ClipboardSpy

<Separator/>分隔线

IDataObject \_dataObject= Clipboard.GetDataObject();

判断包含的内容

Clipboard.ContainsAudio();

Clipboard.ContainsFileDropList();

Clipboard.ContainsImage();

Clipboard.ContainsText();

Returns a list of all formats that the data in this data object is stored in,

or can be converted to.

var formats = \_dataObject.GetFormats();

\_dataObject.GetData(format, true).ToString()

ClipboardViewer

Clipboard.Clear();

Checks to see whether the data is available in, or can be converted to, a specified

// format; the data format is specified by a string.

dataObject.GetDataPresent(DataFormats.WaveAudio);

dataObject.GetDataPresent(DataFormats.FileDrop);

dataObject.GetDataPresent(DataFormats.Bitmap);

dataObject.GetDataPresent(DataFormats.Text);

dataObject.GetDataPresent(DataFormats.Rtf);

dataObject.GetDataPresent(DataFormats.Xaml);

ApplicationShutdown

Application.Current.ShutdownMode = ShutdownMode.OnLastWindowClose;

Application.Current.ShutdownMode =

(ShutdownMode) Enum.Parse(typeof (ShutdownMode), shutdownModeListBox.SelectedValue.ToString());

var exitCode = 0;

int.TryParse(appExitCodeTextBox.Text, out exitCode);

Application.Current.Shutdown(exitCode);

CodeOnlyWindowsApplication

Wpf默认会为程序生成main方法。可以把实例化mainwindow传给app.Run()，或设置app.Mainwindow=window

public static void Main()

{

// Start the WPF application

var app = new App();

app.Run();

}

Application的事件可以使用覆盖的方式处理，或用handler订阅事件

Startup中可以处理对程序的传参，或者在这里指定程序的mainwindow

protected override void OnStartup(StartupEventArgs e)

{

base.OnStartup(e);

foreach (var arg in e.Args)

{}

// Show main application window.

// NOTE: this window is automatically set as

// App.Current.MainWindow and App.Current.Windows[0]

var window = new MainWindow();

window.Show();

}

ExceptionHandlingSecondaryUIThread

在子线程中开启另一个子窗口

子窗口的Dispatcher.Thread.ManagedThreadId和子线程的id相等Thread.CurrentThread.ManagedThreadId;

子线程的方法

private void MethodRunningOnSecondaryUIThread()

{

var secondaryUiThreadId = Thread.CurrentThread.ManagedThreadId;

try

{

// On secondary thread, show a new Window before starting a new Dispatcher

// ie turn secondary thread into a UI thread

var window = new SecondaryUIThreadWindow();

window.Show();

Dispatcher.Run();

}

catch (Exception ex)

{

// Dispatch the exception back to the main ui thread and reraise it

Application.Current.Dispatcher.Invoke(

DispatcherPriority.Send,

(DispatcherOperationCallback) delegate

{

// THIS CODE RUNS BACK ON THE MAIN UI THREAD

string msg = $"Exception forwarded from secondary UI thread {secondaryUiThreadId}.";

throw new Exception(msg, ex);

}

, null);

// NOTE - Application execution will only continue from this point

// onwards if the exception was handled on the main UI thread

// by Application.DispatcherUnhandledException

}

}

}

App的xmal中定义DispatcherUnhandledException="App\_DispatcherUnhandledException"

private void App\_DispatcherUnhandledException(object sender, DispatcherUnhandledExceptionEventArgs e)

{

主窗口的Thread.CurrentThread.ManagedThreadId和app中处理异常e.Dispatcher.Thread.ManagedThreadId的id相等

sb.AppendFormat("Exception handled on main UI thread {0}.", e.Dispatcher.Thread.ManagedThreadId);

// Keep application running in the face of this exception

e.Handled = true;

}

SkinnedApplication

private void App\_Startup(object sender, StartupEventArgs e)

{

Properties["Blue"] = (ResourceDictionary) LoadComponent(new Uri("BlueSkin.xaml", UriKind.Relative));

}

Application.Current.Resources = (ResourceDictionary) Application.Current.Properties[selectedValue];

ADODataSet

\_appPath = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);

\_myDataSet = new DataSet();

adapter.Fill(\_myDataSet, "BookTable");

// myListBox is a ListBox control.

