**[C#开发微信门户及应用(13)-使用地理位置扩展相关应用](http://www.cnblogs.com/wuhuacong/p/3756347.html)**

本文继续上一篇《[C#开发微信门户及应用(12)-使用语音处理](http://www.cnblogs.com/wuhuacong/p/3753834.html)》，继续介绍微信的相关应用。我们知道，地理位置信息可以用来做很多相关的应用，除了我们可以知道用户所在的位置，还可以关联出一些地理位置的应用，如天气，热映影片，附近景点，附近影院，交通事件等等，反正所有和地理位置相关的信息，我们都可以根据需要做一些扩展应用。本文主要介绍利用地理位置信息，如何构建使用这些应用的操作。



**1、微信的地理位置信息**

在使用前，我们先来看看微信的接口，为我们定义了那些关于与地理位置的信息。其实地理位置的信息，微信分为了两个方面，一个是接收用户的地理位置请求，一个是用户允许上报地理位置操作，定时发送的地理位置信息。

本文主要介绍基于第一种，用户上报地理位置后，如何处理的相关应用。

地理位置的上报操作，就是在输入的地方，选择+号进行添加地理位置，然后选择当前或者指定的地理位置地图，具体操作如下所示。

**地理位置消息**

[复制代码](javascript:void(0);)

<xml>

<ToUserName><![CDATA[toUser]]></ToUserName>

<FromUserName><![CDATA[fromUser]]></FromUserName>

<CreateTime>1351776360</CreateTime>

<MsgType><![CDATA[location]]></MsgType>

<Location\_X>23.134521</Location\_X>

<Location\_Y>113.358803</Location\_Y>

<Scale>20</Scale>

<Label><![CDATA[位置信息]]></Label>

<MsgId>1234567890123456</MsgId>

</xml>

[复制代码](javascript:void(0);)

|  |  |
| --- | --- |
| **参数** | **描述** |
| ToUserName | **开发者**微信号 |
| FromUserName | 发送方帐号（一个OpenID） |
| CreateTime | 消息创建时间 （整型） |
| MsgType | location |
| Location\_X | 地理位置维度 |
| Location\_Y | 地理位置经度 |
| Scale | 地图缩放大小 |
| Label | 地理位置信息 |
| MsgId | 消息id，64位整型 |

有了上面的地理位置信息，我们在程序里面，需要在消息传递过来的时候，定义一个实体类信息，承载相关的地理位置信息，方便我们进一步的处理操作。

[复制代码](javascript:void(0);)

/// <summary>

/// 接收的地理位置消息

/// </summary>

[System.Xml.Serialization.XmlRoot(ElementName = "xml")]

public class RequestLocation : BaseMessage

{

public RequestLocation()

{

this.MsgType = RequestMsgType.Location.ToString().ToLower();

}

/// <summary>

/// 消息ID

/// </summary>

public Int64 MsgId { get; set; }

/// <summary>

/// 地理位置维度

/// </summary>

public decimal Location\_X { get; set; }

/// <summary>

/// 地理位置经度

/// </summary>

public decimal Location\_Y { get; set; }

/// <summary>

/// 地图缩放大小

/// </summary>

public int Scale { get; set; }

/// <summary>

/// 地理位置信息

/// </summary>

public string Label { get; set; }

}

[复制代码](javascript:void(0);)

有了这些信息，我们在信息传递的时候，就能很好得到用户的相关数据了。

如果仅仅为了返回给用户，告诉用户目前的地理位置信息，可以用下面的操作就可以了。

[复制代码](javascript:void(0);)

/// <summary>

/// 对地理位置请求信息进行处理

/// </summary>

/// <param name="info">地理位置请求信息实体</param>

/// <returns></returns>

public string HandleLocation(Entity.RequestLocation info)

{

string xml = "";

ResponseText txtinfo = new ResponseText(info);

txtinfo.Content = string.Format("您发送的地理位置是：{0}", info.Label);

xml = txtinfo.ToXml();

return xml;

}

[复制代码](javascript:void(0);)

