Xamarin.Forms. Розробка користувацького інтерфейсу засобами XAML.

BoxView

BoxView представляє прямокутник. Найчастіше BoxView використовується для створення зафарбованих областей, або як декоративне примітивне графічне оформлення до інших елементів. Основні властивості класу BoxView:

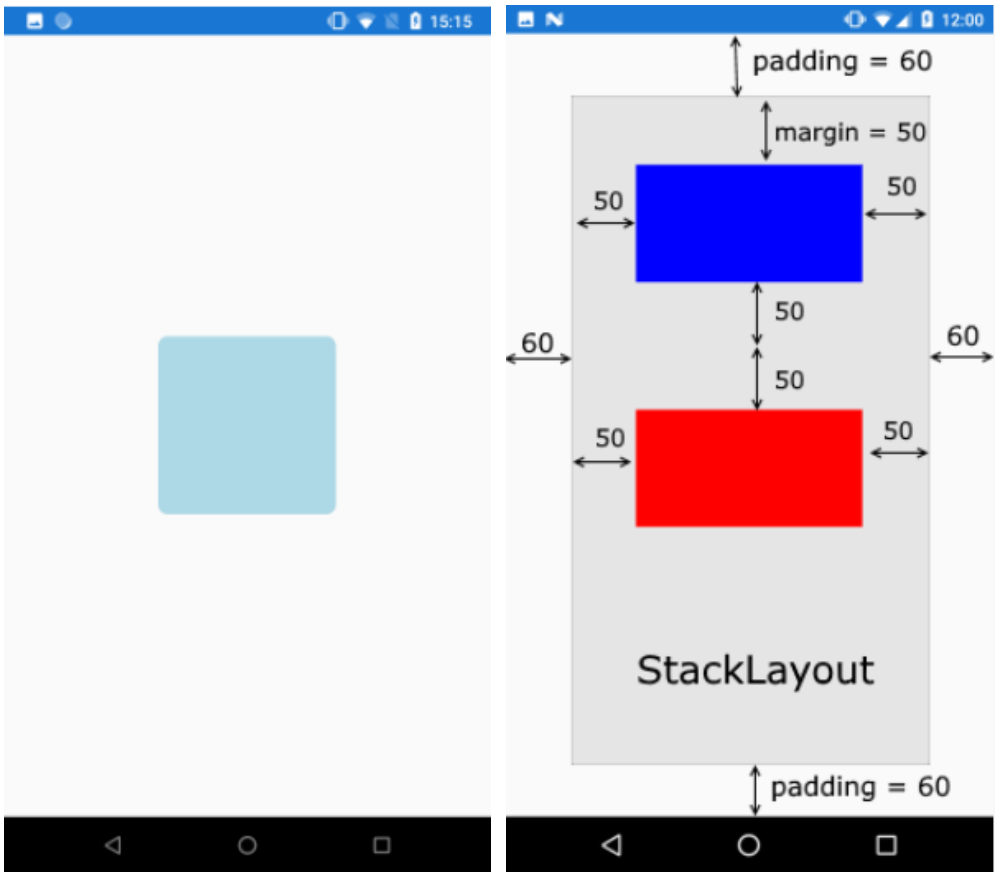
- Color : представляє колір елемента як структури Color.
- CornerRadius : представляє радіус кордону BoxView як значення типу float.
- WidthRequest : представляє ширину елемента (за замовчуванням дорівнює 40 одиниць).
- HeightRequest : представляє висоту елемента (за замовчуванням дорівнює 40 одиниць).

Створення BoxView у XAML:

```
<?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>
        <BoxView Color="LightBlue" WidthRequest="150"
                 HeightRequest="150" CornerRadius="8"
                 HorizontalOptions="Center" VerticalOptions="Center"/>
    </StackLayout>
</ContentPage>
```

```
using Xamarin.Forms;
namespace HelloApp
{
    public partial class MainPage : ContentPage
    {
        public MainPage()
        {
            InitializeComponent();
            BoxView boxView = new BoxView
            {
                Color = Color.LightBlue,
                CornerRadius = 8,
                WidthRequest = 150,
                HeightRequest = 150,
                HorizontalOptions = LayoutOptions.Center,
                VerticalOptions = LayoutOptions.CenterAndExpand
            };
            StackLayout stackLayout = new StackLayout() {
                Children = { boxView }
            };
```

```
            Content = stackLayout;
        }
    }
}
```



Для встановлення відступів у елементів застосовуються властивості **Margin** та **Padding**. Властивість **Margin** визначає зовнішній відступ елемента з інших елементів або контейнера. А властивість **Padding** встановлює внутрішні відступи - від внутрішнього вмісту елемента до його меж. Обидві ці властивості представляють структуру **Thickness** .

