Стили

Последнее обновление: 12.09.2016

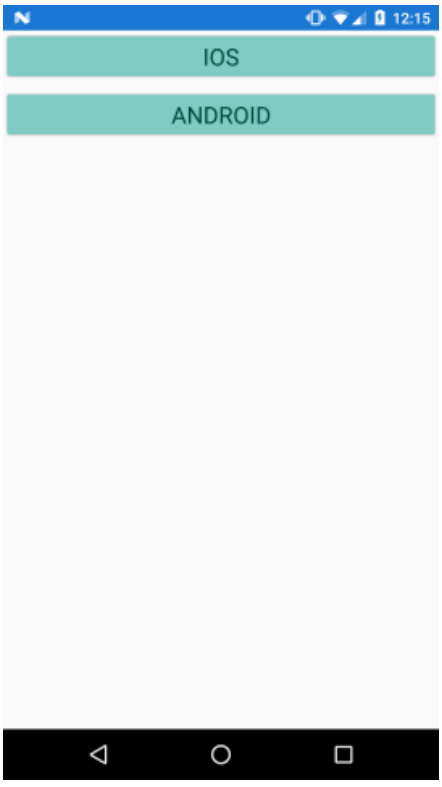
Стили позволяют определить набор некоторых свойств и их значений, которые потом могут применяться к элементам. Основная их задача - создать стилевое единообразие для элементов интерфейса. Стили хранятся в ресурсах и отделяют значения свойств элементов от пользовательского интерфейса.

Чтобы понять, как стили упрощают нам работу, рассмотрим простой пример:

```
<?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="StylesApp.MainPage">
  <StackLayout>
    <Button Text="iOS" TextColor="#004D40" BackgroundColor="#80CBC4" FontSize="Large" />
    <Button Text="Android" TextColor="#004D40" BackgroundColor="#80CBC4" FontSize="Large" />
  </StackLayout>
</ContentPage>
```

Здесь определены две кнопки, которые фактически имеют один и тот же стиль: одни и те же цвет фона и текста, а также размер текста. Единственное отличие состоит в тексте кнопки.

Однако в данном случае мы вынуждены повторяться и повторно определять один и те же свойства и одни и те же значения для каждого из элементов.



Теперь применим стили:

```
<?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="StylesApp.MainPage">
  <ContentPage.Resources>
    <ResourceDictionary>
      <Style x:Key="buttonStyle" TargetType="Button">
        <Setter Property="TextColor" Value="#004D40" />
        <Setter Property="BackgroundColor" Value="#80CBC4" />
        <Setter Property="FontSize" Value="Large" />
      </Style>
    </ResourceDictionary>
  </ContentPage.Resources>
  <StackLayout>
    <Button Text="iOS" Style="{StaticResource buttonStyle}" />
    <Button Text="Android" Style="{StaticResource buttonStyle}" />
  </StackLayout>
</ContentPage>
```

Стиль создается как ресурс с помощью объекта Style и, как любой другой ресурс, он обязательно должен иметь ключ. Атрибут **TargetType** указывает, к какому типу относится стиль. В данном случае это тип `Button`.

С помощью коллекции Setters определяется группа свойств, входящих в стиль. В нее входят объекты Setter, которые имеют следующие свойства:

- **Property**: указывает на свойство, к которому будет применять данный сеттер. При этом свойство должно представлять тип BindableProperty
- **Value**: собственно значение свойства

Иногда свойство может представлять сложный объект, либо же значение формируется сложным способом. Например, у структуры Color есть статический метод `Color.FromRgb()`, который создает цвет по трем значениям. В этом случае мы можем расписать формирование объекта:

```
<Style x:Key="buttonStyle" TargetType="Button">
  <Setter Property="TextColor">
        <Setter.Value>
            <Color>
                <x:Arguments>
                    <x:Double>0</x:Double>
                    <x:Double>0.75</x:Double>
                    <x:Double>0.25</x:Double>
                </x:Arguments>
            </Color>
        </Setter.Value>
    </Setter>
</Style>
```

Также в качестве значения можно устанавливать ссылку на другой ресурс:

```
<ContentPage.Resources>
    <ResourceDictionary>
      <Color x:Key="greenColor">#004D40</Color>
      <Style x:Key="buttonStyle" TargetType="Button">
        <Setter Property="TextColor" Value="{StaticResource Key=greenColor}" />
        <Setter Property="BackgroundColor" Value="#80CBC4" />
        <Setter Property="FontSize" Value="Large" />
      </Style>
    </ResourceDictionary>
</ContentPage.Resources>
```

Если нам надо создать общий стиль для элементов определенного типа, то можно не задавать ключ ресурса, а достаточно установить у стиля атрибут **TargetType**, в который передается тип элементов:

```
<ContentPage.Resources>
  <ResourceDictionary>
    <Color x:Key="greenColor">#004D40</Color>
    <Style TargetType="Button">
      <Setter Property="TextColor" Value="{StaticResource Key=greenColor}" />
      <Setter Property="BackgroundColor" Value="#80CBC4" />
      <Setter Property="FontSize" Value="Large" />
    </Style>
  </ResourceDictionary>
</ContentPage.Resources>
<StackLayout>
  <Button Text="iOS" />
  <Button Text="Android" />
</StackLayout>
```

Теперь у кнопок не надо будет указывать ресурс стиля, так как стиль будет автоматически применяться ко всем объектам типа, который указан в атрибуте TargetType.

