**[C#开发微信门户及应用(24)-微信小店货架信息管理](http://www.cnblogs.com/wuhuacong/p/4203869.html)**

在前面微信小店系列篇《[C#开发微信门户及应用(22)-微信小店的开发和使用](http://www.cnblogs.com/wuhuacong/p/4203050.html)》里面介绍了一些微信小店的基础知识，以及《[C#开发微信门户及应用(23)-微信小店商品管理接口的封装和测试](http://www.cnblogs.com/wuhuacong/p/4203139.html)》里面详细介绍了微信小店商品的接口定义、实现和测试的内容，本文主要介绍微信小店货架信息管理。这个模块是在微信小店对象里面，最为复杂，也是最难理解的一个模块，对于它的对象建模，需要反复测试才能完善起来，因此这个货架管理模块，可以说是最具技术含量的一个模块了。

**1、微信小店货架介绍**

在微信公众号的后台里面，可以对货架信息进行维护，界面如下所示。货架的概念，就是把商品分门别类的很好展示给客户，货架就是类似一个布局良好的展柜，我们可以定义不同的货架，然后公布不同的URL进行体验。



另外，我们一般创建货架，都是基于货架的模板库来构建的，货架的模板给我们快速构建一个货架，提供了可视化的参考界面，货架模板界面如下所示。



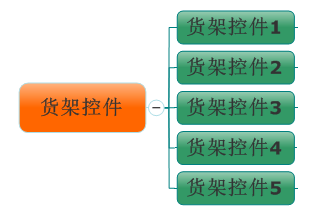
**2、货架管理的开发模型**

对于利用API开发微信店铺，微信小店的货架管理操作接口，和常规的模块差不多，具有下面几个功能操作。



虽然看起来和前面的对象模型差不多，但是货架的信息非常复杂，因此你如果需要根据Json数据把它还原为实体对象的时候，需要反复进行斟酌，否则很容易建模错误。

对应着微信小店管理界面的货架模板，货架的对象信息包括了5个不同的控件模型，它们有的可以进行组合使用。



几个货架的模型展示如下所示。







通过上面5个控件模型，我们可以看到它们分别代表不一样的布局效果，而且它们可以在货架上进行组合使用的。

**3、货架信息的对象建模**

根据微信小店的接口说明，我们最终定义的货架实体对象信息，内容就非常丰富和弹性化。



我们通过参考微信小店的API说明，就可以看到货架的信息JSON数据很复杂，具体定义如下所示。

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifhttp://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

{

"shelf\_data": {

"module\_infos": [

{

"group\_info": {

"filter": {

"count": 2

},

"group\_id": 50

},

"eid": 1

},

{

"group\_infos": {

"groups": [

{

"group\_id": 49

},

{

"group\_id": 50

},

{

"group\_id": 51

}

]

},

"eid": 2

},

{

"group\_info": {

"group\_id": 52,

"img": "http://mmbiz.qpic.cn/mmbiz/4whpV1VZl29nqqObBwFwnIX3licVPnFV5Jm64z4I0TTicv0TjN7Vl9bykUUibYKIOjicAwIt6Oy0Y6a1Rjp5Tos8tg/0"

},

"eid": 3

},

{

"group\_infos": {

"groups": [

{

"group\_id": 49,

"img": "http://mmbiz.qpic.cn/mmbiz/4whpV1VZl29nqqObBwFwnIX3licVPnFV5uUQx7TLx4tB9qZfbe3JmqR4NkkEmpb5LUWoXF1ek9nga0IkeSSFZ8g/0"

},

{

"group\_id": 50,

"img": "http://mmbiz.qpic.cn/mmbiz/4whpV1VZl29nqqObBwFwnIX3licVPnFV5G1kdy3ViblHrR54gbCmbiaMnl5HpLGm5JFeENyO9FEZAy6mPypEpLibLA/0"

},

{

"group\_id": 52,

"img": "http://mmbiz.qpic.cn/mmbiz/4whpV1VZl29nqqObBwFwnIX3licVPnFV5uUQx7TLx4tB9qZfbe3JmqR4NkkEmpb5LUWoXF1ek9nga0IkeSSFZ8g/0"