Наприклад, поставимо обидва відступи в XAML:

```
<?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 Padding="60">
        <BoxView Color="Blue" Margin="50" HeightRequest="100" />
        <BoxView Color="Red"  Margin="50" HeightRequest="100" />
    </StackLayout>
</ContentPage>
```

Встановлення у коді C#.

```
public class MainPage : ContentPage
{
 public MainPage()
 {
   var stackLayout = new StackLayout
   {
     Padding = new Thickness(60),
     Children = {
       new BoxView { Color = Color.Blue, Margin = new Thickness (50)},
       new BoxView { Color = Color.Red, Margin = new Thickness (50)}
     }
   };
   Content = stackLayout;
 }
}
```

Крім того, структура Thickness має властивості Left, Top, Right, Bottom, тому задати її значення можна трьома способами:

- вказати одне значення для відступів з усіх боків
- вказати два значення для відступів по горизонталі (ліворуч та праворуч) та вертикалі (зверху та знизу)
- вказати чотири значення для відступів для кожної із сторін.

Button

Для створення будь-якої дії нам знадобляться кнопки, представлені класом Button. Для кнопки можна задати обробник натискання для події Clicked. Кнопка має багато різних властивостей, з яких слід виділити такі:

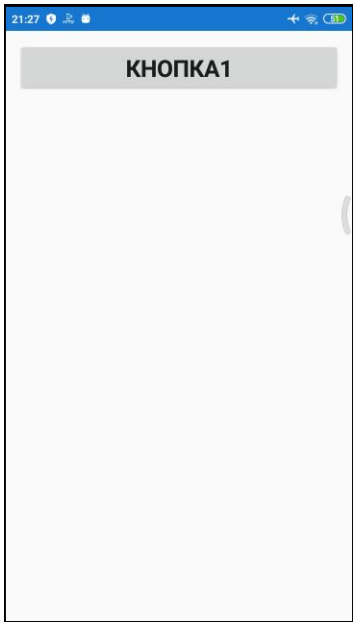
- *FontFamily* : шрифт, який використовується для тексту на кнопці;
- *FontSize* : розмір тексту на кнопці;
- *FontAttributes* : виділення жирним або курсивом тексту на кнопці;
- *TextColor* : колір шрифту;
- *BorderColor* : колір кордону;
- *BorderWidth* : ширина кордону;
- *CornerRadius* : радіус кордону;
- *Image* : дозволяє задати зображення на кнопці.

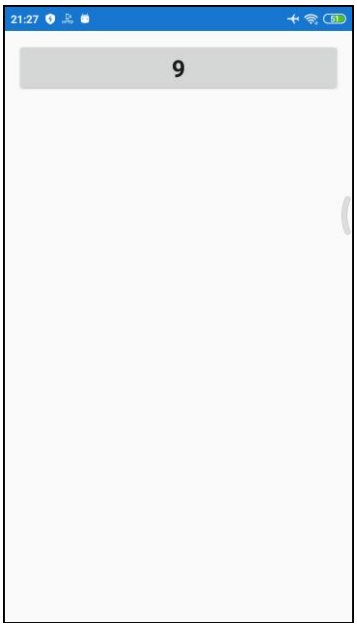
Створимо кнопку в коді:

```
<StackLayout Padding="12, 12, 12, 12">
    <Button x:Name="Button1" Text="Кнопка1"
            FontAttributes="Bold" Clicked="Button1_Click">
    </Button>
</StackLayout>
```

Перейдемо до файлу MainPage.xaml.cs та реалізуємо лічильник натисньь на кнопку

```
public partial class MainPage : ContentPage
{
   int c = 0;
   public MainPage()
   {
      InitializeComponent();
    }
    private void Button1_Click(object sender, EventArgs e)
    {
      c++;
      Button1.Text = Convert.ToString(c);
     }
 }
```

