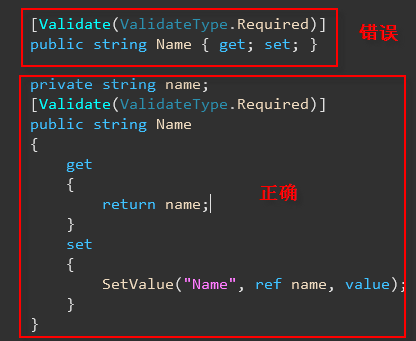
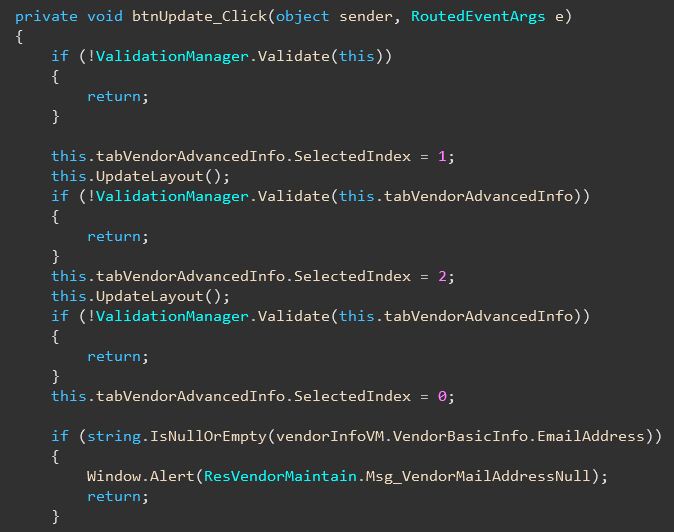
1. **数据有效性验证**
2. 对于数据有效性验证，优先使用Oversea NeweggCentral框架中的ValidationManager，对于使用ValidationManager也可以扩展自定义验证，最后对于需要复杂逻辑检查而无法使用ValidationManager的就只能用Alert框自己来实现验证和错误提示；
3. 使用OverseaNeweggCentral框架中的ValidationManager来做输入数据的有效性验证时，需要注意一下几点：
4. ViewModel需要从ModelBase继承，并且在需要验证数据有效性的属性的set方法中要调用SetValue方法，以及给该属性加上Validate的Attribute，如下面的例子：



1. 在xaml中binding的Mode需要设置为TwoWay，且设置NotifyOnValidationError=true，这样在UI上输入非法数据后VM中验证到错误时，UI才能显示错误
2. 需要调用ValidationManager.Validate方法来进行验证，对于是否验证成功，只需要根据ValidationManager.Validate方法的返回值即可，不再需要去判断VM的ValidationErrors.Count是否大于0
3. 对于ViewModel上的int、decimal等值类型属性，如果需要绑定数据有效性验证，那么都需要将这些字段修改为string类型。因为当用户输入非法的值而不能正确的转换成ViewModel对应属性的类型时，在Silverlight的binding引擎级别就会转换失败，并发生异常，但这个异常会直接被微软的Silverlight的binding引擎给catch掉，然后正常地返回值类型的默认值给ViewModel的对应属性，所以传入到NeweggCentral的UI框架里的将会是值类型的默认值，就会被认为是合法的，从而造成验证失效。
4. 有些特殊场景的验证，比如在一个Page中有TabControl，每个TabItem中的内容都需要验证通过后才能保存，由于每次只有一个TabItem显示，此时其他的TabItem还未在控件树中生成，此时用ValidationManager验证的时候只能验证当前选中的TabItem内容，不能验证全部的TabItem

解决方案：依次对TabControl的TabItem进行验证，如果第一个验证通过了，则改变TabControl的SelectedIndex，并且同时调用UpdateLayout()方法，使UI重绘，让第二个TabItem中的内容立即加载到控件树中，再用ValidationManager对第二个TabItem中的内容进行验证，以此类推；

