# Set StartPage

using System.Linq;

using System.Reflection.Emit;

using System.Text;

using Xamarin.Forms;

namespace HelloApp

{

    public class Page1 : ContentPage

    {

        public Page1()

        {

            Content = new StackLayout

            {

                Children = {

                    new Label { Text = "Hello Page" }

                }

            };

        }

    }

}

namespace HelloApp

{

    public partial class App : Application

    {

        public App()

        {

            InitializeComponent();

            MainPage = new Page1();

        }

        //...........

    }

}

# DifficultProperty

Кроме простых свойств, которые могут устанавливаться в с помощью простой строки, например, <Label Text="Hello World" />, в XAML могут применяться сложные или комплексные свойства. В таких случаях свойство класса может принимать в качестве значения какой-нибудь сложный объект, который в свою очередь имеет некоторый набор свойств. Сложные свойства определяются в формате:

|  |  |
| --- | --- |
| 1  2  3 | <Элемент.Свойство>      <Сложный\_объект />  </Элемент.Свойство> |

Например, у класса Label есть свойство Style, которое в качестве значения принимает сложный объект одноименного класса Style:

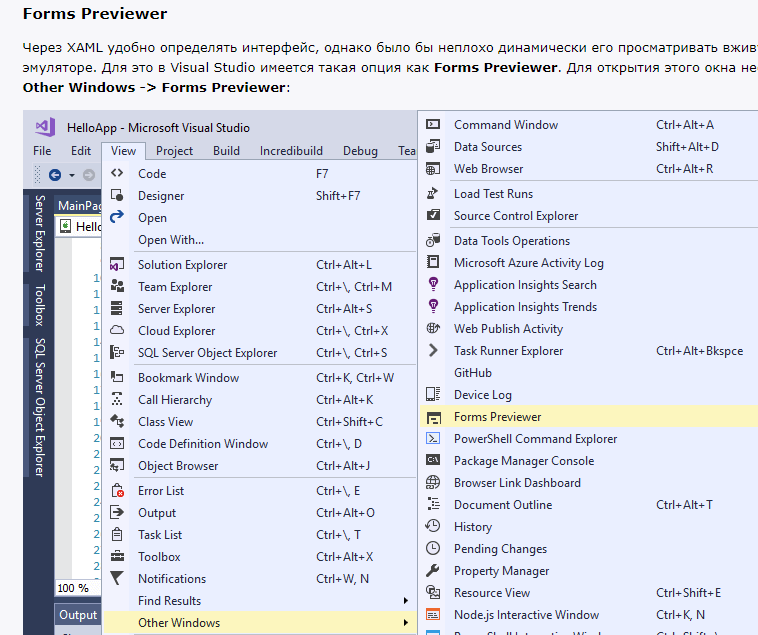
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <Label Text="Hello" HorizontalOptions="Center" VerticalOptions="Center">      <Label.Style>        <Style TargetType="Label">          <Setter Property="TextColor" Value="Red" />          <Setter Property="BackgroundColor" Value="Silver" />        </Style>      </Label.Style>  </Label> |

Элемент Style устанавливает стилевые свойства метки через вложенные элементы Setter.

Также простые свойства можно представить в виде сложных, например вместо VerticalOptions="Center" написать:

|  |  |
| --- | --- |
| 1  2  3 | <Label.VerticalOptions>      <LayoutOptions>Center</LayoutOptions>  </Label.VerticalOptions> |

# FormsPreview



# X:Static

# x:Static

**x:Static** позволяет привязать к атрибуту значение константы или статической переменной. Например, в коде C# определим константу или статическую переменную:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | using Xamarin.Forms;    namespace HelloApp  {      public partial class MainPage : ContentPage      {          public const string HEADER = "Xamarin";            public static double staticVar = 20;            public MainPage()          {              InitializeComponent();            }      }  } |

Тогда мы сможем использовать их значения следующим образом:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="<http://xamarin.com/schemas/2014/forms>"               xmlns:x="<http://schemas.microsoft.com/winfx/2009/xaml>"               xmlns:local="clr-namespace:HelloApp"               x:Class="HelloApp.MainPage">        <Label Text="{x:Static local:MainPage.HEADER}"           FontSize="{x:Static local:MainPage.staticVar}"           VerticalOptions="Center" HorizontalOptions="Center" />  </ContentPage> |

# x:String x:Type

# x:Array и x:Type

Расширение разметки **x:Array** позволяет определить массив данных. А расширение **x:Type** позволяет указать тип данных. Например, воспользуемся элементом ListView для вывода содержимого массива:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="<http://xamarin.com/schemas/2014/forms>"               xmlns:x="<http://schemas.microsoft.com/winfx/2009/xaml>"               xmlns:local="clr-namespace:HelloApp"               x:Class="HelloApp.MainPage">      <ListView>      <ListView.ItemsSource>        <x:Array Type="{x:Type x:String}">          <x:String>iPhone 7</x:String>          <x:String>Samsung Galaxy S7</x:String>          <x:String>HP Elite x3</x:String>        </x:Array>      </ListView.ItemsSource>    </ListView>    </ContentPage> |

В данном случае в определении массива атрибут Type="{x:Type x:String}" указывает, что массив будет содержать данные типа String.