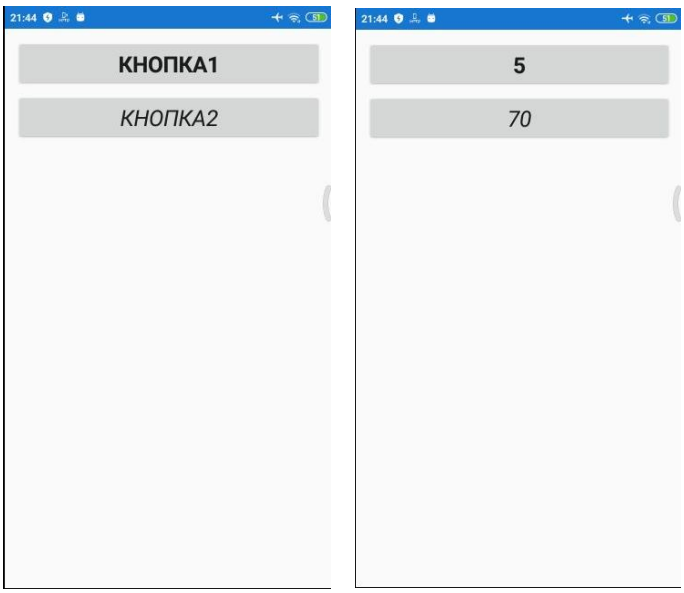




Додаємо кнопку 2, тільки метод обробки буде доданий в коді файлу MainPage.xaml.cs

```
<StackLayout Padding="12, 12, 12, 12">
    <Button x:Name="Button1" Text="Кнопка1"
            FontAttributes="Bold" Clicked="Button1_Click">
    </Button>
    <Button x:Name="Button2" Text="Кнопка2"
            FontAttributes="Italic">
    </Button>
</StackLayout>
```

```
public partial class MainPage : ContentPage
{
     int c = 0;
     int c1 = 0;

     public MainPage()
     {
        InitializeComponent();
        Button2.Clicked += Button2_Click;
     }

     private void Button2_Click(object sender, EventArgs e)
     {
         c1 += 10;
         Button2.Text = Convert.ToString(c1);
      }

     private void Button1_Click(object sender, EventArgs e)
     {
          c++;
          Button1.Text = Convert.ToString(c);
      }
 }
```



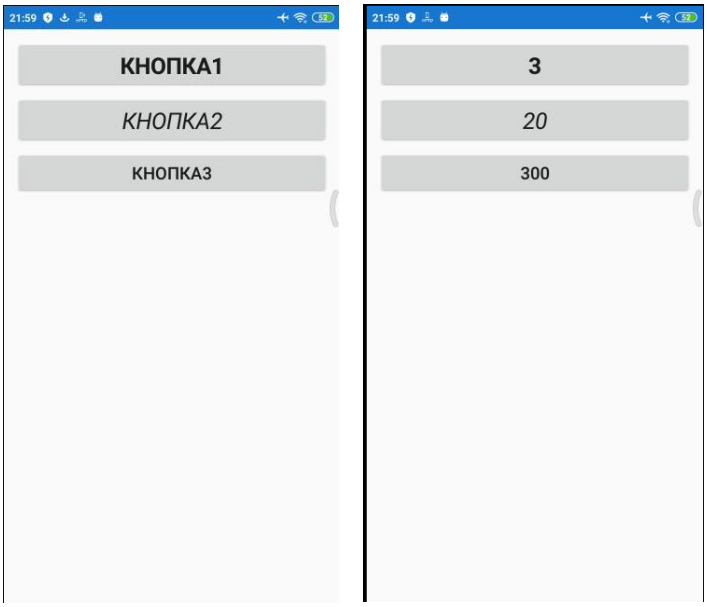
Створюємо обробку кнопки3 через лямбда вираз. Додаємо її в xaml файл

```
<Button x:Name="Button3" Text="Кнопка3"></Button>
```

Обробник події для кнопки 3 у файлі MainPage.xaml.cs

```
public MainPage()
{
    InitializeComponent();

     Button2.Clicked += Button2_Clicked;
     Button3.Clicked += (s, e) =>
     {
        int k = 0;
        if (Int32.TryParse(Button3.Text, out k))
        {
            Button3.Text = Convert.ToString(k + 100);
         }
         else
         {
            Button3.Text = "0";
         }
      };
 }
```

Якщо текст кнопки число, то повертаємо число +100 (out k), інакше присвоюємо значення 0.



DisplayAlert

Створимо проєкт з двома кнопками. По натисненню кнопки 1 появляється інформаційне повідомлення, по натисненню кнопки 2 закривається додаток.

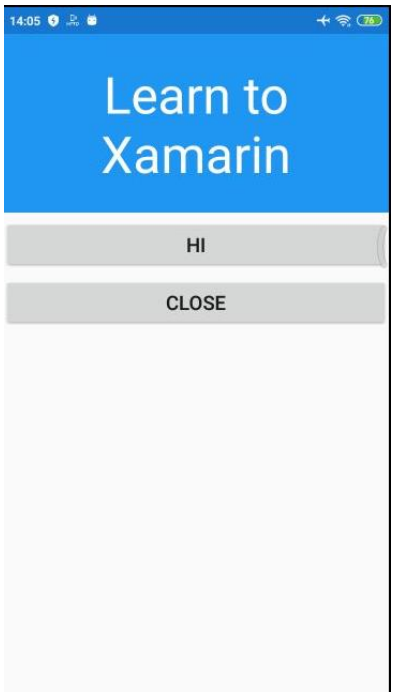
```
<StackLayout>
    <Frame BackgroundColor="#2196F3" Padding="24" CornerRadius="0">
        <Label Text="Welcome to Xamarin.Forms!" FontSize="36"
               HorizontalTextAlignment="Center" TextColor="White"/>
    </Frame>
```

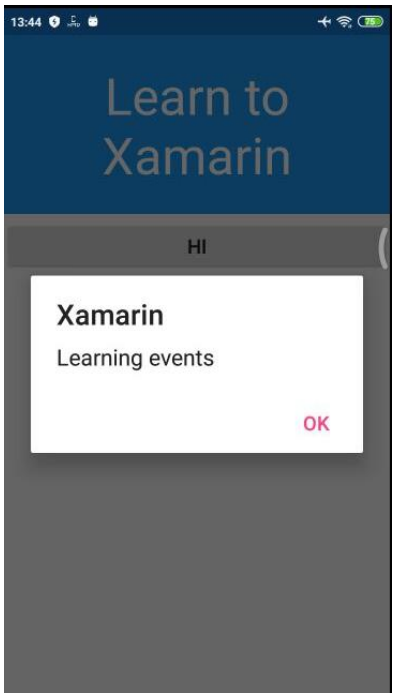
```
        <Button x:Name="button1" Text="Hi"></Button>
        <Button Clicked="Button_Clicked" Text="Close"></Button>
    </StackLayout>
```

