Silverlight开发基本要点

# 1. MVVM

M + V + VM

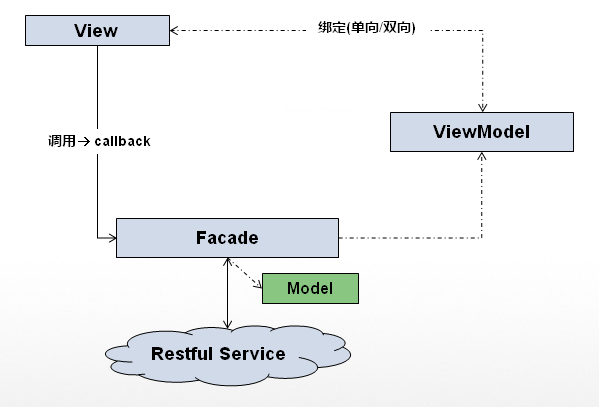
Model + View + ViewModel

## 1.1 标准定义

* View：UI界面
* ViewModel：它是View的抽象，负责View与Model之间信息转换，将View的Command传送到Model；
* Model：数据访问层
* **View与ViewModule连接方式**
* Binding Data：实现数据的传递
* Command：实现操作的调用，Command只能与具有Command属性的控件进行绑定，如按钮，View中使用按钮的Command=”{Binding cmd}”的方式，，ViewModel中申明一个Icommand的属性，然后在构造函数中将cmd注册进去:\_cmd=new ActionCommand(执行方法)；这里ActionCommand是继承ICommand的实现类。
* AttachBehavior：实现控件加载过程中的操作；通常在控件加载时通过Behavior捕获在VM中定义的事件，用于变更需要联动的控件数据。

## 1.2 EC-Central中使用的MVVM模式

为了更易于理解和开发，我们基于MVVM模型提供了一套改造的开发模式：



* ViewModel：继承与ModelBase，只作为一个数据对象（贫血类），并不提供Command和Behavior实现。View通过事件驱动的方式，来调用Façade，与Behavior类似的操作，由Façade来控制。
* Façade：提供Portal与Restful交互的地方，并完成Model与ViewModel的相互转换。
* Model：仅仅是Portal与Restful 进行交互的数据对象。

尤其对于UI操作复杂，控件相互存在互动的View，推荐一种做法：

* Façade中提供一个全局的属性——ViewModel
* View中申明一个全局的Façade；View中的控件实现双向绑定。
* 在PageLoad事件中实例Façade，并将当前View的ViewModel赋给Façade中的VM
* 通过View的事件驱动调用Façade的方法，当存在联动数据操作，直接在Façade中操纵ViewModel的数据，以达到UI上数据自动变换的目的。

总而言之，就是我们尽量在Façade中通过操控ViewModel的数据来达成目的，尽量不要直接修改控件中的值，尽量让数据逻辑放在Façade中，让Façade充当了ViewModel中Behavior的作用。

# 2. RESTful调用

## 2.1 Restful Service Web Method选用及restClient调用方法

1. Get   Portal: restClient.Query

什么时候使用Get：

* 没有条件或者条件是稳定的，这是最重要的先决条件
* 稳定的条件，最多不能超过5个；建议3个以上采用Post方法。

2. Post   Portal: restClient.Query  或  restClient. QueryDynamicData

查询、创建、综合性业务处理，一般使用Post。

3. Put   Portal: restClient.Update

更新数据，对业务处理并发时只关注最后一次处理，使用Put。

4. Delete Portal: restClient.Delete

什么时候用Delete：服务端是非常明确的，并且仅仅是处理物理删除时，采用delete

## 2.2 Restful URL规则

我们并不参照标准RESTful URL规则，而是自定义一套简单、明确的URL规则，规则如下：

URL构成：

BaseURL+      Service路由 +      URL二级路径 + URL三级路径 （如果是Get方法，则在三级路径后面加GET的参数）

例如：

<http://localhost:778> +  CustomerService     +       Customer +      Query       ——Query Customer的URL

<http://localhost:778> +  CustomerService     +       Customer +       Create    ——创建Customer

<http://localhost:778> +  CustomerService     +       Customer +       Update   ——更新Customer

## 2.3 传输数据Data Contract的规则

**原则：**如果能够用BizEntity来在Service端和Client端进行传输（即DataContract），那么则直接使用BizEntity来作为Data Contract；只有当BizEntity无法满足使用时，才定义Request Message和Response Message。

