ONE一个

软件设计文档

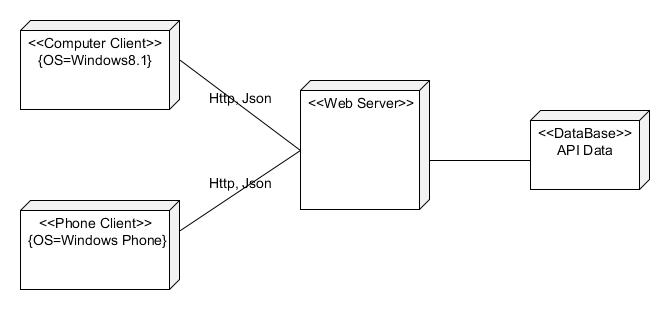
2015年7月

1. 技术选型理由

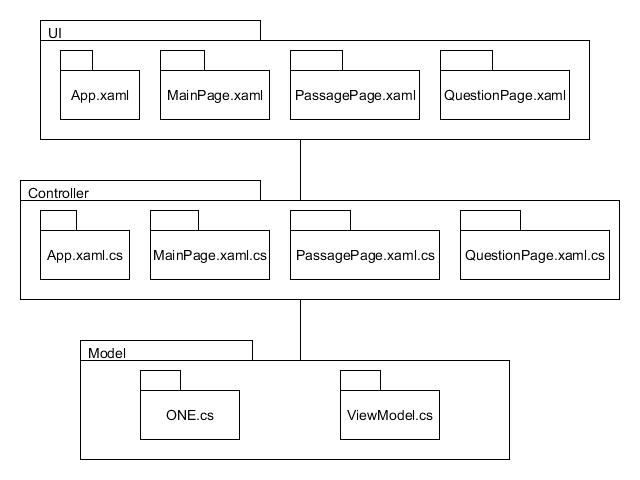
本小组选择Win8通用平台开发，可以实现一个应用同时在Windows平板及PC， 还有Windows Phone上同时发布。一次开发就可以实现多平台通用，高效利用代码。

从需求的角度上看，本应用ONE在其他移动终端平台iOS，Android都十分受欢迎，有数百万用户量，而Windows终端缺少该应用的官方产品。本小组抓住用户需求，在Windows应用平台完善了该产品，积累了一定用户量，证明选择此平台是正确的。

