



Наследование элемента и рендерера

Последнее обновление: 02.10.2016



Модульные Дата-центры

GreenMDC

Высокоэффективный,
надежный, готовый дата
центр за 16 недель!

Нам необязательно создавать новый элемент с нуля. Мы можем унаследовать свой класс элемента от уже готового класса. К примеру, переопределим элемент Button, добавив в Portable-проект следующий класс:

```
public class CustomButton : Button
{
}
```

Пусть этот элемент выводится на страницу:

```
<?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:CustomRendererApp;assembly=CustomRendererApp"
  x:Class="CustomRendererApp.MainPage">

  <local:CustomButton Text="Custom Button" HorizontalOptions="Center"
  VerticalOptions="Center"></local:CustomButton>
</ContentPage>
```

И далее добавим класс рендерера для этого элемента в проект для UWP:

```
using CustomRendererApp;
using CustomRendererApp.UWP;
using Xamarin.Forms.Platform.UWP;
using Windows.UI.Xaml.Media;

[assembly: ExportRenderer(typeof(CustomButton), typeof(CustomButtonRenderer))]
namespace CustomRendererApp.UWP
{
    public class CustomButtonRenderer: ButtonRenderer
    {
        protected override void OnElementChanged(ElementChangedEventArgs<Xamarin.Forms.Button>
e)
        {
            base.OnElementChanged(e);
            if(Control!=null)
            {
                Control.Background = new SolidColorBrush(Windows.UI.Colors.LightBlue);
                Control.BorderBrush = new SolidColorBrush(Windows.UI.Colors.CadetBlue);
            }
        }
    }
}
```

```

    }
  }
}

```

Так как для элемента Button уже есть свой рендерер - ButtonRenderer, то мы можем наследовать от этого класса. И фактически через механизм наследования мы также наследуем всю логику по реализации нативного элемента управления. Поэтому после вызова `base.OnElementChanged(e)` у нас уже будет сформирован нативный элемент, который при желании мы можем подкорректировать, например, как здесь, установив цвет фона и границы. Либо же можно полностью переопределить элемент, создав свой нативный объект Button.

Но в любом случае нам также надо использовать атрибут `ExportRenderer`.

Но в данном случае необязательно наследовать класс рендера от ButtonRenderer. Это может быть и ViewRenderer, особенно если мы хотим сопоставить с элементом CustomButton не нативный класс Button, а какой-то другой. Например:

```

using CustomRendererApp;
using CustomRendererApp.UWP;
using Xamarin.Forms.Platform.UWP;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Media;
using System.ComponentModel;

[assembly: ExportRenderer(typeof(CustomButton), typeof(CustomButtonRenderer))]
namespace CustomRendererApp.UWP
{
    public class CustomButtonRenderer: ViewRenderer<CustomButton, Border>
    {
        protected override void OnElementChanged(ElementChangedEventArgs<CustomButton> e)
        {
            base.OnElementChanged(e);
            if (Control == null)
            {
                Border buttonBorder = new Border
                {
                    CornerRadius = new CornerRadius(25),
                    Padding= new Thickness(4),
                    Background = new SolidColorBrush(Windows.UI.Colors.LightBlue),
                    BorderBrush = new SolidColorBrush(Windows.UI.Colors.CadetBlue),
                    BorderThickness = new Thickness(2)
                };
                ContentPresenter contentPresenter = new ContentPresenter
                {
                    VerticalAlignment = VerticalAlignment.Center,
                    HorizontalAlignment = HorizontalAlignment.Center
                };
                buttonBorder.Child = contentPresenter;
                buttonBorder.Tapped += (o, args) => {
                    ((Xamarin.Forms.IButtonController)Element).SendClicked(); };

                SetNativeControl(buttonBorder);
                if (e.NewElement != null)
                {
                    SetText();
                }
            }
        }

        protected override void OnElementPropertyChanged(object sender,
            PropertyChangedEventArgs e)

```

```
{
    base.OnElementPropertyChanged(sender, e);

    if (e.PropertyName == HeaderView.TextProperty.PropertyName)
    {
        SetText();
    }
}

private void SetText()
{
    ContentPresenter contentPresenter = Control.Child as ContentPresenter;
    contentPresenter.Content = Element.Text;
}
}
```

