Netwoven
Enabling Intelligent Enterprises

Silverlight Tips and Tricks

Silverlight is rapidly emerging as technology of choice for User Interface implementation for web based application. Traditionally desktop based applications provide much richer and intuitive user experience in comparison to web based application. At the onset of internet era the absence of suitable technology to render a desktop like interaction capabilities over a thin client (browser) and the need to make round trips to retrieve data from server over unreliable bandwidth forced web application developers to change the user interaction pattern almost 180^0 when a desktop application was ported to web. Gradually that became standards of web application as users got used to the web application interfaces with their shortcomings. However the last decade saw substantial advancements in technologies dealing with asynchronous communication to server from presentation layer and Silverlight is the latest derivatives of that group from Microsoft. Silverlight literally takes windows on to the web and removes many of the bottle-necks of traditional web based interface. Moreover Silverlight being a cross-browser, cross-platform, cross-device browser plug-in immediately reduces the issues browser compatibility to a great extent.

As Silverlight's popularity grows the developer community has the responsibility to quickly understand the framework and master various tips and tricks. Quickly adopting the framework is all the more important as the technology yet to mature and is emerging at a hectic pace.

The objective of this white paper is to assist the Silverlight developer community with some tips and tricks on some of the most challenging issues that developers frequently encounter while designing applications with Silverlight. Hopefully by the end this paper the developers will have their concepts cleared on these topics as discussed in the remaining sections of the document and will be in a position to try out some sample programs on their own. Following topics have been discussed in the document –

- Data Binding and Dependency Property
- Create Tree View of unlimited depth in Silverlight 3 using HierarchicalDataTemplate
- Implement Row Click event in Silverlight DataGrid
- Implement Mouse Scroll button in SilverLight DataGrid

Data Binding and Dependency Property:

Data binding is single most important concept in Silverlight which makes it unique among similar development platforms. Yet very little is documented on the basics of databinding principle. Any Silverlight control that needs to display some data must be bound to a object as data source. **To bind any object with a Silverlight control in XAML that object must be implemented of type Dependency Property**. **If the bound object is not of type Dependency Property then any runtime changes in the source object data will not reflect in Silverlight control.** The alternate option is to use InotifyHandler but Dependency property is most convenient. The concept and application of Dependency Property can easily be understood following small footsteps and that should be sufficient for most practical requirements. Basics of Dependency Property are explained below and you

can just copy and paste the definition defined as below in your code and change the names as required.

How to define a dependency property:

Step 1: If you want to define a dependency property in a class, you need to inherit the class from DependencyObject. For example -

```
public class Wrapper : DependencyObject
```

Step 2: Register your Dependency Property

```
   public static DependencyProperty ShowPercentProperty =
            DependencyProperty.Register("ShowPercent", typeof(string),
typeof(Wrapper),new PropertyMetadata(PropertyChanged) );
```

Step 3: Wrap in CLR property

```
   public string ShowPercent
        {
            get { return (string)GetValue(ShowPercentProperty); }
            set { SetValue(ShowPercentProperty, value); }
        }
```

Step 4: Define event handler as metadata if required. If tracking the changes in property data is not required you can safely place null in its place.

```
private static void PropertyChanged(object sender,
      DependencyPropertyChangedEventArgs args)
        {

        }
```

Having discussed the basic rules let's go over a typical implementation of dependency property in Silverlight. Here we have used a Text Block control to bind with ShowPercent Dependency Property. If any changes are made to the value of ShowPercent the same will be immediately reflected in the Text Block control.

1. Define the Class

```
public class Wrapper : DependencyObject
    {
public static DependencyProperty ShowPercentProperty =
            DependencyProperty.Register("ShowPercent", typeof(string),
typeof(InfoWrapper), null);

        public string ShowPercent
        {
            get { return (string)GetValue(ShowPercentProperty); }
            set { SetValue(ShowPercentProperty, value); }
        }
}
```

2. Define DataContext in code

```
Wrapper wrapper = new Wrapper();
LayoutRoot.DataContext = wrapper;
```

3. And finally create the XAML

```
<Grid x:Name="LayoutRoot" HorizontalAlignment="Left">
…
<TextBlock x:Name="TextBlockPercent" Text="{Binding ShowPercent}"
Margin="0,0,2,0" VerticalAlignment="Center"></TextBlock>
…
```

That's it and ShowProperty dependency property is ready for use.

