Homework 7 ——

网络访问

基于HTTP的网络访问

- 通过客户端代理访问SOAP Web Service
- 使用HttpClient访问网络数据
- 使用WebSocket访问网络数据

Appxmanifest

在Appxmanifest里的“功能”处勾上"Internet（客户端）"

HttpClient访问网络

- HttpClient类用于通过 HTTP 发送和接收基本要求
- HttpClientHandler类是HttpClient的默认消息处理程序
- HttpResponseMessage类用于声明从 HTTP 请求接收到的 HTTP 响应消息
- HttpContent类是用于声明 HTTP 实体正文和内容标题的基类

创建HttpClientHandler

```
private HttpClient httpClient;
private HttpClientHandler handler;
public MainPage()
{
    this.InitializeComponent();
    handler = new HttpClientHandler();
    handler.AllowAutoRedirect = false;

    httpClient = new HttpClient(handler);
    httpClient.MaxResponseContentBufferSize = 256000;
    httpClient.DefaultRequestHeaders.Add("user-agent",
"Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; WOW64;
Trident/6.0)");
}
```

发送GET请求并检索响应

```
 private async void Start_Click(object sender, RoutedEventArgs e)
{
    try
    {
        string responseBodyAsText;
        OutputView.NavigateToString("");
        StatusText.Text = "Waiting for response ...";
        HttpResponseMessage response = await httpClient.GetAsync(InputAddress.Text);
        response.EnsureSuccessStatusCode();

        StatusText.Text = response.StatusCode + " " + response.ReasonPhrase + Environment.NewLine;
        responseBodyAsText = await response.Content.ReadAsStringAsync();
        OutputView.NavigateToString(responseBodyAsText);
    }
    catch (HttpRequestException hre)
    {
        StatusText.Text = hre.ToString();
    }
    catch (Exception ex)
    {
        StatusText.Text = ex.ToString();
    }
}
```

发送POST请求并检索响应

```
private async void Start_Click(object sender, RoutedEventArgs e) {
   try
   {
        OutputView.NavigateToString("");
        StatusText.Text = "Waiting for response ...";
        content = new FormUrlEncodedContent(new Dictionary<string, string>() {
             {“id”, gameid},
             {“user”, User.Text},
             {“score”, score},
             {“words”, Words.Text}
        });
        HttpResponseMessage response = await httpClient.PostAsync(url, content);
        response.EnsureSuccessStatusCode();
        StatusText.Text = response.StatusCode + " " + response.ReasonPhrase + Environment.NewLine;
        string responseBodyAsText = await response.Content.ReadAsStringAsync();
        OutputView.NavigateToString(responseBodyAsText);
   }
   catch (HttpRequestException hre)
   {
        StatusText.Text = hre.ToString();
   }
   catch (Exception ex)
   {
        StatusText.Text = ex.ToString();
   }
}
```

排行榜使用说明

1.使用GET请求获取排行榜信息

请求url:

http://222.200.185.43:8000/rank/getScore/?id=**YOURGAMEID**&num=**THE_NUMBER_OF_REQUEST**

其中**YOURGAMEID**为相应游戏的id（自己设）， **THE_NUMBER_OF_REQUEST**为返回的条目数

结果按以下xml格式返回

```
<records>
    <item>
        <user>谷阿莫</user>
        <score>100</score>
        <words>科科</words>
    </item>
    <item>
        <user>pml</user>
        <score>99</score>
        <words>做项目什么，最怕大神同学失恋</words>
    </item>
    ……
</records>
```

排行榜使用说明

2.使用POST请求更新排行榜信息

请求url:

http://222.200.185.43:8000/rank/newScore/

表单数据:

id YOURGAMEID（游戏id）

user PLAYERNAME（玩家名，若玩家名已存在并且此次分数较高则更新分数）

score SCORE（得分）

words SOMETHING_PLAYER_SAID（获奖感言）

返回**succeed**或者**invalid operation**以标志操作是否成功

每个游戏ID最多保存前50条记录

排行榜——GET请求示例

```
private async void Get(int gameid, int number)
{
    try
    {
        status.Text = "Waiting for response ...";
        HttpClient httpClient = new HttpClient();
        var headers = httpClient.DefaultRequestHeaders;
        headers.UserAgent.ParseAdd("Mozilla/5.0 (Windows NT 6.2; WOW64; rv:25.0) Gecko/20100101 Firefox/25.0");
        string url = string.Format(getUrl, gameid, number);
        HttpResponseMessage response = await httpClient.GetAsync(url);
        response.EnsureSuccessStatusCode();
        status.Text = response.StatusCode + " " + response.ReasonPhrase + Environment.NewLine;
        string rescontent = await response.Content.ReadAsStringAsync();
        XmlDocument xmlDoc = new XmlDocument();
        xmlDoc.LoadXml(rescontent);
        XmlElement root = xmlDoc.DocumentElement;
        XmlNodeList users = root.SelectNodes("/records/item/user");
        XmlNodeList scores = root.SelectNodes("/records/item/score");
        XmlNodeList wordss = root.SelectNodes("/records/item/words");
        rec = new List<Record>();
        for (int i = 0; i < users.Length; i++)
        {
            rec.Add(new Record() { User = users[i].InnerText, Score = scores[i].InnerText, Words = wordss[i].InnerText });
        }
        recordList.ItemsSource = rec;
    }
    catch (HttpRequestException hre)
```

private string getUrl = "xxx/rank/getScore/?id={0}&num={1}";
字符串格式化构造出请求url

发起异步的get请求

获取返回内容并用XmlDocument解析
用正则表达式或者字符串方法都可以，
自己喜欢

设置listview数据源

排行榜——POST请求示例

post其实只是多了个表单数据而已，使用httpClient.PostAsync方法请求

```
var content = new FormUrlEncodedContent(new Dictionary<string, string>() {
    { "id", gameid},
    { "user", name},
    { "score", mark},
    { "words", words}
});
HttpResponseMessage response = await httpClient.PostAsync(postUrl, content);
```

一些其他

我也不是专业做后台的，只是个临时工，
所以有什么做得不好的多多谅解，
有问题我会尽快修的。

然后也请学了web安全的各位想练手的大哥手下留情
谢谢

作业

- 使用HttpClient进行网络访问

- 实用就行，比如说手机号查归属地，输入城市查天气，单词翻译之类的，自己上网找API

- http://blog.sina.com.cn/s/blog_7bac4707010143o2.html

备注

- 一些HTTP 消息处理程序可以与HttpClient类结合使用
  - **HttpClientHandler**-：HttpClient 所用的默认消a息处理程序
  - **MessageProcessingHandler**：一种基本的 HTTP 消息处理程序。这是最容易进行派生的处理程序，应该作为大多数自定义处理程序的起点
  - **DelegatingHandler**：一种基本的 HTTP 处理程序，可将 HTTP 响应消息的处理委派给其他处理程序。此处理程序十分适合用于测试

资源

- *Accessing app data with the Windows Runtime*
  - *http://technet.microsoft.com/zh-cn/library/hh464917*
- *连接到对等端、 Web 和网络服务*
  - *http://msdn.microsoft.com/zh-cn/library/windows/apps/br211370.aspx*
- *《Programming Windows Sixth Edition》*

  http://shop.oreilly.com/product/0790145369079.do