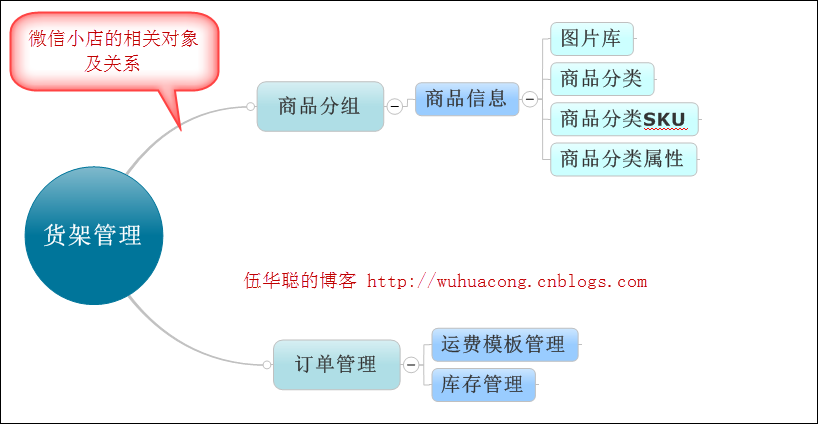
[**C#开发微信门户及应用(23)-微信小店商品管理接口的封装和测试**](http://www.cnblogs.com/wuhuacong/p/4203139.html)

在上篇《[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，以及对应的对象模型，本篇继续微信小店的主题，介绍其中API接口的封装和测试使用。微信小店的相关对象模型，基本上包括了常规的商品、商品分组、货架、库存、订单这些模型，还有商品分类，商品分类属性、商品分类SKU、快递邮寄模板、图片管理等功能。本文介绍的接口封装也就是基于这些内容进行的，并针对接口的实现进行测试和使用。

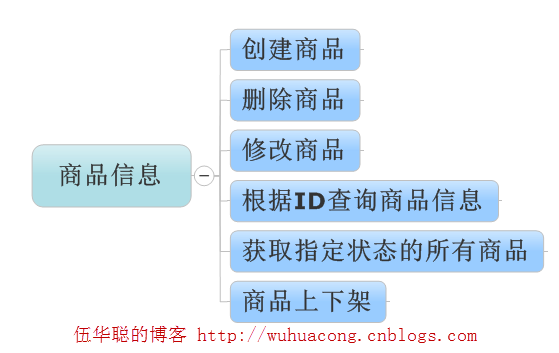
**1、商品管理接口的定义**

前面文章介绍了微信小店的对象模型，如下所示。



这个图形基本上覆盖了微信小店的相关对象，并介绍了它们之间的关系了。

我们从基础的商品信息管理入手，我们知道，商品接口包含了增加、修改、查询、删除等接口，如下所示。



商品信息是所有微店的基础，因此对它的管理操作，我们需要更加清晰和完善。

综上所述的功能，我们可以定义好微信商品的接口如下所示。

[复制代码](javascript:void(0);)

#region 商品信息

/// <summary>

/// 创建商品

/// </summary>

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

/// <param name="merchantJson">商品对象</param>

/// <returns></returns>

AddMerchantResult AddMerchant(string accessToken, MerchantJson merchantJson);

/// <summary>

/// 删除商品

/// </summary>

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

/// <param name="productId">商品ID</param>

/// <returns></returns>

CommonResult DeleteMerchant(string accessToken, string productId);

/// <summary>

/// 修改商品

/// product\_id表示要更新的商品的ID，其他字段说明请参考增加商品接口。

/// 从未上架的商品所有信息均可修改，否则商品的名称(name)、商品分类(category)、商品属性(property)这三个字段不可修改。

/// </summary>

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

/// <param name="merchantJson">修改商品的信息</param>

/// <returns></returns>

CommonResult UpdateMerchant(string accessToken, MerchantJson merchantJson);

/// <summary>

/// 根据ID查询商品信息，如果成功返回MerchantJson信息，否则返回null

/// </summary>

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

/// <param name="productId">商品的Id</param>

/// <returns></returns>

MerchantJson GetMerchant(string accessToken, string productId);

/// <summary>

/// 获取指定状态的所有商品

/// </summary>

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

/// <param name="status">商品状态(0-全部, 1-上架, 2-下架)</param>

/// <returns></returns>

List<MerchantJson> GetMerchantByStatus(string accessToken, int status);

