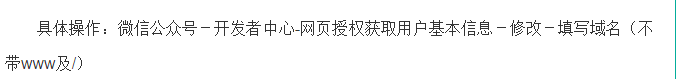
微信公众号要获得相应的AppID和AppSecret，申请微信登录且通过审核后，才可开始接入流程。



登录

(S🡪C) 发送一个请求

(C🡪S) APPID(Utf16类型)【或者测试的时候 前端写死】

前端和微信交互 获取Code

方法一:后端获取openID(有一定编程代码)

(C🡪S) Code

(S🡪微信) 获取openID

方法二:前端自己和微信进行交互，但是不安全



(C🡪S) 发送请求(获取AppSecret)

(S🡪C) AppSecret（或者测试时不和服务器交互，前端写死，但是不安全）

(C🡪微信) 获取openID

支付

//测试时，前端获取openID

（C🡪S） openID（Utf16类型）、支付的钱数(Int类型)、GameID（Byte类型）

(S🡪C) appID( UTF16类型)、timeStamp（Long类型）、nonceStr(UTF16类型)

package(UTF16类型)、paySign(UTF类型);

利用微信的微信支付接口:WeixinJSBridge.invoke调用微信支付接口

1. 准备:

为了识别用户，每个用户针对每个公众号会产生一个安全的OpenID，如果需要在多公众号、移动应用之间做用户共通，则需前往微信开放平台，将这些公众号和应用绑定到一个开放平台账号下，绑定后，一个用户虽然对多个公众号和应用有多个不同的OpenID，但他对所有这些同一开放平台账号下的公众号和应用，只有一个UnionID，可以在用户管理-获取用户基本信息（UnionID机制）文档了解详情。

每个接口都有每日接口调用频次限制，可以在公众平台官网-开发者中心处查看具体频次。

公众平台以access\_token为接口调用凭据，来调用接口，所有接口的调用需要先获取access\_token，access\_token在2小时内有效，过期需要重新获取，但1天内获取次数有限，开发者需自行存储

公众平台接口调用仅支持80端口。

1. UnionID机制

请注意，网页授权获取用户基本信息也遵循UnionID机制。即如果开发者有在多个公众号，或在公众号、移动应用之间统一用户帐号的需求，需要前往微信开放平台（open.weixin.qq.com）绑定公众号后，才可利用UnionID机制来满足上述需求。

UnionID机制的作用说明：如果开发者拥有多个移动应用、网站应用和公众帐号，可通过获取用户基本信息中的unionid来区分用户的唯一性，因为同一用户，对同一个微信开放平台下的不同应用（移动应用、网站应用和公众帐号），unionid是相同的。

参考：

<https://mp.weixin.qq.com/wiki/1/8a5ce6257f1d3b2afb20f83e72b72ce9.html>

在不同的公众账号下openid是不一样的，而他们的unionid却是一样的。其他的信息不具有唯一性，没有比较意义。但基本信息是一致的。

* UnionID的意义:

对于拥有多个账号的企业来说，unionid可以帮助识别不同公众账号下的用户是否是同一个人。这样在不同账号下对该用户提供的服务可以连续起来了。

还可以去除重复关注的用户数，便于统计真实的关注用户总数。

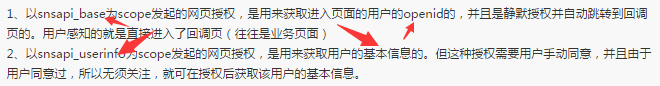
1. 登录:

如果用户在微信客户端中访问第三方网页，公众号可以通过微信网页授权机制，来获取用户基本信息，进而实现业务逻辑。

微信网页授权是通过OAuth2.0机制实现的，在用户授权给公众号后，公众号可以获取到一个网页授权特有的接口调用凭证（网页授权access\_token），通过网页授权access\_token可以进行授权后接口调用，如获取用户基本信息；