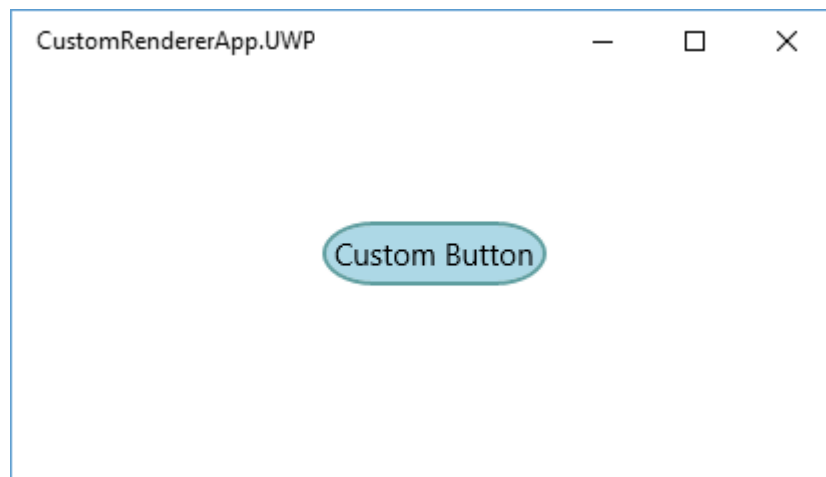
Здесь в качестве нативного элемента выступает объект `Border` - специальный элемент в UWP, представляющий границу вокруг некоторого содержимого. Содержимым же в данном случае выступает объект `ContentPresenter`. В частности через его свойство `Content` устанавливается текст кнопки.

Также стоит отметить, что для поддержки событий нажатия на наш элемент `CustomButton` здесь при нажатии на элемент `Border` срабатывает делегат:

```
buttonBorder.Tapped += (o, args) => { ((Xamarin.Forms.IButtonController)Element).SendClicked();
};
```

Благодаря этому при нажатии сработает событие `Clicked` у элемента `CustomButton`.

В итоге мы получим следующую кнопку:



Аналогичным образом можно создать рендеры и для других платформ.



[Назад](#) [Содержание](#) [Вперед](#)



Купите новый iPhone 8

Оригинальный iPhone 8 в Беларуси. Гарантия от Apple 1 год.
Доставка от 1 до 24ч. Закажите! applecafe.by



2 Комментариев metanit.com

1 Войти ▾

♥ Рекомендовать

🔗 Поделиться

Лучшее в начале ▾



Присоединиться к обсуждению...

ВОЙТИ С ПОМОЩЬЮ

ИЛИ ЧЕРЕЗ DISQUS (?)

**Andrey Lysenko** • 2 месяца назад

Подскажите, пожалуйста! прекомпилятор ругается, красным выделяет определение класса рендерера

```
public class CustomButtonRenderer : ViewRenderer<custombutton, border="">
{
```

пишет что не может использовать CustomButton в качестве аргумента ViewRenderer, так как нет преобразования явной ссылки из<мое пространство имен>.CustomButton в Xamarin.Forms.View

То есть не хочет воспринимать его как элемент xamarin forms

^ | ▾ • Ответить • Поделиться ›

**Andrey Lysenko** ➔ Andrey Lysenko • 2 месяца назад

ахахах, разобрался, сразу же, противоречие в ссылках, Вот дубина, вы же не зря указали Xamarin.Forms.Button - для идиотов, а я подставил Windows.UI.Xaml.Controls

^ | ▾ • Ответить • Поделиться ›

ТАКЖЕ НА METANIT.COM

Go | http.Client

1 комментарий • 2 месяца назад

Roman — Спасибо. Замечательные статьи. Но хотелось бы больше сетевого программирования. Очень буду ждать новых

Django | CRUD. Операции с моделями

3 комментария • 11 дней назад

Metanit — это просто перечисление общих операций с данными. А в каком контексте их использовать, это уже решать вам. Для

Angular в ASP.NET Core | CRUD и маршрутизация. Часть 2

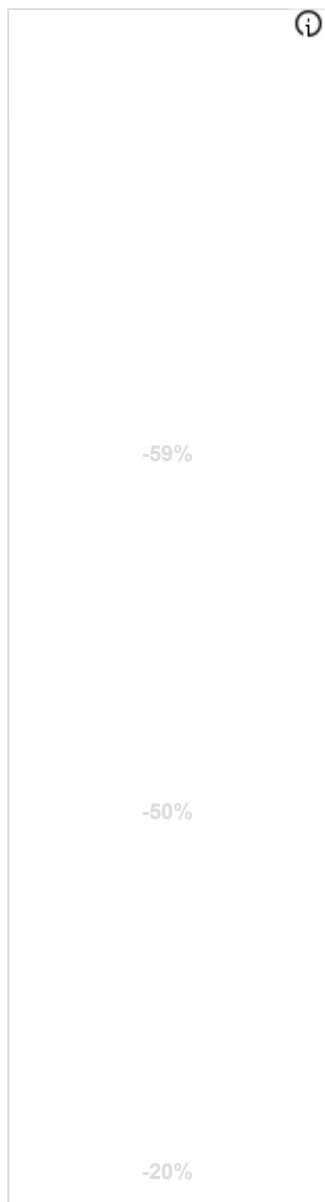
7 комментариев • 3 месяца назад

Igor Teterkin — Поправка - это в IE11. В google chrome всё ок

Go | Синхронизация

1 комментарий • 2 месяца назад

Roman — Получается, что каналы и мапы передаются в функцию по ссылке, верно?



[Вконтакте](#) | [Twitter](#) | [Google+](#) | [Канал сайта на youtube](#) | [Помощь сайту](#)

Контакты для связи: metanit22@mail.ru

Copyright © metanit.com, 2012-2017. Все права защищены.