Работа с элементом управления Button (внешний вид кнопки)

Перейдем к изучению элемента управления Button, который часто используется при разработке пользовательского интерфейса. Элемент управления Button обычно отображает короткую текстовую строку, указывающую на команду, которую необходимо выполнить при нажатии пальцем на элемент управления или при клике мышью по нему. В элементе управления для задания выполняемой команды может также отображаться изображение в виде точечного рисунка или сочетание текста и изображения.

Кнопка Button имеет несколько свойств, влияющих на ее внешний вид:

- TextColor - цвет текста внутри кнопки;
- BackgroundColor - цвет фона кнопки;
- BorderColor - цвет области, окружающей кнопку;
- FontFamily - семейство шрифтов, используемое для текста;
- FontSize - размер текста в кнопке;
- FontAttributes - указывает, выделен ли текст курсивом или полужирным шрифтом;
- BorderWidth - ширина границы кнопки;
- CornerRadius – радиусом скругления углов кнопки;
- CharacterSpacing - интервал между символами текста внутри кнопки;
- TextTransform - регистр текста внутри кнопки.

Свойство IsEnabled для элемента управления Button должно иметь значение true, чтобы Button был доступен для нажатий.

Элемент управления Button инициирует событие Clicked, которое происходит, когда пользователь нажимает на элемент управления Button с помощью пальца или указателя мыши. Помимо события Clicked, элемент управления Button также определяет события Pressed и Released. Событие Pressed возникает, когда палец нажимает на Button или кнопка нажата указателем мыши. Событие Released возникает при отпускании кнопки мыши или пальца. Как правило, событие Clicked также срабатывает одновременно с событием Released.

Для работы с элементами управления Button, Picker, Slider и Switch создайте новый проект ButtonDemo для разработки мультиплатформенного программного приложения Xamarin.Forms с использованием шаблона «Пустой».

Для работы с палитрой цветов в программном приложении сначала создадим класс NamedColor. Для этого подведем курсор мыши к проекту ButtonDemo, нажмем правую кнопку мыши, выберем «Добавить» и далее выберем «Класс». Откроем созданный файл NamedColor.cs и наберем приведенный в презентации программный код.

В программном коде создаются свойства Name, FriendlyName, Color, RgbDisplay, к которым будет обращаться программный код из файлов MainPage.xaml и MainPage.xaml.cs. Приведена функция NamedColor(), которая

формирует массив имеющихся системных цветов, а также соответствующих им названий. Список названий цветов необходим для работы элементов управления Picker, с помощью которых производится выбор цветов из списка (программный код приведен в файле MainPage.xaml)

```
using System;
using System.Collections.Generic;
using System.Reflection;
using System.Text;
using Xamarin.Forms;

namespace ButtonDemo
{
   public class NamedColor
   {
      static StringBuilder stringBuilder = new StringBuilder();

      private NamedColor()
      {
      }

      public string Name { private set; get; }

      public string FriendlyName { private set; get; }

      public Color Color { private set; get; }

      public string RgbDisplay { private set; get; }

      static NamedColor()
      {
         List<NamedColor> all = new List<NamedColor>();

         foreach (PropertyInfo propInfo in typeof(Color).GetRuntimeProperties())
         {
            if (propInfo.GetMethod.IsPublic &&
               propInfo.GetMethod.IsStatic &&
               propInfo.PropertyType == typeof(Color))
            {
               all.Add(CreateNamedColor(propInfo.Name, (Color)propInfo.GetValue(null)));
            }
         }

         foreach (FieldInfo fieldInfo in typeof(Color).GetRuntimeFields())
         {
            if (fieldInfo.IsPublic &&
               fieldInfo.IsStatic &&
               fieldInfo.FieldType == typeof (Color))
            {
```

```
            all.Add(CreateNamedColor(fieldInfo.Name, (Color)fieldInfo.GetValue(null)));
          }
        }
        all.TrimExcess();
        All = all;
      }

      public static IList<NamedColor> All { private set; get; }

      static NamedColor CreateNamedColor(string name, Color color)
      {
        // Преобразование name в FriendlyName.
        stringBuilder.Clear();
        int index = 0;

        foreach (char ch in name)
        {
          if (index != 0 && Char.IsUpper(ch))
          {
            stringBuilder.Append(' ');
          }
          stringBuilder.Append(ch);
          index++;
        }

        NamedColor namedColor = new NamedColor
        {
          Name = name,
          FriendlyName = stringBuilder.ToString(),
          Color = color,
          RgbDisplay = String.Format("{0:X2}-{1:X2}-{2:X2}",
                                     (int)(255 * color.R),
                                     (int)(255 * color.G),
                                     (int)(255 * color.B))
        };
        return namedColor;
      }
    }
}
```

Откроем файл MainPage.xaml и наберем программный код, приведенный далее в презентации. Страница содержит макет StackLayout, в котором располагаются элементы управления Label и Button, а также два макета StackLayout.