**2、地址位置的应用处理**

不过上面的信息，显然不符合我们扩展应用的要求，因此我们进一步进行完善里面对地理位置信息处理的操作。我们进一步把关于地理位置的操作，放到事件处理模块里面进行处理，处理代码如下所示。

[复制代码](javascript:void(0);)

/// <summary>

/// 对地理位置请求信息进行处理

/// </summary>

/// <param name="info">地理位置请求信息实体</param>

/// <returns></returns>

public string HandleLocation(Entity.RequestLocation info)

{

string xml = "";

EventDispatch dispatch = new EventDispatch();

xml = dispatch.DealLocation(info, info.Label, info.Location\_Y, info.Location\_X);

return xml;

}

[复制代码](javascript:void(0);)

在处理的时候，我们需要先保存用户的地理位置信息，把它存储到用户的上下文记录里面。这样我们在处理指令的时候，把它获取到，然后传递给相关的方法就可以实现地理位置的扩展应用了。

//保存经纬度

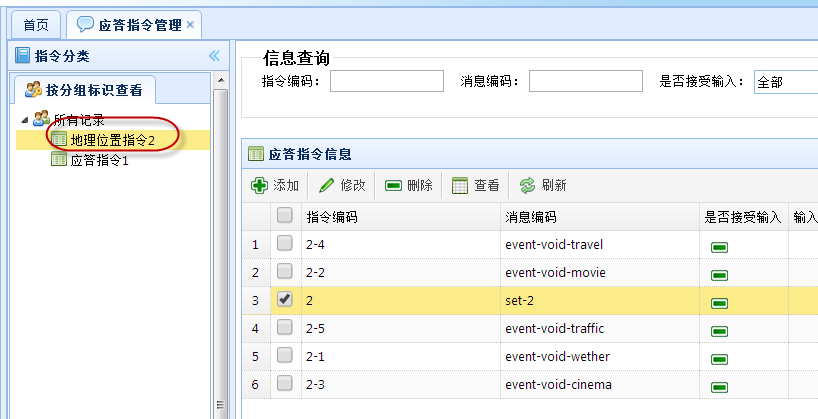
string location = string.Format("{0},{1}", lat, lon);

bool result = BLLFactory<UserSet>.Instance.UpdateUserInput(info.FromUserName, location);

首先对用户地理位置的请求，我根据数据库配置给出了一个用户选择的指令提示，如下所示。



为了对地理位置请求的处理，我定义了一个用于处理这个操作的指令操作



这样整个地理位置的指令操作，就在应答链里面进行很好的跳转管理了。那么为了实现天气、放映影片、附近影院、旅游线路、交通事件等方面的扩展应用，我们应该如何操作呢？

**3、地址位置应用扩展**

我们知道，百度或者腾讯都提供了一些开放平台，给我们进行各种方式的使用。那么我们这里以使用百度LBS平台应用来构建一些模块。





这上面都有很多相关的接口供使用，我们可以根据其提供的数据格式进行封装，然后进行调用处理就可以了。

刚才说了，我配置了一些指令，用来构建相关的应用，指令的最后是一些事件代码的定义，我们对这些末端的事件代码进行处理，就可以给用户返回相关的信息了，总体的操作代码如下所示。

[复制代码](javascript:void(0);)

/// <summary>

/// 其他插件操作，如天气，景点、电影影讯、交通等

/// </summary>

/// <param name="info">基础消息</param>

/// <param name="eventKey">事件标识</param>

/// <returns></returns>

public string DealPlugin(BaseMessage info, string eventKey)

{

//LogTextHelper.Info(eventKey);

string userInput = BLLFactory<UserSet>.Instance.GetUserInput(info.FromUserName);

string xml = "";

switch (eventKey)

{

case "event-void-wether":

xml = new WeatherPlugin().Response(info, userInput);

break;

case "event-void-movie":

xml = new MoviePlugin().Response(info, userInput);

break;

case "event-void-cinema":

xml = new CinemaPlugin().Response(info, userInput);

break;

case "event-void-travel":

xml = new TravelPlugin().Response(info, userInput);

break;

case "event-void-traffic":

xml = new TrafficEventPlugin().Response(info, userInput);

break;

default:

break;

