**[C#开发微信门户及应用(5)--用户分组信息管理](http://www.cnblogs.com/wuhuacong/p/3695351.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。本篇继续上一篇的介绍，主要介绍分组管理方面的开发应用，这篇的内容和上一篇，作为一个完整的用户信息和分组信息管理的组合。

**1、用户分组管理内容**

用户分组的引入，主要是方便管理关注者列表，以及方便向不同的组别发送消息的操作的，一个公众账号，最多支持创建500个分组。

用户分组管理，包含下面几个方面的内容：

1 创建分组  
2 查询所有分组  
3 查询用户所在分组  
4 修改分组名  
5 移动用户分组

微信对于创建分组的定义如下所示。

http请求方式: POST（请使用https协议）

https://api.weixin.qq.com/cgi-bin/groups/create?access\_token=ACCESS\_TOKEN

POST数据格式：json

POST数据例子：{"group":{"name":"test"}}

正常返回的结果如下所示。

{

"group": {

"id": 107,

"name": "test"

}

}

其他接口，也是类似的方式，通过POST一些参数进去URL里面，获取返回的Json数据。

前面随笔定义了GroupJson的实体类信息如下所示。

[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);)

根据以上几个接口的定义，我定义了几个接口，并把它们归纳到用户管理的API接口里面。

[复制代码](javascript:void(0);)

/// <summary>

/// 查询所有分组

/// </summary>

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

/// <returns></returns>

List<GroupJson> GetGroupList(string accessToken);

/// <summary>

/// 创建分组

/// </summary>

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

/// <param name="name">分组名称</param>

/// <returns></returns>

GroupJson CreateGroup(string accessToken, string name);

/// <summary>

/// 查询用户所在分组

/// </summary>

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

/// <param name="openid">用户的OpenID</param>

/// <returns></returns>

int GetUserGroupId(string accessToken, string openid);

/// <summary>

/// 修改分组名

/// </summary>

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

/// <param name="id">分组id，由微信分配</param>

/// <param name="name">分组名字（30个字符以内）</param>

/// <returns></returns>

CommonResult UpdateGroupName(string accessToken, int id, string name);

/// <summary>

/// 移动用户分组

/// </summary>

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

/// <param name="openid">用户的OpenID</param>

/// <param name="to\_groupid">分组id</param>

/// <returns></returns>

CommonResult MoveUserToGroup(string accessToken, string openid, int to\_groupid);

[复制代码](javascript:void(0);)

**2、用户分组管理接口的实现**

**2.1 创建用户分组**

为了解析如何实现创建用户分组的POST数据操作，我们来一步步了解创建用户的具体过程。

首先需要创建一个动态定义的实体类信息，它包含几个需要提及的属性，如下所示。

[复制代码](javascript:void(0);)

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

var data = new

{

group = new

{

name = name

}

};

string postData = data.ToJson();

[复制代码](javascript:void(0);)

其中我们把对象转换为合适的Json数据操作，放到了扩展方法ToJson里面了，这个主要就是方便把动态定义的实体类转换Json内容，主要就是调用Json.NET的序列号操作。

[复制代码](javascript:void(0);)

/// <summary>

/// 把对象为json字符串

/// </summary>

/// <param name="obj">待序列号对象</param>

/// <returns></returns>

public static string ToJson(this object obj)

{

return JsonConvert.SerializeObject(obj, Formatting.Indented);

}

[复制代码](javascript:void(0);)

准备好Post的数据后，我们就进一步看看获取数据并转换为合适格式的操作代码。

[复制代码](javascript:void(0);)

GroupJson group = null;

CreateGroupResult result = JsonHelper<CreateGroupResult>.ConvertJson(url, postData);

if (result != null)

{

group = result.group;

}

[复制代码](javascript:void(0);)

其中POST数据并转换为合适格式实体类的操作，放在了ConvertJson方法里面，这个方法的定义如下所示，里面的HttpHelper是我公用类库的辅助类，主要就是调用底层的httpWebRequest对象方法，进行数据的提交，并获取返回结果。

[复制代码](javascript:void(0);)

/// <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>

/// <param name="postData">POST提交的数据</param>

/// <returns></returns>

public static T ConvertJson(string url, string postData)

{

HttpHelper helper = new HttpHelper();

string content = helper.GetHtml(url, postData, true);

VerifyErrorCode(content);

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

return result;

}

[复制代码](javascript:void(0);)

这样，完整的创建用户分组的操作函数如下所示。

[复制代码](javascript:void(0);)

/// <summary>

/// 创建分组

/// </summary>

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

/// <param name="name">分组名称</param>

/// <returns></returns>

public GroupJson CreateGroup(string accessToken, string name)

{

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

var data = new

{

group = new

{

name = name

}

};

string postData = data.ToJson();

GroupJson group = null;

CreateGroupResult result = JsonHelper<CreateGroupResult>.ConvertJson(url, postData);

if (result != null)

{

group = result.group;

}

return group;

}

[复制代码](javascript:void(0);)

**2.2 查询所有分组**

查询所有分组，可以把服务器上的分组全部获取下来，也就是每个分组的ID和名称。

[复制代码](javascript:void(0);)

/// <summary>

/// 查询所有分组

/// </summary>

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

/// <returns></returns>

public List<GroupJson> GetGroupList(string accessToken)

{

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

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

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

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

{

list.AddRange(result.groups);

}

return list;