* 网页授权的两种Scope的区别:
* 1、以snsapi\_base为scope发起的网页授权，是用来获取进入页面的用户的openid的，并且是静默授权并自动跳转到回调页的。用户感知的就是直接进入了回调页（往往是业务页面）
* 2、以snsapi\_userinfo为scope发起的网页授权，是用来获取用户的基本信息的。但这种授权需要用户手动同意，并且由于用户同意过，所以无须关注，就可在授权后获取该用户的基本信息。
* 默认登录:





* 显性登陆:
* 静默授权:

1. 对于以snsapi\_base为scope的网页授权，就静默授权的，用户无感知；
2. 对于已关注公众号的用户，如果用户从公众号的会话或者自定义菜单进入本公众号的网页授权页，即使是scope为snsapi\_userinfo，也是静默授权，用户无感

* 具体而言，网页授权流程分为四步:

1. 引导用户进入授权页面统一授权，获取code;
2. 通过code换取网页授权access\_token(与基础支持中的access\_token不同)，此时已经成功获取openid(如果Scope是snsapi\_base类型，那么到达此步骤，已经结束);
3. 如果需要，开发者可以刷新网页授权access\_token,避免过期;
4. 通过网页授权access\_token和openid

* 用户同意授权，获取code

在确保微信公众账号拥有授权作用域（scope参数）的权限的前提下（服务号获得高级接口后，默认拥有scope参数中的snsapi\_base和snsapi\_userinfo）

引导关注者打开如下页面:

<https://open.weixin.qq.com/connect/oauth2/authorize?appid=APPID&redirect_uri=REDIRECT_URI&response_type=code&scope=SCOPE&state=STATE#wechat_redirect>

尤其注意：由于授权操作安全等级较高，所以在发起授权请求时，微信会对授权链接做正则强匹配校验，如果链接的参数顺序不对，授权页面将无法正常访问

**参数说明**

|  |  |  |
| --- | --- | --- |
| **参数** | **是否必须** | **说明** |
| appid | 是 | 当前公众号的唯一标识 |
| redirect\_uri | 是 | 授权后重定向的回调链接地址，请使用urlencode对链接进行处理 |
| response\_type | 是 | 返回类型，请填写code |
| scope | 是 | 应用授权作用域，snsapi\_base （不弹出授权页面，直接跳转，只能获取用户openid），snsapi\_userinfo （弹出授权页面，可通过openid拿到昵称、性别、所在地。并且，即使在未关注的情况下，只要用户授权，也能获取其信息） |
| state | 否 | 重定向后会带上state参数，开发者可以填写a-zA-Z0-9的参数值，最多128字节 |
| #wechat\_redirect | 是 | 无论直接打开还是做页面302重定向时候，必须带此参数 |

当scope的值为snsapi\_userinfo时的授权页面如下:



**用户同意授权后**

如果用户同意授权，页面将跳转至 redirect\_uri?code=CODE&state=STATE。若用户禁止授权，则重定向后不会带上code参数，仅会带上state参数redirect\_uri?state=STATE

code说明 ：

code作为换取access\_token的票据，每次用户授权带上的code将不一样，code只能使用一次，5分钟未被使用自动过期。

//显性授权

**window**.**location**.**href**= **'https://open.weixin.qq.com/connect/oauth2/authorize?appid='**+ dataMap.get(**"appID"**)+**'&redirect\_uri=http%3a%2f%2fh5.tongqutongqu.cn%2ftest.html&response\_type=code&scope=snsapi\_userinfo&state=STATE%23wechat\_redirect'**;

* 通过code换取网页授权access\_token

注意:

这里通过code换取的是一个特殊的网页授权access\_token,与基础支持中的access\_token（该access\_token用于调用其他接口）不同。公众号可通过下述接口来获取网页授权access\_token。如果网页授权的作用域为snsapi\_base，则本步骤中获取到网页授权access\_token的同时，也获取到了openid，snsapi\_base式的网页授权流程即到此为止。

**请求方法：**

获取code后，请求以下链接获取access\_token：

<https://api.weixin.qq.com/sns/oauth2/access_token?appid=APPID&secret=SECRET&code=CODE&grant_type=authorization_code>