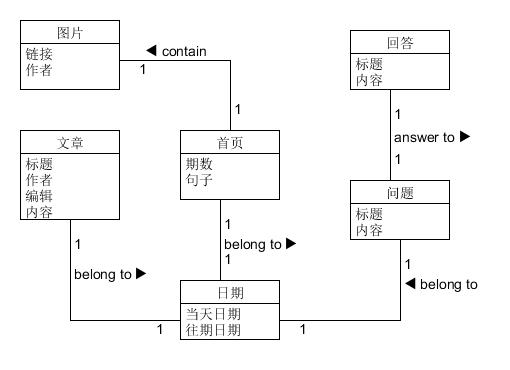
1. 架构设计
   1. 整体架构（逻辑图）



* 1. 前端架构（包图）



1. 全局数据结构说明
   1. 领域模型



* 1. 数据结构

在**ONE/ONE.Shared/Strings/ONE.cs**中，定义了ONE类，存储该应用的基本数据：

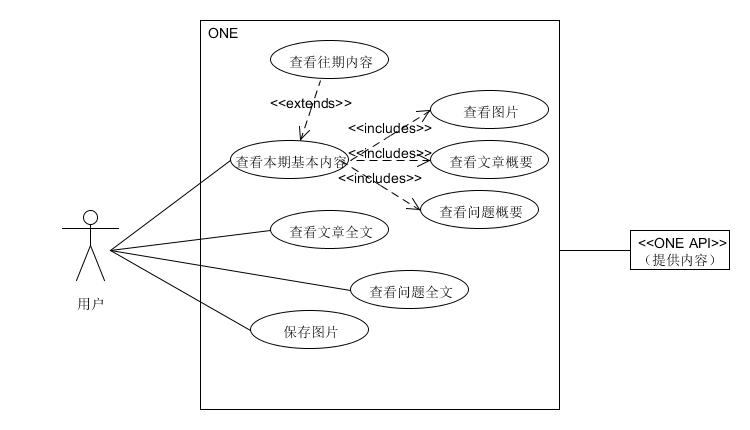
|  |
| --- |
| public class ONE  {  public ONE()  {  datelist = new List<string>();  }  public List<string> datelist { get; set; } //最近日期的数组  public string date{get;set;} //日期yyyy-mm-dd  public string HomepagestrAuthor { get; set; } //首页图片作者  public string HomepagestrContent { get; set; } //首页句子内容  public string HomepagestrHpTitle { get; set; } //首页期数  public string HomepagestrOriginalImgUrl { get; set; } //首页图片链接  public BitmapImage Homepageimage { get; set; }  public string ContentstrContTitle { get; set; } //文章标题  public string ContentstrContAuthor { get; set; } //文章作者  public string ContentstrContAuthorIntroduce { get; set; } //文章编辑  public string ContentstrContent { get; set; } //文章内容  public string ContentsGW { get; set; } //文章句子  public string strQuestionTitle { get; set; } //问题标题  public string strQuestionContent { get; set; } //问题内容  public string strAnswerTitle { get; set; } //回答标题  public string strAnswerContent { get; set; } //回答内容  } |

* 1. 常量

在**ONE/ONE.Shared/Strings/ViewModel.cs**中，定义了网络接口为常量

|  |
| --- |
| //主页、文章和问题的api  private const string HomepageLink  = @"http://211.152.49.184:7001/OneForWeb/one/getHpinfo?strDate=";  //private const string ContentLink  // = "http://211.152.49.184:7001/OneForWeb/one/getOneContentInfo?strDate=";  //ContentLink之前的API接口有问题，数据无法解析，之后改了另外一个接口  private const string ContentLink = @"http://onewp.sinaapp.com/";  private const string QuestionLink  = @"http://211.152.49.184:7001/OneForWeb/one/getOneQuestionInfo?strDate="; |

1. 模块设计
   1. 用例图



* 1. 模块划分

根据代码功能，划分为以下几个重要模块：网络连接模块、数据加载模块、逻辑跳转模块、保存图片模块、处理文本溢出模块。

* + 1. 网络连接模块

在**ONE/ONE.Shared/Strings/ViewModel.cs**中，定义了HttpClient并在构造函数中初始化：

|  |
| --- |
| class ViewModel  {  private HttpClient httpClient;  …  public ViewModel(string \_date)  {  httpClient = new HttpClient();  httpClient.MaxResponseContentBufferSize = 2560000;  httpClient.DefaultRequestHeaders.Add("User-Agent", "Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; WOW64; Trident/6.0)");  …  } |

在函数GetDataAsync()中，异步获取API远程数据

|  |
| --- |
| private async Task<String> GetDataAsync(string uriString)  {  try  {  HttpResponseMessage response = await httpClient.GetAsync(uriString);  response.EnsureSuccessStatusCode();  string res = await response.Content.ReadAsStringAsync();  return res;  }  catch(Exception)  {  isDataGet = false;  return string.Empty;  }  } |

* + 1. 数据加载模块

在**ONE/ONE.Shared/Strings/ViewModel.cs**中，通过LoadData()函数获取并加载数据：

|  |
| --- |
| //加载数据  public async Task<bool> LoadData()  {  //获取api上的Json数据  Homepage = await GetDataAsync(HomepageLink + one.date);  Content = await GetDataAsync(ContentLink + one.date + "/content.html");  Question = await GetDataAsync(QuestionLink + one.date);  if (isDataGet == true)  {  //处理获得的Json数据  LoadHomepage(Homepage);  LoadContent(Content);  LoadQuestion(Question);  }  } |

子函数LoadHomepage, LoadContent, LoadQuestion分别将数据加载到ONE对象的数据中：

举一个例子：

|  |
| --- |
| //处理获得的Homepage的Json数据  private void LoadHomepage(string data)  {  JsonObject obj = JsonObject.Parse(data);  string result = obj.GetNamedString("result");  if(result == "SUCCESS")  {  JsonObject hpentity = obj.GetNamedObject("hpEntity");  one.HomepagestrAuthor = hpentity.GetNamedString("strAuthor");  one.HomepagestrContent = " " + hpentity.GetNamedString("strContent");  one.HomepagestrHpTitle = hpentity.GetNamedString("strHpTitle");  one.HomepagestrOriginalImgUrl = hpentity.GetNamedString("strOriginalImgUrl");  }  else  {  isDataLoad = false;  }  } |