// Set the DataContext of the ListBox to myDataSet

myListBox.DataContext = \_myDataSet;

<ListBox Name="myListBox" Height="200"

ItemsSource="{Binding Path=BookTable}"

ItemTemplate ="{StaticResource BookItemTemplate}"/>

DataTemplate中的binding只有path是已ItemsSource为源？

BindConversion

public class MyData : INotifyPropertyChanged{

OnPropertyChanged都要调用PropertyChanged?

public DateTime TheDate

{

get { return \_thedate; }

set

{

\_thedate = value;

OnPropertyChanged("TheDate");

}

}

// Declare event

public event PropertyChangedEventHandler PropertyChanged;

// OnPropertyChanged method to update property value in binding

private void OnPropertyChanged(string info)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(info));

}

}

<TextBlock Name="myconvertedtext"

Foreground="{Binding Path=TheDate,

Converter={StaticResource MyConverterReference}}">

<TextBlock.Text>

<Binding Path="TheDate"

Converter="{StaticResource MyConverterReference}"/>

</TextBlock.Text>

</TextBlock>

// 参数:

// value:

// The value produced by the binding source.

//

// targetType:

// The type of the binding target property.

//

// parameter:

// The converter parameter to use.

//

// culture:

// The culture to use in the converter.

public class MyConverter : IValueConverter

{

public object Convert(object o, Type type,

object parameter, CultureInfo culture)

{

var date = (DateTime) o;

switch (type.Name)

{

case "String":

return date.ToString("F", culture);

case "Brush":

return Brushes.Red;

default:

return o;

}

}

BindingDPToDP

DependencyProperty

<ComboBoxItem>Green</ComboBoxItem>

<Canvas.Background>

<Binding ElementName="myComboBox" Path="SelectedItem.Content"/>

</Canvas.Background>

BindingToMethod

<ObjectDataProvider ObjectType="{x:Type local:TemperatureScale}"

MethodName="ConvertTemp" x:Key="ConvertTemp">

<ObjectDataProvider.MethodParameters>

<system:Double>0</system:Double>

<local:TempType>Celsius</local:TempType>

</ObjectDataProvider.MethodParameters>

</ObjectDataProvider>

<TextBox.Text>

<Binding Source="{StaticResource ConvertTemp}" Path="MethodParameters[0]"

BindsDirectlyToSource="true" UpdateSourceTrigger="PropertyChanged"

Converter="{StaticResource DoubleToString}">

<Binding.ValidationRules>

<local:InvalidCharacterRule/>

</Binding.ValidationRules>

</Binding>

</TextBox.Text>

BindsDirectlyToSource获取或设置一个值，该值指示是相对于数据项还是 [DataSourceProvider](https://docs.microsoft.com/zh-cn/dotnet/api/system.windows.data.datasourceprovider?view=netframework-4.8) 对象计算 [Path](https://docs.microsoft.com/zh-cn/dotnet/api/system.windows.data.binding.path?view=netframework-4.8#System_Windows_Data_Binding_Path)。

若要相对于数据项自身计算路径，则为 false；否则为 true。 默认值为 false。

BindValidation

Target

<Label Grid.Column="0" Grid.Row="1" FontSize="15" Margin="2"

Target="{Binding ElementName=textBox1}">TextBox with \_custom ErrorTemplate and ToolTip:</Label>

都是使用Validation附加属性

<Style x:Key="TextBoxInError" TargetType="{x:Type TextBox}">

<Style.Triggers>

<Trigger Property="Validation.HasError" Value="true">

<Setter Property="ToolTip"

Value="{Binding RelativeSource={x:Static RelativeSource.Self},

Path=(Validation.Errors)[0].ErrorContent}"/>

</Trigger>

</Style.Triggers>

</Style>

Validation.Error事件

Validation.Errors依赖属性

<StackPanel Name="stackPanel1" Margin="10"