Demo代码：



1. **控件使用**
2. 在使用ComboBox时要注意SelectionChanged事件的触发机制。对于ComboBox控件，在后台代码里面改变其SelectedIndex时，会触发SelectionChanged事件，在做多个ComboBox 联动的那种控件（比如品牌选择控件，省市区控件）的时候，要注意处理此事件。（你可能改变1个comboBox的SelectedIndex，但会触发其它所有相关的ComboBox的SelectionChanged事件）
3. DataGrid列要实现多语言，应该用附加属性ovsDataControls:DataGridAttached.Header，而不能直接使用原生的Header属性，因为DataGridColumn原生的Header属性不是依赖属性，不支持binding。
4. 对于Oversea NeweggCentral框架里的DataGrid控件，在指定ColumnName的时候，如果有2列的Name是相同的，那么在UI上你点击其中一个Header进行排序，会抛“数组中存在相同元素..”这类的异常，所以要保证一个Grid里面的每一个ColumnName都是唯一的。
5. 对于只读的DataGrid，一定要记得设置：IsReadOnly="True"。
6. DataGrid控件有文字列（DataGridTextColumn）和模板列（DataGridTemplageColumn），如果要设置列的颜色等样式需要使用模板列，并在模板列中放置展示控件。
7. 对于DataGrid表头需要合并的(例如下面图片里的表头)，需要自己写列模板来实现（包括HeaderTemplate的样式，和CellTemplate）。



实现的XAML代码：

|  |
| --- |
| <ovsDataControls:DataGridTemplateColumn CanUserSort="False" CanUserResize="False">  <!—Header，包括样式和Header文字-->  <ovsDataControls:DataGridTemplateColumn.HeaderStyle>  <Style x:Key="IVNTDataGridHeader" targetType="dataprimitives:DataGridColumnHeader">  ……  ……  </Style>  </ovsDataControls:DataGridTemplateColumn.HeaderStyle>  <!—Cell，里面嵌套一个Grid,可以设置多列 -->  <ovsDataControls:DataGridTemplateColumn.CellTemplate>  <DataTemplate>  <Grid>  <Grid.ColumnDefinitions>  <ColumnDefinition Width="80"></ColumnDefinition>  <ColumnDefinition Width="1"></ColumnDefinition>  <ColumnDefinition Width="80"></ColumnDefinition>  <ColumnDefinition Width="1"></ColumnDefinition>  <ColumnDefinition Width="80"></ColumnDefinition>  </Grid.ColumnDefinitions>  ……  ……  </DataTemplate>  </ovsDataControls:DataGridTemplateColumn.CellTemplate>  ……  </ovsDataControls:DataGridTemplateColumn> |

1. Grid控件中，如果涉及到在控件中的数据编辑，则需要建立相应的ViewModel，并使用双向绑定模式，这样可以在控件上数据更新时，同时更新到ViewModel当中，方便批量保存。
2. Grid控件中，一般涉及到批量操作的，都在第一列添加CheckBox控件，来选择相应的数据集合，通过将此checkbox列绑定到ViewModel的bool类型的属性来识别选中状态。
3. 当Checkbox 的IsChecked属性绑定到ViewModel上某个属性时，如：<CheckBox IsChecked="{Binding Path=IsHomePageShow, Mode= TwoWay }"></CheckBox>；如果IsHomePageShow属性为可空类型，且IsHomePageShow的值为null时，会抛出 system.invalidoperationexception异常，异常信息是 cannot resolve targetname，这个异常没办法调试很不好定位；解决办法是让IsHomePageShow为不可空类型。
4. **页面布局**
5. 如果选项较少，比如只有固定的三四项同时页面上布局空间足够，建议就不要用ComboBox而是用连续并排的几个RadionButton代替。
6. Silverlight支持三种布局方式Grid，StackPanel，Canvas；根据场景不同选择合适的布局方式。比如放一排的RadioButton，CheckBox，Button等适合用StackPanel；复杂的布局用Grid；而Canvas在业务系统中应用较少，主要用来做一个拖拽，绝对定位等。
7. 对于在Grid布局中，因为一些列设置为Auto的方式，而导致其它列的输入控件（如TextBox）会因为用户输入的内容增多而自动撑大，为了保证这些输入控件在输入内容增多后能保持原有的宽度，我们需要设置其Width，具体的做法是在PageLoad之后设置其Width，如下：

|  |
| --- |
| public override void OnPageLoad(object sender, EventArgs e)  {          base.OnPageLoad(sender, e);          //页面加载后固定跨列的文本框的宽度          this.txtTitle.Width = this.txtTitle.ActualWidth;          ……  } |

1. **与Restful Service交互**
2. 如果发现RestFul Service 不能访问，如类似下面的错误：

*Call Service http://localhost:778/SOService/SO/Query Failed.*

*Error Detail:System.Net.WebException: 远程服务器返回了错误: NotFound。 ---> System.Net.WebException: 远程服务器返回了错误: NotFound。*