}

[复制代码](javascript:void(0);)

**2.3 查询用户所在分组**

每个用户都属于一个分组，默认在 **未分组** 这个分组里面，我们可以通过API获取用户的分组信息，也就是获取所在用户分组的ID。

[复制代码](javascript:void(0);)

/// <summary>

/// 查询用户所在分组

/// </summary>

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

/// <param name="openid">用户的OpenID</param>

/// <returns></returns>

public int GetUserGroupId(string accessToken, string openid)

{

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

var data = new

{

openid = openid

};

string postData = data.ToJson();

int groupId = -1;

GroupIdJsonResult result = JsonHelper<GroupIdJsonResult>.ConvertJson(url, postData);

if (result != null)

{

groupId = result.groupid;

}

return groupId;

}

[复制代码](javascript:void(0);)

**2.4 修改分组名称**

也可以在实际中，调整用户所在的分组，操作代码如下。

[复制代码](javascript:void(0);)

/// <summary>

/// 修改分组名

/// </summary>

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

/// <param name="id">分组id，由微信分配</param>

/// <param name="name">分组名字（30个字符以内）</param>

/// <returns></returns>

public CommonResult UpdateGroupName(string accessToken, int id, string name)

{

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

var data = new

{

group = new

{

id = id,

name = name

}

};

string postData = data.ToJson();

return Helper.GetExecuteResult(url, postData);

}

[复制代码](javascript:void(0);)

这里的返回值CommonResult是，一个实体类，包含了bool的成功与否的标志，以及String类型的错误信息（如果有的话）。

对于这个GetExecuteResult函数体，里面主要就是提交数据，然后获取结果，并根据结果进行处理的函数。

[复制代码](javascript:void(0);)

/// <summary>

/// 通用的操作结果

/// </summary>

/// <param name="url">网页地址</param>

/// <param name="postData">提交的数据内容</param>

/// <returns></returns>

public static CommonResult GetExecuteResult(string url, string postData = null)

{

CommonResult success = new CommonResult();

try

{

ErrorJsonResult result;

if (postData != null)

{

result = JsonHelper<ErrorJsonResult>.ConvertJson(url, postData);

}

else

{

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

}

if (result != null)

{

success.Success = (result.errcode == ReturnCode.请求成功);

success.ErrorMessage = result.errmsg;

}

}

catch (WeixinException ex)

{

success.ErrorMessage = ex.Message;

}

return success;

}

}

[复制代码](javascript:void(0);)

上面红色部分的意思，就是转换为实体类的时候，如果错误是微信里面定义的，那么记录错误信息，其他异常我不处理（也就是抛出去）。

**2.5 移动用户到新的分组**

移动用户到新的分组的操作和上面小节的差不多，具体看代码。

[复制代码](javascript:void(0);)

/// <summary>

/// 移动用户分组

/// </summary>

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

/// <param name="openid">用户的OpenID</param>

/// <param name="to\_groupid">分组id</param>

/// <returns></returns>

public CommonResult MoveUserToGroup(string accessToken, string openid, int to\_groupid)

{

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

var data = new

{

openid = openid,

to\_groupid = to\_groupid

};

string postData = data.ToJson();

return Helper.GetExecuteResult(url, postData);

}

[复制代码](javascript:void(0);)

**3、用户分组接口的调用**

上面小节，定义并实现了用户分组的各类接口，所有的用户相关的都已经毫无保留贴出代码，它的调用操作如下代码所示（测试代码）。

[复制代码](javascript:void(0);)

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

{

IUserApi userBLL = new UserApi();

List<GroupJson> list = userBLL.GetGroupList(token);

foreach (GroupJson info in list)

{

string tips = string.Format("{0}:{1}", info.name, info.id);

Console.WriteLine(tips);

}

}

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

{

IUserApi userBLL = new UserApi();

int groupId = userBLL.GetUserGroupId(token, openId);

string tips = string.Format("GroupId:{0}", groupId);

Console.WriteLine(tips);

}

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

{

IUserApi userBLL = new UserApi();

GroupJson info = userBLL.CreateGroup(token, "创建测试分组");

if (info != null)

{

string tips = string.Format("GroupId:{0} GroupName:{1}", info.id, info.name);

Console.WriteLine(tips);

string newName = "创建测试修改";

CommonResult result = userBLL.UpdateGroupName(token, info.id, newName);

Console.WriteLine("修改分组名称：" + (result.Success ? "成功" : "失败:" + result.ErrorMessage));

}

}

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

{

int groupId = 111;

string newName = "创建测试修改";

IUserApi userBLL = new UserApi();

CommonResult result = userBLL.UpdateGroupName(token, groupId, newName);

Console.WriteLine("修改分组名称：" + (result.Success ? "成功" : "失败:" + result.ErrorMessage));

}

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

{

int togroup\_id = 111;//输入分组ID

if (togroup\_id > 0)

{

IUserApi userBLL = new UserApi();

CommonResult result = userBLL.MoveUserToGroup(token, openId, togroup\_id);

Console.WriteLine("移动用户分组名称：" + (result.Success ? "成功" : "失败:" + result.ErrorMessage));

}

}

[复制代码](javascript:void(0);)

了解了上面的代码和调用规则，我们就能通过API进行用户分组信息的管理了。通过在应用程序中集成相关的接口代码，我们就能够很好的控制我们的关注用户列表和用户分组信息。从而为我们下一步用户的信息推送打好基础。