Файл MainPage.xaml.cs

```
protected override void OnAppearing()
{
    button1.Clicked += Button1_Clicked;
}

private async void Button1_Clicked(object sender, EventArgs e)
{
   await DisplayAlert("Xamarin", "Learning events", "Ok");
}

private void Button_Clicked(object sender, EventArgs e)
{
    Environment.Exit(0);
}
```

Метод **OnAppearing()** обробляється при запуску та появі сторінки. Він override (перевизначається), так як успадковується від сторінки ContentPage. Метод **Button1_Clicked** робимо його асинхронним (використання асинхронних методів потрібно для того, щоб не гальмувати інтерфейс користувача) так коли використовуємо *async await* код запускається в новому потоці і не відбувається затримок та зависань програми). З допомогою оператора *await* викликаємо метод **DisplayAlert()**, в який потрібно передати 3 параметри: *заголовок повідомлення, текст повідомлення, напис на кнопці*, яка буде закривати повідомлення.





Розглянемо проєкт, в якому автоматично створюється 10 кнопок з викликом інформаційного повідомлення з номером кнопки, що натиснута.

Файл MainPage.xaml

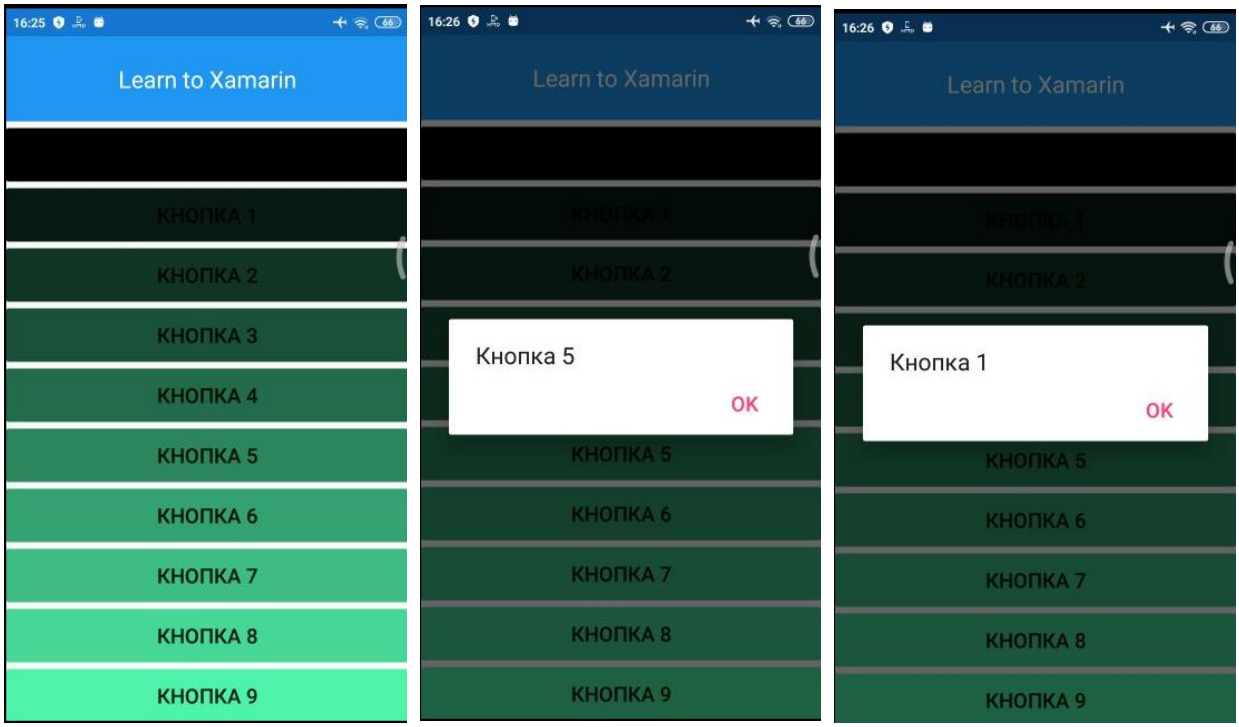
```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="ButtonApp.MainPage">

    <StackLayout x:Name="stackLayout1">
       <Frame BackgroundColor="#2196F3" Padding="16" CornerRadius="0">
            <Label Text="Learn to Xamarin" FontSize="16"
             HorizontalTextAlignment="Center" TextColor="White" />
       </Frame>
    </StackLayout>

</ContentPage>
```