Loaded="StackPanel1\_Loaded"

Validation.Error="ItemError">

自定义错误模板

<TextBox Name="textBox1" Width="50" FontSize="15"

Validation.ErrorTemplate="{StaticResource ValidationTemplate}"

Style="{StaticResource TextBoxInError}"

Grid.Row="1" Grid.Column="1" Margin="2" />

<ControlTemplate x:Key="ValidationTemplate">

<DockPanel>

<TextBlock Foreground="Red" FontSize="20">!</TextBlock>

<AdornedElementPlaceholder/>

</DockPanel>

</ControlTemplate>

<Binding.ValidationRules>

<local:AgeRangeRule Min="21" Max="130"/>

</Binding.ValidationRules>

<Binding.ValidationRules>

<ExceptionValidationRule />

</Binding.ValidationRules>

public override ValidationResult Validate(object value, CultureInfo cultureInfo)

{

var age = 0;

try

{

if (((string) value).Length > 0)

age = int.Parse((string) value);

}

catch (Exception e)

{

return new ValidationResult(false, "Illegal characters or " + e.Message);

}

if ((age < Min) || (age > Max))

{

return new ValidationResult(false,

"Please enter an age in the range: " + Min + " - " + Max + ".");

}

return new ValidationResult(true, null);

}

Model实现IDataErrorInfo接口

// 参数:

// columnName:

// The name of the property whose error message to get.

//

// 返回结果:

// The error message for the property. The default is an empty string ("").

string this[string columnName] { get; }

// 返回结果:

// An error message indicating what is wrong with this object. The default is an

// empty string ("").

string Error { get; }

public string this[string name]

{

get

{

string result = null;

if (name == "Age")

{

if (Age < 0 || Age > 150)

{

result = "Age must not be less than 0 or greater than 150.";

}

}

return result;

}

}

<TextBox.Text>

<!--By setting ValidatesOnExceptions to True, it checks for exceptions

that are thrown during the update of the source property.

An alternative syntax is to add <ExceptionValidationRule/> within

the <Binding.ValidationRules> section.-->

<Binding Path="Age" Source="{StaticResource Data}"

ValidatesOnExceptions="True"

UpdateSourceTrigger="PropertyChanged">

<Binding.ValidationRules>

<!--DataErrorValidationRule checks for validation

errors raised by the IDataErrorInfo object.-->

<!--Alternatively, you can set ValidationOnDataErrors="True" on the Binding.-->

<DataErrorValidationRule/>

</Binding.ValidationRules>

</Binding>

</TextBox.Text>

CodeOnlyBinding

清除目标元素上的binding

BindingOperations.ClearBinding(MyText, TextBlock.TextProperty);

CollectionBinding

IsSynchronizedWithCurrentItem

// Gets or sets a value that indicates whether a System.Windows.Controls.Primitives.Selector

// should keep the System.Windows.Controls.Primitives.Selector.SelectedItem synchronized

// with the current item in the System.Windows.Controls.ItemsControl.Items property.

<ListBox Width="200" IsSynchronizedWithCurrentItem="True"

ItemsSource="{Binding Source={StaticResource MyFriends}}"/>

ContentControl默认绑到集合的当前元素？

<ContentControl Content="{Binding Source={StaticResource MyFriends}}"

ContentTemplate="{StaticResource DetailTemplate}"/>

等价

Content="{Binding Path=/, Source={StaticResource MyFriends}}"

<DataTemplate x:Key="DetailTemplate">

<TextBlock Grid.Row="0" Grid.Column="1" Text="{Binding Path=FirstName}"/>

</DataTemplate>

CollectionViewSource

Xaml中的xml数据

<XmlDataProvider x:Key="MyTasks" XPath="Tasks/Task">

<x:XData>

<Tasks xmlns="">

<Task Name="Groceries" Priority="2" Type="Home">

<Description>Pick up Groceries and Detergent</Description>

</Task>

</Tasks>

</x:XData>

</XmlDataProvider>

<CollectionViewSource x:Key="MySortedTasks"