В программном коде у элемента управления Button приведены свойства TextColor, BackgroundColor, BorderColor, привязанные с помощью атрибута x:Reference к элементам управления Picker. Элемент управления Button также

связан с обработчиками событий Pressed и Released, программный код для которых приведен в файле MainPage.xaml.cs.

Первый из внутренних макетов StackLayout содержит два элемента управления Label, в которых отображается текст со значениями свойства FontAttributes, которое будет применено к тексту в элементе управления Button, а также элемент управления Switch для переключения между значениями свойства FontAttributes. Элемент управления Switch также связан с обработчиком события OnSwitchToggled, программный код которого приведен в файле MainPage.xaml.cs.

Второй внутренний макет StackLayout содержит три элемента управления Slider для выбора размера шрифта текста внутри кнопки, для выбора толщины границы кнопки, а также для выбора радиуса закругления углов кнопки. Программный код содержит привязку макета StackLayout к элементу управления Button с помощью BindingContext. Далее элементы управления Slider используют привязку Binding к свойствам FontSize, BorderWidth и CornerRadius элемента управления Button.

Также в программном коде приведены три элемента управления Label, предназначенные для отображения значений, выбранных при помощи элементов управления Slider. Для этого с помощью атрибута x:Reference организуется привязка элементов управления Label к соответствующим элементам управления Slider.

Элементы управления Picker предназначены для выбора цвета для свойств TextColor, BackgroundColor, BorderColor из списка, создаваемого с помощью класса NamedColor, программный код которого уже приведен в файле NamedColor.cs.

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
        xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        xmlns:local="clr-namespace:ButtonDemo"
        x:Class="ButtonDemo.MainPage"
        BackgroundColor="Lime">
  <StackLayout>
    <Label x:Name="label"
         Text="Вращение Label"
         FontSize="20"  FontAttributes="Bold"
         VerticalOptions="CenterAndExpand"
         HorizontalOptions="Center"
         BackgroundColor="Red"  TextColor="Yellow"
         Padding="10"
         Margin="50"/>

    <Button x:Name="button"
         Text="Pressed and Released"
         VerticalOptions="CenterAndExpand"
         HorizontalOptions="Center"
         TextColor="{Binding Source={x:Reference textColorPicker},
                     Path=SelectedItem.Color}"
```

```
      BackgroundColor="{Binding Source={x:Reference backgroundColorPicker},
                                                Path=SelectedItem.Color}"
      BorderColor="{Binding Source={x:Reference borderColorPicker},
                                                Path=SelectedItem.Color}"
      Pressed="OnButtonPressed"
      Released="OnButtonReleased"/>
<StackLayout Orientation="Horizontal">
  <Label Text="Button.FontAttributes=None"
       VerticalOptions="Center"
       HorizontalOptions="CenterAndExpand"
       FontSize="12" FontAttributes="Bold"
       TextColor="Black"/>
  <Switch x:Name="BoldSwitch"
        Toggled="OnSwitchToggled"
        HorizontalOptions="CenterAndExpand"/>
  <Label Text="Button.FontAttributes=Bold"
       VerticalOptions="Center"
        HorizontalOptions="CenterAndExpand"
       FontSize="12" FontAttributes="Bold"
       TextColor="Black"/>
</StackLayout>
<StackLayout BindingContext="{x:Reference button}"
         Padding="10">

  <Slider x:Name="fontSizeSlider"
       Maximum="30"
       Minimum="1"
       Value="{Binding FontSize}"
       MinimumTrackColor="Yellow"
       MaximumTrackColor="Blue"/>

  <Label Text="{Binding Source={x:Reference fontSizeSlider},
           Path=Value,
           StringFormat='FontSize = {0:F0}'}"
       HorizontalTextAlignment="Center"
       FontSize="16"
       TextColor="Black"/>

  <Slider x:Name="borderWidthSlider"
       Minimum="-1"
       Maximum="12"
       Value="{Binding BorderWidth}"
        MinimumTrackColor="Yellow"
       MaximumTrackColor="Blue"/>

  <Label Text="{Binding Source={x:Reference borderWidthSlider},
           Path=Value,
           StringFormat='BorderWidth = {0:F0}'}"
       HorizontalTextAlignment="Center"
```

```
        FontSize="16"
       TextColor="Black"/>

<Slider x:Name="cornerRadiusSlider"
          Minimum="-1"
         Maximum="24"
         Value="{Binding CornerRadius}"
        MinimumTrackColor="Yellow"
       MaximumTrackColor="Blue"/>
<Label Text="{Binding Source={x:Reference cornerRadiusSlider},
          Path=Value,
          StringFormat='CornerRadius = {0:F0}'}"
      HorizontalTextAlignment="Center"
      FontSize="16"
      TextColor="Black"/>
<Grid>
   <Grid.RowDefinitions>
      <RowDefinition Height="Auto" />
      <RowDefinition Height="Auto" />
      <RowDefinition Height="Auto" />
   </Grid.RowDefinitions>