}

return xml;

}

[复制代码](javascript:void(0);)

这里以天气为例，说明该如何调用百度的接口的，首先我们封装一下相关的接口调用。

[复制代码](javascript:void(0);)

/// <summary>

/// 根据参数调用百度接口，获取相关的结果数据

/// </summary>

/// <param name="location">地理位置</param>

/// <param name="ak">API调用键</param>

/// <returns></returns>

public BaiduWeatherResult Execute(string location, string ak)

{

location = HttpUtility.UrlEncode(location);

var url = string.Format("http://api.map.baidu.com/telematics/v3/weather?location={0}&output=json&ak={1}", location, ak);

BaiduWeatherResult result = BaiduJsonHelper<BaiduWeatherResult>.ConvertJson(url);

return result;

}

[复制代码](javascript:void(0);)

其中的BaiduWeatherResult 是我根据调用返回的Json结果，构建的一个实体类，用来存储返回的内容。具体代码如下所示。

[复制代码](javascript:void(0);)

/// <summary>

/// 天气请求结果Json对象

/// </summary>

public class BaiduWeatherResult : BaiduResult

{

/// <summary>

/// 天气预报信息

/// </summary>

public List<BaiduWeatherData> results = new List<BaiduWeatherData>();

}

/// <summary>

/// 城市的天气信息

/// </summary>

public class BaiduWeatherData

{

/// <summary>

/// 当前城市

/// </summary>

public string currentCity { get; set; }

/// <summary>

/// 天气预报信息

/// </summary>

public List<BaiduWeatherJson> weather\_data = new List<BaiduWeatherJson>();

}

/// <summary>

/// 天气预报的单条记录Json信息

/// </summary>

public class BaiduWeatherJson

{

/// <summary>

/// 天气预报时间

/// </summary>

public string date { get; set; }

/// <summary>

/// 白天的天气预报图片url

/// </summary>

public string dayPictureUrl { get; set; }

/// <summary>

/// 晚上的天气预报图片url

/// </summary>

public string nightPictureUrl { get; set; }

/// <summary>

/// 天气状况

/// </summary>

public string weather { get; set; }

/// <summary>

/// 风力

/// </summary>

public string wind { get; set; }

/// <summary>

/// 温度

/// </summary>

public string temperature { get; set; }

}

[复制代码](javascript:void(0);)

为了构建返回给客户的图文数据，我们需要构建一个News对象，然后生成XML数据返回给服务器进行处理即可。

[复制代码](javascript:void(0);)

/// <summary>

/// 响应用户请求，并返回相应的XML数据

/// </summary>

/// <param name="info">微信基础信息</param>

/// <param name="location">地理位置：经纬度坐标或者地名</param>

/// <returns></returns>

public string Response(BaseMessage info, string location)

{

string xml = "";

//"广州" 或者 "116.305145,39.982368"

if (!string.IsNullOrEmpty(location))

{

BaiduWeatherResult result = Execute(location, baiduAK);

if (result != null && result.results.Count > 0)

{

BaiduWeatherData data = result.results[0];

if (data != null)

{

ArticleEntity first = new ArticleEntity();

first.Title = string.Format("{0} 天气预报", data.currentCity);

ResponseNews news = new ResponseNews(info);

news.Articles.Add(first);

int i = 0;

foreach (BaiduWeatherJson json in data.weather\_data)

{

ArticleEntity article = new ArticleEntity();

article.Title = string.Format("{0}\n{1} {2} {3}", json.date, json.weather, json.wind, json.temperature);

if (i++ == 0)

{

article.PicUrl = IsDayTime() ? json.dayPictureUrl : json.nightPictureUrl;

}

else

{

article.PicUrl = json.dayPictureUrl;

}

news.Articles.Add(article);

}

xml = news.ToXml();

}

}

}

return xml;

}

[复制代码](javascript:void(0);)

这样就很好实现了整体的功能了，具体界面功能可以访问我的微信（广州爱奇迪）进行了解，下面是功能截图供参考。