/// <summary>

/// 商品上下架

/// </summary>

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

/// <param name="status">商品上下架标识(0-下架, 1-上架)</param>

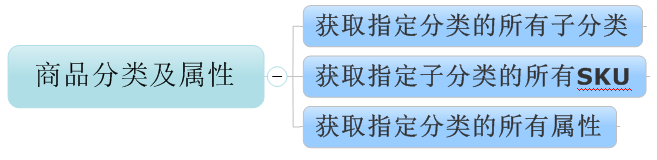
/// <returns></returns>

CommonResult UpdateMerchantStatus(string accessToken, string productId, int status);

#endregion

[复制代码](javascript:void(0);)

当然，微信的商品还包含了分类、分类属性、分类SKU的基础管理，因此商品管理还需要增加这个内容



它们的功能接口定义如下所示。通过下面的接口，我们就很容易实现商品分类（不是商品分组）、SKU信息、和分类属性等信息的获取操作了。

[复制代码](javascript:void(0);)

#region 商品分类及属性

/// <summary>

/// 获取指定分类的所有子分类

/// </summary>

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

/// <param name="cateId">大分类ID(根节点分类id为1)</param>

/// <returns></returns>

List<SubCategory> GetSub(string accessToken, int cate\_id);

/// <summary>

/// 获取指定子分类的所有SKU

/// </summary>

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

/// <param name="cateId">商品子分类ID</param>

/// <returns></returns>

List<SubCategorySku> GetSku(string accessToken, int cate\_id);

/// <summary>

/// 获取指定分类的所有属性

/// </summary>

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

/// <param name="cateId">分类ID</param>

/// <returns></returns>

List<SubCategoryProperty> GetProperty(string accessToken, int cate\_id);

#endregion

[复制代码](javascript:void(0);)

**2、商品管理接口的实现**

上面的接口定义了对应商品的接口。

对于接口的实现，我们一般都是根据官方网站的接口说明，提交到那个URL，并且是POST那些数据，然后整理成一个常规的处理方式，获得结果并转换为对应的对象即可，如添加商品操作的实现代码如下所示。

[复制代码](javascript:void(0);)

/// <summary>

/// 创建商品

/// </summary>

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

/// <param name="merchantJson">商品对象</param>

/// <returns></returns>

public AddMerchantResult AddMerchant(string accessToken, MerchantJson merchantJson)

{

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

string postData = merchantJson.ToJson();

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

}

[复制代码](javascript:void(0);)

而返回结果，这是定义一个对象来获得添加商品的ID等内容，如下所示。

[复制代码](javascript:void(0);)

/// <summary>

/// 创建商品信息的返回结果

/// </summary>

public class AddMerchantResult : ErrorJsonResult

{

/// <summary>

/// 商品ID

/// </summary>

public string product\_id { get; set; }

}

[复制代码](javascript:void(0);)

而基类这是常规的响应内容

[复制代码](javascript:void(0);)

/// <summary>

/// 微信返回Json结果的错误数据

/// </summary>

public class ErrorJsonResult

{

/// <summary>

/// 返回代码

/// </summary>

public ReturnCode errcode { get; set; }

/// <summary>

/// 错误消息

/// </summary>

public string errmsg { get; set; }

}

[复制代码](javascript:void(0);)

通过这些对象的定义，添加商品后，我们就知道操作是否成功，如果添加成功，返回了一个刚刚创建的ID给我们使用，我们可以进行查询具体的商品信息或者进行修改、删除等操作的。

而对商品信息的修改或者删除的操作，都是返回一个是否成功的记录就可以了，因此我们定义了一个统一的回应对象CommonResult。商品修改、删除的接口实现代码如下所示。

由于代码我都进行高度的完善和整理，对于各种处理的代码都相对比较容易理解的了。

[复制代码](javascript:void(0);)

/// <summary>

/// 删除商品

/// </summary>

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

/// <param name="productId">商品ID</param>

/// <returns></returns>

public CommonResult DeleteMerchant(string accessToken, string productId)

{

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

var data = new

{

product\_id = productId

};

string postData = data.ToJson();

return Helper.GetExecuteResult(url, postData);

}

/// <summary>

/// 修改商品

/// product\_id表示要更新的商品的ID，其他字段说明请参考增加商品接口。

/// 从未上架的商品所有信息均可修改，否则商品的名称(name)、商品分类(category)、商品属性(property)这三个字段不可修改。

/// </summary>

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

/// <param name="merchantJson">修改商品的信息</param>

/// <returns></returns>

public CommonResult UpdateMerchant(string accessToken, MerchantJson merchantJson)

