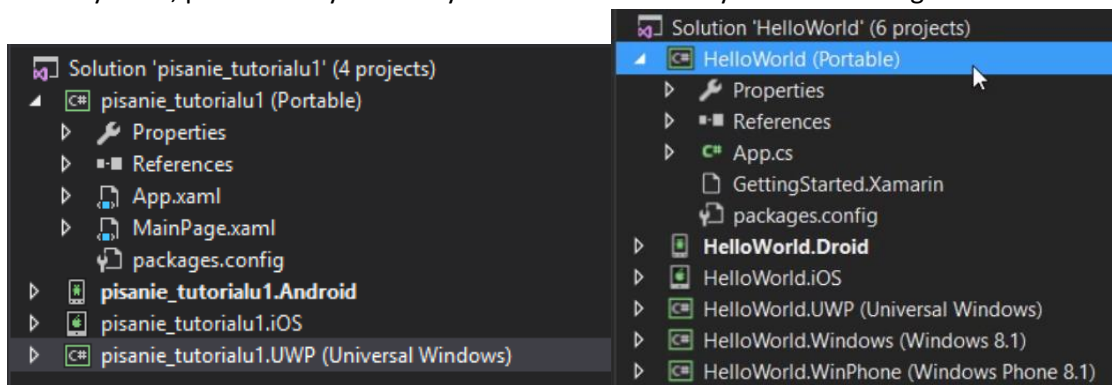
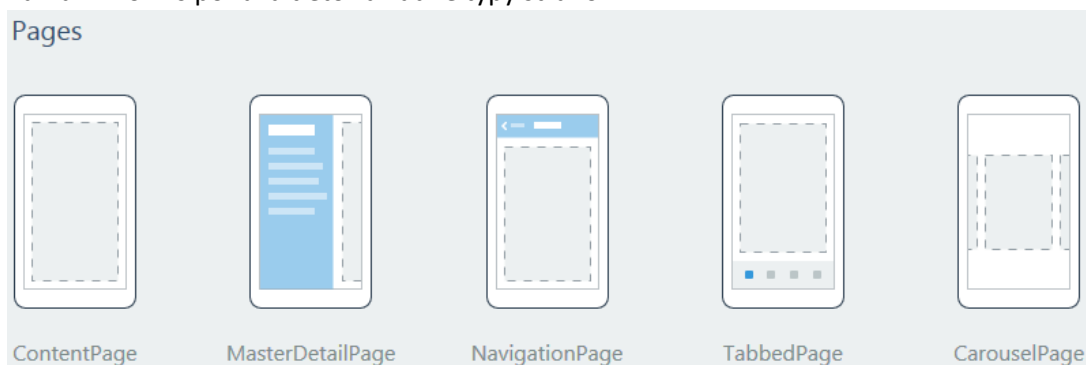


1. Vytvoríme nový projekt vo Visual Studio: V menu **File – New - Project**. Vyberieme umiestnenie a názov projektu podľa vlastného výberu.
2. Vyberieme **Cross-Platform – Cross-Platform App (Xamarin.Forms)**. V inej verzii to môže byť **Blank App (Xamarin.Forms Portable)**.
3. Na ďalšej obrazovke (ak sa objaví) vyberieme **Blank App** a **UI Technology – Xamarin.Forms** a **Code Sharing Strategy – Portable Class Library (PCL)**
4. Zatláčime OK a vytvorí sa projekt. Zavrieme okno, kde nám ponúka Mac Agentu a odsúhlasíme target verzie UWP podprojektu.
5. Mali by sa nám vytvoriť minimálne 4 podprojekty (Portable, Android, iOS a UWP). Prípadne tam môžeme mať ešte Windows 8.1 a Windows Phone 8.1. Existujú 2 varianty, ako sa nám to môže vytvoriť, po novom by to mal byť ten naľavo už s nachystanou MainPage.



