

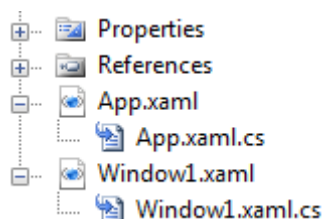
1. WPF - Hello 'XAML' World

Objektově orientované programování je pro psaní uživatelského rozhraní velice těžkopádné. Proto součástí WPF je nový programovací jazyk uživatelského rozhraní - XAML

- eXtensible Application Markup Language
- čti "zamml"
- jedná se o značkovací jazyk
- slouží pouze pro psaní uživatelského rozhraní aplikace

Značkovací jazyky (markup languages) jsou pro definování vzhledu ideálním řešením - rychle, stručně, přehledně a jednoduše můžete nadefinovat vlastnosti, umístění a spojitosti jednotlivých elementů. Mezi značkovací jazyky patří například HTML, který má obdobnou funkci v oblasti vývoje webových stránek.

Další výhodou XAMLu je oddělení programové části od uživatelského rozhraní. Ve WPF aplikaci se každé okno skládá ze 2 souborů, jednoho ".xaml" a připojeném ".cs" souboru (záleží na použitém .NET jazyku).



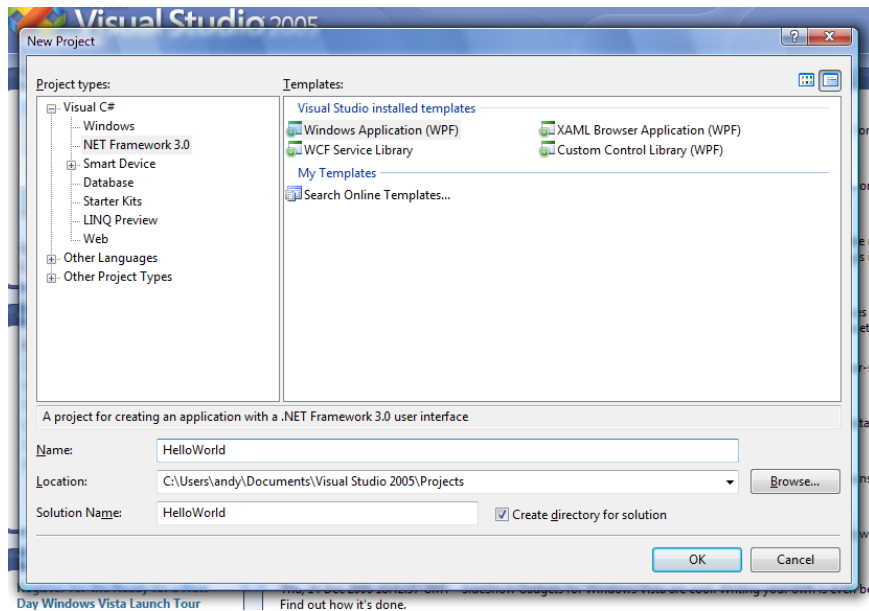
V prvním souboru je v XAMLu definován vzhled okna, tj. jednotlivé elementy jako tlačítka, textová pole, labely, jejich umístění a vzhled (barvy, barevné gradieny, písma, ...), dále animace a chování jednotlivých prvků. V druhém připojeném souboru ".xaml.cs" je programová část aplikace. Zde můžete použít jakýkoliv .NET jazyk, nejčastěji používané jsou C# a VB.NET. Dále v tomto článku budu používat C#.

** Abyste mohli vyvíjet WPF aplikace ve Visual Studiu 2005, musíte mít nainstalované odpovídající vývojové rozšíření ve vašem počítači. Popis a seznam naleznete v předchozím tutoriálu.*