Файл MainPage.xaml.cs

```
namespace ButtonApp
{
    public partial class MainPage : ContentPage
    {
        public MainPage()
        {
            InitializeComponent();
        }

        protected override void OnAppearing()
        {
            for (int i =0; i < 10; i++)
            {
                Button btn = new Button();
                btn.Text = $"Кнопка {i}";
                btn.BackgroundColor = Color.FromRgb(i*9, i*27, i*19);
                btn.TextColor = Color.WhiteSmoke;

                btn.Clicked += Button_Clicked;
                stackLayout1.Children.Add(btn);
            }
        }

        private async void Button_Clicked(object sender, EventArgs e)
        {
          await DisplayAlert("", $"{(sender as Button).Text}", "Ok");
        }
    }
}
```

Створюємо 10 кнопок з різними кольором Color.FromRgb (R, G, B). При натисненні кнопки будемо виводити її ім’я. У методі DisplayAlert приводимо

відправника (sender) до типу Button та на отриманому об’єкті беремо властивість Text.



Stepper та Slider

Stepper – це дві кнопки -, + які дозволяють встановлювати числове значення. Stepper дозволяє встановити ряд властивостей, які налаштовують його поведінку:

- *Minimum* : встановлює мінімальне значення;
- *Maximum* : встановлює максимальне значення;
- *Increment* : встановлює крок зміни значення.

Для відстеження зміни значення можна обробити подію ValueChanged.

Slider – це горизонтальним повзунок. Серед його властивостей можна виділити такі:

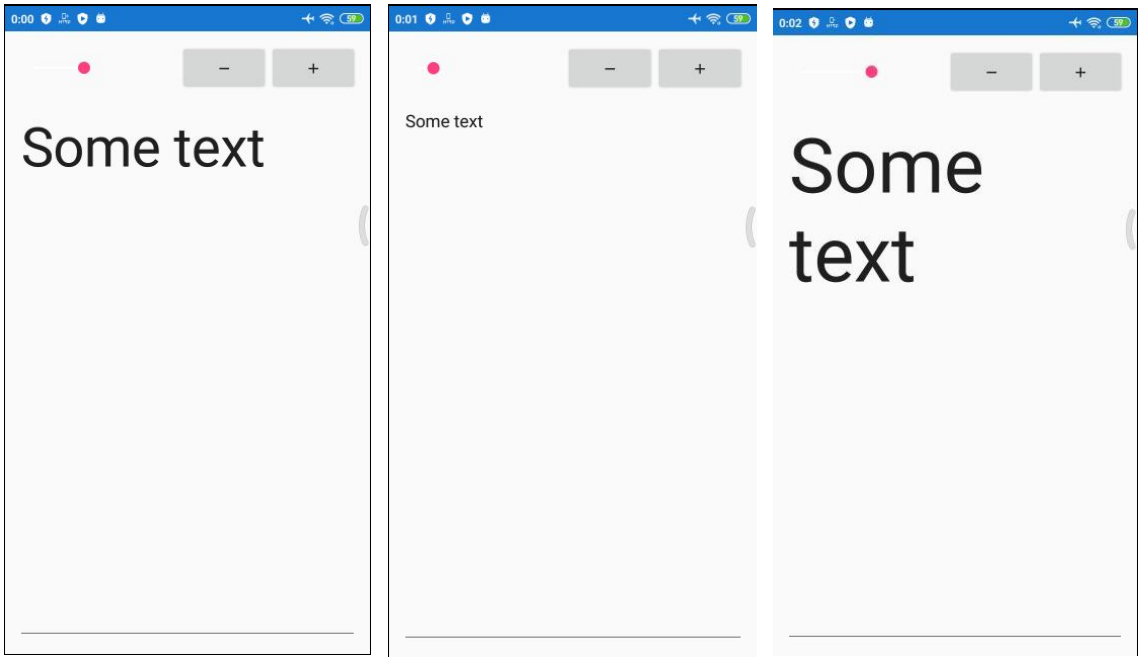
- *Minimum* : встановлює мінімальне значення;
- *Maximum* : встановлює максимальне значення;
- *Value* : поточне значення;
- *ThumbColor* : колір покажчика поточного значення;
- *MinimumTrackColor* : колір повзунка до вказівника значення;
- *MaximumTrackColor* : колір повзунка після вказівника значення.

Створюємо новий додаток. На екрані ліворуч знаходиться елемент Slider, у правого краю Stepper. Зміна положення елементів змінює розмір шрифта в елементі Editor, який займає всю решту частину екрану. Створюємо контейнер StackLayout, до якого вкладаємо ще один StackLayout з горизонтальною