6. Klikneme pravým tlačidlom na **Solution**, ktorý je umiestnený nad podprojektami a vyberieme **Manage NuGet Packages for Solution**.
7. Vyberieme položku Update a updatneme knižnice, aby sme boli aktuálni. **POZOR – niekedy toto môže byť skôr kontraproduktívne...**
8. Ideme urobiť jednoduchú aplikáciu, ktorá nám po kliknutí na tlačidlo zobrazí Hello World.
9. Ak sa nám projekt vytvoril bez MainPage.xaml (variant vpravo v bode 5), tak klikneme pravým tlačidlom na Portable podprojekt a vyberieme **Add – New item**. Tu vyberieme **Forms Blank Content Page Xaml**. Ak nie je napísané vyslovene **blank**, tak vyberieme **Forms Xaml Page**. Pomenujeme ju napríklad **GreetPage**.
10. Ak sa nám projekt vytvoril s **MainPage.xaml**, tak pracujeme rovno s ním. Ďalej už budeme používať v návode tento názov, hoci ho môžete mať nazvaný aj **GreetPage.xaml** z bodu 9.
11. Spolu s **MainPage.xaml** máme aj **MainPage.xaml.cs**. Prvý súbor určuje, ako bude obrazovka vyzeráť a druhý bude obsahovať zdrojový kód v jazyku C#.
12. Všimnime si, že v XAML súbore máme **<ContentPage**. Znamená to, že pracujeme s obrazovkou (stránkou-page) tohto typu. Page je niečo ako Activity v Androide. Xamarin.Forms ponúka tieto základné typy stránok:



13. Pod tag `ContentPage` (teda medzi `<ContentPage...` a `</ContentPage>`) pridáme kód:

```
<Button HorizontalOptions="Center" VerticalOptions="Center"
Text="Click Me" Clicked="Button_Clicked"></Button>
```

14. Automaticky by sa nám v kóde mala vytvoriť metóda:

```
private void Button_Clicked(object sender, EventArgs e)
{
}
}
```

15. Napíšeme do nej kód: `DisplayAlert("Title", "Hello World", "OK");`

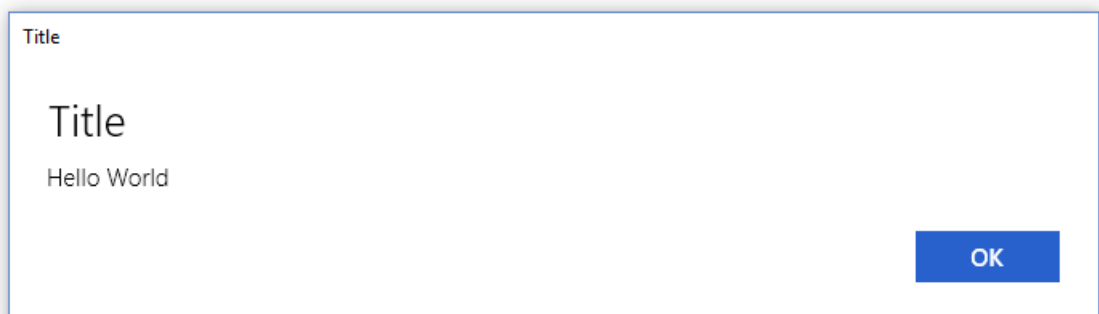
16. V prípade, že v bode 5 sme mali pravý variant (bez Xaml a s `App.cs`), tak treba v `App.cs` nastaviť `MainPage` na nami vytvorenú page. Napríklad teda takto (ak sa volá naša stránka `GreetPage`):

```
namespace HelloWorld
{
    public partial class App : Application
    {
        public App()
        {
            InitializeComponent();

            MainPage = new GreetPage();
        }

        protected override void OnStart()
        {
            // Handle when your app starts
        }
    }
}
```

17. Projekt odskúšame – napríklad ako UWP aplikáciu.



18. Teraz z `MainPage.xaml` vymažeme náš button. Tiež jeho metódu v kóde.

19. V `MainPage.xaml` pod tag `ContentPage` (teda medzi `<ContentPage...` a `</ContentPage>`) pridáme kód: `<Label HorizontalOptions="Center" VerticalOptions="Center" Text="Hello World" />`

20. Keď odskúšame aplikáciu, tak vidíme, že nám vypísalo `Hello World` do stredu obrazovky. Vieme toto isté UI dosiahnuť v kóde a nie v XAML?

