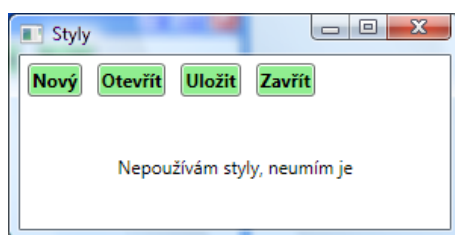


2. WPF - Styly

Často když vytváříme uživatelské rozhraní a chceme zachovat jednotný vzhled, jsme nuceni psát některé vlastnosti pořád dokola. Například pokud bychom chtěli, aby text ve všech textových polích měl červenou barvu, byl 11 px vysoký a podtržený, museli bychom tyto vlastnosti psát u každého pole znovu ... a nebo použít styly.

Bez stylů bychom se sice obešli, ale proč je lepší styly používat? Podívejme se na jednoduchou aplikaci.

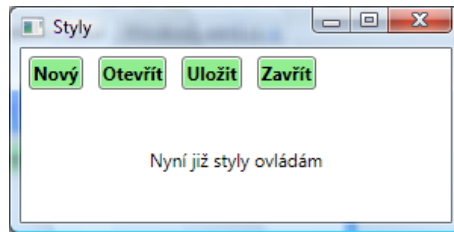


```
<DockPanel>
  <StackPanel Orientation="Horizontal" DockPanel.Dock="Top">
    <Button Background="LightGreen"
      Margin="5"
      FontWeight="Bold">Nový</Button>
    <Button Background="LightGreen"
      Margin="5"
      FontWeight="Bold">Otevřít</Button>
    <Button Background="LightGreen"
      Margin="5"
      FontWeight="Bold">Uložit</Button>
    <Button Background="LightGreen"
      Margin="5"
      FontWeight="Bold">Zavřít</Button>
  </StackPanel>
  <Label VerticalAlignment="Center"
    HorizontalAlignment="Center">
    Nepoužívám styly, neumím to
  </Label>
</DockPanel>
```

I takto triviální aplikace má základní nedostatky. Část kódu se zde neustále opakuje, takže psaní bylo časově náročnější, kód se stal méně přehledný a pokud bychom se rozhodli změnit barvu pozadí tlačítek, musíme změnu provést u každé ho tlačítka zvlášť. Zde se jedná pouze o jednoduchou aplikaci, ale u větších programů by už taková změna mohl být problém.

Řešením je použití stylů. Styl je skupina vlastností jako například barva pozadí, velikost a styl písma a nebo šířka a barva rámečku, kterou můžeme aplikovat na vybrané kontroly.

Zde je stejná aplikace ale s použitím stylů



```
<DockPanel>
  <DockPanel.Resources>
    <Style x:Key="GreenButtonStyler" TargetType="Button">
      <Setter Property="Background" Value="LightGreen" />
      <Setter Property="Margin" Value="5" />
      <Setter Property="FontWeight" Value="Bold" />
    </Style>
  </DockPanel.Resources>

  <StackPanel Orientation="Horizontal" DockPanel.Dock="Top">
    <Button Style="{StaticResource GreenButtonStyler}">Nový</Button>
    <Button Style="{StaticResource GreenButtonStyler}">Otevřít</Button>
    <Button Style="{StaticResource GreenButtonStyler}">Uložit</Button>
    <Button Style="{StaticResource GreenButtonStyler}">Zavřít</Button>
  </StackPanel>
  <Label VerticalAlignment="Center"
    HorizontalAlignment="Center">
    Nyní již styly ovládám
  </Label>
</DockPanel>
```

Výhody použití stylů jsou jednoznačné, pokud bychom nyní chtěli změnit barvu pozadí tlačítek, stačí provést změnu pouze v daném stylu. Používání stylů tedy dělá kód flexibilní a zároveň přehlednější, protože se vyvarujeme opisování kódu.

Jak to funguje

Používání stylů ve WPF funguje obdobně jako CSS (Cascading Style Sheets) u webových stránek.

V naší aplikaci jsme si styl nejdříve nadefinovali mezi elementy `<Style> ... </Style>`. Atribut **x:Key** slouží k pojmenování stylu. Podle jména k tomuto stylu přistupujeme.

TargetType určuje na jaký prvek může být tento styl aplikován. Jedná se o nepovinný atribut, pokud ho ale vynecháme, musíme použít před každou vlastností typ kontroly.

