XAML - MARKUP EXTENSIONS

In XAML applications, markup extensions are a method/technique to gain a value that is neither a specific XAML object nor a primitive type. Markup extensions can be defined by opening and closing curly braces and inside that curly braces, the scope of the markup extension is defined.

Data binding and static resources are markup extensions. There are some predefined XAML markup extensions in **System.xaml** which can be used.

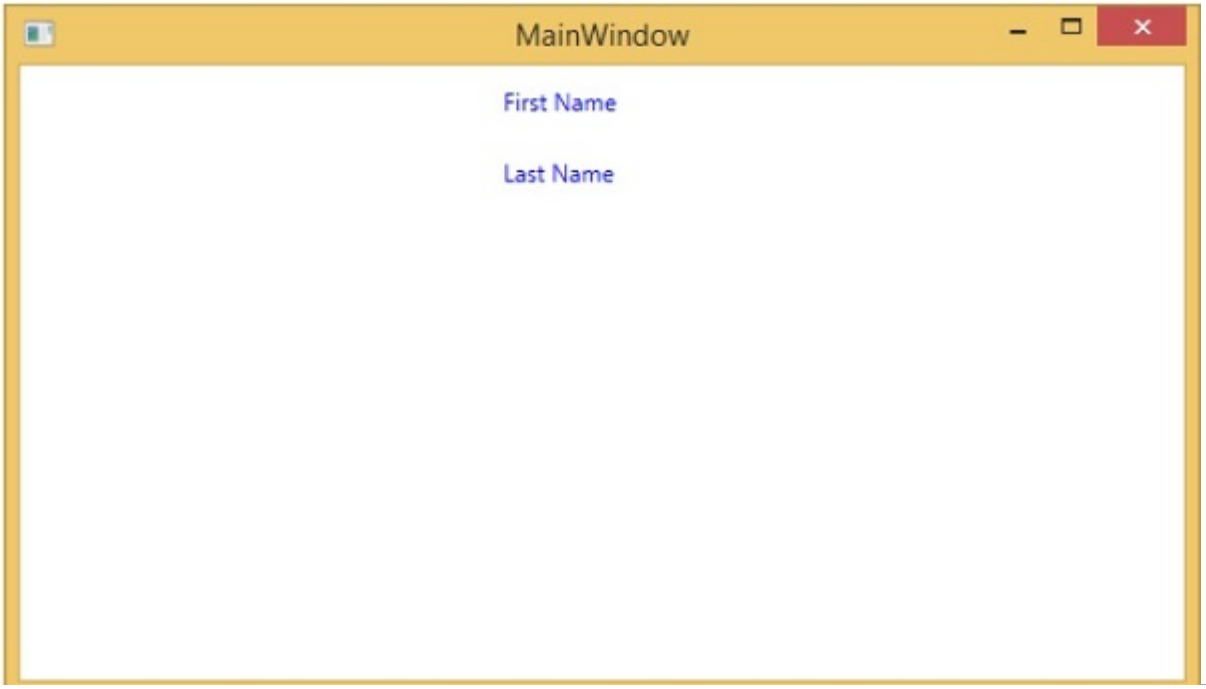
Let's have a look at a simple example where **StaticResources** markup extension is used which is a predefined XAML markup extension.

The following XAML code creates two text blocks with some properties and their foreground is defined in **Window.Resources**.

```
<Window x:Class = "XAMLStaticResourcesMarkupExtension.MainWindow"
   xmlns = "http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x = "http://schemas.microsoft.com/winfx/2006/xaml"
   Title = "MainWindow" Height = "350" Width = "525">

   <Window.Resources>
      <SolidColorBrush Color = "Blue" x:Key = "myBrush"></SolidColorBrush>
   </Window.Resources>

   <Grid>
      <StackPanel Orientation = "Vertical">
         <TextBlock Foreground = "{StaticResource myBrush}" Text = "First Name" Width =
"100" Margin = "10" />
         <TextBlock Foreground = "{StaticResource myBrush}" Text = "Last Name" Width =
"100" Margin = "10" />
      </StackPanel>
   </Grid>

</Window>
```

In **Window.Resources**, you can see **x:Key** is used which uniquely identifies the elements that are created and referenced in an XAML defined dictionary to identify a resource in a resource dictionary.

When you compile and execute the above code, it will produce the following MainWindow. You can see the two text blocks with blue foreground color.



In XAML, custom markup extensions can also be defined by inheriting MarkupExtension class and overriding the ProvideValue method which is an abstract method in the MarkupExtension class.

Let's have a look at a simple example of custom markup extension.

```
<Window x:Class = "XAMLMarkupExtension.MainWindow"
   xmlns = "http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x = "http://schemas.microsoft.com/winfx/2006/xaml"
   xmlns:my = "clr-namespace:XAMLMarkupExtension"
   Title = "MainWindow" Height = "350" Width = "525">

   <Grid>
      <Button Content = "{my:MyMarkupExtension FirstStr = Markup, SecondStr = Extension}"
Width = "200" Height = "20" />
   </Grid>

</Window>
```

In the above XAML code, a button is created with some properties and for the content value, a custom markup extension ***my:MyMarkupExtension*** has been used with two values "Markup" and "Extension" which are assigned to FirstStr and SecondStr respectively.

Actually, `MyMarkupExtension` is a class which is derived from `MarkupExtension` as shown below in the C# implementation. This class contains two string variables, FirstStr and SecondStr, which are concatenated and return that string from the ProvideValue method to the Content of a button.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Markup;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace XAMLMarkupExtension {
   /// <summary>
      /// Interaction logic for MainWindow.xaml
   /// </summary>

   public partial class MainWindow : Window {
      public MainWindow() {
         InitializeComponent();
      }
   }

   public class MyMarkupExtension : MarkupExtension {
      public MyMarkupExtension() { }
      public String FirstStr { get; set; }
      public String SecondStr { get; set; }

      public override object ProvideValue(IServiceProvider serviceProvider) {
        return FirstStr + " " + SecondStr;
      }
   }
}
```

Let's run this application and you can see immediately in our MainWindow that "markup extension"

has been successfully used as the content of the button.