21. Po kóde `InitializeComponent();` pridáme kód (vlastne ide o property):

```
Content = new Label
{
    HorizontalOptions = LayoutOptions.Center,
    VerticalOptions = LayoutOptions.Center,
    Text = "Hello World"
};
```

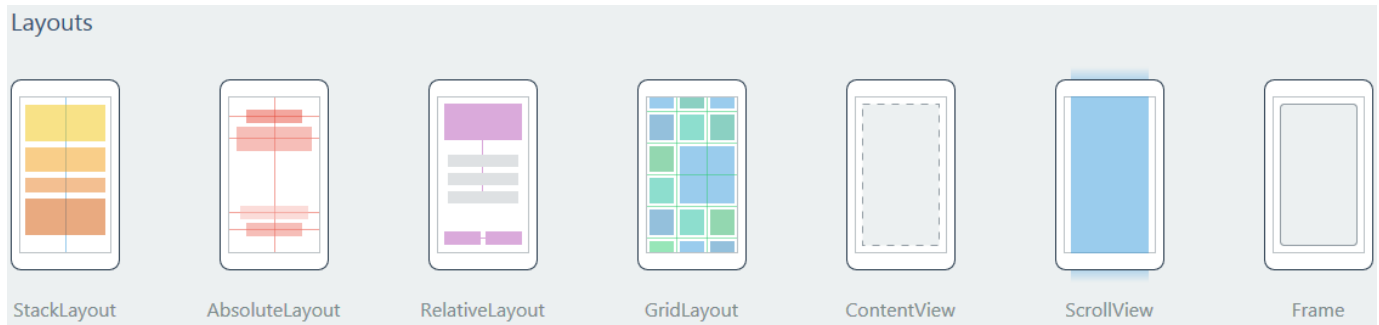
22. Kódom vyššie sme dosiahli presne to isté ako predtým v XAML. Kód sa používa, keď chceme UI meniť dynamicky.

23. Teraz kód z bodu 21 zmažeme

24. Do **MainPage.xaml** namiesto **Label** pridáme:

```
<StackLayout HorizontalOptions="Center" VerticalOptions="Center">  
    <Label Text="Hello World" />  
    <Slider />  
</StackLayout>
```

25. Museli sme pridať StackLayout, nakoľko chceme na stránke usporiadať viacero prvkov a nie iba jeden. Typy rozložení (layouts) sú:



26. Teraz chceme, aby mi hodnotu zo slajdera zobrazovalo.

27. Slajder teda v XAML doplníme: `<Slider ValueChanged="Slider_ValueChanged" />`

28. Ak sme správne klikli, tak do kódu nám doplnilo:

```
private void Slider_ValueChanged(object sender, ValueChangedEventArgs e)  
{  
}
```

29. Vrátime sa do XAML a namiest Label napíšeme: `<Label Text="Hello World" x:Name="label"/>`

30. Vrátime sa do kódu a do metódy Slider\_ValueChanged napíšeme:

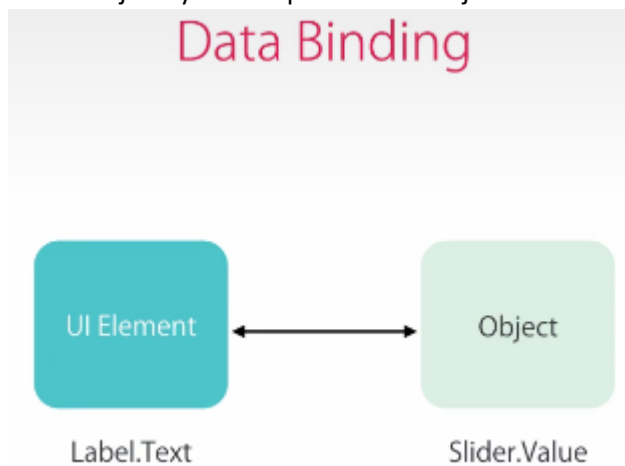
```
label.Text = String.Format("Value is {0:F2}", e.NewValue);
```

31. Spustíme aplikáciu. Vidíme, že pri spustení nám stále píše Hello World, ale my tam hneď chceme hodnotu zo slajdera.

32. Vrátime sa do XAML a namiest Slider napíšeme: `<Slider ValueChanged="Slider_ValueChanged" x:Name="slider" />`

33. Do kódu sa InitializeComponent napíšeme: `slider.Value = 0.5;`

34. Spustíme aplikáciu a vidíme, že to funguje. Teraz si ukážeme **Data Binding**. Týmto konceptom efektívnejšie vyriešime previazanie slajdera a UI elementu, ako sme to urobili teraz.

