[**C#开发微信门户及应用(4)--关注用户列表及详细信息管理**](http://www.cnblogs.com/wuhuacong/p/3695213.html)

在上个月的对C#开发微信门户及应用做了介绍，写过了几篇的随笔进行分享，由于时间关系，间隔了一段时间没有继续写这个系列的博客了，并不是对这个方面停止了研究，而是继续深入探索这方面的技术，为了更好的应用起来，专心做好底层的技术开发。

微信的很重要的一个特点就是能够利用其平台庞大的用户群体，因此很容易整合在CRM（客户关系管理）系统里面，服务号和订阅好都能够向关注者推送相关的产品消息，还能和48小时内响应消息和事件的活跃用户进行交互对话，因此用户信息是微信API里面非常重要的一环，本随笔主要介绍获取关注用户、查看用户信息、分组管理等方面的开发应用。

**1、关注用户列表及用户分组信息**

在微信的管理平台上，我们可以看到自己账号的关注者用户，以及用户分组信息，如下所示。



上面的管理界面，能看到关注者用户的基础信息，但是使用微信API获取到的是一个称之为OpenID的列表，我们先了解这个东西是什么？微信API的说明给出下面的解析：

关注者列表由一串OpenID（加密后的微信号，每个用户对每个公众号的OpenID是唯一的。对于不同公众号，同一用户的openid不同）组成。公众号可通过本接口来根据OpenID获取用户基本信息，包括昵称、头像、性别、所在城市、语言和关注时间。

上面的解析意思很清楚了，就是一个用户关注我们的公众号，那么不管他是第几次关注，对我们公众号来说，都是一个确定的值；但是，一个用户对其他公众号，却有着其他不同的OpenID。

微信提供了为数不多的几个关键字信息，用来记录用户的相关内容，根据用户的相关定义，我们定义一个实体类，用来放置获取回来的用户信息。

[复制代码](javascript:void(0);)

/// <summary>

/// 高级接口获取的用户信息。

/// 在关注者与公众号产生消息交互后，公众号可获得关注者的OpenID

/// （加密后的微信号，每个用户对每个公众号的OpenID是唯一的。对于不同公众号，同一用户的openid不同）。

/// 公众号可通过本接口来根据OpenID获取用户基本信息，包括昵称、头像、性别、所在城市、语言和关注时间。

/// </summary>

public class UserJson : BaseJsonResult

{

/// <summary>

/// 用户是否订阅该公众号标识，值为0时，代表此用户没有关注该公众号，拉取不到其余信息。

/// </summary>

public int subscribe { get; set; }

/// <summary>

/// 用户的标识，对当前公众号唯一

/// </summary>

public string openid { get; set; }

/// <summary>

/// 用户的昵称

/// </summary>

public string nickname { get; set; }

/// <summary>

/// 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

/// </summary>

public int sex { get; set; }

/// <summary>

/// 用户的语言，简体中文为zh\_CN

/// </summary>

public string language { get; set; }

/// <summary>

/// 用户所在城市

/// </summary>

public string city { get; set; }

/// <summary>

/// 用户所在省份

/// </summary>

public string province { get; set; }

/// <summary>

/// 用户所在国家

/// </summary>

public string country { get; set; }

/// <summary>

/// 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

/// </summary>

public string headimgurl { get; set; }

/// <summary>

/// 用户关注时间，为时间戳。如果用户曾多次关注，则取最后关注时间

/// </summary>

public long subscribe\_time { get; set; }

}

[复制代码](javascript:void(0);)

根据分组信息定义，我们定义一个分组的实体类信息。

[复制代码](javascript:void(0);)

/// <summary>

/// 分组信息

/// </summary>

public class GroupJson : BaseJsonResult

{

/// <summary>

/// 分组id，由微信分配

/// </summary>

public int id { get; set; }

/// <summary>

/// 分组名字，UTF8编码

/// </summary>

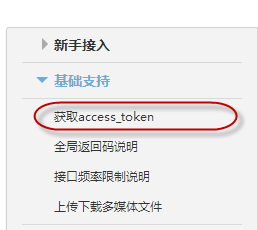
public string name { get; set; }

}

[复制代码](javascript:void(0);)