在**ONE/ONE.Windows/MainPage.xaml.cs**中，调用LoadData()函数：

|  |
| --- |
| viewModel = new ViewModel(date);  bool flag = await viewModel.LoadData(); |

* + 1. 逻辑跳转模块

在**ONE/ONE.Shared/Strings/App.xaml.cs**中，OnLaunched()函数将启动界面导航至Mainpage:

|  |
| --- |
| protected async override void OnLaunched(LaunchActivatedEventArgs e)  {  …  //日期的初始化  DateTime dt = DateTime.Now;  date = dt.ToString("yyyy-MM-dd");  ...  // 当未还原导航堆栈时，导航到第一页，  // 并通过将所需信息作为导航参数传入来配置参数  if (!rootFrame.Navigate(typeof(MainPage), date))  {  throw new Exception("Failed to create initial page");  }  …  } |

在**ONE/ONE.Windows/MainPage.xaml.cs**中，下面几个函数实现了逻辑跳转：

|  |
| --- |
| private void itemClick(object sender, ItemClickEventArgs e)  {  string date = e.ClickedItem.ToString();  Frame.Navigate(typeof(MainPage),date); //导航到指定日期的主页  }  private void Grid\_Tapped(object sender, TappedRoutedEventArgs e)  {  Frame.Navigate(typeof(PassagePage), viewModel); //导航到内容页  }  private void TextBlock\_Tapped(object sender, TappedRoutedEventArgs e)  {  Frame.Navigate(typeof(QuestionPage), viewModel); //导航到问题页  } |

* + 1. 保存图片模块

在**ONE/ONE.Windows/MainPage.xaml.cs**中，Image\_Tapped()函数可以将图片保存在本地：

|  |
| --- |
| private async void Image\_Tapped(object sender, TappedRoutedEventArgs e)  {  FileSavePicker savepicker = new FileSavePicker();  savepicker.SuggestedStartLocation = PickerLocationId.PicturesLibrary;  savepicker.FileTypeChoices.Add("Image", new List<string> { ".jpg" });  savepicker.SuggestedFileName = viewModel.one.date;  StorageFile savefile = await savepicker.PickSaveFileAsync();  if(savefile != null)  {  viewModel.stream.Seek(0);  IRandomAccessStream input = viewModel.stream;  Stream inputstream  = WindowsRuntimeStreamExtensions.AsStreamForRead(input);  IRandomAccessStream ouput  = await savefile.OpenAsync(FileAccessMode.ReadWrite);  Stream ouputstream  = WindowsRuntimeStreamExtensions.AsStreamForWrite(ouput);  await inputstream.CopyToAsync(ouputstream);  ouputstream.Dispose();  inputstream.Dispose();  }  } |

* + 1. 处理文本溢出模块

在**ONE/ONE.Windows/PassagePage.xaml.cs**及**ONE/ONE.Windows/QuestionPage.xaml.cs**中，loadcontent()函数可以处理因文本长度不定导致的文本溢出问题：