Vynechání atributu *TargetType* v elementu *Style*

```
<Style x:Key="GreenButtonStyler">
  <Setter Property="Button.Background" Value="LightGreen" />
  <Setter Property="Button.Margin" Value="5" />
  <Setter Property="Button.FontWeight" Value="Bold" />
</Style>
```

Nejdůležitější jsou elementy **Setter**. Ty nastavují některou z vlastností (*Property*) kontroly na konkrétní hodnotu (*Value*).

```
<Setter Property="Button.FontWeight" Value="Bold" />
```

Celý styl jsme vložili do *DockPanel.Resources*. Jedná se o místo, kam můžeme ukládat různé objekty. *Resources* budou probírány dále v tomto tutoriálu.

Použití

Chceme-li, aby kontrola používala některý z našich stylů, stačí tomuto prvku zadat do vlastnosti *Style* jméno tohoto stylu.

```
<Button Style="{StaticResource GreenButtonStyler}">Nový</Button>
```

Zde se setkáváme s novým způsobem zápisu. Jedná se o kratší zápis tohoto kódu

```
<Button>Zavřít
  <Button.Style>
    <StaticResource ResourceKey="GreenButtonStyler" />
  </Button.Style>
</Button>
```

StaticResource znamená, že styl bude načten z *Resources* pouze jednou, hned po spuštění aplikace. Jinou možností je *DynamicResource*, která se ale ve spojení se styly nepoužívá.

Resources

Styly ukládáme do tzv. *Resources* neboli zdrojů, tj. úložiště, kam můžeme ukládat různé objekty a přistupovat k nim přes unikátní identifikátor *x:Key*.

Ve WPF má většina kontrol svůj zdroj, to který použijeme záleží pouze na tom, v jakém rozsahu má být styl/objekt přístupný. Ke stylům uloženým v *Resources* můžou přistupovat pouze vnořené prvky. V našem případě jsme použili *DockPanel.Resources*, takže styl je přístupný pouze tlačítkům uvnitř tohoto *DockPanelu*, kdybychom chtěli styl přístupný všem prvkům v daném okně, použili bychom *Window.Resources*, pro celou aplikaci by to byl *Application.Resources*, který se nachází v souboru *App.xaml*.

```

<Application x:Class="Styly.App"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  StartupUri="Window1.xaml"
>
  <Application.Resources>
    <Style x:Key="mujStyl">
      ...
    </Style>
  </Application.Resources>
</Application>

```

Pokud kontrola používá styl, hledání tohoto stylu probíhá nejdříve v *Resources* rodičovského elementu, pokud zde není, pokračuje o úroveň výš až se dostane k poslednímu, nejvyššímu elementu v hierarchii a tím je element *Application*, pokud ani v *Application.Resources* styl neexistuje, nahlásí chybu.

Pokud v hierarchii existují dva a více stylů se stejným jménem, použije se nejbližší.

Příklad:



```

<StackPanel>
  <StackPanel.Resources>
    <!-- 1. styl - ButtonStyler (zelena) -->
    <Style x:Key="ButtonStyler" TargetType="Button">
      <Setter Property="Background" Value="LightGreen" />
    </Style>
  </StackPanel.Resources>
  <WrapPanel>
    <WrapPanel.Resources>
      <!-- 2. styl - ButtonStyler (modra) -->
      <Style x:Key="ButtonStyler" TargetType="Button">
        <Setter Property="Background" Value="LightBlue" />
      </Style>
    </WrapPanel.Resources>
    <Button Style="{StaticResource ButtonStyler}">Nový</Button>
    <Button Style="{StaticResource ButtonStyler}">Otevřít</Button>
    <Button Style="{StaticResource ButtonStyler}">Uložit</Button>
    <Button Style="{StaticResource ButtonStyler}">Zavřít</Button>
  </WrapPanel>
  <WrapPanel>
    <Button Style="{StaticResource ButtonStyler}">Zkopírovat</Button>
    <Button Style="{StaticResource ButtonStyler}">Vložit</Button>
    <Button Style="{StaticResource ButtonStyler}">Odstranit</Button>
  </WrapPanel>
</StackPanel>

```

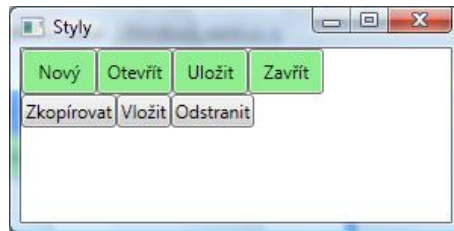
1. WrapPanel

2. WrapPanel

V naší aplikaci jsou nedefinovány dva styly se stejným jménem *ButtonStyler*, ale každý v jiném *Resources*. Tlačítka v prvním *WrapPanelu* používají 2. styl, protože je v hierarchii blíže než 1. styl. Tlačítka v druhém *WrapPanelu* nemají přístup k 2. stylu, používají 1. styl.