**2、获取AIP调用者的的Token**

在做微信API的开发，很多时候，我们都需要传入一个AccessToken，这个就是区分调用者和记录会话信息的字符串，因此，在学习所有API开发之前，我们需要很好理解这个访问控制参数。



这个对象的定义，我们可以从微信的API说明中了解。

access\_token是公众号的全局唯一票据，公众号调用各接口时都需使用access\_token。正常情况下**access\_token有效期为7200秒**，重复获取将导致上次获取的access\_token失效。**由于获取access\_token的api调用次数非常有限，建议开发者全局存储与更新access\_token，频繁刷新access\_token会导致api调用受限，影响自身业务。**

根据上面的说明定义，我们可以看到，它是一个和身份，以及会话时间有关的一个参数，而且它的产生次数有限制，因此要求我们需要对它进行缓存并重复利用，在会话到期之前，我们应该尽可能重用这个参数，避免反复请求，增加服务器压力，以及调用的时间。

我定义了一个方法，用来构造生成相关的Access Token，而且它具有缓存的功能，但具体如何缓存及使用，对我API的调用是透明的，我们只要用的时候，就对它调用就是了。

/// 获取凭证接口

/// </summary>

/// <param name="appid">第三方用户唯一凭证</param>

/// <param name="secret">第三方用户唯一凭证密钥，既appsecret</param>

string GetAccessToken(string appid, string secret);

缓存主要是基于.NET4增加的类库MemoryCache，这个是一个非常不错的缓存类。

我的获取AccessToken的操作实现代码如下所示。

[复制代码](javascript:void(0);)

/// <summary>

/// 获取每次操作微信API的Token访问令牌

/// </summary>

/// <param name="appid">应用ID</param>

/// <param name="secret">开发者凭据</param>

/// <returns></returns>

public string GetAccessToken(string appid, string secret)

{

//正常情况下access\_token有效期为7200秒,这里使用缓存设置短于这个时间即可

string access\_token = MemoryCacheHelper.GetCacheItem<string>("access\_token", delegate()

{

string grant\_type = "client\_credential";

var url = string.Format("https://api.weixin.qq.com/cgi-bin/token?grant\_type={0}&appid={1}&secret={2}",

grant\_type, appid, secret);

HttpHelper helper = new HttpHelper();

string result = helper.GetHtml(url);

string regex = "\"access\_token\":\"(?<token>.\*?)\"";

string token = CRegex.GetText(result, regex, "token");

return token;

},

new TimeSpan(0, 0, 7000)//7000秒过期

);

return access\_token;

}

[复制代码](javascript:void(0);)

由于我们知道，AccessToken默认是7200秒过期，因此在这个时间段里面，我们尽可能使用缓存来记录它的值，如果超过了这个时间，我们调用这个方法的时候，它会自动重新获取一个新的值给我们了。

**3、获取关注用户列表**

获取关注用户列表，一次拉取API调用，最多拉取10000个关注者的OpenID，可以通过多次拉取的方式来满足需求。微信的接口定义如下所示。

http请求方式: GET（请使用https协议）  
https://api.weixin.qq.com/cgi-bin/user/get?access\_token=ACCESS\_TOKEN&next\_openid=NEXT\_OPENID

这个接口返回的数据是

{"total":2,"count":2,"data":{"openid":["","OPENID1","OPENID2"]},"next\_openid":"NEXT\_OPENID"}

根据返回的Json数据定义，我们还需要定义两个实体类，用来存放返回的结果。

[复制代码](javascript:void(0);)