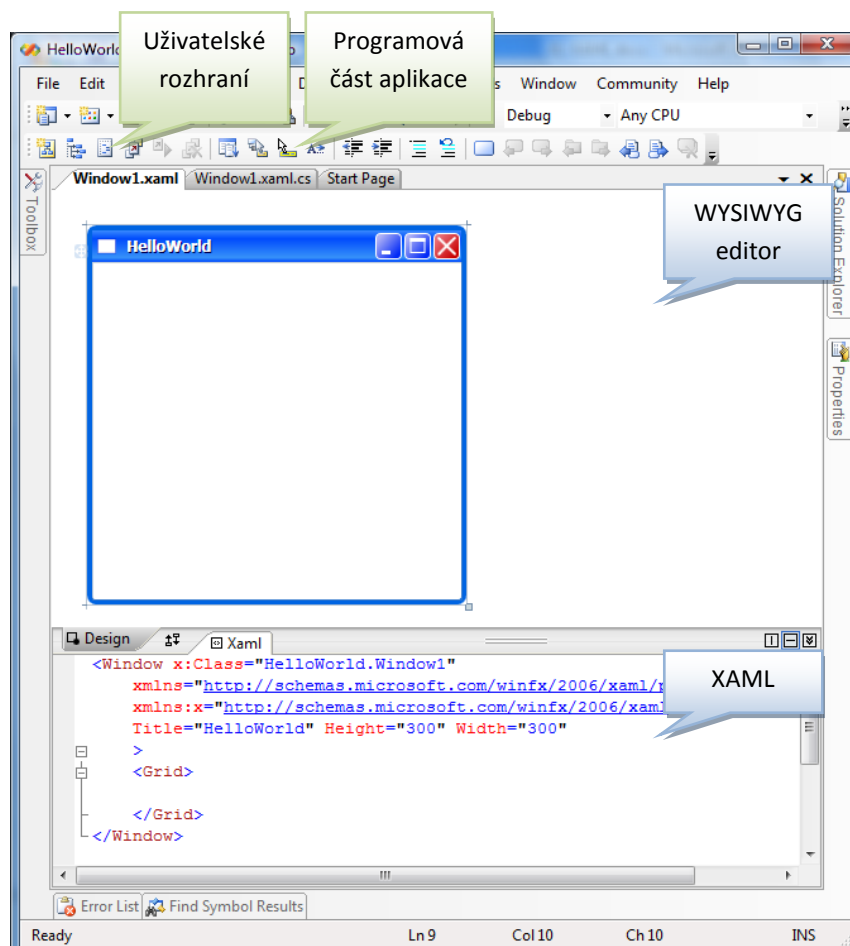
Hello World

Začněme tedy nejjednodušší aplikací "Hello World". Nejdříve vytvoříme nový WPF projekt

File > New Project > Visual C# > NET Framework 3.0 > Windows Application (WPF) pojmenovaný "HelloWorld"



Po vytvoření se otevře okno rozdělené na dvě části. V té vrchní je WYSIWYG editor, stejný jako u WinForms aplikací a nově ve spodní části je XAML kód, programátor si tedy může vybrat, zda chce používat WYSIWYG nebo psát XAML.



Pokud bychom nyní aplikaci spustili (F5), otevřelo by se nám naše prázdné okno *Window1.xaml*.

Druhý soubor, který byl vytvořen je *App.xaml*, pro nás zatím nezajímavý, jediné co stojí za zmínku je vlastnost *StartupUri*, která určuje počáteční okno aplikace. V našem případě

```
StartupUri="Window1.xaml"
```

Přejdemě teď zpět do souboru *Window1.xaml*, odstraníme tagy `<Grid>` a `</Grid>`, které nyní nepotřebujeme a pro zobrazení textu "Hello World!" použijeme *Label* (jednoduchá kontrola pro zobrazování textu), dále nastavíme barvu pozadí a styl a velikost písma.

```
<Window x:Class="HelloWorld.Window1"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="HelloWorld" Height="300" Width="300"
  >
  <Label Background="LightBlue"
    FontWeight="Bold"
    FontSize="20">Hello World!</Label>
</Window>
```

.. a nyní spustíme



* WYSIWYG editor automaticky nereflektuje změny provedené v XAML kódu, proto pro obnovení na WYSIWYG editor klikněte

Do většiny kontrol (Window, Button, TextBlock, ...) můžete vkládat další prvky.

```
<Label><TextBox>zadej jmeno</TextBox></Label>
```

nebo

```
<Button><Image Source="Andy.png" /></Button>
```

Problém ale nastává, pokud budeme chtít vložit více elementů najednou

```
<Button>
  <Image Source="OpenIco.png" />
  <Label>Otevřít</Label>
</Button>
```

^^ tento XAML kód vám nahlásí chybu

WPF totiž v této situaci neví, jak má prvky uvnitř tlačítka umístit, zda všechny prvky přes sebe roztáhnout nebo vložit do levého horního rohu, vedle sebe nebo třeba vertikálně pod sebe? Možností je mnoho a tak dovnitř vložit pouze jeden element, takže se neobejdete bez tzv. ...