}

]

},

"eid": 4

},

{

"group\_infos": {

"groups": [

{

"group\_id": 43

},

{

"group\_id": 44

},

{

"group\_id": 45

},

{

"group\_id": 46

}

],

"img\_background": "http://mmbiz.qpic.cn/mmbiz/4whpV1VZl29nqqObBwFwnIX3licVPnFV5uUQx7TLx4tB9qZfbe3JmqR4NkkEmpb5LUWoXF1ek9nga0IkeSSFZ8g/0"

},

"eid": 5

}

]

},

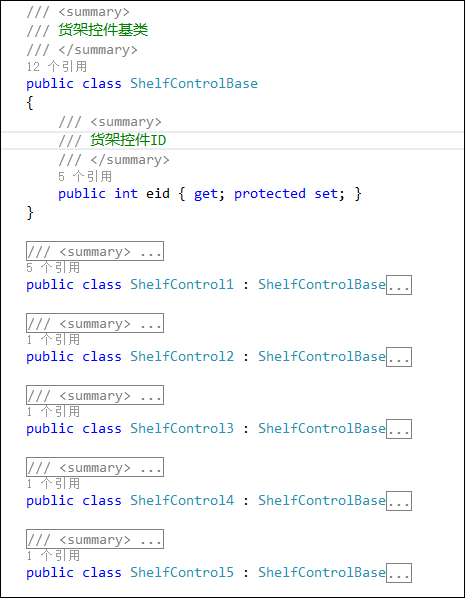
"shelf\_banner": "http://mmbiz.qpic.cn/mmbiz/4whpV1VZl2ibrWQn8zWFUh1YznsMV0XEiavFfLzDWYyvQOBBszXlMaiabGWzz5B2KhNn2IDemHa3iarmCyribYlZYyw/0",

"shelf\_name": "测试货架"

}

View Code

我们根据JSON数据的定义，定义了几个货架控件的对象，他们的关系如下所示。



我们可以根据JSON数据进行实体对象的建模，然后有了这些对象，我们就可以进一步定义好货架的相关操作接口了，接口定义如下所示。

[复制代码](javascript:void(0);)

#region 货架管理

/// <summary>

/// 增加货架

/// </summary>

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

/// <param name="shelfBanner">货架招牌图片Url</param>

/// <param name="shelfName">货架名称</param>

/// <param name="controls">货架控件1,2,3,4,5类型的集合</param>

/// <returns></returns>

AddShelfResult AddShelf(string accessToken, string shelfBanner, string shelfName, List<ShelfControlBase> controls);

/// <summary>

/// 删除货架

/// </summary>

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

/// <param name="shelfId">货架Id</param>

/// <returns></returns>

CommonResult DeleteShelf(string accessToken, int shelfId);

/// <summary>

/// 修改货架

/// </summary>

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

/// <param name="shelfId">货架Id</param>

/// <param name="shelfBanner">货架招牌图片Url</param>

/// <param name="shelfName">货架名称</param>

/// <param name="controls">货架控件1,2,3,4,5类型的集合</param>

/// <returns></returns>

CommonResult UpdateShelf(string accessToken, int shelfId, string shelfBanner, string shelfName, List<ShelfControlBase> controls);

/// <summary>

/// 获取所有货架

/// </summary>

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

/// <returns></returns>

List<ShelfJson> GetAllShelf(string accessToken);

/// <summary>

/// 根据货架ID获取货架信息

/// </summary>

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

/// <param name="shelfId">货架Id</param>

/// <returns></returns>

ShelfJson GetShelfById(string accessToken, int shelfId);

#endregion

[复制代码](javascript:void(0);)

有了这些接口的定义，我们就需要实现对应的接口，从而实现我们向微信API的封装处理了。

微信小店的货架管理实现内容如下所示（部分内容，增删改）。

[复制代码](javascript:void(0);)

/// <summary>

/// 增加货架

/// </summary>

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

/// <param name="shelfBanner">货架招牌图片Url</param>

/// <param name="shelfName">货架名称</param>

/// <param name="controls">货架控件1,2,3,4,5类型的集合</param>

/// <returns></returns>

public AddShelfResult AddShelf(string accessToken, string shelfBanner, string shelfName, List<ShelfControlBase> controls)