Source="{StaticResource MyTasks}">

<CollectionViewSource.SortDescriptions>

<componentModel:SortDescription PropertyName="@Priority" />

</CollectionViewSource.SortDescriptions>

<CollectionViewSource.GroupDescriptions>

<PropertyGroupDescription PropertyName="@Priority" />

</CollectionViewSource.GroupDescriptions>

/CollectionViewSource>

？

<ListBox ItemsSource="{Binding Source={StaticResource Cvs}}"

DisplayMemberPath="CityName" Name="lb">

<ListBox.GroupStyle>

<x:Static Member="GroupStyle.Default"/>

</ListBox.GroupStyle>

</ListBox>

Colors

<RadioButton GroupName="SortBy" Checked="OnSortByChanged">Name</RadioButton>

<RadioButton GroupName="SortBy" Checked="OnSortByChanged">Luminance</RadioButton>

获取RadioButton继承的DataContext，并从中得到default view

var cv = (CollectionView) CollectionViewSource.GetDefaultView((IEnumerable) rb.DataContext);

cv.SortDescriptions.Clear();

cv.SortDescriptions.Add(new SortDescription(sortBy, ListSortDirection.Descending));

ItemsSource="{Binding}"从datacontext中获取binding？

<ListBox Name="colorListBox" Height="455" Width="200" DockPanel.Dock="Left"

ItemsSource="{Binding}"

ItemTemplate = "{StaticResource ColorItemTemplate}"

ItemContainerStyle="{StaticResource ListItemStyle}"

IsSynchronizedWithCurrentItem="True"/>

明细和汇总都共用父元素的datacontext

var newItem = new ColorItem((ColorItem) cv.CurrentItem);

colorList.Add(newItem);

cv.MoveCurrentTo(newItem);

CompositeCollections

<ListBox.ItemsSource>

<CompositeCollection>

<CollectionContainer

Collection="{Binding Source={StaticResource GreekGodsData}}" />

<CollectionContainer

Collection="{Binding Source={StaticResource GreekHeroesData}}" />

<ListBoxItem Foreground="Red">Other Listbox Item 1</ListBoxItem>

<ListBoxItem Foreground="Red">Other Listbox Item 2</ListBoxItem>

</CompositeCollection>

</ListBox.ItemsSource>

<DataTemplate DataType="{x:Type local:GreekGod}">

<TextBlock Text="{Binding Path=Name}" Foreground="Gold"/>

</DataTemplate>

<DataTemplate DataType="Hero">

<TextBlock Text="{Binding XPath=@Name}" Foreground="Blue"/>

</DataTemplate>

DataBindingToStringFomat

<GridViewColumn DisplayMemberBinding="{Binding Path=Price, StringFormat=Now {0:c}!}"/>

<TextBlock.Text>

<MultiBinding StringFormat="{}{0} -- Now only {1:C}!">

<Binding Path="Description"/>

<Binding Path="Price"/>

</MultiBinding>

</TextBlock.Text>

DataTemplatingIntro

DataTrigger在datatemplate中

<DataTemplate.Triggers>

<DataTrigger Binding="{Binding Path=TaskType}">

<DataTrigger.Value>

<local:TaskType>Home</local:TaskType>

</DataTrigger.Value>

<Setter TargetName="border" Property="BorderBrush" Value="Yellow"/>

</DataTrigger>

</DataTemplate.Triggers>

<ListBox Width="400" Margin="10"

ItemsSource="{Binding Source={StaticResource MyTodoList}}"

ItemTemplateSelector="{StaticResource MyDataTemplateSelector}"/>

FindResource用法

public override DataTemplate

SelectTemplate(object item, DependencyObject container)

{

if (item != null && item is Task)

{

var taskitem = (Task) item;

var window = Application.Current.MainWindow;

if (taskitem.Priority == 1)

return

window.FindResource("ImportantTaskTemplate") as DataTemplate;

return

window.FindResource("MyTaskTemplate") as DataTemplate;

}

return null;

}

MultiBinding

ImultiValueConverter

object Convert(object[] values, Type targetType, object parameter, CultureInfo culture);