*…….*

则可在浏览器中直接访问Restful Service，如上面的错误就直接访问地址：http://localhost:778/SOService/SO/Query ，就会看到错误的真正原因（需要设置浏览器，在IE的Tools 🡪 Internet Options 🡪 Advanced里，去掉“显示友好的http错误信息“这个checkbox前面的勾选）。

这种问题主要可能是有以下几个原因引起的：

1. 同一个Restful Service中 有方法的重载，这是不允许的；
2. Restful Service 中如果有通过 "GET"方式取得数据的方法，其方法的参数的类型只能是string类型；
3. Service的方法的头标记写错：UriTemplate是有规范的，需要注意；
4. 因为实体对象的属性为值类型，且没有给对象属性赋值，那么会导致其JOSN序列化时出错（解决办法是将属性改为值类型的Nullable类型）；
5. 使用了错误的http method，比如对于删除一定要delete方式的HttpMethod；

对于出现了该错误的情况下，还可以在ECCentral.Service.Utility.WCF工程中的RestServiceErrorHandler类型的public bool HandleError(Exception error)方法里加断点来跟踪debug，也可以看到真实的出错原因；

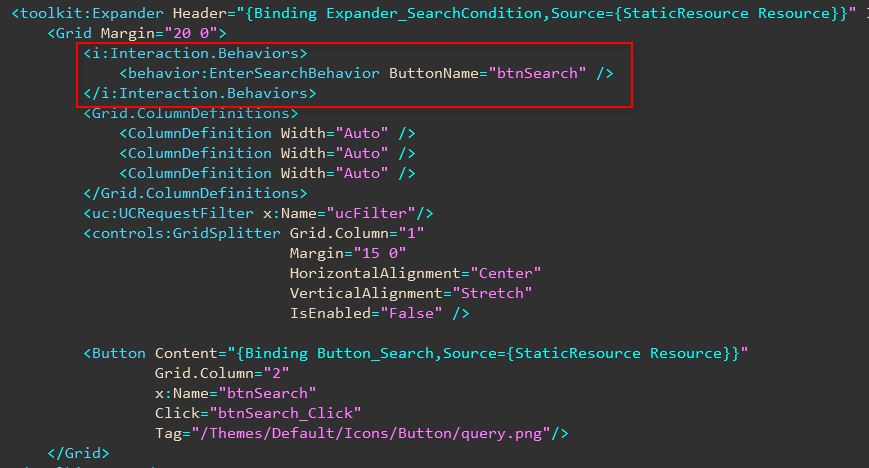
1. 用RestClient.QueryDynamicData请求到的数据类型是Dynamic(DynamicXml)的，目前只支持简单类型的属性，不支付像Silverlight中的Brush,Visibility等类型。遇到这种场景只能转换思路，用简单的数据类型结合Converter来实现。比如根据单据状态等不同在DataGrid中显示不同的颜色等。
2. 由于Silverlight的数据请求异步加载特性，在同时加载多个存在依赖关系的数据源时，需要特别注意数据获取动作的先后顺序，可以使用异步回调方法中嵌套调用数据获取方法，或者用计数器等控制并发。
3. 对于调用Restful Service后的回调方法，我们一般会通过Facade里的一个方法，在XAML的cs里将对应事件传入进去，如下面的代码将EventHandler<RestClientEventArgs<SOInfo>> callback传入：

|  |
| --- |
| public void UpdateSO(SOVM soViewModel, EventHandler<RestClientEventArgs<SOInfo>> callback)  {          string relativeUrl = "/SOService/SO/Update";          SOInfo soInfo = ConvertTOSOInfoFromSOVM(soViewModel);          restClient.Update<SOInfo>(relativeUrl, soInfo, callback);  } |

这里建议不要使用上面代码的方式定义的callback参数的类型，因为restClient.Update<SOInfo>(relativeUrl, soInfo, callback)的callback返回值的类型为SOInfo，我们会将其转换为ViewModel类型：SOVM ，上面代码的写法将会出现，只要在调用UpdateSO方法的地方都要写一遍从SOInfo转换为SOVM的代码，所以为了避免重复代码的问题我们可以用如下方式来解决：更改callback参数的类型为Action，直接返回SOVM：

|  |
| --- |
| public void UpdateSO(SOVM soViewModel, Action<SOVM> callback)  {          string relativeUrl = "/SOService/SO/Update";          SOInfo soInfo = ConvertTOSOInfoFromSOVM(soViewModel);          restClient.Update<SOInfo>(relativeUrl, soInfo, (obj, args) =>              {                  if (!args.FaultsHandle() && callback != null)                  {                      SOVM vm = null;                      if (args.Result != null)                      {                          vm = ConvertTOSOVMFromSOInfo(args.Result);                      }                      callback(vm);                  }              });  } |