{

var url = string.Format("https://api.weixin.qq.com/merchant/shelf/add?access\_token={0}", accessToken);

var data = new

{

shelf\_data = new

{

module\_infos = controls

},

shelf\_banner = shelfBanner,

shelf\_name = shelfName

};

string postData = data.ToJson();

return JsonHelper<AddShelfResult>.ConvertJson(url, postData);

}

/// <summary>

/// 删除货架

/// </summary>

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

/// <param name="shelfId">货架Id</param>

/// <returns></returns>

public CommonResult DeleteShelf(string accessToken, int shelfId)

{

var url = string.Format("https://api.weixin.qq.com/merchant/shelf/del?access\_token={0}", accessToken);

var data = new

{

shelf\_id = shelfId

};

string postData = data.ToJson();

return Helper.GetExecuteResult(url, postData);

}

/// <summary>

/// 修改货架

/// </summary>

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

/// <param name="shelfId">货架Id</param>

/// <param name="shelfBanner">货架招牌图片Url</param>

/// <param name="shelfName">货架名称</param>

/// <param name="controls">货架控件1,2,3,4,5类型的集合</param>

/// <returns></returns>

public CommonResult UpdateShelf(string accessToken, int shelfId, string shelfBanner, string shelfName, List<ShelfControlBase> controls)

{

var url = string.Format("https://api.weixin.qq.com/merchant/shelf/mod?access\_token={0}", accessToken);

var data = new

{

shelf\_id = shelfId,

shelf\_data = new

{

module\_infos = controls

},

shelf\_banner = shelfBanner,

shelf\_name = shelfName

};

string postData = data.ToJson();

return Helper.GetExecuteResult(url, postData);

}

[复制代码](javascript:void(0);)

**4、微信小店货架管理的接口测试**

由于货架管理的对象和接口定义比较复杂一些，一定需要进行反复的测试才能正式使用，如果不注意有可能你定义的实体类，获取不到某个字段信息。

我为了方便，创建了一个Winform项目，分别对各个接口进行测试。



对于货架管理内容的接口测试，测试代码如下所示。

[复制代码](javascript:void(0);)

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

{

IMerchantApi api = new MerchantApi();

List<ShelfJson> list = api.GetAllShelf(token);

Console.WriteLine(list.ToJson());

foreach(ShelfJson json in list)

{

Console.WriteLine("货架信息：");

ShelfJson getJson = api.GetShelfById(token, json.shelf\_id.Value);

Console.WriteLine(getJson.ToJson());

}

string shelf\_banner = "http://mmbiz.qpic.cn/mmbiz/mLqH9gr11Gyb2sgiaelcsxYtQENGePp0RgeNlAQicfZQokjbJMUq4h8MHtjpekJNEWKuMN3gdRz5RxfkYb7NlIrw/0";

string shelf\_name = "测试货架";

ShelfControl1 c11 = new ShelfControl1(6, 202797386);

ShelfControl1 c12 = new ShelfControl1(4, 202797397);

List<ShelfControlBase> controlList = new List<ShelfControlBase>(){c11, c12};

AddShelfResult result = api.AddShelf(token, shelf\_banner, shelf\_name, controlList);

if (result != null && result.shelf\_id > 0)

{

Console.WriteLine("增加的货架信息：");

ShelfJson getJson = api.GetShelfById(token, result.shelf\_id);

Console.WriteLine(getJson.ToJson());

shelf\_name = "测试货架-修改";

controlList = new List<ShelfControlBase>(){c11};

CommonResult updateReuslt = api.UpdateShelf(token, result.shelf\_id, shelf\_banner, shelf\_name, controlList);

Console.WriteLine("修改货架操作：{0}", updateReuslt.Success ? "成功" : "失败");

CommonResult deleteResult = api.DeleteShelf(token, result.shelf\_id);

Console.WriteLine("删除货架操作：{0}", deleteResult.Success ? "成功" : "失败");

}

}

[复制代码](javascript:void(0);)