|  |
| --- |
| private void loadcontent()  {  stPanel.Children.Clear();  // 为了支持文本分块，使用RichTextBlock  RichTextBlock tbContent = new RichTextBlock();  tbContent.Width = CT\_WIDTH;  tbContent.Height = CT\_HEIGHT;  tbContent.TextWrapping = TextWrapping.Wrap;  tbContent.Margin = new Thickness(CT\_MARGIN, 0, CT\_MARGIN, CT\_MARGIN + 20);  Paragraph ph = new Paragraph();  Run txtRun = new Run();  txtRun.Text = \_viewmodel.one.ContentstrContent;  ph.Inlines.Add(txtRun);  tbContent.Blocks.Add(ph);  if(Window.Current.Bounds.Height == 1080)  tbContent.FontSize = Convert.ToDouble(32);  else if (Window.Current.Bounds.Height == 1440)  tbContent.FontSize = Convert.ToDouble(38);  else  tbContent.FontSize = Convert.ToDouble(20);  stPanel.Children.Add(tbContent);  // 更新一下状态，方便获取是否有溢出的文本  tbContent.UpdateLayout();  bool isflow = tbContent.HasOverflowContent;  // 因为除了第一个文本块是RichTextBlock，  // 后面的都是RichTextBlockOverflow一个一个接起来的  // 所以我们需要两个变量  RichTextBlockOverflow oldFlow = null, newFlow = null;  if (isflow)  {  oldFlow = new RichTextBlockOverflow();  oldFlow.Width = CT\_WIDTH;  oldFlow.Height = CT\_HEIGHT;  oldFlow.Margin = new Thickness(CT\_MARGIN, 0, CT\_MARGIN, CT\_MARGIN + 20);  tbContent.OverflowContentTarget = oldFlow;  stPanel.Children.Add(oldFlow);  oldFlow.UpdateLayout();  // 继续判断是否还有溢出  isflow = oldFlow.HasOverflowContent;  }  while (isflow)  {  newFlow = new RichTextBlockOverflow();  newFlow.Height = CT\_HEIGHT;  newFlow.Width = CT\_WIDTH;  newFlow.Margin = new Thickness(CT\_MARGIN, 0, CT\_MARGIN, CT\_MARGIN + 20);  oldFlow.OverflowContentTarget = newFlow;  stPanel.Children.Add(newFlow);  newFlow.UpdateLayout();  // 继续判断是否还有溢出的文本  isflow = newFlow.HasOverflowContent;  // 当枪一个变量填充了文本后，  // 把第一个变量的引用指向当前RichTextBlockOverflow  // 确保OverflowContentTarget属性可以前后相接  oldFlow = newFlow;  }  } |

1. 软件设计技术
   1. 面向对象开发技术

本软件采用C#面向对象开发语言开发

C#的技术要点包括C#语言基础、C#的面向对象特征、异常处理、委托与事件、集合、泛型、字符串与正则表达式、特性、反射、程序集与版本控制、.NET框架的内存管理、多线程、Windows窗体、GDI+、与非托管代码互操作等，对每一个知识点都提供了典型实例。

C#是一门可以用来做面向对象开发的语言，以下给出C#类的一个例子

|  |
| --- |
| namespace ONE.Common  {  /// <summary>  /// 专门用于中继自身功能的命令  /// 通过调用委托分配给其他对象。  ///CanExecute 方法的默认返回值为“true”。  /// 在下列情况中，始终需要调用 <see cref="RaiseCanExecuteChanged"/>  /// <see cref="CanExecute"/> 应返回其他的值。  /// </summary>  public class RelayCommand : ICommand  {  private readonly Action \_execute;  private readonly Func<bool> \_canExecute;  /// <summary>  /// 调用 RaiseCanExecuteChanged 时引发。  /// </summary>  public event EventHandler CanExecuteChanged;  /// <summary>  /// 创建可始终执行的新命令。  /// </summary>  /// <param name="execute">执行逻辑。</param>  public RelayCommand(Action execute)  : this(execute, null)  {  }  /// <summary>  /// 创建新命令。  /// </summary>  /// <param name="execute">执行逻辑。</param>  /// <param name="canExecute">执行状态逻辑。</param>  public RelayCommand(Action execute, Func<bool> canExecute)  {  if (execute == null)  throw new ArgumentNullException("execute");  \_execute = execute;  \_canExecute = canExecute;  }  /// <summary>  /// 确定此 <see cref="RelayCommand"/> 是否可在其当前状态下执行。  /// </summary>  /// <param name="parameter">  /// 命令使用的数据。如果不需要向命令传递数据，则可将此对象设置为 null。  /// </param>  /// <returns>如果可执行此命令，则返回 true；否则返回 false。</returns>  public bool CanExecute(object parameter)  {  return \_canExecute == null ? true : \_canExecute();  }  /// <summary>  /// 对当前命令目标执行 <see cref="RelayCommand"/>。  /// </summary>  /// <param name="parameter">  /// 命令使用的数据。如果不需要向命令传递数据，则可将此对象设置为 null。  /// </param>  public void Execute(object parameter)  {  \_execute();  }  /// <summary>  /// 用于引发 <see cref="CanExecuteChanged"/> 事件的方法  /// 执行 <see cref="CanExecute"/> 的返回值  /// 方法已更改。  /// </summary>  public void RaiseCanExecuteChanged()  {  var handler = CanExecuteChanged;  if (handler != null)  {  handler(this, EventArgs.Empty);  }  }  }  } |