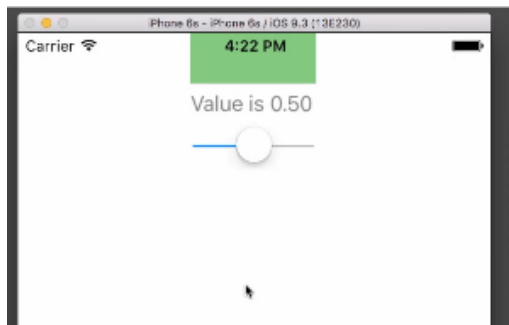


35. Zmažeme z kódu metódu Slider\_ValueChanged.

36. Zo XAML zmažeme `ValueChanged="Slider_ValueChanged"`.
37. Label zmažeme a namiesto neho dáme: `<Label Text="{Binding Source={x:Reference slider}, Path=Value}" x:Name="label"/>`
38. Týmto sme text Labelu napojili na hodnotu objektu **slider** a odovzdávame mu jeho hodnotu **Value**.
39. Vyskúšame a vidíme, že funguje.
40. Pre krajšie formátovanie použijeme: `<Label Text="{Binding Source={x:Reference slider}, Path=Value, StringFormat='Value is {0:F2}}'" x:Name="label"/>`
41. Čo keby sme chceli s hodnotou slidera zviazať aj s priehľadnosť prvku.
42. Napíšeme namiesto Label toto:
- ```
<Label Text="{Binding
    Source={x:Reference slider},
    Path=Value,
    StringFormat='Value is {0:F2}}'"

    Opacity="{Binding
        Source={x:Reference slider},
        Path=Value}"
    x:Name="label"/>
```
43. Ak chceme kód sprehľadniť, tak použijeme BindingContext, aby sme nemuseli písať 2 razy `x:Reference slider`:  
Kód v XAML teda bude:
- ```
<Label BindingContext="{x:Reference slider}"
    Text="{Binding Value,
        StringFormat='Value is {0:F2}}'"

    Opacity="{Binding Value}"
    x:Name="label"/>
```
44. Naučíme sa ďalší prvok. Pred `<Label` pridáme: `<BoxView Color="Green" Opacity="{Binding Source={x:Reference slider}, Path=Value}" />`
45. Pre krajší kód môžeme urobiť dedenie BindingContextu zo StackLayout:
46. `<StackLayout BindingContext="{x:Reference slider}" HorizontalOptions="Center" VerticalOptions="Center">`  
`<BoxView Color="Green" Opacity="{Binding Value}" />`  
`<Label`  
`Text="{Binding Value,`  
`StringFormat='Value is {0:F2}}'"`  
`Opacity="{Binding Value}"`  
`/>`  
`<Slider x:Name="slider" />`  
`</StackLayout>`
47. Ideme nastavovať rôzny vzhľad pre rôzne platformy. Ak by sme zmazali `VerticalOptions="Center"`, tak na Androide to je v poriadku, ale na iOS by sa obdĺžnik kryl s časom:



48. Toto je nežiaduce a chceme teda nastaviť padding (okraj) špeciálne pre iOS zariadenia.

49. Dá sa to urobiť v kóde a aj v XAML. Najprv to spravíme v kóde za `slider.Value=0.5;` (hodnoty pre WinPhone a Android sú ilustračné):

```
if (Device.OS == TargetPlatform.iOS)
    Padding = new Thickness(0, 20, 0, 0);
else if (Device.OS == TargetPlatform.Android)
    Padding = new Thickness(10, 20, 0, 0);
else if (Device.OS == TargetPlatform.WinPhone)
    Padding = new Thickness(10, 20, 0, 0);
```

50. Takýto kód je škaredý, urobíme to krajšie, a to pomocou generických typov:

```
Padding = Device.OnPlatform(
    iOS: new Thickness(0, 20, 0, 0),
    Android: new Thickness(10, 20, 0, 0),
    WinPhone: new Thickness(30, 20, 0, 0)
);
```

51. Ďalšia možnosť je:

```
Device.OnPlatform(
    iOS: () => {
        Padding = new Thickness(0, 20, 0, 0);
    },
    Android: () => {
        Padding = new Thickness(10, 20, 0, 0);
    },
    WinPhone: () => {
        Padding = new Thickness(30, 20, 0, 0);
    }
);
```