   <Grid.ColumnDefinitions>
      <ColumnDefinition Width="*" />
      <ColumnDefinition Width="*" />
   </Grid.ColumnDefinitions>

   <Grid.Resources>
      <Style TargetType="Label">
         <Setter Property="VerticalOptions" Value="Center" />
      </Style>
   </Grid.Resources>

   <Label Text="Text Color:"
         Grid.Row="0" Grid.Column="0"
         TextColor="Black"
          FontSize="16"/>

   <Picker x:Name="textColorPicker"
          ItemsSource="{Binding Source={x:Static local:NamedColor.All}}"
          ItemDisplayBinding="{Binding FriendlyName}"
          SelectedIndex="0"
          Grid.Row="0" Grid.Column="1" />

   <Label Text="Background Color:"
         Grid.Row="1" Grid.Column="0"
         TextColor="Black"
          FontSize="16"/>
```

```
        <Picker x:Name="backgroundColorPicker"
                ItemsSource="{Binding Source={x:Static local:NamedColor.All}}"
                ItemDisplayBinding="{Binding FriendlyName}"
                SelectedIndex="0"
                Grid.Row="1" Grid.Column="1" />

        <Label Text="Border Color:"
               Grid.Row="2" Grid.Column="0"
               TextColor="Black"
               FontSize="16"/>

        <Picker x:Name="borderColorPicker"
                ItemsSource="{Binding Source={x:Static local:NamedColor.All}}"
                ItemDisplayBinding="{Binding FriendlyName}"
                SelectedIndex="0"
                Grid.Row="2" Grid.Column="1" />
      </Grid>
    </StackLayout>
  </StackLayout>
</ContentPage>
```

Откроем файл MainPage.xaml.cs и наберем программный код, приведенный в презентации. В программном коде приведены три обработчика события, на которые производятся ссылки в файле MainPage.xaml.

При нажатии на элемент управления Button и возникновении события Pressed происходит вращение элемента управления Label, к которому в файле MainPage.xaml применен атрибут x:Name и установлено значение label для свойства Name.

Для реализации вращения запускается таймер с использованием возможностей отсчета времени, имеющихся у устройства. Интервал между выполнениями срабатывания таймера равен 20 миллисекундам. Для определения угла поворота элемента управления Button используется экземпляр класса Stopwatch.

При отпускании кнопки возникает событие Released, и вращение элемента управления Label прекращается.

При переключении Switch и возникновении события Toggled срабатывает обработчик события OnSwitchToggled. Событию Toggled соответствует объект ToggledEventArgs, который имеет свойство с именем Value. При возникновении события значение свойства Value отражает новое значение свойства IsToggled.

```
using System;
using System.Diagnostics;
using Xamarin.Forms;
namespace ButtonDemo
{
        public partial class MainPage: ContentPage
        {
```

```
    bool animationInProgress = false;
    bool bold = false;
    Stopwatch stopwatch = new Stopwatch();

            public MainPage()
            {
                InitializeComponent ();
            }
    void OnButtonPressed(object sender, EventArgs args)
    {
      stopwatch.Start();
      animationInProgress = true;
      Device.StartTimer(TimeSpan.FromMilliseconds(20), () =>
      {
        label.Rotation = 360 * (stopwatch.Elapsed.TotalSeconds % 1);

        return animationInProgress;
      });
    }
    void OnButtonReleased(object sender, EventArgs args)
    {
      animationInProgress = false;
      stopwatch.Stop();
    }


    void OnSwitchToggled(object sender, ToggledEventArgs args)
    {
      if (bold)
      {
        bold = false;
        button.FontAttributes = FontAttributes.None;
      }
      else
      {
        bold = true;
        button.FontAttributes = FontAttributes.Bold;
      }

    }
  }
}
```

Запустим программное приложение для работы с элементами управления Button, Picker, Switch и Slider. Программное приложение позволяет экспериментировать с внешним видом элемента управления Button за счет задания значений его свойствам с помощью элементов управления Picker, Switch и Slider. С помощью элементов управления Picker задаются цвет фона элемента управления Button, цвет отображаемого текста, а также цвет границы. С помощью элементов управления Slider задаются размер шрифта

текста, отображаемого в элементе управления Button, радиус закругления углов и толщина границы, обрамляющей элемент управления Button. С помощью элемента управления Switch задается стиль отображения текста в элементе управления Button.

При нажатии на элемент управления Button и удержании нажатия происходит вращение элемента управления Label. При прекращении нажатия на элемент управления Button вращение элемента управления Label прекращается.