Create Tree View of unlimited depth in Silverlight 3 using HierarchicalDataTemplate

TreeView control is ideal to display hierarchical data and populating a treeview control in Silverlight is also fairly straight forward. You can use HierarchicalDataTemplate to populate a tree.

Steps to Implement Treeview control with unlimited depth in Silverlight:

Step 1: Create a Class DocTree

```
public class DocTree
    {
        public FileName File {get;set}
        public List<DocTree> ChildTreeList{get;set;}
    }
```

Step 2: Create a Dependency Property for tree in the class Wrapper

```
public class Wrapper : DependencyObject
    {
public static DependencyProperty docTreeCollectionProperty =
        DependencyProperty.Register("docTreeCollection",
typeof(ObservableCollection<DocTree>), typeof(Wrapper), null);

        public ObservableCollection<DocTree> docTreeCollection
        {
            get { return
(ObservableCollection<DocTree>)GetValue(docTreeCollectionProperty); }
            set { SetValue(docTreeCollectionProperty, value); }
        }
}
```

Step 3: In XAML use the treeview control with HierarchicalDataTemplate Template. You can also use HierarchicalDataTemplate Template as static resource and use this resource as static resource in TreeView control

```
<local:DragDropTree x:Name="TreeViewDest" ItemsSource="{Binding
docTreeCollection}">
            <controls:TreeView.ItemTemplate>
                <common:HierarchicalDataTemplate ItemsSource="{Binding
ChildTreeList}">
                    <StackPanel Orientation="Horizontal" >
                     <TextBlock Text="{Binding FileName></TextBlock>
                    </StackPanel>
                </common:HierarchicalDataTemplate>
            </controls:TreeView.ItemTemplate>
        </local:DragDropTree>
```

Step 4: Set DataContext

```
Wrapper wrapper = new Wrapper();
TreeViewDest.DataContext = wrapper;
```

Step 5: Populate the docTreeCollection defined in Step 2 above.

As soon as docTreeCollection is populated it is shown in the silverlight treeview control, because docTreeCollection is implemented as Dependency Property. If you do not use Dependency Property then each time the docTreeCollection object is changed you have to execute the code TreeViewDest.ItemSource = docTreeCollection to refresh the treeview and reflect your updates.

Implement Row Click event in Silverlight DataGrid

DataGrid in Silverlight 3 does not provide Row Click event out of the box. This is because in DataGrid visible rows are loaded. If you scroll the mouse then automatically those rows which are invisible are unloaded. Since the row index is not available with respect of the first row in the DataGrid the conventional click event of DataGrid could not be implemented. But fortunately there is a workaround as explained below:

Step 1:
In XAML provide the datagrid control as follows:

```
<data:DataGrid x:Name="DataGridDetails"
                          ItemsSource="{Binding SelectedItem.ChildTreeList,
ElementName=TreeViewAccounts}"
LoadingRow="DataGridDetails_LoadingRow"
UnloadingRow="DataGridDetails_UnloadingRow"
                                      IsReadOnly="True"
                          AutoGenerateColumns="False"  FrozenColumnCount="3"
                <data:DataGrid.Columns>
                    <data:DataGridTextColumn Header="Acc Code"
Binding="{Binding AccCode}" Width="86" ElementStyle="{StaticResource
StringSmall}"></data:DataGridTextColumn>
                    <data:DataGridTextColumn Header="Name" Binding="{Binding
AccName}" ElementStyle="{StaticResource StringSmall}"
Width="97"></data:DataGridTextColumn>
                    <data:DataGridTextColumn Header="Op Bal"
Binding="{Binding AccOp,Converter={StaticResource DecimalFormatter}}"
ElementStyle="{StaticResource NumericSmallDisplay}"
Width="73"></data:DataGridTextColumn>
```

```
            </data:DataGrid.Columns>
        </data:DataGrid>
```

