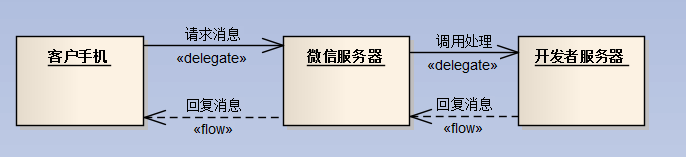
**[C#开发微信门户及应用(2)--微信消息的处理和应答](http://www.cnblogs.com/wuhuacong/p/3614175.html)**

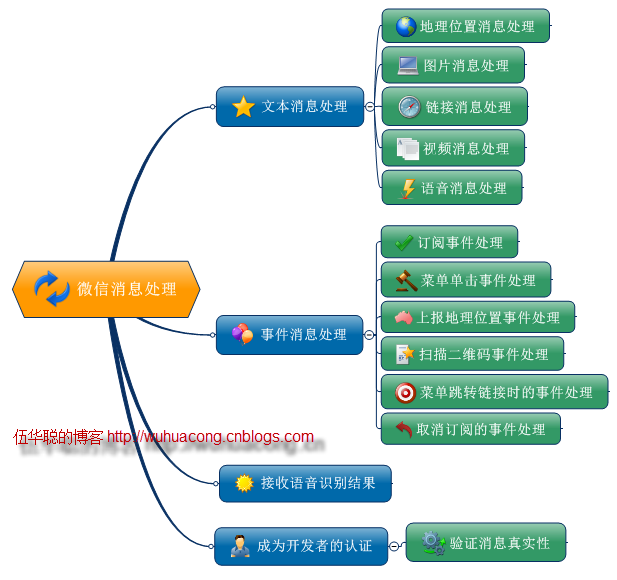
微信应用如火如荼，很多公司都希望搭上信息快车，这个是一个商机，也是一个技术的方向，因此，有空研究下、学习下微信的相关开发，也就成为计划的安排事情之一了。本系列文章希望从一个循序渐进的角度上，全面介绍微信的相关开发过程和相关经验总结，希望给大家了解一下相关的开发历程。本篇随笔主要基于上一篇《[C#开发微信门户及应用(1)--开始使用微信接口](http://www.cnblogs.com/wuhuacong/p/3613826.html)》的基础上进行深入的介绍，介绍微信消息的处理和应答的过程。

**1、微信的消息应答交互**

我们知道，微信的服务器架起了客户手机和开发者服务器的一个桥梁，通过消息的传递和响应，实现了与用户的交互操作，下面是它的消息流程图。



微信向开发者服务器请求的消息包含了多种类型，不过基本来说，分为了文本消息处理、事件消息处理、语音消息的识别，以及成为开发者之前的那个消息认证操作基本分类，下面是我绘制的一个消息分类图，其中介绍了这几种关系，以及各自的消息细化分类。



对于这些消息的请求，我们在开发服务器端，需要编写相关的逻辑进行对应给的处理，然后给微信服务器平台回应消息即可。

在前一篇的随笔里面我贴过代码，介绍微信消息处理的入口操作，代码如下所示。

[复制代码](javascript:void(0);)

public void ProcessRequest(HttpContext context)

{

//WHC.Framework.Commons.LogTextHelper.Info("测试记录");

string postString = string.Empty;

if (HttpContext.Current.Request.HttpMethod.ToUpper() == "POST")

{

using (Stream stream = HttpContext.Current.Request.InputStream)

{

Byte[] postBytes = new Byte[stream.Length];

stream.Read(postBytes, 0, (Int32)stream.Length);

postString = Encoding.UTF8.GetString(postBytes);

}

if (!string.IsNullOrEmpty(postString))

{

**Execute(postString);**

}

}

else

{

Auth();

}

}

[复制代码](javascript:void(0);)

其中的Execute(postString);就是对消息的处理函数，它实现了对不同消息的分发处理过程。‘

[复制代码](javascript:void(0);)

/// <summary>

/// 处理各种请求信息并应答（通过POST的请求）

/// </summary>

/// <param name="postStr">POST方式提交的数据</param>

private void Execute(string postStr)

{

WeixinApiDispatch dispatch = new WeixinApiDispatch();

string responseContent = dispatch.**Execute**(postStr);