IValueConverter

object Convert(object value, Type targetType, object parameter, CultureInfo culture);

<TextBlock.Text>

<MultiBinding Converter="{StaticResource MyNameConverter}"

ConverterParameter="FormatLastFirst">

<Binding Path="FirstName"/>

<Binding Path="LastName"/>

</MultiBinding>

</TextBlock.Text>

public object Convert(object[] values, Type targetType, object parameter, CultureInfo culture)

{

string name;

switch ((string) parameter)

{

case "FormatLastFirst":

name = values[1] + ", " + values[0];

break;

default:

name = values[0] + " " + values[1];

break;

}

return name;

}

DataTrigger

<Style TargetType="ListBoxItem">

<Style.Triggers>

<DataTrigger Binding="{Binding Path=State}" Value="WA">

<Setter Property="Foreground" Value="Red" />

</DataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding Path=Name}" Value="Portland" />

<Condition Binding="{Binding Path=State}" Value="OR" />

</MultiDataTrigger.Conditions>

<Setter Property="Background" Value="Cyan" />

</MultiDataTrigger>

</Style.Triggers>

</Style>

DirectionalBinding

OneWay Binding, with TargetUpdated event handling

<TextBlock Grid.Row="1" Grid.Column="1" Name="RentText"

Text="{Binding Path=Rent, Mode=OneWay, NotifyOnTargetUpdated=True}"

TargetUpdated="OnTargetUpdated"/>

后台更新binding源的值

var bindingExpression =

BindingOperations.GetBindingExpression(SavingsText, TextBlock.TextProperty);

var sourceData = (NetIncome) bindingExpression.DataItem;

sourceData.Rent = (int) ((1 + i/100)\*sourceData.Rent);

<!-- TwoWay binding example (default for TextBox), with

UpdateSourceTrigger=LostFocus (default for TextBox) -->

<!-- OneWay binding example (which is default for Text) -->

<TextBlock Grid.Row="4" Grid.Column="1" Name="SavingsText" Text="{Binding Path=Savings}"/>

Text="{Binding Path=TotalIncome, Mode=OneTime}"

EditingCollections

有多个属性变化，NotifyPropertyChanged不传参数名？

public void CancelEdit()

{

\_currentData = \_copyData;

NotifyPropertyChanged("");

}

实现接口

public interface IEditableObject

{

void BeginEdit();

void CancelEdit();

void EndEdit();

}

private ItemData \_copyData;

private ItemData \_currentData;

ItemData 是结构体类型，没有引用，都是值复制

private struct ItemData

{

internal string Description;

internal DateTime OfferExpires;

internal double Price;

}

var editableCollectionView = itemsControl.Items as IEditableCollectionView;

if (!editableCollectionView.CanAddNew)

{

MessageBox.Show("You cannot add items to the list.");

return;

}

// Create a window that prompts the user to enter a new

// item to sell.

var win = new ChangeItem {DataContext = editableCollectionView.AddNew()};

//Create a new item to be added to the collection.

// If the user submits the new item, commit the new

// object to the collection. If the user cancels

// adding the new item, discard the new item.

if ((bool) win.ShowDialog())

{

editableCollectionView.CommitNew();

}

else

{

editableCollectionView.CancelNew();

}

删除

if (!editableCollectionView.CanRemove)

{

MessageBox.Show("You cannot remove items from the list.");

return;

}

if (MessageBox.Show("Are you sure you want to remove " + item.Description,

"Remove Item", MessageBoxButton.YesNo) == MessageBoxResult.Yes)

{

editableCollectionView.Remove(itemsControl.SelectedItem);

}

editableCollectionView.EditItem(itemsControl.SelectedItem);

if ((bool) win.ShowDialog())

{

editableCollectionView.CommitEdit();

}

else

{

editableCollectionView.CancelEdit();

}

Grouping

\_myView = (CollectionView) CollectionViewSource.GetDefaultView(myItemsControl.ItemsSource);

if (\_myView.CanGroup)

{

var groupDescription

= new PropertyGroupDescription("@Type");