Step 2: Implement the `DataGridDetails_LoadingRow"` and `DataGridDetails_UnloadingRow"` events as defined in XAML. You also need to define the MouseLeftButtonDown handler as in following code:

```
#region MouseLeftButtonDown
private void MouseLeftButtonDown(object sender, MouseButtonEventArgs e)
{
   AccTree accTree = (DataGridDetails.SelectedItem as AccTree);
   wrapper.IsBusy = true;
   ChildWindowLedger childLedger = new ChildWindowLedger();
   childLedger.wrapper = wrapper;
   childLedger.Show();
   wrapper.client.GetLedgerInfoAsync(accTree.AccID);
}
#endregion

#region DataGridDetails_LoadingRow
private void DataGridDetails_LoadingRow(object sender, DataGridRowEventArgs
e)
{
  try
  {
    e.Row.AddHandler(MouseLeftButtonDownEvent, new
MouseButtonEventHandler(MouseLeftButtonDown), true);
  }
  catch (Exception ex)
  {
     MessageBox.Show(ex.Message);
  }
 }
 #endregion

 #region DataGridDetails_UnloadingRow
 private void DataGridDetails_UnloadingRow(object sender,
DataGridRowEventArgs e)
 {
   try
   {
      e.Row.RemoveHandler(MouseLeftButtonDownEvent, new
MouseButtonEventHandler(MouseLeftButtonDown));//(, new
MouseButtonEventHandler(), true);
   }
   catch (Exception ex)
   {
      MessageBox.Show(ex.Message);
   }
 }
 #endregion
```

That's it; the row click event is now implemented for Silverlight DataGrid.

Implement Mouse Scroll button in SilverLight DataGrid

Silverlight 3 datagrid OTB does not support mouse scroll button. However this can be accomplished following some simple steps. Following code implements the mouse wheel handler in Silverlight 3. You need to create an event for MouseWheel in XAML. The page behind code is as follows:

```
public void MouseWheelHandler(object sender,
System.Windows.Input.MouseWheelEventArgs e)
{
    DataGrid dataGrid = sender as DataGrid;
    int iCount = 0;
    if (dataGrid.ItemsSource == null)
    {
        return;
    }
    iCount = dataGrid.ItemsSource.Cast<object>().Count() - 1;
    if (dataGrid != null)
    {
        if (e.Delta < 0)
        {
            if (dataGrid.SelectedIndex == iCount)
            {
                return;
            }
            dataGrid.SelectedIndex++;
        }
        else
        {
            if (dataGrid.SelectedIndex <= 0)
            {
                return;
            }
            dataGrid.SelectedIndex--;
        }
    }
    dataGrid.ScrollIntoView(dataGrid.SelectedItem, dataGrid.Columns[0]);
}
```

That's it. The scroll mouse button is implemented for your APP.

About the author

Sushant Agarwal is the CTO of Netwoven (India) and a founder of the Netwoven's India office. He has twenty years of experience in the software industry working on a variety of solutions in the ERP and other custom applications area. He has extensive experience working with both Java and Microsoft .NET platforms developing large scale applications. During the last several years, he has developed several customs and SharePoint based .NET applications for clients in the United States. Developing software has been his passion since the early days of his career. Sushant holds a BS in Computer Science from Birla Institute of Technology (BIT), Ranchi, India. Sushant can be reached at sagarwal@netwoven.com

About Netwoven

Netwoven is a premier professional services firm based in the United States with office in Kolkata, India. Netwoven is a Microsoft Gold Partner specializing in the design and implementation of solutions in the areas of Enterprise Content Management, Business Intelligence and Workflows. Our team of experts is well versed in SharePoint, SQL Server, .NET 4.0, Silverlight and other Microsoft technologies. Please visit our website at http://www.netwoven.com for additional information. Netwoven can also be contacted at info@netwoven.com.