{

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

string postData = merchantJson.ToJson();

return Helper.GetExecuteResult(url, postData);

}

[复制代码](javascript:void(0);)

为了获取商品的详细信息，我们需要定义一个商品的实体对象，以便我们把获取到的信息转换为实体类信息，方便使用和处理。

商品的信息，包含了不少细小定义的类，他们构成了商品的各个部分的内容，主体的实体类信息如下所示。



定义好相对比较复杂的商品信息实体后，我们就可以通过对象进行处理了。

获取商品详细信息的实现代码如下所示。

[复制代码](javascript:void(0);)

/// <summary>

/// 根据ID查询商品信息，如果成功返回MerchantJson信息，否则返回null

/// </summary>

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

/// <param name="productId">商品的Id</param>

/// <returns></returns>

public MerchantJson GetMerchant(string accessToken, string productId)

{

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

var data = new

{

product\_id = productId

};

string postData = data.ToJson();

MerchantJson merchant = null;

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

if (result != null)

{

merchant = result.product\_info;

}

return merchant;

}

[复制代码](javascript:void(0);)

虽然商品的实体信息很复杂，但是一旦我们定义好，我们就很容易对结果进行转换并处理了，上面的代码并不是很难理解，主要就是提交数据后，对数据进行转换而已。

当然，我们还可以获取不同状态的商品列表内容，如下代码所示。

[复制代码](javascript:void(0);)

/// <summary>

/// 获取指定状态的所有商品

/// </summary>

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

/// <param name="status">商品状态(0-全部, 1-上架, 2-下架)</param>

/// <returns></returns>

public List<MerchantJson> GetMerchantByStatus(string accessToken, int status)

{

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

var data = new

{

status = status

};

string postData = data.ToJson();

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

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

if (result != null)

{

list = result.products\_info;

}

return list;

}

[复制代码](javascript:void(0);)

我们添加商品的时候，商品的分类信息、分类属性、分类SKU信息也都是很重要的内容，我们需要指定对应商品分类才能添加到微信小店里面。

获取商品分类的操作实现代码如下所示。

[复制代码](javascript:void(0);)

/// <summary>

/// 获取指定分类的所有子分类

/// </summary>

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

/// <param name="cateId">大分类ID(根节点分类id为1)</param>

/// <returns></returns>

public List<SubCategory> GetSub(string accessToken, int cate\_id)

{

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

var data = new

{

cate\_id = cate\_id

};

string postData = data.ToJson();

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

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

if(result != null)

{

list = result.cate\_list;

}

return list;

}

[复制代码](javascript:void(0);)

**3、商品管理接口的测试**

为了验证我们开发的接口，我们需要增加一个测试项目，方便对我们编写的API进行测试，测试完全成功后，我们才能正式在项目中使用。

我为了方便，创建了一个Winform项目，分别对各个接口进行测试。



本篇主要介绍商品管理方面的接口，因此下面主要介绍其中商品管理部分的接口测试代码，以及对应的结果。

其中商品常规管理的接口测试代码如下所示。

[复制代码](javascript:void(0);)

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

{

//商品管理

IMerchantApi api = new MerchantApi();

//获取所有商品信息

Console.WriteLine("获取所有商品信息");

List<MerchantJson> list = api.GetMerchantByStatus(token, 0);

foreach(MerchantJson json in list)

{

Console.WriteLine(json.ToJson());

Console.WriteLine();

}

//更新商品状态

Console.WriteLine("更新商品状态");

foreach (MerchantJson json in list)

{

CommonResult result = api.UpdateMerchantStatus(token, json.product\_id, 1);

Console.WriteLine("商品ID：{0},商品名称：{1}， 操作：{2}",

json.product\_id, json.product\_base.name, result.Success ? "成功" : "失败");

}

Thread.Sleep(1000);

//根据商品ID获取商品信息

Console.WriteLine("根据商品ID获取商品信息");

foreach (MerchantJson json in list)

{

MerchantJson getJson = api.GetMerchant(token, json.product\_id);

if(json != null)

{

Console.WriteLine("商品ID：{0},商品名称：{1}", getJson.product\_id, getJson.product\_base.name);

}

}

}

[复制代码](javascript:void(0);)