* 1. .NET平台开发

.NET就是微软用来实现XML，Web Services，SOA（面向服务的体系结构service-oriented architecture）和敏捷性的技术。

.NET作为Microsoft XML Web services 平台。XML Web services 允许应用程序通过 Internet 进行通讯和共享数据，而不管所采用的是哪种操作系统、设备或编程语言。

通过.NET平台和C#语言的紧密，win8应用开发变得更加方便和强大。

由于是平台性的东西，所以比较难通过代码体现，下面放部分Manifest XML代码。C#代码如上。

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <Package xmlns="http://schemas.microsoft.com/appx/2010/manifest" xmlns:m2="http://schemas.microsoft.com/appx/2013/manifest" xmlns:build="http://schemas.microsoft.com/developer/appx/2012/build" IgnorableNamespaces="build">  <!--  此包清单文件由生成过程生成。  如果重新生成此文件，将丢失对其所做的更改。若要更正此文件中的错误，请编辑源 .appxmanifest 文件。  有关程序包清单文件的详细信息，  请参阅 http://go.microsoft.com/fwlink/?LinkID=241727  -->  <Identity Name= "49117jtZhang.one"  Publisher="CN=6E9F3C3E-70E1-4B95-B599-FB093861F027"  Version="1.1.0.0" ProcessorArchitecture="neutral" />  <Properties>  <DisplayName>one•一个</DisplayName>  <PublisherDisplayName>jtZhang</PublisherDisplayName>  <Logo>Assets\StoreLogo.png</Logo>  </Properties>  <Prerequisites>  <OSMinVersion>6.3.0</OSMinVersion>  <OSMaxVersionTested>6.3.0</OSMaxVersionTested>  </Prerequisites>  <Resources>  <Resource Language="ZH-CN" />  </Resources>  <Applications>  <Application Id="App" Executable="ONE.Windows.exe" EntryPoint="ONE.Windows.App">  <m2:VisualElements  DisplayName="one•一个"  Square150x150Logo="Assets\Logo.png"  Square30x30Logo="Assets\SmallLogo.png"  Description="ONE.Windows" ForegroundText="light" BackgroundColor="#464646">  <m2:DefaultTile>  <m2:ShowNameOnTiles>  <m2:ShowOn Tile="square150x150Logo" />  </m2:ShowNameOnTiles>  </m2:DefaultTile>  <m2:SplashScreen Image="Assets\SplashScreen.png" />  </m2:VisualElements>  </Application>  </Applications> |

* 1. MVC架构

本应用采用的的是MVC架构。M表示Model， V表示View， C表示Controller。

**Model**：在本应用中表示的是数据类：

示例：ViewModel.cs以及Mainpage.xaml.cs的部分内容

|  |
| --- |
| class ViewModel  {    public bool isDataLoad; //判断数据是否获得  public bool isDataGet; //判断数据是否正确处理  public IRandomAccessStream stream;  private string Homepage;  private string Content;  private string Question;  private HttpClient httpClient;  private string date;  //主页、文章和问题的api  private const string HomepageLink  = @"http://211.152.49.184:7001/OneForWeb/one/getHpinfo?strDate=";  //private const string ContentLink  = "http://211.152.49.184:7001/OneForWeb/one/getOneContentInfo?strDate=";  private const string QuestionLink  = @"http://211.152.49.184:7001/OneForWeb/one/getOneQuestionInfo?strDate=";  private const string ContentLink = @"http://onewp.sinaapp.com/";    public ViewModel()  {  }  //构造函数，初始化数据  public ViewModel(string \_date)  {  httpClient = new HttpClient();  httpClient.MaxResponseContentBufferSize = 2560000;  httpClient.DefaultRequestHeaders.Add("User-Agent",  "Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; WOW64; Trident/6.0)");  one = new ONE();  date = \_date;  one.date = \_date;  isDataLoad = true;  isDataGet = true;  getdatelist();    } |