Obecné styly, styly nepojmenované

Tyto styly nemají jméno a jsou aplikované na všechny kontroly typu uvedeného v *TargetType* (ten je při absenci jména povinný). Přesněji tyto styly jsou aplikované na všechny kontroly v rozsahu *Resources*.



```
<StackPanel>
  <WrapPanel>
    <WrapPanel.Resources>
      <!-- Obecný styl, nepojmenovaný -->
      <Style TargetType="Button">
        <Setter Property="Background" Value="LightGreen" />
        <Setter Property="Width" Value="50" />
        <Setter Property="Height" Value="30" />
      </Style>
    </WrapPanel.Resources>
    <Button>Nový</Button>
    <Button>Otevřít</Button>
    <Button>Uložit</Button>
    <Button>Zavřít</Button>
  </WrapPanel>
  <WrapPanel>
    <Button>Zkopírovat</Button>
    <Button>Vložit</Button>
    <Button>Odstranit</Button>
  </WrapPanel>
</StackPanel>
```

Všimněte si, že styl je použit pouze na tlačítkách uvnitř prvního *WrapPanelu*, ostatní tlačítka nemají k tomuto stylu přístup.

Podobně bychom mohli změnit tlačítka v celé aplikaci

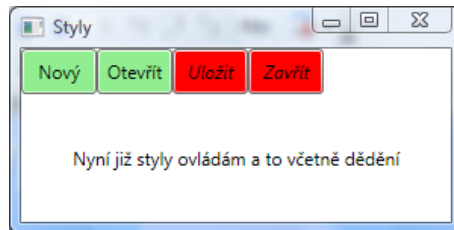
```
<Application.Resources>
  <Style TargetType="Button">
    <Setter Property="Background" Value="LightGreen" />
  </Style>
</Application.Resources>
```

... v souboru *App.xaml*.

Dědičnost

Pokud chceme založit styl na vlastnostech jiného stylu, použijeme dědičnost. Vytvoříme tedy nový styl, který podědí vlastnosti z jiného, již vytvořeného stylu, a tyto vlastnosti ještě upraví nebo přidá nové.

Příklad: chceme styl tlačítka se zeleným pozadím, velikostí písma 12 a velikostí tlačítka 50x30. Druhý styl má být stejný až na to, že barva pozadí má být červená a nově text kurzívou. Druhý styl tedy podědíme od prvního a tak převeze jeho vlastnosti. Ty nakonec ještě upravíme.



```
<DockPanel>
  <DockPanel.Resources>
    <!-- První styl -->
    <Style x:Key="GreenButtonStyler" TargetType="Button">
      <Setter Property="Background" Value="LightGreen" />
      <Setter Property="Width" Value="50" />
      <Setter Property="Height" Value="30" />
    </Style>

    <!-- Druhý styl, poděděný -->
    <Style x:Key="RedButtonStyler" TargetType="Button"
          BasedOn="{StaticResource GreenButtonStyler}">
      <Setter Property="Background" Value="Red" />
      <Setter Property="FontStyle" Value="Italic" />
    </Style>
  </DockPanel.Resources>

  <StackPanel Orientation="Horizontal" DockPanel.Dock="Top">
    <Button Style="{StaticResource GreenButtonStyler}">Nový</Button>
    <Button Style="{StaticResource GreenButtonStyler}">Otevřít</Button>
    <Button Style="{StaticResource RedButtonStyler}">Uložit</Button>
    <Button Style="{StaticResource RedButtonStyler}">Zavřít</Button>
  </StackPanel>

  <Label VerticalAlignment="Center"
        HorizontalAlignment="Center">
    Nyní již styly ovládám a to včetně dědění
  </Label>
</DockPanel>
```

Parametr *BasedOn* v elementu *Style* určí, ze kterého stylu mají být vlastnosti nového stylu zděděny.

Dědění z obecných (nepojmenovaných) stylů lze provést takto:

```
<Style TargetType="Button">
  <Setter Property="Background" Value="LightGreen" />
</Style>
<Style BasedOn="{StaticResource {x:Type Button}}">
  ...
</Style>
```

Pozor! Styl, ze kterého dědíme musí být vždy uveden dřív. Zde je ukázka špatného použití.

```
<Style x:Key="RedButtonStyler" TargetType="Button"
  BasedOn="{StaticResource GreenButtonStyler}">
  ...
</Style>
<Style x:Key="GreenButtonStyler" TargetType="Button">
  ...
</Style>
```

^^ tento XAML kód vám nahlásí chybu, protože podděný RedButtonStyler je dřív než GreenButtonStyler

Závěr

Stejně jako u moderních webových stránek se už neobejdeme bez CSS, tak ani ve WPF se neobejdeme bez použití stylů, které nám ulehčují práci při psaní aplikace.

Více informací o WPF naleznete na www.unitedstatesof.net/wpf

Aleš Šturala, 21. 1. 2007

<http://hidentity.org/hid/CZ123456>