测试后结果如下所示（就是返回我微店铺里面的商品信息），一切正常。

返回的商品Json数据如下所示：

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

{

"product\_id": "pSiLnt6FYDuFtrRRPMlkdKbye-rE",

"product\_base": {

"category\_id": [

"537103312"

],

"property": [

{

"id": "类型",

"vid": "软件产品设计"

}

],

"name": "代码生成工具Database2Sharp",

"sku\_info": [],

"main\_img": "http://mmbiz.qpic.cn/mmbiz/mLqH9gr11Gyb2sgiaelcsxYtQENGePp0Rb3AZKbjkicnKTUNBrEdo7Dyic97ar46SoAfKRB5x2R94bDUdNpgqiaZzA/0",

"img": [

"http://mmbiz.qpic.cn/mmbiz/mLqH9gr11Gyb2sgiaelcsxYtQENGePp0RiaheJmVXm7tbvTYUQV7OF3DgfGiaQVMh3WbeEcGDOQQiajQXGKK9tfoeA/0"

],

"detail": [],

"buy\_limit": 0,

"detail\_html": ""

},

"sku\_list": [

{

"sku\_id": "",

"ori\_price": 100000,

"price": 50000,

"icon\_url": "",

"quantity": 1100,

"product\_code": ""

}

],

"attrext": {

"location": {

"country": "中国",

"province": "广东",

"city": "广州",

"address": ""

},

"isPostFree": 1,

"isHasReceipt": 0,

"isUnderGuaranty": 0,

"isSupportReplace": 0

},

"delivery\_info": {

"delivery\_type": 0,

"template\_id": 175807970,

"express": [

{

"id": 10000027,

"price": 0

},

{

"id": 10000028,

"price": 0

},

{

"id": 10000029,

"price": 0

}

]

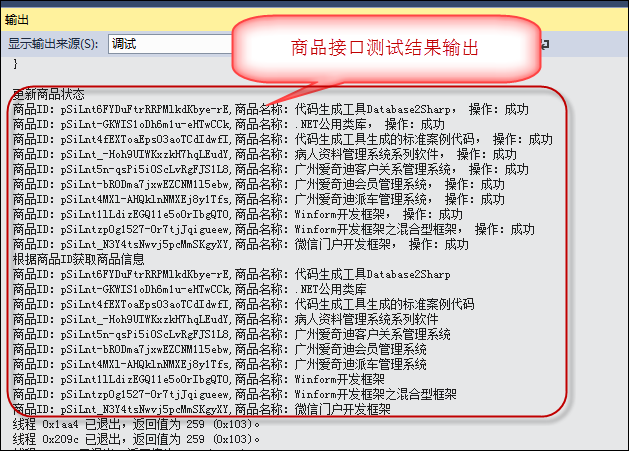
},

"status": 1

}

View Code

测试的部分结果输出如下所示。



另外，“商品维护管理”的功能测试主要就是测试商品的增加、修改、删除操作，具体代码如下所示。

[复制代码](javascript:void(0);)

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

{

IMerchantApi api = new MerchantApi();

string img1 = "http://mmbiz.qpic.cn/mmbiz/4whpV1VZl2iccsvYbHvnphkyGtnvjD3ulEKogfsiaua49pvLfUS8Ym0GSYjViaLic0FD3vN0V8PILcibEGb2fPfEOmw/0";

string img2 = "http://mmbiz.qpic.cn/mmbiz/4whpV1VZl2iccsvYbHvnphkyGtnvjD3ul1UcLcwxrFdwTKYhH9Q5YZoCfX4Ncx655ZK6ibnlibCCErbKQtReySaVA/0";

string img3 = "http://mmbiz.qpic.cn/mmbiz/4whpV1VZl28bJj62XgfHPibY3ORKicN1oJ4CcoIr4BMbfA8LqyyjzOZzqrOGz3f5KWq1QGP3fo6TOTSYD3TBQjuw/0";

//商品增删改处理

MerchantJson merchant = new MerchantJson();

merchant.product\_base = new Merchant\_base();

merchant.product\_base.name = "测试产品";

merchant.product\_base.category\_id.Add("537074298");

merchant.product\_base.img = new List<string>() { img1, img2, img3 };

merchant.product\_base.main\_img = img1;

merchant.product\_base.detail.AddRange(new List<MerchantDetail>() {

new MerchantDetail()

{

text = "test first"

},

new MerchantDetail()