|  |
| --- |
| public sealed partial class MainPage : Page  {  string date;  ViewModel viewModel;    public MainPage()  {  this.InitializeComponent();    }  protected override async void OnNavigatedTo(NavigationEventArgs e)  {  base.OnNavigatedTo(e);  date = (string)e.Parameter;    viewModel = new ViewModel(date);  bool flag = await viewModel.LoadData();  if (flag == true)  {  pageRoot.DataContext = viewModel; //数据绑定  }  else  {  //无网络连接弹出对话  MessageDialog dialog = new MessageDialog("无网络连接！");  await dialog.ShowAsync();    }    } |

**View**：在WIN8应用中, View采用xml文件形式呈现。

示例：Mainpage.xaml 部分代码

|  |
| --- |
| <Page  x:Name="pageRoot"  x:Class="ONE.MainPage"  DataContext="{Binding DefaultViewModel, RelativeSource={RelativeSource Self}}"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:local="using:ONE"  xmlns:data="using:ONE.Data"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  mc:Ignorable="d">  <Page.Resources>  <x:String x:Key="ChevronGlyph">&#xE26B;</x:String>  <!-- TODO: 如果在 App.xaml 中声明了键 AppName，则删除此行-->  <x:String x:Key="AppName">One 一个</x:String>  </Page.Resources> |

**Controller:** 即控制类

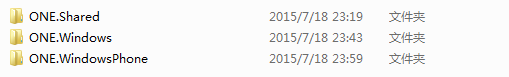
示例：ONE.cs部分代码

|  |
| --- |
| public class ONE  {  public ONE()  {  datelist = new List<string>();  }  public List<string> datelist { get; set; }  /// <summary>  /// 首页Json格式  /// {"result":"SUCCESS",  /// "hpEntity":{  /// "strAuthor":"光年2011&绘图\/枣",  /// "strContent":"。 by 河井宽次郎",  /// "strHpId":"604",  /// "strHpTitle":"VOL.587",  /// "strMarketTime":"2014-05-17",  /// "strOriginalImgUrl":"http:\/\/pic.yupoo.com\/hanapp\/DKIwiYM7\/f2Fs8.jpg",  /// "strThumbnailUrl":"http:\/\/pic.yupoo.com\/hanapp\/DKIwiYM7\/f2Fs8.jpg",  /// }}  /// </summary>  public string date{get;set;} //日期yyyy-mm-dd  public string HomepagestrAuthor { get; set; } //首页图片作者  public string HomepagestrContent { get; set; } //首页句子内容  public string HomepagestrHpTitle { get; set; } //首页期数  public string HomepagestrOriginalImgUrl { get; set; } //首页图片链接  public BitmapImage Homepageimage { get; set; } |

