Základy programování 3: C#

Martin Kauer

Palacký University
Olomouc
Czech Republic

7. listopadu 2018

Okenní aplikace

- *Windows Forms* je původní okenní systém, který je postaven na *Win32* API.
- Pokud k tomu není dobrý důvod, není potřeba používat a lépe je použít nástupce Windows Forms a to *Windows Presentation Foundation* (*WPF*) nebo *Universal Windows Platform* (*UWP*).
- Je možné použít další UI frameworky: *GTK*, platformově specifické vrstvy *Xamarin*, ...
- Budeme se zabývat zejména WPF.
- I zběžné prozkoumání WPF by vydalo na celý semestr, tato prezentace je jen přehled vybraných základů.

WPF

- Velmi rozsáhlý UI framework, který podporuje mnoho soudobých programovacích technik okenních aplikací.
- Podporuje HW akceleraci (vykreslování pomocí DirectX) i SW vykreslování.
- Většinu částí najdete v namespacu `System.Windows`.
- Dokáže oddělit práci designera a programátora.
- Vizuální stránku aplikace lze specifikovat pomocí *Extensible Application Markup Language* (*XAML*), což je odnož XML pro deklarativní vytváření okenních aplikací.
- XAML můžeme editovat jako klasické XML v textové podobě, nebo použít tzv. designery.
- Visual Studio má vestavěný WYSIWYG designer pro XAML.
- Existuje i speciální nástroj *Blend for Visual Studio*.

WPF - cvičení

- Ve Visual Studiu si vytvořte nový projekt pod záložkou C# → *Wpf App* a nazvěte jej **WpfShowcase**.
- Otevřete si v designeru vytvořený soubor *MainWindow.xaml* a vložte do něj následující kód. Projekt zkompilujte a spusťte.

```
<Window x:Class="WpfShowcase.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/
            presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="200" Width="400">
    <TextBlock Text="Hello World!"/>
</Window>
```

WPF - cvičení

- Do *MainWindow.xaml* vložte následující kód.

```
<Window x:Class="WpfShowcase.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/
            presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="200" Width="400">
    <Button Content="Hello Button" Name="HelloButton" Click="
        HelloButton_OnClick"/>
</Window>
```

- Do *MainWindow.xaml.cs* přidejte následující metodu a program vyzkoušejte.

```
private void HelloButton_OnClick(object sender, RoutedEventArgs e)
{
    MessageBox.Show("Hello world!", "Hello button clicked",
        MessageBoxButton.OK);
}
```

WPF Layouts

- WPF obsahuje rozsáhlou škálu základních kontrol.
- Začněme vybranými layouty, všechny dědí po abstraktní třídě `Panel` a mají veřejnou vlastnost `Children`, určující obsažené elementy:
  - `Canvas` - obsažené prvky se pozicují absolutně.
  - `DockPanel` - obsažené prvky lze "dokovat", tzn. skládat horizontálně, vertikálně, či relativně k sobě.
  - `Grid` - skládá prvky do mřížky, kterou si definujeme.
  - `StackPanel` - skládá prvky do posloupnosti buď horizontální či vertikální.
  - `VirtualizingPanel` - abstraktní třída, základ pro layouty podporující virtualizaci obsažených prvků.
  - `WrapPanel` - jako stack panel, ale s podporou zalomení.

WPF Controls

- Vybrané základní kontroly:
  - `TextBlock` - vykresluje řetězec z vlastnosti `Text`.
  - `TextBox, RichTextBox` - umožňuje zadávání řetězce.
  - `Button` - událost kliknutí `Click` a jako obsah tlačítka vykresluje hodnotu vlastnosti `Content`. To může být text, obrázek, `DateTime`, ...
  - `CheckBox` - může být i třístavový.
  - `RadioButton` - automaticky seskupuje prvky ve stejném kontejneru nebo můžeme určit seskupení pomocí vlastnosti `GroupName`.
  - `Image` - vykresluje obrázek ve formátu .bmp, .gif, .ico, .jpg, .png, .wdp, nebo .tiff.
  - `MediaElement` - pro přehrávání audia/videa. Jedná se jen o přehrávač, jeho ovládání je nutno dodělat.
  - `Menu`, `MenuItem` - obecné menu a jeho prvky, lze použít jako kontextové i hlavní menu aplikace.
  - `ProgressBar` - pro zobrazení postupu nějaké operace.
  - ...