HttpContext.Current.Response.ContentEncoding = Encoding.UTF8;

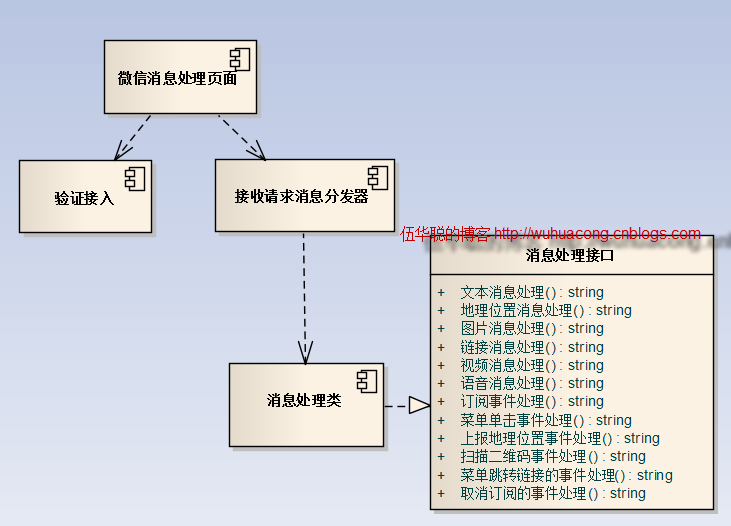
HttpContext.Current.Response.Write(responseContent);

}

[复制代码](javascript:void(0);)

里面的WeixinApiDispatch就是一个分发的管理类，它提取请求消息的内容，并构建不同类型的消息参数，传递给不同的响应函数进行处理，然后返回封装好的XML内容，作为响应。

具体的代码处理逻辑如下图所示。



这个消息处理接口，其实就是定义好一系列的对请求消息的处理操作，参数是不同给的消息对象，具体的代码定义如下所示（由于篇幅原因，省略部分接口，具体可以参考上图）。

[复制代码](javascript:void(0);)

/// <summary>

/// 客户端请求的数据接口

/// </summary>

public interface IWeixinAction

{

/// <summary>

/// 对文本请求信息进行处理

/// </summary>

/// <param name="info">文本信息实体</param>

/// <returns></returns>

string HandleText(RequestText info);

/// <summary>

/// 对图片请求信息进行处理

/// </summary>

/// <param name="info">图片信息实体</param>

/// <returns></returns>

string HandleImage(RequestImage info);

...........................

/// <summary>

/// 对订阅请求事件进行处理

/// </summary>

/// <param name="info">订阅请求事件信息实体</param>

/// <returns></returns>

string HandleEventSubscribe(RequestEventSubscribe info);

/// <summary>

/// 对菜单单击请求事件进行处理

/// </summary>

/// <param name="info">菜单单击请求事件信息实体</param>

/// <returns></returns>

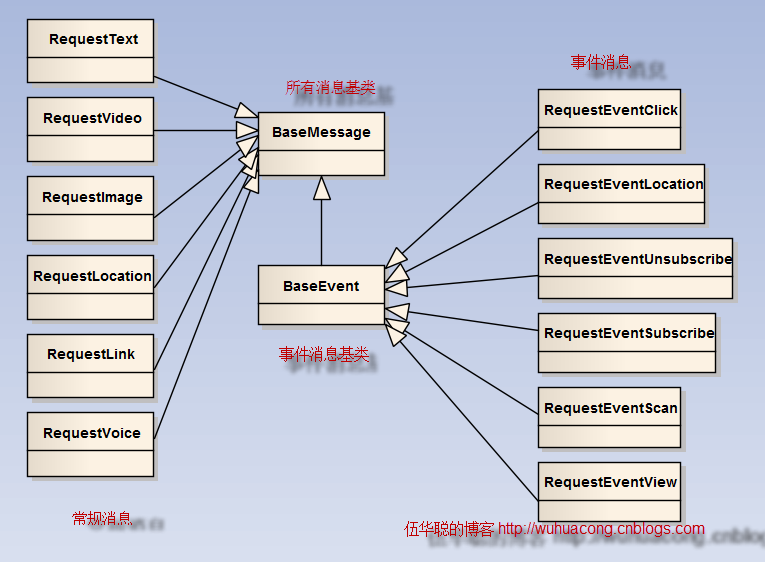
string HandleEventClick(RequestEventClick info);

..............................