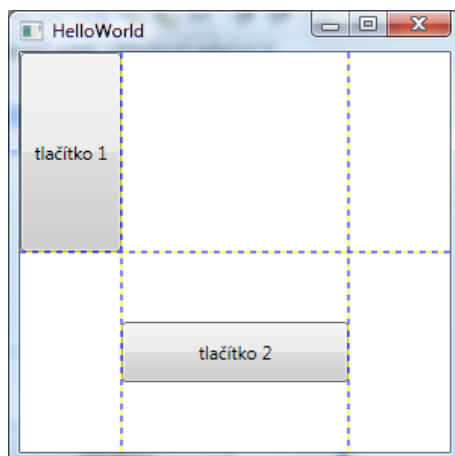
Layout controls

Jedná se o speciální kontroly, do kterých je možné vložit "nekonečně" mnoho dalších prvků a které mají svá specifická pravidla na chování elementů uvnitř.

Grid

Nejpoužívanější. Jedná se o tabulku nebo spíše mřížku podobnou HTML tabulce. Vnitřním prvkům nastavujeme řádek a sloupec, do kterého patří. Pokud prvky nemají explicitně nastavenou velikost, jsou roztaženy přes celou buňku.

```
<Window x:Class="HelloWorld.Window1"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="HelloWorld" Height="300" Width="300"
  >
  <Grid ShowGridLines="True">
    <Grid.RowDefinitions>
      <RowDefinition />
      <RowDefinition />
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
      <ColumnDefinition />
      <ColumnDefinition Width="150" />
      <ColumnDefinition />
    </Grid.ColumnDefinitions>
    <Button Grid.Column="0" Grid.Row="0">tlačítko 1</Button>
    <Button Grid.Column="1" Grid.Row="1" Height="40">tlačítko 2</Button>
  </Grid>
</Window>
```



* Vyzkoušejte si chování této tabulky při změně velikosti okna

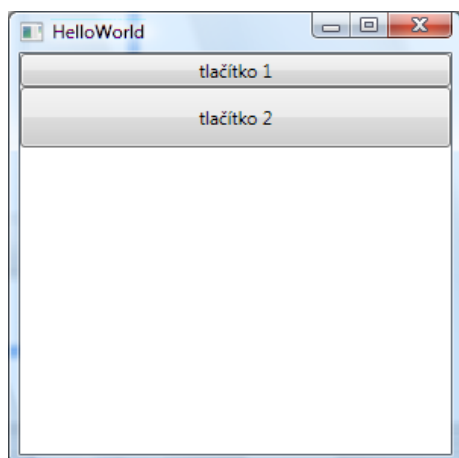
StackPanel

Elementy jsou vkládány jeden za druhým a to buď vertikálně

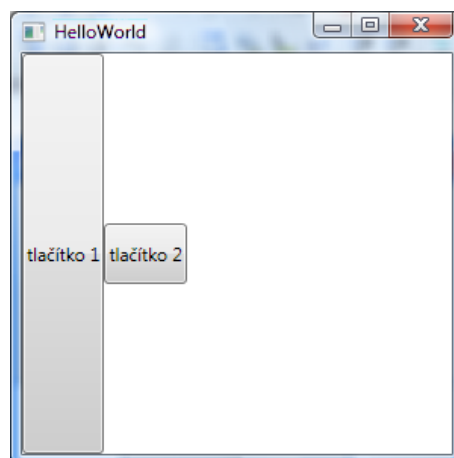
```
<StackPanel>  
  <Button>tlačítko 1</Button>  
  <Button Height="40">tlačítko 2</Button>  
</StackPanel>
```

nebo horizontálně

```
<StackPanel Orientation="Horizontal">  
  <Button>tlačítko 1</Button>  
  <Button Height="40">tlačítko 2</Button>  
</StackPanel>
```