WPF Controls

- Vybrané kontroly zobrazující kolekce prvků:
  - `ComboBox` - kontrola pro výběr z předem definované kolekce prvků, kterou lze zobrazit či skrýt pomocí tlačítka.
  - `ListBox` - zobrazuje kolekci prvků s možností vybrat jeden, či více prvků.
  - `ListView` - potomek `ListBoxu`, přidává vlastnost `View` pro další možnost přizpůsobení zobrazení prvků kolekce.
  - `DataGrid` - zejména vhodné pro zobrazení databázových tabulek apod.
- Všechny zmíněné kontroly dědí po třídě `ItemsControl`, která obsahuje vlastnost `IEnumerable ItemsSource`, sloužící pro nastavení zobrazované kolekce prvků. Popř. můžeme použít vlastnost `ItemsCollection Items`. Tyto dvě vlastnosti ale nelze používat dohromady.
- Pokud se kolekce změní a nejedná se o kolekci implementující `INotifyCollectionChange`, je nutné aktualizovat kontrolu např. pomocí `kontrola.Items.Refresh()`.

WPF cvičení

- Vytvořte následující okno a upravte jej, aby se při výběru z listboxu vybraný prvek vypsal do `TextBlocku` SelectedItemTextBlock:

```
<Window x:Class="WpfShowcase.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/
            presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="200" Width="400">
    <StackPanel>
        <TextBlock Name="SelectedItemTextBlock"/>
        <ListBox SelectionMode="Single">
            <ListBoxItem>Item 1</ListBoxItem>
            <ListBoxItem>Item 2</ListBoxItem>
            <ListBoxItem>Item 3</ListBoxItem>
            <ListBoxItem>Item 4</ListBoxItem>
            <ListBoxItem>Item 5</ListBoxItem>
        </ListBox>
    </StackPanel>
</Window>
```

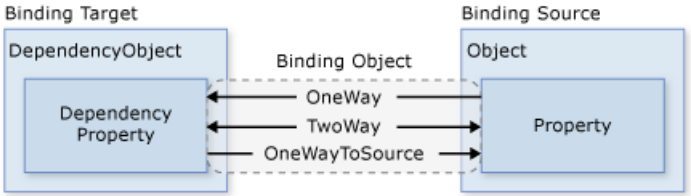
- Upravte program aby jste kolekci prvků nastavili přes vlastnost `ItemsSource` v kódu.

WPF přizpůsobení

- Vizuální stránka všech WPF kontrol je velice dobře přizpůsobitelná.
- Pro přizpůsobení vzhledu kontroly lze použít vlastnost `Template`, což úplně změní vizuální strom dané kontroly.
- Můžeme použít `Style`, což je kolekce hodnot, pro dané vlastnosti kontrol. Pomocí stylů můžeme nastavovat veřejné vlastnosti kontroly a lze je použít lokálně na jednotlivé kontroly nebo globálně na všechny kontroly (daného typu) v celém kontejneru/okně/aplikaci.
- Kontroly pro zobrazování kolekcí mají navíc vlastnost `DataTemplate`, která určuje jak se budou zobrazovat prvky kolekce.
  - Pro každý prvek v kolekci se vytvoří kontrola předepsaná tímto templatem.
  - Při psaní templatu v XAML můžeme použít jako hodnotu {`Binding`}, což je aktuální prvek, ke kterému se vytváří vizuální podoba pomocí tohoto templatu.
  - Pomocí {`Binding`} lze specifikovat i vlastnost aktuálního prvku: {`Binding Path=SomeProperty.SomeSubproperty`}

WPF Binding

- WPF obsahuje systém pro svázání aplikačních dat s jejich vizualizací.
- Tato vazba je reprezentována pomocí instance třídy `Binding`.
- Pokud je vazba tak nastavena, změny v datech se automaticky projeví ve vizualizaci a změny lze promítnout i obráceně.
- Typ vazby určuje vlastnost `Mode`, která může mít hodnoty `Default`, `OneTime`, `OneWay`, `OneWayToSource`, `TwoWay`.