}

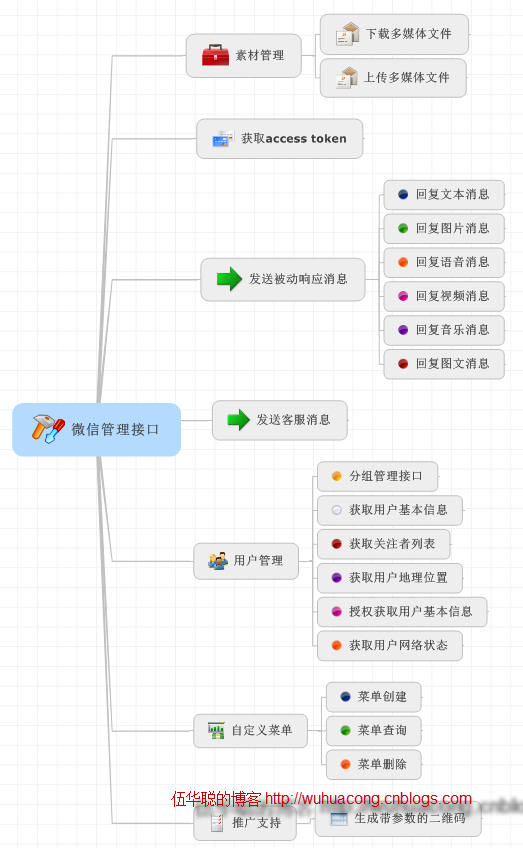
[复制代码](javascript:void(0);)

从上面的代码可以看出，不同的消息，到处理函数这里，就以不同的消息实体类的方式传递过来了（**注意：实体类是我根据程序开发需要自己定义的，非微信本身的实体类**），这样非常方便我们处理操作，否则每次需要解析不同的消息内容，很容易出现问题，这样强类型的数据类型，提高了我们开发微信应用的强壮型和高效性。这些实体类的对象有一定的继承关系的，他们的继承关系如下所示。

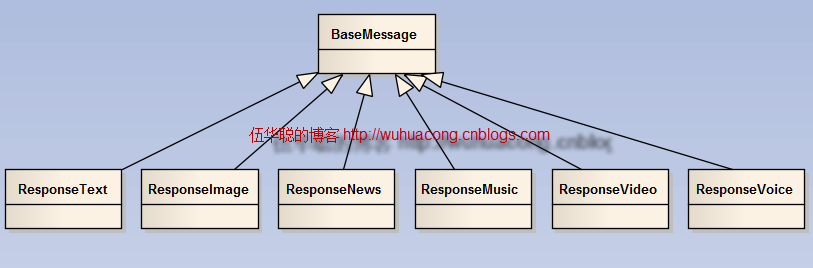


**2、微信的管理接口**

上面的消息分类是微信服务器向开发者服务器发送的消息请求操作，还有一种消息，是我们开发者服务器向微信服务器进行的消息请求或者响应，这种这里暂且称之为微信的管理接口，它表明了我们可以通过这些接口进行相关的消息回复或者数据管理操作。它的分类图如下所示。

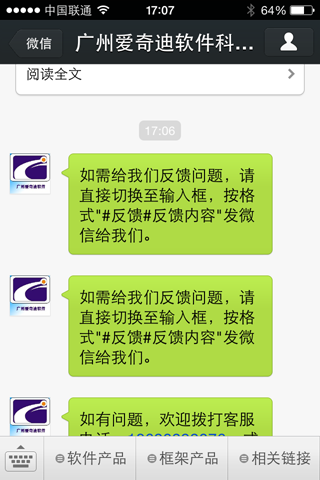


微信的回复消息处理，它也和上面小节的信息一样，它也是继承自BaseMessage实体类的（**同样，下图的实体类及其继承关系也是自定义的，方便程序开发**），它的关系如下所示



回复的消息，一般用的最多的是文本消息和图文消息。

文本消息的效果如下所示。



图文消息，可以增加图片，还可以增加详细的链接页面，是非常好看的一种效果，对于一些内容比较多，希望展现更好效果的，一般采用这种，效果如下所示。