орієнтацією та Editor. Основними властивостями Slider є мінімальне, максимальне значення, поточне значення (Value) та обробник події; до властивостей Stepper ще додається крок зміни. В коді створюємо метод OnAppearing(), у якому задаємо початкове значення розміру шрифта та 2 методи обробника подій

```
<StackLayout Padding="12, 12, 12, 12">
        <StackLayout Orientation="Horizontal"
                     HorizontalOptions="FillAndExpand">
            <Slider x:Name="Slider1" Value="15"
                    Minimum="0" Maximum="100"
                    ValueChanged="Slider1_ValueChanged"
                    HorizontalOptions="FillAndExpand"></Slider>

            <Stepper x:Name="Stepper1" Value="8" Increment="2"
                     Minimum="0" Maximum="100"
                     ValueChanged="Stepper1_ValueChanged"
                     HorizontalOptions="End"> </Stepper>
        </StackLayout>
        <Editor x:Name="Editor1" Text="Some text"
                VerticalOptions="FillAndExpand">  </Editor>
 </StackLayout>


public MainPage()
        {
            InitializeComponent();
        }

        protected override void OnAppearing()
        {
            base.OnAppearing();
            Editor1.FontSize = Slider1.Value;
        }

        private void Slider1_ValueChanged(object sender,
ValueChangedEventArgs e)
        {
            if(e.NewValue> 0)
            {
                Editor1.FontSize = e.NewValue;
                Stepper1.Value = e.NewValue;
            }
        }
        private void Stepper1_ValueChanged(object sender,
ValueChangedEventArgs e)
        {
            if (e.NewValue > 0)
            {
```

```
                Editor1.FontSize = e.NewValue;
                Slider1.Value = e.NewValue;
            }
        }
```



Елемент Image

Для виведення зображення є елемент Image. Перед виведенням зображення його треба додати в проекти, причому в проект для кожної окремої ОС. У файлі MainPage.xaml видаляємо частину коду, створюємо елемент Image

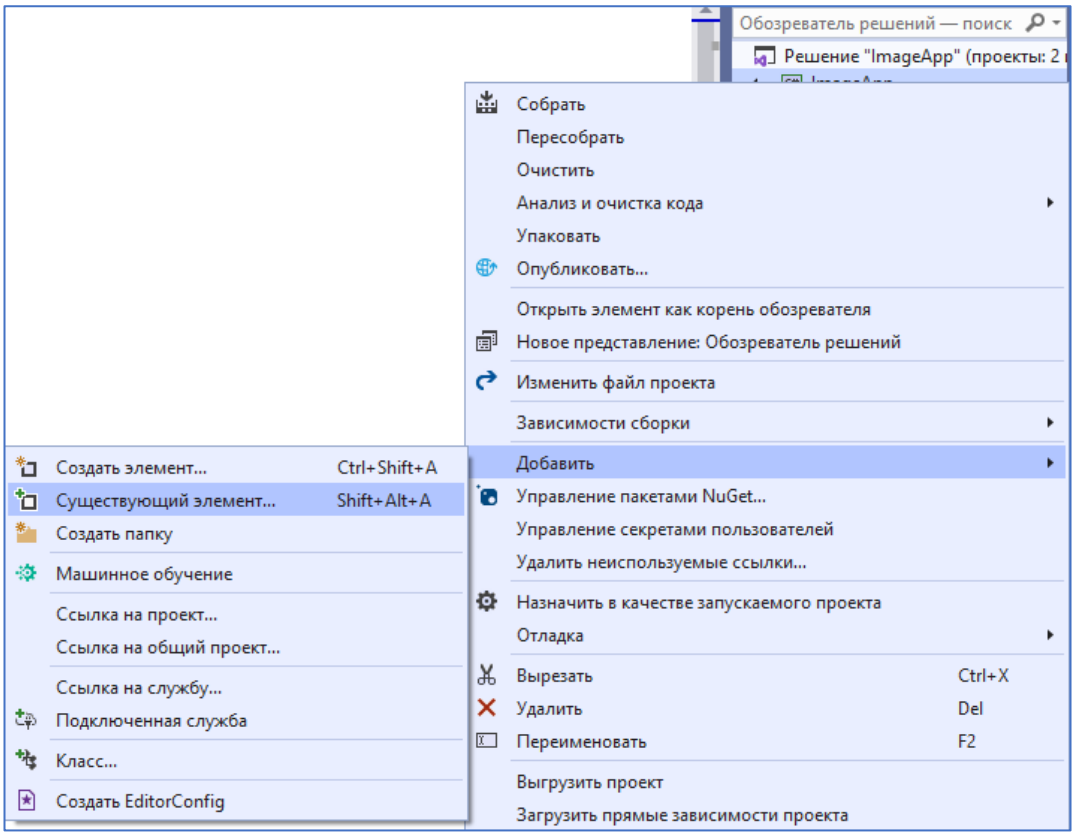
```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="ImageApp.MainPage">

    <Image x:Name="Image1" VerticalOptions="FillAndExpand"
           HorizontalOptions="FillAndExpand"></Image>
</ContentPage>
```