vertikálně

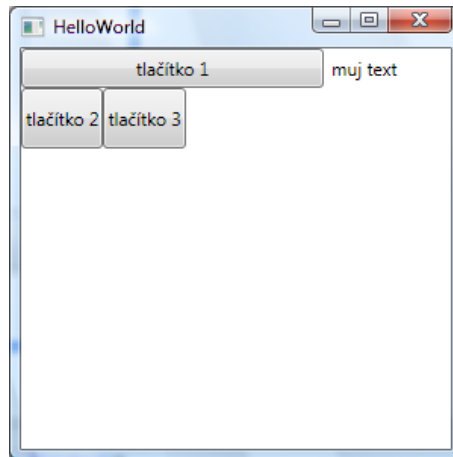


horizontálně

WrapPanel

Obdobně jako u textu jsou zde jednotlivé prvky zobrazeny vedle sebe na řádek, pokud se některý prvek již na řádek nevejde, dojde k zalomení řádku.

```
<WrapPanel>
  <Button Width="200">tlačítko 1</Button>
  <Label>muj text</Label>
  <Button Height="40">tlačítko 2</Button>
  <Button>tlačítko 3</Button>
</WrapPanel>
```



DockPanel

Obdobně jako ve WinForms slouží DockPanel k "přilepení" elementů uvnitř k jedné ze čtyř stran. Poslední prvek zabere volné místo.

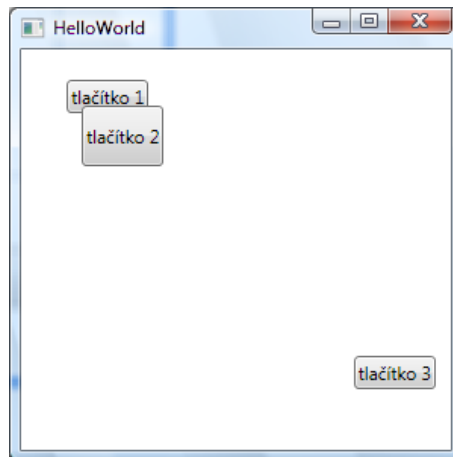
```
<DockPanel>
  <Button DockPanel.Dock="Top">tlačítko 1</Button>
  <Button DockPanel.Dock="Left">tlačítko 2</Button>
  <Label BorderBrush="Blue" BorderThickness="2">Text label</Label>
</DockPanel>
```



Canvas

Poslední z pěti layout kontrol je Canvas. Prvky uvnitř se absolutně pozicují a proto byste Canvas měli používat co nejméně.

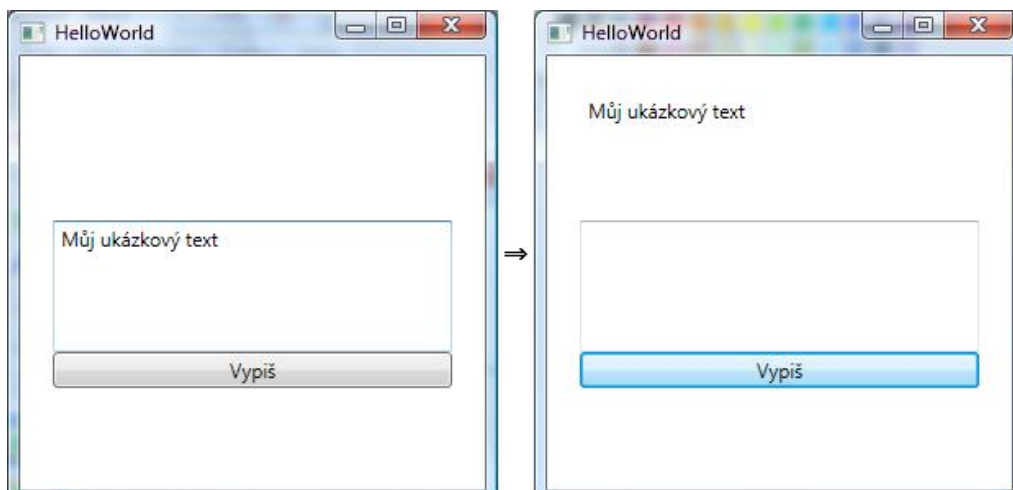
```
<Canvas>
  <Button Canvas.Top="20"
    Canvas.Left="30">tlačítko 1</Button>
  <Button Canvas.Top="37"
    Canvas.Left="40"
    Height="40">tlačítko 2</Button>
  <Button Canvas.Bottom="40"
    Canvas.Right="10">tlačítko 3</Button>
</Canvas>
```



Volání metod

XAML slouží pouze k vytváření uživatelského rozhraní, takže zde musí existovat způsob interakce s programovou částí aplikace.