\_myView.GroupDescriptions.Add(groupDescription);

}

<ItemsControl.GroupStyle>

<GroupStyle>

<GroupStyle.HeaderTemplate>

<DataTemplate>

<TextBlock FontWeight="Bold" FontSize="15"

Text="{Binding Path=Name}"/>

</DataTemplate>

</GroupStyle.HeaderTemplate>

</GroupStyle>

</ItemsControl.GroupStyle>

HierarchicalDataTemplate

<TreeView>

<TreeViewItem ItemsSource="{Binding Source={StaticResource MyList}}" Header="My Soccer Leagues" />

</TreeView>

<HierarchicalDataTemplate DataType = "{x:Type local:League}"

ItemsSource = "{Binding Path=Divisions}">

<TextBlock Text="{Binding Path=Name}"/>

</HierarchicalDataTemplate>

<HierarchicalDataTemplate DataType = "{x:Type local:Division}"

ItemsSource = "{Binding Path=Teams}">

<TextBlock Text="{Binding Path=Name}"/>

</HierarchicalDataTemplate>

<DataTemplate DataType="{x:Type local:Team}">

<TextBlock Text="{Binding Path=Name}"/>

</DataTemplate>

ListBox选择项的类型？

int.Parse(((sender as ListBox).SelectedItem as ListBoxItem).Content.ToString());

MasterDetail

<StackPanel>

<Label>My Soccer Leagues</Label>

<ListBox ItemsSource="{Binding}"

IsSynchronizedWithCurrentItem="true"/>

</StackPanel>

<StackPanel>

<Label Content="{Binding Path=Name}"/>

<ListBox ItemsSource="{Binding Path=Divisions}"

IsSynchronizedWithCurrentItem="true"/>

</StackPanel>

<StackPanel>

<Label Content="{Binding Path=Divisions/Name}"/>

<ListBox ItemsSource="{Binding Path=Divisions/Teams}"/>

</StackPanel>

PriorityBinding

字段用attr设置

<local:AsyncDataSource SlowestDp="Slowest Value" SlowerDp="Slower Value"

FastDp="Fast Value" x:Key="AsyncDs" />

public string SlowerDp

{

get

{

// This simulates a lengthy time before the

// data being bound to is actualy available.

Thread.Sleep(3000);

return \_slowerDp;

}

set { \_slowerDp = value; }

}

<TextBlock.Text>

<PriorityBinding FallbackValue="defaultvalue">

<Binding Path="SlowestDp" IsAsync="True"/>

<Binding Path="SlowerDp" IsAsync="True"/>

<Binding Path="FastDp" />

</PriorityBinding>

</TextBlock.Text>

SortFilter

MyCollectionView = (ListCollectionView) CollectionViewSource.GetDefaultView(rootElement.DataContext);

MyCollectionView.SortDescriptions.Add(new SortDescription("OrderItem",

ListSortDirection.Ascending));

// Gets or sets a method that is used to determine whether an item is suitable for

// inclusion in the view.

case "Filter":

MyCollectionView.Filter = Contains;

break;

case "Unfilter":

MyCollectionView.Filter = null;

case "Previous":

if (MyCollectionView.MoveCurrentToPrevious())

{

feedbackText.Text = "";

}

else

{

MyCollectionView.MoveCurrentToFirst();

feedbackText.Text = "At first record";

}

break;

case "Next":

if (MyCollectionView.MoveCurrentToNext())

{

feedbackText.Text = "";

}

else

{

MyCollectionView.MoveCurrentToLast();

feedbackText.Text = "At last record";

}

UpdateSource

// itemNameTextBox is an instance of a TextBox

var be = itemNameTextBox.GetBindingExpression(TextBox.TextProperty);

be.UpdateSource();

ValidateItemSample

When a ValidationRule

is added to a BindingGroup, the rule can get the properties of

the source item in the Validate method.