{

img = img2

}, new MerchantDetail()

{

text = "test again"

}

});

merchant.product\_base.property.AddRange(new List<MerchantProperty>(){

new MerchantProperty

{

id= "1075741879",

vid="1079749967"

},

new MerchantProperty{

id= "1075754127",

vid= "1079795198"

},

new MerchantProperty(){

id= "1075777334",

vid= "1079837440"

}

});

merchant.product\_base.sku\_info.AddRange(new List<MerchantSku>(){

new MerchantSku{

id= "1075741873",

vid = new List<string>() {

"1079742386",

"1079742363"

}

}

});

merchant.product\_base.buy\_limit = 10;

//merchant.product\_base.detail\_html = "<div class=\"item\_pic\_wrp\" style=\"margin-bottom:8px;font-size:0;\"><img class=\"item\_pic\" style=\"width:100%;\" alt=\"\" src=\"http://mmbiz.qpic.cn/mmbiz/4whpV1VZl2iccsvYbHvnphkyGtnvjD3ulEKogfsiaua49pvLfUS8Ym0GSYjViaLic0FD3vN0V8PILcibEGb2fPfEOmw/0\" ></div><p style=\"margin-bottom:11px;margin-top:11px;\">test</p><div class=\"item\_pic\_wrp\" style=\"margin-bottom:8px;font-size:0;\"><img class=\"item\_pic\" style=\"width:100%;\" alt=\"\" src=\"http://mmbiz.qpic.cn/mmbiz/4whpV1VZl2iccsvYbHvnphkyGtnvjD3ul1UcLcwxrFdwTKYhH9Q5YZoCfX4Ncx655ZK6ibnlibCCErbKQtReySaVA/0\" ></div><p style=\"margin-bottom:11px;margin-top:11px;\">test again</p>";

merchant.sku\_list.AddRange(new List<MerchantSku\_list>()

{

new MerchantSku\_list(){

sku\_id="1075741873:1079742386",

price=30,

icon\_url="http://mmbiz.qpic.cn/mmbiz/4whpV1VZl2iccsvYbHvnphkyGtnvjD3ulEKogfsiaua49pvLfUS8Ym0GSYjViaLic0FD3vN0V8PILcibEGb2fPfEOmw/0",

quantity=800,

product\_code="testing",

ori\_price=9000000

},

new MerchantSku\_list(){

sku\_id="1075741873:1079742363",

price=30,

icon\_url="http://mmbiz.qpic.cn/mmbiz/4whpV1VZl28bJj62XgfHPibY3ORKicN1oJ4CcoIr4BMbfA8LqyyjzOZzqrOGz3f5KWq1QGP3fo6TOTSYD3TBQjuw/0",

quantity=800,

product\_code="testingtesting",

ori\_price=9000000

}

});

merchant.attrext = new MerchantAttrext()

{

location = new MerchantLocation()

{

country = "中国",

province = "广东省",

city = "广州市",

address = "T.I.T创意园"

},

isPostFree = 0,

isHasReceipt = 1,

isUnderGuaranty = 0,

isSupportReplace = 0

};

merchant.delivery\_info = new MerchantDelivery()

{

delivery\_type = 0,

template\_id = 0,

express = new List<MerchantExpress>(){

new MerchantExpress() {

id=10000027,

price=100

},

new MerchantExpress(){

id=10000028,

price=100

},

new MerchantExpress(){

id=10000029,

price=100

}}

};

Console.WriteLine(merchant.ToJson());

AddMerchantResult result = api.AddMerchant(token, merchant);

Console.WriteLine("添加商品：{0}", result.product\_id);

if (!string.IsNullOrEmpty(result.product\_id))

{

//更新商品

merchant.product\_id = result.product\_id;

merchant.product\_base.name = "测试产品22";

CommonResult updateResult = api.UpdateMerchant(token, merchant);

Console.WriteLine("更新商品：{0}", updateResult.Success ? "成功" : "失败");

CommonResult deleteResult = api.DeleteMerchant(token, merchant.product\_id);

Console.WriteLine("删除商品：{0}", deleteResult.Success ? "成功" : "失败");

}

}

[复制代码](javascript:void(0);)

测试的输出结果如下所示（一切成功）。



以上就是我对商品管理接口的API定义和实现，以及对接口进行测试的阐述，基本上把所有相关的内容都贴出来了，希望大家能够对微店开发部分，有更深入的了解和认识。