To ukážu přímo na aplikaci, která bude mít tlačítko, textové pole a textbox. Text z textboxu se po kliknutí na tlačítko "Vypiš" vypíše do textového pole a textbox se vymaže.



Window1.xaml

```
<Window x:Class="HelloWorld.Window1"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="HelloWorld" Height="300" Width="300"
>
<StackPanel Margin="20">
    <Label Name="myLabel" Height="80" />
    <TextBox Name="myTextBox" Height="80" />
    <Button Click="OnButtonClick">Vypiš</Button>
</StackPanel>
</Window>
```

Window1.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace HelloWorld
{
    public partial class Window1 : System.Windows.Window
    {
        public Window1()
        {
            InitializeComponent();
        }

        public void OnButtonClick(object sender, RoutedEventArgs e)
        {
            myLabel.Content = myTextBox.Text;
            myTextBox.Text = "";
        }
    }
}
```

- 1) pojmenovali jsme si textové pole (*myLabel*) a textbox (*myTextBox*). Přes tyto názvy můžeme přistupovat k vlastnostem, metodám a událostem těchto kontrol v rámci dané třídy *Window1*
- 2) nadefinovali jsme, že metoda "OnButtonClick" se má zavolat po kliknutí na tlačítko (událost *Click* tlačítka)
- 3) v metodě "OnButtonClick" volané po kliknutí na tlačítko jsme přiřadili hodnotu *myTextBox.Text* (zde se nachází vepsaný text) do našeho Labelu (textového pole)

* Pokud pojmenujete některou z kontrol v XAMLu, intellisense v C# tuto kontrolu ještě nebude znát. Aktualizace je možné dosáhnout "buildnutím" aplikace - tlačítko F6, poté již bude tato kontrola v intellisence k dispozici.

Nakonec ještě srovnání - C# vs XAML

Už víme, že XAML je nový značkovací jazyk pro psaní uživatelského rozhraní, ale ve WPF máme stále možnost psát uživatelské rozhraní i starým způsobem, stejně jako ve WinForms - programově v C# nebo jiném .NET jazyce.

Zde je pro porovnání ukázka stejné části aplikace v XAMLu a v C#.

XAML

```
<DockPanel>
  <Label Content="Hello World"
        FontStyle="Italic"
        FontWeight="Bold"
        Foreground="Red"
        HorizontalAlignment="Center"
        VerticalAlignment="Center" />
</DockPanel>
```

C#

```
DockPanel myDockPanel = new DockPanel();

Label myLabel = new Label();
myLabel.Content = "Hello World";
myLabel.FontStyle = FontStyles.Italic;
myLabel.FontWeight = FontWeights.Bold;
myLabel.Foreground = Brushes.Red;
myLabel.HorizontalAlignment = HorizontalAlignment.Center;
myLabel.VerticalAlignment = VerticalAlignment.Center;

myDockPanel.Children.Add(myLabel);
```

XAML je tedy kratší a přehlednější, v některých situacích si ale neporadí (např. vypisování tlačítek v cyklu) a musíme použít C#.

Závěr

WPF obsahuje spoustu kontrol, potřebných při vývoji aplikací o kterých ale nebylo v mých silách napsat. Vyzkoušejte si ty nejdůležitější, mezi které určitě patří TextBox, Label, TextBlock, Button, Border, Rectangle a další. Mezi nejpoužívanější vlastnosti (property) se jistě řadí Width, Height, HorizontalAlignment, VerticalAlignment, Background, Fill, BorderThickness, BorderBrush ...

Na Internetu naleznete jistě spoustu příkladů a ukázkových souborů. Seznam některých webů naleznete na <http://www.unitedstatesof.net/wpf>.