 文章正文 |

### 返回值

无

### 代码例子

```
GlinkUnity.executeArticlePost(28290504, "文章标题","文章正文" );
```

## executeArticlePostWithImage()

### 说明

附加图片的状态下打开留言板写文章画面。

### 文章版面

```
public static void executeArticlePostWithImage(int menuId, string subject, string content, string filePath) {
        sharedInstance().executeArticlePostWithImage (menuId, subject, content, filePath);
    }
```

**参数**

| 参数 | 类型 | 是否必需 | 说明 |
| --- | --- | --- | --- |
| menuId | int | Y | 留言板 ID（基本值: -1）。留言板 ID 是 NAVER Cafe 留言板 URL 中 menuid 参数的值。 |
| subject | string | Y | 文章标题 |
| content | string | Y | 文章正文 |
| filePath | string | Y | 图片文件的径路（URI 形式） |

**返回值**

无

**代码例子**

```
{
GlinkUnity.executeArticlePostWithImgae(28290504, "    文章标题", "文章内容","/navercafesdk/glink.png" );
}
```

## executeArticlePostWithVideo()

**说明**

附加视频的状态下打开留言板写文章画面。

**文章版面**

```
    public static void executeArticlePostWithVideo(int menuId, string subject, string content, string filePath) {
        sharedInstance().executeArticlePostWithVideo (menuId, subject, content, filePath);
    }
```

**参数**

| 参数 | 类型 | 是否必需 | 说明 |
| --- | --- | --- | --- |
| menuId | int | Y | 留言板 ID（基本值: -1）。留言板 ID 是 NAVER Cafe 留言板 URL 中 menuid 参数的值。 |
| subject | string | Y | 文章标题 |
| content | string | Y | 文章正文 |
| filePath | string | Y | 视频文件的径路（URI 形式） |

**返回值**

无

**代码例子**

```
{
GlinkUnity.executeArticlePostWithImgae(28290504, "文章标题", "文章内容","/navercafesdk/glink.avi" );
}
```

## executeMain()

**说明**

打开 NAVER Cafe SDK 画面。

### 文章版面

```
public static void executeMain() {
            sharedInstance().executeMain ();
    }
```

### 参数

无

### 返回值

无

### 代码例子

```
{
  GlinkUnity.executeMain ();
}
```