**参数说明**

|  |  |  |
| --- | --- | --- |
| **参数** | **是否必须** | **说明** |
| appid | 是 | 公众号的唯一标识 |
| secret | 是 | 公众号的appsecret |
| code | 是 | 填写第一步获取的code参数 |
| grant\_type | 是 | 填写为authorization\_code |

**返回说明**

正确时返回的JSON数据包如下：

{

"access\_token":"ACCESS\_TOKEN",

"expires\_in":7200,

"refresh\_token":"REFRESH\_TOKEN",

"openid":"OPENID",

"scope":"SCOPE"

}

|  |  |
| --- | --- |
| **参数** | **描述** |
| access\_token | 网页授权接口调用凭证,注意：此access\_token与基础支持的access\_token不同 |
| expires\_in | access\_token接口调用凭证超时时间，单位（秒） |
| refresh\_token | 用户刷新access\_token |
| openid | 用户唯一标识，请注意，在未关注公众号时，用户访问公众号的网页，也会产生一个用户和公众号唯一的OpenID |
| scope | 用户授权的作用域，使用逗号（,）分隔 |

错误时微信会返回JSON数据包如下（示例为Code无效错误）:

{"errcode":40029,"errmsg":"invalid code"}

* 刷新access\_token（如果需要）

由于access\_token拥有较短的有效期，当access\_token超时后，可以使用refresh\_token进行刷新，refresh\_token拥有较长的有效期（7天、30天、60天、90天），当refresh\_token失效的后，需要用户重新授权。

**请求方法**

获取第二步的refresh\_token后，请求以下链接获取access\_token：

<https://api.weixin.qq.com/sns/oauth2/refresh_token?appid=APPID&grant_type=refresh_token&refresh_token=REFRESH_TOKEN>

**参数**

|  |  |  |
| --- | --- | --- |
| **参数** | **是否必须** | **说明** |
| appid | 是 | 公众号的唯一标识 |
| grant\_type | 是 | 填写为refresh\_token |
| refresh\_token | 是 | 填写通过access\_token获取到的refresh\_token参数 |

**返回说明**

正确时返回的JSON数据包如下：

{

"access\_token":"ACCESS\_TOKEN",

"expires\_in":7200,

"refresh\_token":"REFRESH\_TOKEN",

"openid":"OPENID",

"scope":"SCOPE"

}

|  |  |
| --- | --- |
| **参数** | **描述** |
| access\_token | 网页授权接口调用凭证,注意：此access\_token与基础支持的access\_token不同 |
| expires\_in | access\_token接口调用凭证超时时间，单位（秒） |
| refresh\_token | 用户刷新access\_token |
| openid | 用户唯一标识 |
| scope | 用户授权的作用域，使用逗号（,）分隔 |

错误时微信会返回JSON数据包如下（示例为Code无效错误）:

{"errcode":40029,"errmsg":"invalid code"}

* 拉取用户信息(需scope为 snsapi\_userinfo)

如果网页授权作用域为snsapi\_userinfo，则此时开发者可以通过access\_token和openid拉取用户信息了。  
**请求方法**

http：GET（请使用https协议）

<https://api.weixin.qq.com/sns/userinfo?access_token=ACCESS_TOKEN&openid=OPENID&lang=zh_CN>

**参数说明:**

|  |  |
| --- | --- |
| **参数** | **描述** |
| access\_token | 网页授权接口调用凭证,注意：此access\_token与基础支持的access\_token不同 |
| openid | 用户的唯一标识 |
| lang | 返回国家地区语言版本，zh\_CN 简体，zh\_TW 繁体，en 英语 |

**返回说明**

正确时返回的JSON数据包如下：

{

"openid":" OPENID",

" nickname": NICKNAME,

"sex":"1",

"province":"PROVINCE"

"city":"CITY",

"country":"COUNTRY",

"headimgurl": "<http://wx.qlogo.cn/mmopen/g3MonUZtNHkdmzicIlibx6iaFqAc56vxLSUfpb6n5WKSYVY0ChQKkiaJSgQ1dZuTOgvLLrhJbERQQ4eMsv84eavHiaiceqxibJxCfHe/46>",