/// <summary>

/// 获取关注用户列表的Json结果

/// </summary>

public class UserListJsonResult : BaseJsonResult

{

/// <summary>

/// 关注该公众账号的总用户数

/// </summary>

public int total { get; set; }

/// <summary>

/// 拉取的OPENID个数，最大值为10000

/// </summary>

public int count { get; set; }

/// <summary>

/// 列表数据，OPENID的列表

/// </summary>

public OpenIdListData data { get; set; }

/// <summary>

/// 拉取列表的后一个用户的OPENID

/// </summary>

public string next\_openid { get; set; }

}

/// <summary>

/// 列表数据，OPENID的列表

/// </summary>

public class OpenIdListData

{

/// <summary>

/// OPENID的列表

/// </summary>

public List<string> openid { get; set; }

}

[复制代码](javascript:void(0);)

为了获取相关的用户信息，我定义了一个接口，用来获取用户的信息，接口定义如下所示。

[复制代码](javascript:void(0);)

/// <summary>

/// 微信用户管理的API接口

/// </summary>

public interface IUserApi

{

/// <summary>

/// 获取关注用户列表

/// </summary>

/// <param name="accessToken">调用接口凭证</param>

/// <param name="nextOpenId">第一个拉取的OPENID，不填默认从头开始拉取</param>

/// <returns></returns>

List<string> GetUserList(string accessToken, string nextOpenId = null);

/// <summary>

/// 获取用户基本信息

/// </summary>

/// <param name="accessToken">调用接口凭证</param>

/// <param name="openId">普通用户的标识，对当前公众号唯一</param>

/// <param name="lang">返回国家地区语言版本，zh\_CN 简体，zh\_TW 繁体，en 英语</param>

UserJson GetUserDetail(string accessToken, string openId, Language lang = Language.zh\_CN);

[复制代码](javascript:void(0);)

然后在实现类里面，我们分别对上面两个接口进行实现，获取用户列表信息如下所示。

[复制代码](javascript:void(0);)

/// <summary>

/// 获取关注用户列表

/// </summary>

/// <param name="accessToken">调用接口凭证</param>

/// <param name="nextOpenId">第一个拉取的OPENID，不填默认从头开始拉取</param>

/// <returns></returns>

public List<string> GetUserList(string accessToken, string nextOpenId = null)

{

List<string> list = new List<string>();

string url = string.Format("https://api.weixin.qq.com/cgi-bin/user/get?access\_token={0}", accessToken);

if (!string.IsNullOrEmpty(nextOpenId))

{

url += "&next\_openid=" + nextOpenId;

}

UserListJsonResult result = JsonHelper<UserListJsonResult>.ConvertJson(url);

if (result != null && result.data != null)

{

list.AddRange(result.data.openid);

}

return list;

}

[复制代码](javascript:void(0);)

我们看到，转换的逻辑已经放到了JsonHelper里面去了，这个辅助类里面分别对数值进行了获取内容，验证返回值，然后转换正确实体类几个部分的操作。

获取内容，通过辅助类HttpHelper进行，这个在我的公用类库里面，里面的逻辑主要就是通过HttpRequest进行数据的获取操作，不在赘述。

HttpHelper helper = new HttpHelper();

string content = helper.GetHtml(url);

由于返回的内容，我们需要判断它是否正确返回所需的结果，如果没有，抛出自定义的相关异常，方便处理，具体如下所示。

[复制代码](javascript:void(0);)

/// <summary>

/// 检查返回的记录，如果返回没有错误，或者结果提示成功，则不抛出异常

/// </summary>

/// <param name="content">返回的结果</param>

/// <returns></returns>

private static bool VerifyErrorCode(string content)

{

if (content.Contains("errcode"))

{

ErrorJsonResult errorResult = JsonConvert.DeserializeObject<ErrorJsonResult>(content);

//非成功操作才记录异常，因为有些操作是返回正常的结果（{"errcode": 0, "errmsg": "ok"}）

if (errorResult != null && errorResult.errcode != ReturnCode.请求成功)

{

string error = string.Format("微信请求发生错误！错误代码：{0}，说明：{1}", (int)errorResult.errcode, errorResult.errmsg);