52. Teraz to isté urobíme v XAML. Zmažeme teda kódy z bodov 49,50 a 51.

53. Najprv do kódu ilustračne za `slider.Value=0.5;` doplníme:

```
var x = new OnPlatform<Thickness>
{
    Android = new Thickness(0),
    iOS = new Thickness(0,20,0,0)
};
Padding = x;
```

54. Kód môžeme zmazať. Podobnú logiku chceme dosiahnuť v XAML, a teda pred `<StackLayout` pridáme:

```
<ContentPage.Padding>
    <OnPlatform
        iOS="0,20,0,0"
        Android="0,40,0,0">
    </OnPlatform>
</ContentPage.Padding>
```

---

55. Ak sa chcete zdokonaľiť v tvorbe rôznych layoutov, tak si pozrite tieto dobre spracované videá:

Stack Layout v XAML:

<https://drive.google.com/file/d/0ByqSuaFYkqNwZnFTVTNveDRZTWM/view?usp=sharing>

Stack Layout v kóde:

<https://drive.google.com/file/d/0ByqSuaFYkqNwc1ZMeEVZSFNIUzQ/view?usp=sharing>

Grid Layout v XAML:

<https://drive.google.com/file/d/0ByqSuaFYkqNwMkUwZjRyOFBHOWc/view?usp=sharing>

Grid Layout v kóde:

<https://drive.google.com/file/d/0ByqSuaFYkqNwT3dpVXZ0T2x3RIU/view?usp=sharing>

Absolute Layout v XAML:

<https://drive.google.com/file/d/0ByqSuaFYkqNwQWFSOGdLQnJmbU0/view?usp=sharing>

Absolute Layout v kóde:

<https://drive.google.com/file/d/0ByqSuaFYkqNwTFBMb21FZVhzR0E/view?usp=sharing>

Relative Layout v XAML:

<https://drive.google.com/file/d/0ByqSuaFYkqNwb2FURWNqU2RONGc/view?usp=sharing>

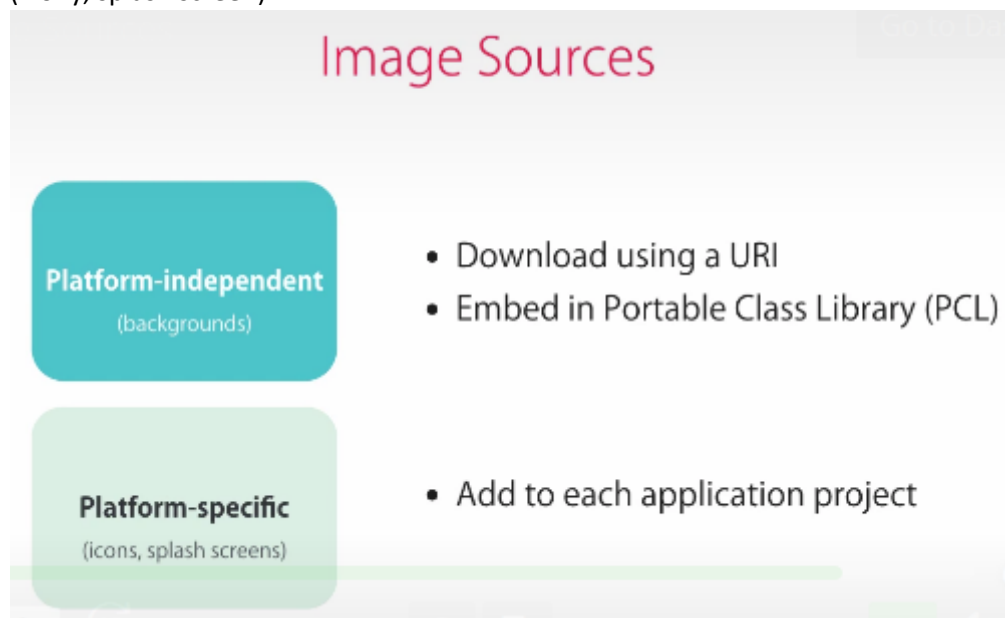
Relative Layout v kóde:

<https://drive.google.com/file/d/0ByqSuaFYkqNwb1EyQzJhdEhRQXc/view?usp=sharing>

---

56. Teraz sa budeme venovať **práci s obrázkami**.

57. V Xamarin.Forms máme 2 typy obrázkov: platformovo-nezávislé a platformovo-špecifické (ikony, splash screen)



58. Vytvoríme si nový projekt podľa bodov 1 až 10.

59. Do XAML pred `</ContentPage>` pridáme: `<Image`

`Source="http://lorempixel.com/1920/1080/sports/7/" x:Name="image"/>`

60. Do kódu po `InitializeComponent` pridáme:

```
var imageSource = new UriImageSource { Uri = new  
Uri("http://lorempixel.com/1920/1080/sports/7/") };
```

```
imageSource.CachingEnabled = false; // defaultne je nastavene na 24 hod  
imageSource.CacheValidity = TimeSpan.FromHours(1);
```

```
image.Source = imageSource;
```

61. Zo XAML môžeme teda vymazať

```
Source="http://lorempixel.com/1920/1080/sports/7/"
```

62. Po spustení Android aplikácie vidíme, že obrázok je takto. Čo ak ho chceme napasovať na plochu aplikácie inak?



63. Pridajme do kódu: `image.Aspect = Aspect.Fill`; (vieme použiť aj XAML)

64. Výsledok vyzerá takto:



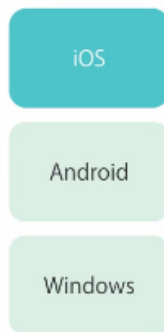
65. Zameňme kód za : `image.Aspect = Aspect.AspectFill`; (vieme použiť aj XAML)

66. Výsledok vyzerá takto:



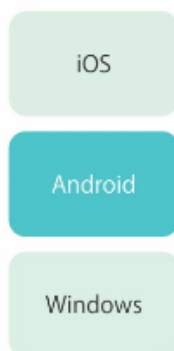
67. Chceme teraz pridať indikátor načítavania obrázka. Do XAML napíšeme (medzi tagy `ContentPage`):
- ```
<ActivityIndicator IsRunning="true"
  AbsoluteLayout.LayoutBounds="0.5,0.5,100,100"
  AbsoluteLayout.LayoutFlags="PositionProportional" />
<Image IsVisible="false" x:Name="image" Aspect="AspectFill"/>
```
68. Po spustení aplikácie vidíme, že sa indikátor točí. Teraz sa však točí stále. My chceme, aby sa zobrazoval len vtedy, keď sa obrázok načítava.
69. Do `IsRunning` napíšeme: `IsRunning="{Binding Source={x:Reference image}, Path=IsLoading}"`
70. `IsVisible="false"` zmažeme. Po spustení však vidíme, že obrázok nevypĺňa celú obrazovku. Je to tým, že ho máme v `Absolute Layoute` a nemáme nastavenú jeho veľkosť.
71. Preto tag obrázok zmeníme na:
- ```
<Image x:Name="image" Aspect="AspectFill"
  AbsoluteLayout.LayoutBounds="0,0,1,1"
  AbsoluteLayout.LayoutFlags="All"/>
```
72. Teraz si ukážeme, ako vložiť obrázky do projektu. V našom **Portable** podprojekte vytvoríme adresár **Images**. Vložíme do neho nejaký obrázok. V našom prípade sa môže volať napríklad **silence.jpg**. Klikneme naňho pravým tlačidlom a vyberieme **Properties**. V otvorenom podmenu nastavíme **Build Action – Embedded Resource**.
73. Z kódu zmažeme všetko v našej triede okrem `InitializeComponent()`. V XAML zmažeme `ActivityIndicator` a `Absolute Layout`. Tiež z tagu obrázka dáme preč `Absolute Layout`.
74. Do kódu napíšeme: `image.Source =`
- ```
ImageSource.FromResource("pisanie_tutorialu1.Images.silence.jpg");
```
75. Namiesto `pisanie_tutorialu1` napíšeme reálny názov nášho projektu (solution).
76. Vidíme, že obrázok sa bez problémov načíta. Poďme teraz na platformovo-špecifické obrázky (ikony a podobne). Budeme potrebovať súbor:
- [http://www.uamt.fei.stuba.sk/MVI/sites/default/files/Resources\\_cv4.zip](http://www.uamt.fei.stuba.sk/MVI/sites/default/files/Resources_cv4.zip)
77. Ikony v súbore pochádzajú zo stránky [icons8.com](http://icons8.com), kde viete nájsť veľa ikon.