* 1. WIN8通用平台开发模式

通过win8通用平台，可以实现一个应用同时在window平板及PC， 还有windowsphone上同时发布。

如目录结构所示：



将数据共享，然后分别在两个平台上使用数据，实现通用平台开发。

* 1. 代码复用

在windows和windowsPhone上的通用开发实现了代码复用。如上图ONE.shared文件夹就包含了共享的类和数据。

如该目录下的ONE.cs文件，就包含了数据获取，解析等功能，这些代码处理后的API所获得的数据，能同时被两个不同的平台使用，从而实现了代码复用

示例：ONE.cs

|  |
| --- |
| public class ONE  {  public ONE()  {  datelist = new List<string>();  }  public List<string> datelist { get; set; }  /// <summary>  /// 首页Json格式  /// {"result":"SUCCESS",  /// "hpEntity":{  /// "strAuthor":"光年2011&绘图\/枣",  /// "strContent":"。 by 河井宽次郎",  /// "strHpId":"604",  /// "strHpTitle":"VOL.587",  /// "strMarketTime":"2014-05-17",  /// "strOriginalImgUrl":"http:\/\/pic.yupoo.com\/hanapp\/DKIwiYM7\/f2Fs8.jpg",  /// "strThumbnailUrl":"http:\/\/pic.yupoo.com\/hanapp\/DKIwiYM7\/f2Fs8.jpg",  /// }}  /// </summary>  public string date{get;set;} //日期yyyy-mm-dd  public string HomepagestrAuthor { get; set; } //首页图片作者  public string HomepagestrContent { get; set; } //首页句子内容  public string HomepagestrHpTitle { get; set; } //首页期数  public string HomepagestrOriginalImgUrl { get; set; } //首页图片链接  public BitmapImage Homepageimage { get; set; }  /// <summary>  /// {"result":"SUCCESS",  /// "contentEntity":{  /// "sGW":"我相信，人生下来，以为是完整吗，其实是为了分裂。",  /// "strContAuthor":"周耀辉",  /// "strContAuthorIntroduce":"（责任编辑：薛诗汉）",  /// "strContDayDiffer":"5",  /// "strContMarketTime":"2014-05-15",  /// "strContTitle":"骨",  /// "strContent":"balabala"  /// }}  /// </summary>  public string ContentstrContTitle { get; set; } //文章标题  public string ContentstrContAuthor { get; set; } //文章作者  public string ContentstrContAuthorIntroduce { get; set; } //文章编辑  public string ContentstrContent { get; set; } //文章内容  public string ContentsGW { get; set; } //文章句子  /// <summary>  /// {"result":"SUCCESS",  /// "questionAdEntity":{  /// "strAnswerContent"",  /// "strAnswerTitle":"网友答@一个App工作室：",  /// "strQuestionContent":"@一个App工作室问网友：你有哪些无聊的技能？",  /// "strQuestionTitle":"你有哪些无聊的技能？"  /// }}  /// </summary>  public string strQuestionTitle { get; set; } //问题标题  public string strQuestionContent { get; set; } //问题内容  public string strAnswerTitle { get; set; } //回答标题  public string strAnswerContent { get; set; } //回答内容 |

* 1. 数据解析

本应用使用API获取数据，获得的数据格式是JSON格式，并且对数据进行解析。

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。它基于ECMAScript的一个子集。 JSON采用完全独立于语言的文本格式，但是也使用了类似于C语言家族的习惯（包括C、C++、C#、Java、JavaScript、Perl、Python等）。这些特性使JSON成为理想的数据交换语言。 易于人阅读和编写，同时也易于机器解析和生成(网络传输速率)。

示例：ViewModel.cs

|  |
| --- |
| public async Task<bool> LoadData()  {    //获取api上的Json数据  Homepage = await GetDataAsync(HomepageLink + one.date);  Content = await GetDataAsync(ContentLink + one.date + "/content.html");  Question = await GetDataAsync(QuestionLink + one.date);  if (isDataGet == true)  {  //处理获得的Json数据  LoadHomepage(Homepage);  LoadContent(Content);  LoadQuestion(Question);  }  else  {  return false;  }    if(isDataLoad == true)  {  //加载图片  bool res = await LoadImage(one.HomepagestrOriginalImgUrl);  }  else  {  return false;  }  return true;  } |

* 1. 数据绑定

Win8平台开发中，采用了数据绑定技术，将前端的组件和后台的数据进行了绑定。

下面给出首页绑定的代码。

示例：MainPage.xaml

|  |
| --- |
| <!-- One首页 -->  <HubSection Width="600" Header="One 首页" Name="mainpage">  <DataTemplate>  <Grid>  <Grid.RowDefinitions>  <RowDefinition Height="Auto" />  <RowDefinition Height="Auto" />  <RowDefinition Height="Auto" />  </Grid.RowDefinitions>  <Image Grid.Row="0"  IsTapEnabled="True" Tapped="Image\_Tapped"  Source="{Binding one.Homepageimage}" Width="500"/>  <TextBlock Grid.Row="1"  Text ="{Binding one.HomepagestrAuthor}" FontSize="16"  Height="24" TextAlignment="Right" Margin="0,5,0,0"/>  <TextBlock Grid.Row="2"  Text="{Binding one.HomepagestrContent}"  FontSize="20" Height="150" TextWrapping="Wrap" Margin="0,8,0,0" />  </Grid>  </DataTemplate>  </HubSection> |