LogTextHelper.Error(errorResult);

throw new WeixinException(error);//抛出错误

}

}

return true;

}

[复制代码](javascript:void(0);)

然后转换为相应的格式，就是通过Json.NET的类库进行转换。

T result = JsonConvert.DeserializeObject<T>(content);

return result;

这样我们就可以在ConvertJson函数实体里面，完整的进行处理和转换了，转换完整的函数代码如下所示。

[复制代码](javascript:void(0);)

/// <summary>

/// Json字符串操作辅助类

/// </summary>

public class JsonHelper<T> where T : class, new()

{

/// <summary>

/// 检查返回的记录，如果返回没有错误，或者结果提示成功，则不抛出异常

/// </summary>

/// <param name="content">返回的结果</param>

/// <returns></returns>

private static bool VerifyErrorCode(string content)

{

if (content.Contains("errcode"))

{

ErrorJsonResult errorResult = JsonConvert.DeserializeObject<ErrorJsonResult>(content);

//非成功操作才记录异常，因为有些操作是返回正常的结果（{"errcode": 0, "errmsg": "ok"}）

if (errorResult != null && errorResult.errcode != ReturnCode.请求成功)

{

string error = string.Format("微信请求发生错误！错误代码：{0}，说明：{1}", (int)errorResult.errcode, errorResult.errmsg);

LogTextHelper.Error(errorResult);

throw new WeixinException(error);//抛出错误

}

}

return true;

}

/// <summary>

/// 转换Json字符串到具体的对象

/// </summary>

/// <param name="url">返回Json数据的链接地址</param>

/// <returns></returns>

public static T ConvertJson(string url)

{

HttpHelper helper = new HttpHelper();

string content = helper.GetHtml(url);

VerifyErrorCode(content);

T result = JsonConvert.DeserializeObject<T>(content);

return result;

}

}

[复制代码](javascript:void(0);)

调用这个API的界面层代码如下所示（测试代码）

IUserApi userBLL = new UserApi();

List<string> userList = userBLL.GetUserList(token)

**4、获取用户详细信息**

上面的获取列表操作，相对比较简单，而且不用POST任何数据，因此通过Get协议就能获取到所需的数据。

本小节继续介绍获取用户详细信息的操作，这个操作也是通过GET协议就可以完成的。

这个API的调用定义如下所示：

http请求方式: GET

https://api.weixin.qq.com/cgi-bin/user/info?access\_token=ACCESS\_TOKEN&openid=OPENID&lang=zh\_CN

通过传入一个OpenId，我们就能很好获取到用户的相关信息了。

前面小节我们已经定义了它的接口，说明了传入及返回值，根据定义，它的实现函数如下所示。

[复制代码](javascript:void(0);)

/// <summary>

/// 获取用户基本信息

/// </summary>

/// <param name="accessToken">调用接口凭证</param>

/// <param name="openId">普通用户的标识，对当前公众号唯一</param>

/// <param name="lang">返回国家地区语言版本，zh\_CN 简体，zh\_TW 繁体，en 英语</param>

public UserJson GetUserDetail(string accessToken, string openId, Language lang = Language.zh\_CN)

{

string url = string.Format("https://api.weixin.qq.com/cgi-bin/user/info?access\_token={0}&openid={1}&lang={2}",

accessToken, openId, lang.ToString());

UserJson result = JsonHelper<UserJson>.ConvertJson(url);

return result;

}

[复制代码](javascript:void(0);)

最后，我们结合获取用户列表和获取用户详细信息的两个API，我们看看调用的代码（测试代码）。

[复制代码](javascript:void(0);)

private void btnGetUsers\_Click(object sender, EventArgs e)

{

IUserApi userBLL = new UserApi();

List<string> userList = userBLL.GetUserList(token);

foreach (string openId in userList)

{

UserJson userInfo = userBLL.GetUserDetail(token, openId);

if (userInfo != null)

{

string tips = string.Format("{0}:{1}", userInfo.nickname, userInfo.openid);

Console.WriteLine(tips);

}

}

}