Article

Kickstart Windows Presentation Foundation (WPF)

By **Abdul Rasheed Feroz Khan** on **May 19 2016**

I am heremoving on towards Windows Presentation Foundation (WPF). This is a basic article about WPF.

- About WPF
- Trees in WPF
- Content Model in WPF
- XAML
- Comparision of XAML and without XAML
- Designing WPF User Interface
- Transformations in WPF

About WPF

Windows Presentation Foundation (WPF) application uses hardware acceleration of graphic cards so it will improve your application performance. WPF introduced new markup language called XAML which allows you to keep separate UI and code behind so designers and developers both can work together.

WPF also supports 2D and 3D Graphics, Flow Documents and multimedia. You can create Standalone Applications and Browser Hosted Applications (XBAP) using WPF. It also helps us to overcome Graphics hardware problem.

Trees in WPF

We have two type of Trees in WPF as Logical Trees and Visual Trees.

Logical Trees

Logical Trees refers to the logical representation of UI (User Interface) Control. Logical Trees in WPF helps us to represent the hierarchy of UI controls and WPF elements present in the project.

Example: Windows holds a Stackpanel and Stackpanel holds button.

Visual Trees

Visual Trees refers to the visual representation of UI (User Interface) Control with inner elements. It contains elements or controls which are internally used.

Content Model in WPF

The Content Model under WPF holds the following item:

- ContentControl
- ItemsControl

- HeaderedContentControl
- HeaderedItemsControl
- Panel
- Decorator

ContentControl Class

It is used to display a single item or element as content. Elements of ContentControl Class are Button, CheckBox, Label, Window, etc.,

- **ItemsControl:** It contains multiple items, these elements can be referred to add one by one such as menu, Selector, StatusBar, Listbox, etc.,
- **HeaderedContentControl:** It is used to display content with header.
- **HeaderedItemsControl:** It is used to display multiple items with control such as MenuItem, Toolbar, etc.,
- **Panel:** Panel is used to arrange and position its child element. Canvas, Grid, tabPanel, etc., comes under Panel.
- **Decorator:** It is used to apply additional features and special effects for child elements. Border, viewbox, adornerdecorator, etc., comes under Dectorator.

XAML

XAMLstands for Extensible Application Markup Language. XAML comes with Silverlight , Windows Form, Type Conversion. XAML works with a file compilation process which contains xaml and xaml.cs, XAML file compiles into BAML and CS file compiles into g.cs

BAML – Binary Application Markup Language and g stands for generated, BAML is more efficient for loading and parsing at runtime. It helps with faster loading at runtime.

XAML Namespaces and Root Elements

XAML contains Namespaces and Root Elements. Root element for XAML file can be Window, Page, ResourceDictionary, Application, etc.,

- **x:Key –** Used to set unique key for resource in resource dictionary.
- **x:Name –** You can specify name of an element at runtime.
- **x:Type –** This is used to specify attributes which accepts type.
- **x:Static –** You can refer static members directly in XAML using this prefix.

Markup Extensions

Markup Extensions comes with curly braces '{}'. The markup extensions in WPF are *Binding, StaticResource, DynamicResources, RelativeSource, TemplateBinding* etc., We can create own Markup Extensions by deriving MarkupExtension class. Nested Markup Extensions is also possible in XAML. Comparision of XAML and without XAML:

With XAML:

```
<Grid>
    <Button Name="TestButton" Content="Click Me!" Height="35" Width="150">
   </Button>
</Grid>
```

Without XAML:

```
1. Grid grd = new Grid();
2. Button btn1 = new Button();
3. btn1.Content = "Click Me!";
4. btn1.Name = "TestButton";
5. btn1.Height = 40;
6. btn1.Width = 150;
7. Grid.SetColumn(btn1, 0);
8. Grid.SetRow(btn1, 0);
9. grd.Children.Add(btn1);
10. this.AddChild(grd);
```

Markup Extensions in XAML:

```
1. <Window.Resources>
2.     <Style x:Key="MyStyle" TargetType="{x:Type Button}">
3.         <Setter Property="Background" Value="Red" />
4.     </Style>
5. </Window.Resources>
6.
7. <StackPanel>
8.     <Button Content="Click Me!" Style="{StaticResource MyStyle}" />
9. </StackPanel>
```

Designing WPF User Interface

Layout Panels in WPF:

- StackPanel
- WrapPanel
- DockPanel
- Grid
- UniformGrid
- Canvas

StackPanel: It places elements either horizontally or vertically in a stack. The orientation property used in the StackPanel by default is vertical.

WrapPanel: Elements under WrapPanel are wrapped by default. The orientation property used in the WrapPanel by default is horizontal.

DockPanel: DockPanel aligns the elements to the edge of the container. It placesthe element to TOP, BOTTOM, LEFT and RIGHT edge.

Grid

Grid panel places elements in row and column format similar to table.

Grids contains the following alignments:

- Grid.ColumnDefinitions and Grid.RowDefinitions
- Grid.Row and Grid.Column
- Grid.RowSpan and Grid.ColumnSpan

UniformGrid

It is used to format uniformly, UnifromGrid works with flow direction property. By default UniformGrid

works with the flow direction of left to right.

Canvas

Canvas is for using exact coordinates. It contains Canvas.Left, Canvas.Top, Canvas.Right and Canvas.Bottom

Transformations in WPF

The different sort of transformations in WPF are:

- Rotate Transform
- Scale Transform
- Skew Transform
- Translate Transform
- Matrix Transform

Rotate Transform: It rotates an element to the specified Angle.
Scale Transform: It scales an element by specified ScaleX and ScaleY values.
Skew Transform: It skews an element by specified AngleX and AngleY values.