## 2.4 调用Restful服务示例

统一使用ECCentral.Portal.Common.Utilities的RESTClient对象，初始化时传入RESTful接口的Base URL作为参数，所有Domain Service的Base URL都必须按照Domain在Control Panel > Settings > Silverlight Config 中进行配置。

例如：

|  |  |  |
| --- | --- | --- |
|  | **Method** | **Service URL** |
| GetCustomerInfo | [**Get**](http://10.16.230.42/NESO.CustomerCallService/CustomerCallService/help/operations/GetCallReasonTree) | http://localhost:778/CustomerService/Customer/{customerid} |

红色为BaseUrl，蓝色为路由，紫色为相对路径。

需要在Silverlight Configuration中配置BaseUrl：



// Declare Client

string customerCallServiceServiceBase = Window.Configuration.GetConfigValue("Customer", "ServiceBaseUrl");

string routePrefix = "CustomerService";

string preUrl= customerCallServiceServiceBase + routePrefix ;

RestClient restClient = new RestClient(preUrl, this);

// Query CallReasonNotes with POST method

string relativeUrl = string.format("/Customer/{0}",customerID);

restClient.Query<CustomerInfo>( relativeUrl, (obj, result) =>{});

# 3. View代码结构和ViewModel介绍

## 3.1 View代码结构

View代码由三部分构成：

* 资源引用部分：相当于CS文件中Namespace引用，如下所示：

<ovs:PageBase x:Class="ECCentral.Portal.UI.Customer.Views.CustomerQuery"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" xmlns:CommonConverter="clr-namespace:ECCentral.Portal.Common;assembly=ECCentral.Portal.Common" xmlns:controls="clr-namespace:System.Windows.Controls;assembly=System.Windows.Controls"

* 资源使用部分：将需要使用的对象引入，并赋予一个别名，这样正文部分就可以通过别名使用了。
* 正文内容：通常以<Grid x:Name="LayoutRoot">开始，该Grid作为基本容器，为使用者提供内容布局。

## 3.2 ViewModel介绍

* ViewModel必须继承ModelBase
* ModelBase继承ComplexObject : INotifyPropertyChanged, IEditableObject, InotifyDataErrorInfo。
* ModeBase提供SetValue方法来对属性进行赋值，SetValue方法InotifyPropertyChanged，以及NotifyDataError相关的实现。只有基于SetValue赋值的属性，实现双向绑定时，才会发生ViewModel数据与View中控件数据同步变化。

# 4. 常用布局知识

* Grid

宽度、高度：数字，Auto，\*

* StackPanel：简单布局，通过Orientation控制方向
* ovsControls:FunctionPanel
  + - Content
    - Anchors
    - FunctionContent

如果容器中含有ovsDataGrid，那么下列容器必须如下设置：

* Grid

ContentHorizontalScrollBarVisibility="Disabled"

ContentVerticalScrollBarVisibility="Disabled"

* FunctionContent

ContentHorizontalScrollBarVisibility="Disabled" ContentVerticalScrollBarVisibility="Disabled"

# 5. 其它与WebPage开发中常见控件区别

* CheckBox

<CheckBox Name="ckb\_BuyCount" Grid.Row="2" Grid.Column="3" Click="ckb\_BuyCount\_Click" Content="{Binding Path=Label\_BuyCount, Source={StaticResource CustomerQueryResource}}" IsChecked="{Binding Path=IsBuyCountCheck, Mode=TwoWay}"/>

<CheckBox Margin="2 0 10 0" x:Name="chkAgent" IsChecked="{Binding ElementName=expAgentInfo,Path=IsExpanded,Mode=TwoWay}" />

* RadioButton

<RadioButton Name="rbtnBuyCountEndpointValue" Content="{Binding Path=Label\_BuyCountEndpointValue, Source={StaticResource CustomerQueryResource}}" GroupName="S1" IsChecked="{Binding Path=IsBuyCountEndpointValue, Mode=TwoWay}" />

* Combox

<ovsControls:Combox Name="cbCompanySysNo" DisplayMemberPath="CompanyName" ItemsSource="{Binding Path=CompanyList, Mode=TwoWay}" SelectedValue="{Binding Path=CompanyCode, Mode=TwoWay}" SelectedValuePath="CompanyCode" />