Установка стилей в коде

Для создания стиля в коде используется объект Style:

```
using Xamarin.Forms;

namespace StylesApp
{
    public partial class MainPage : ContentPage
    {
        public MainPage()
        {
            Resources = new ResourceDictionary
            {
                {
                    "buttonStyle", new Style(typeof(Button))
                    {
                        Setters =
                        {
                            new Setter
                            {
                                Property = Button.TextColorProperty,
                                Value = Color.FromRgb(0, 77, 64)
                            },
                            new Setter
                            {
                                Property = Button.BackgroundColorProperty,
                                Value = Color.FromRgb(128, 203, 196)
                            },
                            new Setter
                            {
                                Property = Button.FontSizeProperty,
                                Value = Device.GetNamedSize(NamedSize.Large, typeof(Button))
                            }
                        }
                    }
                }
            };

            Button button1 = new Button { Text = "iOS", Style=(Style)Resources["buttonStyle"] };
            Button button2 = new Button { Text = "Android", Style = (Style)Resources["buttonStyle"] };

            Content = new StackLayout
            {
                Children = {button1, button2}
            };
        }
    }
}
```

В конструктор объекта Style передается тип, для которого предназначен данный стиль - аналогично использованию атрибута TargetType в XAML.

При создании стиля в коде следует учитывать, что в качестве свойств указываются именно BindableProperty (как правило называется по имени обычного свойства с суффиксом *Property*). Например:

```
Property = Button.TextColorProperty
```

А не просто TextColor. Причем в начале идет тип (в данном случае Button), а потом идет название свойства (здесь TextColorProperty).

Правда, здесь надо отметить, что в коде помещать стиль в ресурсы не имеет смысла, так как мы можем напрямую присвоить кнопке или другому элементу определенный стиль:

```
public partial class MainPage : ContentPage
{
    public MainPage()
    {
        Style buttonStyle = new Style(typeof(Button))
        {
            Setters =
            {
                new Setter
                {
                    Property = Button.TextColorProperty,
                    Value = Color.FromRgb(0, 77, 64)
```

```
                },
                new Setter
                {
                    Property = Button.BackgroundColorProperty,
                    Value = Color.FromRgb(128, 203, 196)
                },
                new Setter
                {
                    Property = Button.FontSizeProperty,
                    Value = Device.GetNamedSize(NamedSize.Large, typeof(Button))
                }
            }
        };
        Button button1 = new Button { Text = "iOS", Style=buttonStyle};
        Button button2 = new Button { Text = "Android", Style = buttonStyle };

        Content = new StackLayout
        {
            Children = {button1, button2}
        };
    }
}
```

Наследование стилей

С помощью свойства **BasedOn** можно наследовать один стиль от другого. Например:

```
<?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="StylesApp.MainPage">
  <ContentPage.Resources>
    <ResourceDictionary>
      <Style x:Key="baseButtonStyle" TargetType="Button">
        <Setter Property="FontSize" Value="Large" />
        <Setter Property="FontFamily" Value="Verdana" />
      </Style>
      <Style x:Key="greenButtonStyle" TargetType="Button" BasedOn="{StaticResource Key=baseButtonStyle}">
        <Setter Property="TextColor" Value="#004D40" />
        <Setter Property="BackgroundColor" Value="#80CBC4" />
      </Style>
    </ResourceDictionary>
  </ContentPage.Resources>
  <StackLayout>
    <Button Text="iOS" Style="{StaticResource Key=greenButtonStyle}" />
    <Button Text="Android" Style="{StaticResource Key=greenButtonStyle}" />
  </StackLayout>
</ContentPage>
```

Здесь у стиля `greenButtonStyle` атрибут `BasedOn` указывает на другой стиль `baseButtonStyle`. И таким образом, стиль greenButtonStyle будет перенимать все установки от baseButtonStyle.

Наследование в коде осуществляется с помощью установки у стиля свойства BasedOn:

```
Style basedStyle = new Style(typeof(Button));
Style childStyle = new Style(typeof(Button))
{
    BasedOn = basedStyle
};
```

Встроенные стили

Xamarin Forms предоставляет ряд встроенных стилей, которые мы можем использовать:

- `BodyStyle`: стиль для текста страницы
- `CaptionStyle`: стиль для подписей к изображениям и т.д.
- `ListItemDetailTextStyle`: стиль для детального описания в списке
- `ListItemTextStyle`: стиль для текста элемента списка
- `SubtitleStyle`: стиль для подзаголовка
- `TitleStyle`: стиль для заголовка страницы

В то же время эти стили имеют ограничения: они применяются только к элементу Label и устанавливают стилевые настройки только для текста метки.

Применение стилей:

```
<Label Text="Заголовок" Style="{DynamicResource TitleStyle}" />
```


Назад Содержание Вперед