<StackPanel.BindingGroup>

<BindingGroup NotifyOnValidationError="True">

<BindingGroup.ValidationRules>

<local:ValidateDateAndPrice ValidationStep="ConvertedProposedValue" />

</BindingGroup.ValidationRules>

</BindingGroup>

</StackPanel.BindingGroup>

<LineBreak/>

public override ValidationResult Validate(object value, CultureInfo cultureInfo)

{

var bg = value as BindingGroup;

// Get the source object.

var item = bg?.Items[0] as PurchaseItem;

}

private void StackPanel1\_Loaded(object sender, RoutedEventArgs e)

{

// Set the DataContext to a PurchaseItem object.

// The BindingGroup and Binding objects use this as

// the source.

stackPanel1.DataContext = new PurchaseItem();

// Begin an edit transaction that enables

// the object to accept or roll back changes.

stackPanel1.BindingGroup.BeginEdit();

}

private void Submit\_Click(object sender, RoutedEventArgs e)

{

if (stackPanel1.BindingGroup.CommitEdit())

{

MessageBox.Show("Item submitted");

stackPanel1.BindingGroup.BeginEdit();

}

}

private void Cancel\_Click(object sender, RoutedEventArgs e)

{

// Cancel the pending changes and begin a new edit transaction.

stackPanel1.BindingGroup.CancelEdit();

stackPanel1.BindingGroup.BeginEdit();

}

<ControlTemplate TargetType="HeaderedContentControl">

<DockPanel LastChildFill="False">

<ContentPresenter ContentSource="Header" DockPanel.Dock="Left" Focusable="False" VerticalAlignment="Center"/>

<ContentPresenter ContentSource="Content" Margin="5,0,0,0" DockPanel.Dock="Right" VerticalAlignment="Center"/>

</DockPanel>

</ControlTemplate>

private void ItemError(object sender, ValidationErrorEventArgs e)

{

if (e.Action == ValidationErrorEventAction.Added)

{

MessageBox.Show(e.Error.ErrorContent.ToString());

}

}

实现了IEditableObject接口的数据源，需要手动提交，texbox默认的twoway和 UpdateSourceTrigger="PropertyChanged"都不管用？

ValidateItemsInItemsControl

The ValidationRule assigned to ItemsControl.ItemBindingGroup checks

multiple properties in the item

<ObjectDataProvider MethodName="GetValues"

ObjectType="{x:Type sys:Enum}"

x:Key="RegionValues">

<ObjectDataProvider.MethodParameters>

<x:Type TypeName="local:Region" />

</ObjectDataProvider.MethodParameters>

</ObjectDataProvider>

Container是ContentPresenter

var container = (FrameworkElement) customerList.ContainerFromElement(btn);

if (container.BindingGroup.ValidateWithoutUpdate())

{

container.BindingGroup.UpdateSources();

\_bindingGroupInError = null;

MessageBox.Show("Item Saved");

}

public override ValidationResult Validate(object value, CultureInfo cultureInfo)

{

var bg = value as BindingGroup;

var cust = bg.Items[0] as Customer;

if (cust == null)

{

return new ValidationResult(false, "Customer is not the source object");

}

var region = (Region) bg.GetValue(cust, "Location");

var rep = bg.GetValue(cust, "ServiceRepresentative") as ServiceRep;

}

将错误显示在label元素上

<ItemsControl.ItemContainerStyle>

<Style TargetType="{x:Type ContentPresenter}">

<Setter Property="Validation.ValidationAdornerSite"

Value="{Binding ElementName=validationErrorReport}"/>

</Style>

</ItemsControl.ItemContainerStyle>

<Label Name="validationErrorReport"

Content="{Binding RelativeSource={RelativeSource Self},

Path=(Validation.ValidationAdornerSiteFor).(Validation.Errors)[0].ErrorContent}"

Margin="5" Foreground="Red" HorizontalAlignment="Center"/>

Xpath用来指定使用数据源中哪些节点数据

<XmlDataProvider x:Key="InventoryData" XPath="Inventory/Books">

Xpath用来过滤数据

<ListBox.ItemsSource>

<Binding Source="{StaticResource InventoryData}"

XPath="\*[@Stock='out'] | \*[@Number>=8 or @Number=3]"/>

</ListBox.ItemsSource>