1. **其他**
2. 在页面的构造函数里面无法使用CPApplication.Current.CurrentPage，此时的CurrentPage为null，所以对于需要用到CPApplication.Current.CurrentPage的初始化动作都应该放到 OnPageLoad事件里面进行；
3. 在查询区域实现回车查询的功能：目前ECCentral框架中已经实现了一个该功能的behavior，使用的时候只需要把该behavior附加到查询条件的容器上，并指定回车要触发按钮的名字即可，不需要再写额外的代码
4. 引入两个namespace：xmlns:i="clr-namespace:System.Windows.Interactivity;assembly=System.Windows.Interactivity" xmlns:behavior="clr-namespace:ECCentral.Portal.Basic.Behaviors;assembly=ECCentral.Portal.Basic"
5. 把该behavior附加到查询条件的容器上；



1. 如果需要写自己的Page基类或UserControl基类，除了在cs文件中继承自己的基类而外，还需要在xaml中修改基类名称。如：<ovs:PageBase></ovs:PageBase>修改成你的基类；
2. 在编辑Resource文件后，要刻修改其对应的类的构造函数的访问修饰符为public(默认是internal，否则无法在xaml中绑定)；
3. 配置菜单时，相应页面类上要加头标记[view]的attribute；
4. 绑定表达式，命名空间声明等中不能有空格，如果有空格，会导致编译不过。比如下面的绑定表达式会得到一个对应的编译错误，如：

<TextBlock Grid.Row="0" Grid.Column="0" Text="{Binding TextBlock\_WebChannel,Source={StaticResource Res}} "/>



1. ViewModel职责要单一，比如Query和Maintain两种场景最好各定义一个ViewModel,虽然它们有时间属性差不多一样。
2. 当运行时出现页面加载失败时，可能会有如下情况：
3. 页面当中控件Name属性存在相同的控件；
4. 不存在相应的资源文件名称；
5. 控件的操作方法没有实现；
6. 控件binding的属性路径有误；
7. 其他的UI界面的异常；
8. Silverlight 中控件双向绑定不成功，可能原因如下：
9. 没有给控件或父级控件的DataContext 赋值；
10. 控件或父级控件的DataContext 已经赋值，如下代码：

|  |
| --- |
| public partial class SOProcesser : UserControl  {          private SOVM \_soVM=new SOVM ();          private SOVM CurrentSOVM          {              get { return \_soVM; }              set              {                  \_soVM =value;              }          }          public SOProcessor(IPage page, int soSysNo)          {                 InitializeComponent();                 Loaded += new RoutedEventHandler(SOProcesser\_Loaded);          }          private void SOProcesser\_Loaded(object sender, RoutedEventArgs e)          {                 this.DataContext = CurrentSOVM;                 new SOQueryFacade().QuerySOInfo(1005689, vm =>                 {                         CurrentSOVM = vm;                 });          }  } |

此种情况下将会出现相关控件数据绑定不上的错误，原因是：CurrentSOVM 并没有真正的赋值给 this.DataContext。可以设计如下代码来避免上面的错误：

|  |
| --- |
| private SOVM CurrentSOVM          {              get { return \_soVM; }              set              {                  \_soVM =value;                  this.DataContext = \_soVM;              }          } |

1. 对于NoBiz Query，很常见的一种情形是将返回的dynamic数据集合需要绑定到DataGrid上，而DataGrid的第一列需要是Checkbox以支持多选批量操作的情况；对于这种情况，我们需要动态的为返回的dynamic对象增加一个bool的属性，用来绑定到第一列Checkbox上，具体做法如下：

|  |
| --- |
| this.dataGrid1.ItemsSource = args.Result.Rows.ToList("IsChecked", false);  this.dataGrid1.TotalCount = args.Result.TotalCount; |

上面的代码通过ToList方法向返回的dynamic数据集合中的每一个dynamic对象都添加了一个IsChecked的属性，并且赋值为false，这样在Xaml里为DataGrid的第一列的Checkbox绑定到IsChecked就可以了，而且支持双向绑定。