Переходимо до файлу MainPage.xaml.cs

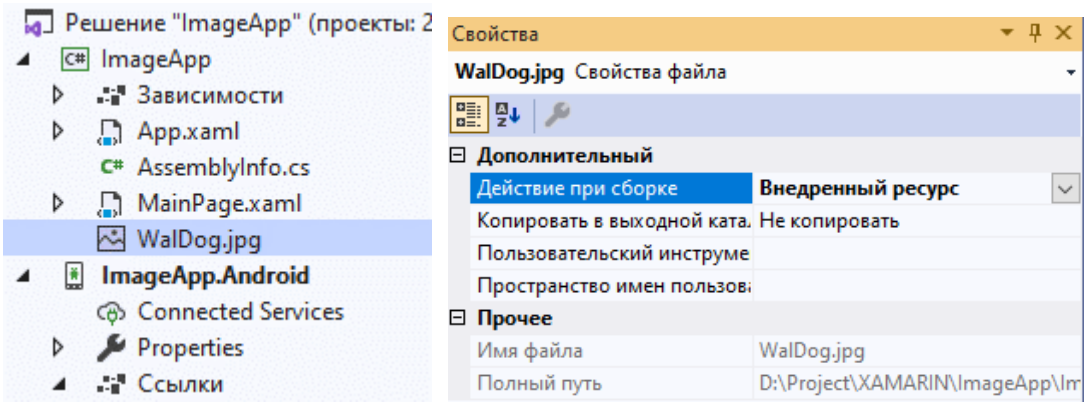
1-й спосіб через вбудований ресурс. Завантажуємо рисунок до проекту (ім'я файлу не містить спецсимволи).

Звернення до рисунка *ім'я_сборки.ім'я_файла.розширення*

```
public MainPage()
{
   InitializeComponent();
   Image1.Source = ImageSource.FromResource("ImageApp.WalDog.jpg");
   Image1.Aspect = Aspect.AspectFit;
}
```



Змінюємо властивість «Действие при сборке» на «Внедренный ресурс»

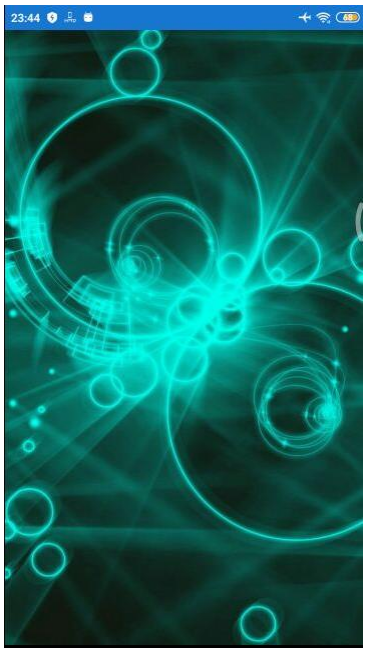


Властивість **Aspect** дозволяє встановити принцип масштабування зображення при його виведенні на екран. Ця властивість може бути встановлена в одне з значень:

– AspectFit: значення за замовчуванням. Якщо стандартне зображення не вписується в екран (наприклад, його ширина більше ширини екрана), воно масштабується зі збереженням аспектного відношення (відношення ширини до довжини)
– Fill: розтягує зображення по ширині або довжині без збереження аспектного відношення

– AspectFill: зберігає аспектне відношення, але вирізає з нього частину, яка вписується в екран