WPF Binding

- Binding Source se v XAML určuje pomocí atributu `Source`, `RelativeSource`, `ElementName`, ...
- Binding Source může být statická vlastnost/člen, prvek `Resource`, aktuální prvek zobrazované kolekce, ...

```
{Binding Source={StaticResource myDataSource}}
{Binding ElementName=SomeElementName}
{Binding RelativeSource={RelativeSource FindAncestor, AncestorType
    ={x:Type ListBox}}}
```

- Aby se změna vlastnosti projevila ve vizualizaci, je potřeba aby daný objekt implementoval `INotifyPropertyChanged` a správně tento mechanismus používal pro danou vlastnost.
- Opačně, aby se změna v kontrole reprezentující vizuální část promítla do hodnoty dané vlastnosti objektu, musí být správně nastavena vlastnost `UpdateSourceTrigger`. Ta je ve výchozím nastavení často buď `PropertyChange`, což změnu promítne hned při změně hodnoty vlastnosti, nebo `LostFocus`, což změnu promítne až při ztátě focusu danou kontrolou.

WPF cvičení

- Vezměte příklad s `ListBoxem` a `TextBoxem` a zajistěte, aby se v `TextBoxu` zobrazoval aktuálně vybraný prvek, ale jen s použitím XAML (a Bindings).

WPF cvičení

- Vyzkoušejte následující příklad:

```
<Window x:Class="WpfShowcase.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/
            presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="200" Width="400">
    <Window.Resources>
        <Style TargetType="Button">
            <Setter Property="Control.Background" Value="Yellow"/>
        </Style>
    </Window.Resources>
    <StackPanel>
        <Button Content="Button 1" />
        <Button Content="Button 2" />
        <Button Content="Button 3" />
    </StackPanel>
</Window>
```

WPF cvičení

- Vezměte příklad s ListBoxem a ItemsSource a přidejte ItemTemplate.

```
<Window x:Class="WpfShowcase.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/
            presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="200" Width="400">
    <ListBox SelectionMode="Single" HorizontalContentAlignment="
        Stretch">
        <ListBox.ItemTemplate>
            <DataTemplate>
                <StackPanel Orientation="Horizontal">
                    <TextBlock Text="Item:" Background="Green"
                        Margin="10,5"/>
                    <TextBlock Text="{Binding}" Background="Yellow"
                         Margin="10,5" />
                </StackPanel>
            </DataTemplate>
        </ListBox.ItemTemplate>
    </ListBox>
</Window>
```

WPF objektové stromy

- Obecně se kontroly organizují do stromu.
- Ve WPF existují dva organizační stromy.
- Logický strom je prakticky ten který vytváříme v XAML zanořováním tagů. Pro práci s logickým stromem existuje třída `LogicalTreeHelper`.
- Vizuální strom navíc zohledňuje kontroly vytvořené vizuálním přizpůsobením (templaty, apod.). Pro práci s vizuálním stromem existuje třída `VisualTreeHelper`.

WPF cvičení

1. Vytvořte WPF aplikaci, která bude vizualizovat instanci třídy `List<Person>`.
2. Pro vizualizaci použijte nějakou kontrolu pro zobrazení kolekcí.
3. V kódu vytvořte instanci `List<Person>`, naplňte ji několika prvky a zobrazte ji.
4. Použijte templaty, aby se čitelně zobrazovali údaje o zadaných osobách.
5. Do okna přidejte sekci, pomocí které bude možné přidávat prvky do našeho listu. Zobrazení listu by mělo reflektovat aktuální obsah listu.
6. Místo `Listu` použijte `ObservableCollection`. Co se změní?
7. Zajistěte, aby třída `Person` implementovala `INotifyPropertyChanged` a ohlašovala změny svých vlastností.
8. Umožněte editaci vybrané položky (instance Person).