# IMarkupExtension

Иногда может возникнуть необходимость создать какое-то свое расширение для XAML. В этом случае нам надо создать класс, который будет реализовать интерфейс **IMarkupExtension**.

К примеру, мы хотим чтобы через XAML можно было установить для фона элемента цвет из четырех компонент - ARGB. В этом случае мы можем создать следующий класс:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | using System;  using Xamarin.Forms;  using Xamarin.Forms.Xaml;    namespace HelloApp  {      public class ARGBColor : IMarkupExtension      {          public int Alpha { get; set; }          public int Red { get; set; }          public int Green { get; set; }          public int Blue { get; set; }            public object ProvideValue(IServiceProvider serviceProvider)          {              return Color.FromRgba(Red, Green, Blue, Alpha);          }      }  } |

Затем используем этот класс в XAML:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="<http://xamarin.com/schemas/2014/forms>"               xmlns:x="<http://schemas.microsoft.com/winfx/2009/xaml>"               xmlns:local="clr-namespace:HelloApp"               x:Class="HelloApp.MainPage">    <Button Text="Xamarin Forms"           FontSize="Large"           HorizontalOptions="Center"           VerticalOptions="Center"             TextColor="{local:ARGBColor Alpha=255, Red=0, Green=77, Blue=64}"             BackgroundColor="{local:ARGBColor Alpha=255, Red=128, Green=203, Blue=196}">    </Button> |

# StackLayout

Аналогично в XAML мы могли бы написать:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="<http://xamarin.com/schemas/2014/forms>"               xmlns:x="<http://schemas.microsoft.com/winfx/2009/xaml>"               x:Class="HelloApp.MainPage">      <StackLayout x:Name="stackLayout">      <StackLayout.Children>        <Label Text="Первая метка" />        <Label Text="Вторая метка" />      </StackLayout.Children>    </StackLayout>  </ContentPage> |

Либо даже сократить:

|  |  |
| --- | --- |
| 1  2  3  4 | <StackLayout x:Name="stackLayout">      <Label Text="Первая метка" />      <Label Text="Вторая метка" />  </StackLayout> |

Поскольку коллекция Children представляет собой обычный список, то он поддерживает операции по управлению элементами. В частности, мы можем динамически добавить новые элементы. Например:

|  |  |
| --- | --- |
| 1 | stackLayout.Children.Add(new Label { Text = "Третья метка" }); |

Хотя в данном случае коллекция элементов рассматривалась на примере контейнера StackLayout, но тот же принцип работы с элементами будет характерен и для других элементов компоновки.

# ScrollView

# ScrollView

При создании стека или любого другого элемента компоновки может сложиться ситуация, когда не все элементы будут помещаться на экране. В этом случае необходимо создать прокрутку с помощью элемента **ScrollView**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | class MainPage : ContentPage  {      public MainPage()      {          StackLayout stackLayout = new StackLayout();          for (int i = 1; i < 20; i++)          {              Label label = new Label              {                  Text = "Метка " + i,                  FontSize = Device.GetNamedSize(NamedSize.Large, typeof(Label))              };              stackLayout.Children.Add(label);          }          ScrollView scrollView = new ScrollView();          scrollView.Content = stackLayout;          this.Content = scrollView;      }  } |

Или в xaml:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="<http://xamarin.com/schemas/2014/forms>"                         xmlns:x="<http://schemas.microsoft.com/winfx/2009/xaml>"                         xmlns:local="clr-namespace:HelloApp"                         x:Class="HelloApp.MainPage">  <ScrollView>      <StackLayout>        <Label Text="Метка 1" FontSize="Large" />        <Label Text="Метка 2" FontSize="Large" />        <!--..................................................-->        <Label Text="Метка 20" FontSize="Large" />      </StackLayout>    </ScrollView>  </ContentPage> |

# AbsoluteLayout

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="<http://xamarin.com/schemas/2014/forms>"

             xmlns:x="<http://schemas.microsoft.com/winfx/2009/xaml>"

             x:Class="HelloApp.MainPage">

  <AbsoluteLayout>

    <BoxView Color="Red" AbsoluteLayout.LayoutBounds=".1, .2, 50, 80"

        AbsoluteLayout.LayoutFlags="PositionProportional" />

    <BoxView Color="Green" AbsoluteLayout.LayoutBounds="1, .2, 50, 80"

        AbsoluteLayout.LayoutFlags="PositionProportional" />

    <BoxView Color="Blue" AbsoluteLayout.LayoutBounds=".4, .8, .2, .2"

        AbsoluteLayout.LayoutFlags="All" />

 </AbsoluteLayout>

</ContentPage>

# Grid

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

             xmlns:local="clr-namespace:HelloApp"

             x:Class="HelloApp.MainPage">

  <Grid>

    <Grid.ColumnDefinitions>

      <ColumnDefinition Width="\*" />

      <ColumnDefinition Width="\*" />

    </Grid.ColumnDefinitions>

    <Grid.RowDefinitions>

      <RowDefinition Height="\*" />

      <RowDefinition Height="\*" />

    </Grid.RowDefinitions>

    <BoxView Color="Red" Grid.Column="0" Grid.Row="0" />

    <BoxView Color="Blue" Grid.Column="1" Grid.Row="0" />

    <BoxView Color="Yellow" Grid.Column="0" Grid.Row="1" Grid.ColumnSpan="2" />

  </Grid>

</ContentPage>