WPF – Styles / Templates

Petr Novák / 2021-01-15

Obsah

1 Úvod

„Style“ a „Template“ jsou velmi užitečné, pokud je potřeba například vytvořit stejnou grafickou podobu pro několik prvků v GUI a tu případně později upravovat. Samozřejmě lze definovat například barevný podklad, font textu, styl zobrazení prvku při najetí myší v každém GUI prvku samostatně. Pokud je však vyžadován (dostatečně) stejný vzhled / chování těchto prvků je vhodné použít „Style“ nebo „Template“ a tím v podstatě vytvořit vzor / chování pro všechny požadované prvky pouze jednou. Jde tedy nejen o úsporu času, ale i o přehlednost kódů. Současně například takto definovaný „Style“ lze i programově upravit nebo dokonce vyměnit za jiný a tím v podstatě změnit vzhled aplikace podle potřeb uživatele. Styly je možno i dědit, lze tedy vytvořit základní styl obsahující například barvu textu a jeho font společný všech prvkům a poté od něho podědit jeden styl pro tlačítka, druhý styl pro texty atd.
„Style“ a „Template“ poskytují skutečně velké možnosti a proto budou probírány postupně a zeména na zá

2 Style vs Template

Jsou dva (základní) způsoby jak upravit jakýkoli GUI prvek a to pomocí „Style“ nebo pomocí „Template“ (samozřejmě i současně). Lze říci, že každý GUI prvek obsahuje vlastnosti (barva, font, …) a strukturu (rámeček, tvar, …). Pomocí „Style“ se mění (hlavně) vlastnosti a pomocí „Template“ se mění (hlavně) struktura. „Style“ tedy určuje nastavení / změnu jeho vlastností v podstatě tedy změnu základního vzhledu, ale nikoli struktury (z čeho je prvek složen / vykreslován). Naopak pomocí „Templeta“ je možno (pře)definovat strukturu prvku (nové položky pro jeho složení).
Použitím „Style“ nebo „Template“ není samozřejmě nutno nastavovat všechny vlastnosti znova, nebo definovat celou novou strukturu prvku. „Style“ v podstatě udává co vše se má nastavit pokud není nastaveno přímo v prvku (pokud je nějaká vlastnost nastavena přímo v prvku, tak přepisuje nastaveni ve „Style“). „Template“ udává, jaká bude nová struktura prvku, ale současně co vše se z původní struktury prvku použije (pře-použít lze cokoli podle potřeby). „Template“ poskytuje samozřejmě mnohem větší možnosti pro změnu prvku na který je aplikován.
Pro vysvětlení jednoduchý příklad. Pro „Style“:

```
<!-- definovani stylu s nazvem „MyButtonStyle“ pro „Button“ -->
<Style x:Key="MyButtonStyle" TargetType="Button">
  <!-- nastaveni vlastnosti „Backgroud“ na cervenou barvu -->
  <Setter Property="Background" Value="Red"/>
</Style>

<!-- „Button“ kteremu je prirazen styl „MyButtonStyle“ -->
<Button Style="{StaticResource MyButtonStyle}"/>
```

Pro „Template“:

```
<!-- definovani vzoru s nazvem „MyButtonTemplate“ pro „Button“ -->
<ControlTemplate x:Key="MyButtonTemplate" TargetType="Button">
  <!-- vnitrek „Button“ bude tvořit „Grid“ -->
  <Grid>
    <!-- jako pozadi bude „Rectangle“ -->
    <Rectangle Fill="Green"/>
    <!-- pres pozadi bude puvodni obsah „Button.Content“ (vnitrek z „Button“) -->
    <ContentPresenter/>
  </Grid>
</ControlTemplate>

<!-- „Button“ kteremu je prirazen vzor „MyButtonTemplate“ -->
<Button Template="{StaticResource MyButtonTemplate}"/>
```

Pro současné použití „Style“ a „Template“ v jednom:

```
<!-- definovani stylu s nazvem „MyButtonNew“ pro „Button“ -->
<Style x:Key="MyButtonNew" TargetType="Button">
  <!-- nastaveni vlastnosti „Backgroud“ na cervenou barvu -->
  <Setter Property="Background" Value="Red"/>
  <!-- nastaveni / vytvoreni vzoru -->
  <Setter Property="Template">
    <Setter.Value>
      <!-- vzor je pro „Button“ -->
      <ControlTemplate TargetType="Button">
        <!-- vnitrek „Button“ bude tvořit „Grid“ -->
        <Grid>
          <!-- jako pozadi bude „Rectangle“ -->
          <!-- barva je prevzata ze stylu nebo prvku na který je aplikovan -->
          <Rectangle Fill="{TemplateBinding Background}"/>
         <!-- pres pozadi bude puvodni obsah „Button.Content“ (vnitrek z „Button“) -->
          <ContentPresenter/>
        </Grid>
      </ControlTemplate>
    </Setter.Value>
  </Setter>
</Style>
```

Toto na úvod pro prvotní pochopení „Style“ a „Template“. Ty lze definovat nejen na samostatný prvky GUI, ale rovněž pomocí nich definovat / upravovat například položky / políčka v tabulkách nebo seznamech.

Ahoj
…

```
1. <MultiTrigger>
2.       <MultiTrigger.Conditions>
3.           <Condition Property="IsMouseOver" Value="True"/>

4.           <Condition Property="IsKeyboardFocused" Value="True"/
   >
5.       </MultiTrigger.Conditions>
6.       <MultiTrigger.Setters>
```

```
7.            <Setter Property="Content" Value="Trigger Applied"/>

8.             </MultiTrigger.Setters>
9.    </MultiTrigger>
```

https://markheath.net/post/creating-custom-wpf-button-template-in