78. Pre iOS platia nasledovné zvyklosti pri pomenúvaní obrázkov (za zavináčom sa nachádza označenie, že ide o ikonu pre displeje s väčším DPI):



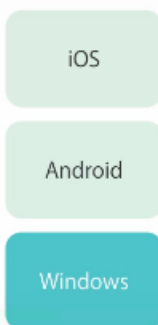
- clock.png (32x32)
- clock@2x.png (64x64)
- clock@3x.png (96x96)

79. Pre Android platia nasledovné zvyklosti pri pomenúvaní obrázkov (máme viacero adresárov, ktoré sú pomenované podľa toho, pre aké DPI displeja tieto adresáre obsahujú obrázky):



- drawable/ (32x32)
- drawable-hdpi/ (48x48)
- drawable-xhdpi/ (64x64)
- drawable-xxhdpi/ (96x96)

80. Pre Windows máme tentokrát len jeden obrázok:



- clock.png (32x32)

81. Rozbalíme si teda stiahnutý súbor s obrázkami na disk (nie do projektu).

82. V iOS podprojekte nájdeme priečinok **Resources**, kam **skopírujeme obrázky z rozbaleného priečinka** (z toho, ktorý sme stiahli) **iOS**.

83. V Android podprojekte nájdeme priečinok **Resources**, kam **skopírujeme obrázky z rozbaleného priečinka** (z toho, ktorý sme stiahli) **Android**.

84. V UWP podprojekte nájdeme priečinok **Assets (prípadne ho vytvoríme)**, kam **skopírujeme obrázok z rozbaleného priečinka** (z toho, ktorý sme stiahli) **Windows**.

85. V XAML si necháme iba ContentPage otvárací (a jeho obsah) a zatvárací tag. Ostatné zmažeme. Do XAML potom napíšeme:

```
<Button Image="clock.png" />
```

86. V Androide aj iOS nám ikonu zobrazia. Avšak v UWP aplikácií nie, nakoľko sme obrázok dali do priečinka Images. UWP ale žiada, aby obrázky boli defaultne umiestnené v koreňovom

priečinku projektu, čo je ale nepraktické. Preto sme ho dali do priečinka Images. Ako toto vyriešime pri zobrazovaní ikony?

87. Do XAML napíšeme: `<Button Image="clock.png" x:Name="btn" />`

88. V kóde necháme iba `InitializeComponent` a za neho napíšeme:

```
btn.Image = (FileImageSource) ImageSource.FromFile(Device.OnPlatform(  
    iOS: "clock.png",  
    Android: "clock.png",  
    WinPhone: "Assets/clock.png"  
));
```

89. V XAML by to vyzeralo takto:

```
<Button x:Name="btn" >  
    <Button.Image>  
        <OnPlatform x:TypeArguments="FileImageSource"  
            iOS="clock.png"  
            Android="clock.png"  
            WinPhone="Assets/clock.png"  
        />  
    </Button.Image>  
</Button>
```

91. Ak chceme vytvárať okrúhle obrázky, pozrite si návod k pluginu:

<https://drive.google.com/file/d/0ByqSuaFYkqNwSDhsVHBMM2xwME0/view?usp=sharing>



## CVIČENIE Č.4 – BONUSOVÁ ÚLOHA

### Úloha (2,5 bonusové boda)

Úloha slúži pre zopakovanie princípov tvorby Xamarin.Forms aplikácie, ktoré sme preberali na cvičení.

Vytvorte aplikáciu (inšpirácia vpravo), kde si užívateľ bude môcť prezerať zoznam výrokov slávnych osobností zvolenou veľkosťou písma na základe slajderu. Medzi výroky môžeme listovať pomocou tlačidla **Next**. Prípadne vymyslite inú zaujímavú techniku pre prepínanie výrokov.

#### Pomôcky:

- Vytvorte si novú stránku nazvanú napríklad **QuotesPage**. Nastavte si ju ako hlavnú stránku v súbore **App.cs**.  
`MainPage = new QuotesPage();`
- Výroky definujte pomocou poľa `string[]`.
- Použite pre túto stránku 20 jednotiek paddingu (20 z každej strany). Toto sa dá nastaviť priamo v prvku `ContentPage: Padding="20"`
- „Overridnite“ padding pre Android a Windows. Pre Android použite **20, 30, 20, 20** a pre Windows **20, 40, 20, 20**.
- Nastavte **maximum** a **minimum** slajdera: `<Slider Maximum="50" Minimum="16"/>`