"privilege":[

"PRIVILEGE1"

"PRIVILEGE2"

],

"unionid": "o6\_bmasdasdsad6\_2sgVt7hMZOPfL"

}

|  |  |
| --- | --- |
| **参数** | **描述** |
| openid | 用户的唯一标识 |
| nickname | 用户昵称 |
| sex | 用户的性别，值为1时是男性，值为2时是女性，值为0时是未知 |
| province | 用户个人资料填写的省份 |
| city | 普通用户个人资料填写的城市 |
| country | 国家，如中国为CN |
| headimgurl | 用户头像，最后一个数值代表正方形头像大小（有0、46、64、96、132数值可选，0代表640\*640正方形头像），用户没有头像时该项为空。若用户更换头像，原有头像URL将失效。 |
| privilege | 用户特权信息，json 数组，如微信沃卡用户为（chinaunicom） |
| unionid | 只有在用户将公众号绑定到微信开放平台帐号后，才会出现该字段。详见：[获取用户个人信息（UnionID机制）](https://open.weixin.qq.com/cgi-bin/frame?t=resource/res_main_tmpl&lang=zh_CN&target=res/app_wx_login) |

错误时微信会返回JSON数据包如下（示例为openid无效）:

{"errcode":40003,"errmsg":" invalid openid "}

* 检验授权凭证（access\_token）是否有效

**请求方法**

http：GET（请使用https协议）

<https://api.weixin.qq.com/sns/auth?access_token=ACCESS_TOKEN&openid=OPENID>

**参数说明**

|  |  |
| --- | --- |
| **参数** | **描述** |
| access\_token | 网页授权接口调用凭证,注意：此access\_token与基础支持的access\_token不同 |
| openid | 用户的唯一标识 |

**返回说明**

正确的Json返回结果：

{ "errcode":0,"errmsg":"ok"}

错误时的Json返回示例：

{ "errcode":40003,"errmsg":"invalid openid"}

1. 支付:

方法一:使用微信API

//必须在页面中，引用微信API.js

wx.chooseWXPay({

timestamp: 0, // 支付签名时间戳，注意微信jssdk中的所有使用timestamp字段均为小写。但最新版的支付后台生成签名使用的timeStamp字段名需大写其中的S字符

nonceStr: '', // 支付签名随机串，不长于 32 位

package: '', // 统一支付接口返回的prepay\_id参数值，提交格式如：prepay\_id=\*\*\*）

signType: '', // 签名方式，默认为'SHA1'，使用新版支付需传入'MD5'

paySign: '', // 支付签名

success: function (res) {

// 支付成功后的回调函数

}

});

备注：prepay\_id 通过微信支付统一下单接口拿到，paySign 采用统一的微信支付 Sign 签名生成方法，注意这里 appId 也要参与签名，appId 与 config 中传入的 appId 一致，即最后参与签名的参数有appId, timeStamp, nonceStr, package, signType。

方法二:统一下单

在微信浏览器里面打开H5网页中执行JS调起支付。接口输入输出数据格式为JSON。

注意：WeixinJSBridge内置对象在其他浏览器中无效。

**列表中参数名区分大小，大小写错误签名验证会失败。**

网页端接口参数列表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 变量名 | 必填 | 类型 | 示例值 | 描述 |
| 公众号id | appId | 是 | String(16) | wx8888888888888888 | 商户注册具有支付权限的公众号成功后即可获得 |
| 时间戳 | timeStamp | 是 | String(32) | 1414561699 | 当前的时间，其他详见[时间戳规则](https://pay.weixin.qq.com/wiki/doc/api/jsapi.php?chapter=4_2) |
| 随机字符串 | nonceStr | 是 | String(32) | 5K8264ILTKCH16CQ2502SI8ZNMTM67VS | 随机字符串，不长于32位。推荐[随机数生成算法](https://pay.weixin.qq.com/wiki/doc/api/jsapi.php?chapter=4_3) |
| 订单详情扩展字符串 | package | 是 | String(128) | prepay\_id=123456789 | 统一下单接口返回的prepay\_id参数值，提交格式如：prepay\_id=\*\*\* |
| 签名方式 | signType | 是 | String(32) | MD5 | 签名算法，暂支持MD5 |
| 签名 | paySign | 是 | String(64) | C380BEC2BFD727A4B6845133519F3AD6 | 签名，详见[签名生成算法](https://pay.weixin.qq.com/wiki/doc/api/jsapi.php?chapter=4_3) |