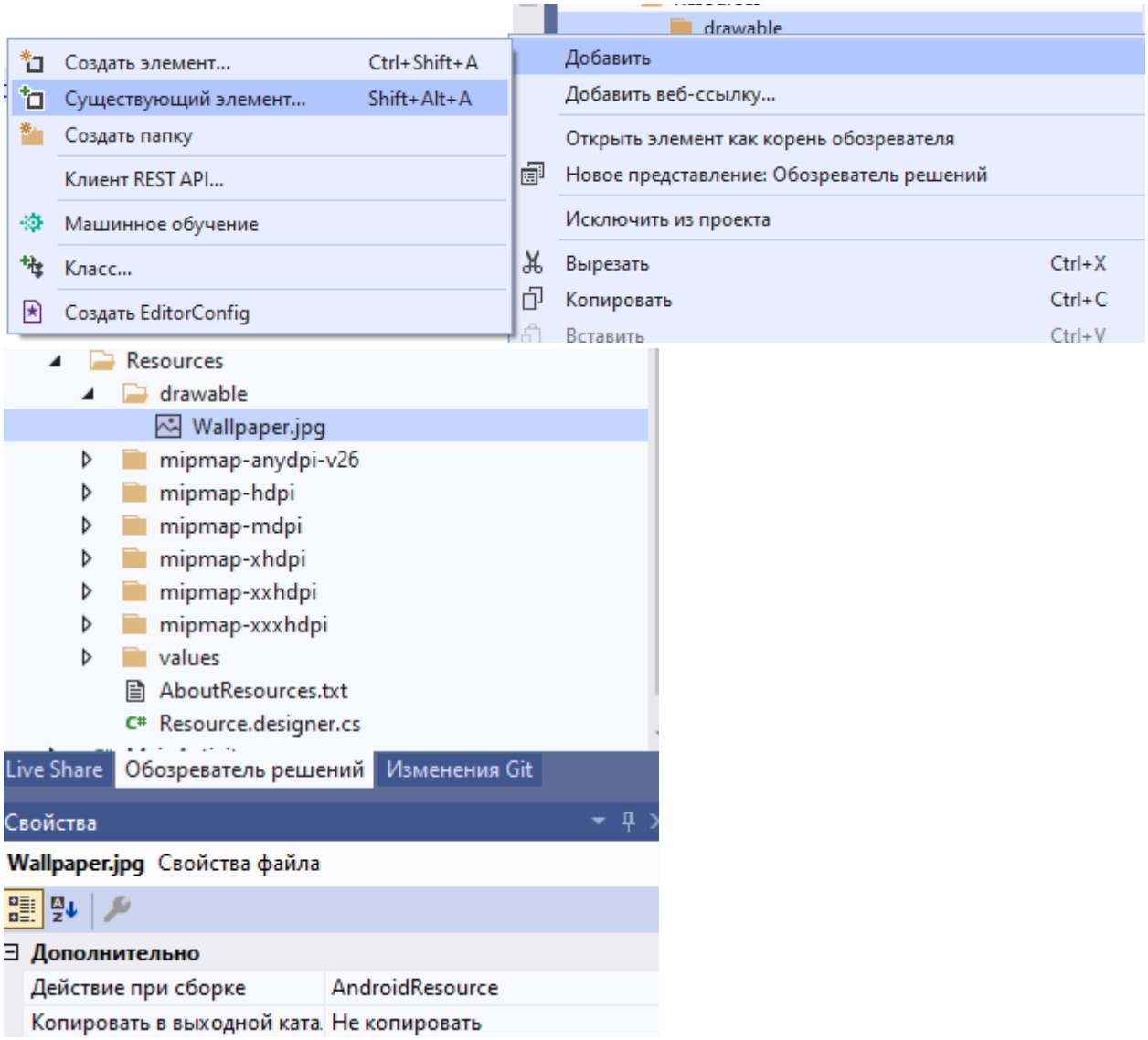

AspectFit


Fill

2-й спосіб. Завантажуємо рисунок з URI

```
Image1.Source = ImageSource.FromUri(new
Uri("https://wallpaper.dog/large/20370946.jpg"));
```

3-й спосіб. Додаємо рисунок в папку ресурсів Drawable




```
Image1.Source = ImageSource.FromFile("Wallpaper.jpg");
```

WebView

WebView є міні-веббраузером, що дозволяє переглядати різні типи контенту:

- в першу чергу WebView дозволяє завантажувати із середовища інтернет веб-сайти та взаємодіяти з ними. WebView має повну підтримку HTML, CSS та JavaScript;
- різні типи документів, наприклад документи pdf. Однак, кожна платформа може підтримувати свій набір типів документів. Наприклад, на iOS та Android ми зможемо переглянути файл pdf, а ось на Windows Phone/Windows Mobile ні;
- рядки, які містять код html, під час перегляду будуть належним чином інтерпретовані, як у звичайному браузері;

– локальні файли - ми можемо додати в проект локальні файли html та запускати їх у WebView.

Створюємо новий проєкт. Шукаємо посилання на відео. Копіюємо код вставки і в ньому вибираємо link з словом embed

<iframe width="1280" height="720" src="**https://www.youtube.com/embed/to3_JLiPaNo**" title="Whiskey Blues - Music medicine for the soul and relieve stress - Best Slow Blues Music" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>

На сторінці MainPage.xaml додаємо елемент WebView та налаштвоуємо його властивості.

```
<StackLayout>
        <WebView
            Source="https://www.youtube.com/embed/to3_JLiPaNo"
            WidthRequest="360"
            HeightRequest="240"
            MinimumWidthRequest="360"
            MinimumHeightRequest="240">
        </WebView>
    </StackLayout>
```

Таж сама реалізація в коді C# на сторінці MainPage.xaml.cs

```
protected override void OnAppearing()
{
    WebView webView1 = new WebView()
    {
        Source = "https://www.youtube.com/embed/to3_JLiPaNo",
        WidthRequest = 360,
        HeightRequest = 240,
```

```
            MinimumWidthRequest = 360,
            MinimumHeightRequest = 240
        };
        StackLayout stackLayout1 = new StackLayout();
        stackLayout1.Children.Add(webView1);
        Content = stackLayout1;
 }
```

Додаємо кнопку та створюємо метод обробки кнопки

```
protected override void OnAppearing()
{
    WebView webView1 = new WebView()
      {
         Source = "https://www.youtube.com/embed/to3_JLiPaNo",
         WidthRequest = 360,
         HeightRequest = 240,
         MinimumWidthRequest = 360,
         MinimumHeightRequest = 240
        };

        Button button = new Button();
        button.Text = "Open URL";
        button.Clicked += OnButtonClick;

        StackLayout stackLayout1 = new StackLayout();
        stackLayout1.Children.Add(webView1);
        stackLayout1.Children.Add(button);

        Content = stackLayout1;
}

private async void OnButtonClick(object sender, EventArgs e)
{
    Uri uri = new Uri("https://www.dou.ua");
    await Browser.OpenAsync(uri);
}
```