网页内支付接口err\_msg返回结果值说明

|  |  |
| --- | --- |
| 返回值 | 描述 |
| get\_brand\_wcpay\_request：ok | 支付成功 |
| get\_brand\_wcpay\_request：cancel | 支付过程中用户取消 |
| get\_brand\_wcpay\_request：fail | 支付失败 |

注：JS API的返回结果get\_brand\_wcpay\_request：ok仅在用户成功完成支付时返回。由于前端交互复杂，get\_brand\_wcpay\_request：cancel或者get\_brand\_wcpay\_request：fail可以统一处理为用户遇到错误或者主动放弃，不必细化区分。

示例代码如下:

WeixinJSBridge.invoke(  
 **'getBrandWCPayRequest'**, {  
 **"appId"** : dataMap.get(**"appID"**), *//公众号名称，由商户传入* **"timeStamp"**:dataMap.get(**"timeStamp"**), *//时间戳，自1970年以来的秒数* **"nonceStr"** : dataMap.get(**"nonceStr"**), *//随机串* **"package"** : dataMap.get(**"package"**),  
 **"signType"** : **"MD5"**, *//微信签名方式：* **"paySign"** : dataMap.get(**"paySign"**) *//微信签名* },  
 **function**(res){  
 alert(**"微信支付:"**+ res.errr\_msg);  
 *// 使用以上方式判断前端返回,微信团队郑重提示：res.err\_msg将在用户支付成功后返回 ok，但并不保证它绝对可靠。  
 onResponseByWebChat*(res);  
 }  
);

* 后台开发中遇到的坑:

1. 微信后台签名通不过，报错:“签名报错”:

* 环境:

字符串拼接完成之后，以UTF8的格式，发送给微信。

* 原因:

拼接的字符串中，包含中文字符，因此要将生成的xml请求参数转为字节数组后，用“ISO8859-1”的编码格式进行编码为字符串即可。

解决:

**new** String(sb2.toString().getBytes(), "ISO8859-1");

1. 微信web上签名通不过，报错:“支付验证签名失败”:

* 环境:

后台的签名算法，已经通过微信签名校验工具(https://pay.weixin.qq.com/wiki/doc/api/jsapi.php?chapter=20\_1)校验通过。

所以，后台遇到这个问题，一直认为签名算法已经通过，不关乎后台的问题，应该是前台地问题。

而自己，也错误的认为，是自己这边调用接口问题。

* 解决:

在支付过程中，需要两次签名，第二次签名算法，后台使用的是第二次签名时的时间戳。但是此处，却应该使用第一次签名时的时间戳。

1. 微信web上报错:“缺少timeStamp”

同理，“缺少total\_free”

签名时，拼接的xml字符串缺少上述元素，或者语法错误。

1. 微信web上,支付界面一闪而过:

可能原因是:

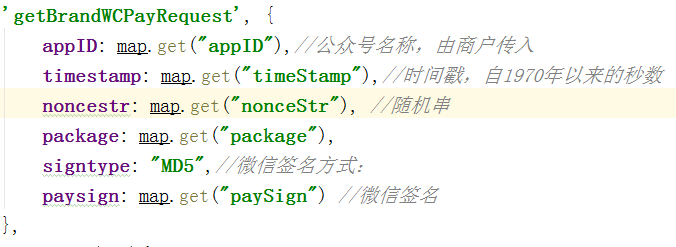
微信支付调用的参数名大小写不一致。

参数名可以不是字符串、timeStamp必须是字符串，唯独参数名的大小写，要一致。



# 微信支付显示当前页面的URL未注册





其中第二、三种写法，不正确，微信支付界面一闪而过。



timeStamp是数值类型