Rozdział 13.
Szablony kontrolek

Autor: Jacek Matulewski

e-mail: jacek@phys.uni.torun.pl

Fragment książki „Visual Studio 2017. Tworzenie aplikacji Windows w języku C#” (Helion, 2018) udostępniony studentom Wydziału Fizyki, Astronomii i Informatyki Stosowanej UMK w semestrze zimowym 2020/2021.

Uwaga! W tym rozdziale kontynuowany jest rozwój projektu prezentowanego w serii filmów o XAML

Usuńmy z kodu XAML okna (plik *MainWindow.xaml*) naszej przykładowej aplikacji *XamlWpf* wszystkie omówione w poprzednich rozdziałach formatowania, pozostawiając w oknie cztery przyciski: pierwszy z logo i napisem oraz trzy „puste” (listing 13.1). Przyciski są formatowane stylem przypisanym do typu `Button`. Styl ten jest umieszczony w zasobach globalnych (plik *App.xaml*). W zasobach tych znajdują się również wykorzystywane przez styl pędzle ze stopniową zmianą koloru.

Listing 13.1. Kod XAML okna oczyszczony z animacji

```
<Window x:Class="XamlWpf.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:XamlWpf"
        mc:Ignorable="d"
        Title="MainWindow" Height="350" Width="525">
    <StackPanel Orientation="Vertical">
        <Button Grid.Column="0" Grid.Row="0" Margin="10,10,10,10"
                HorizontalAlignment="Center" VerticalAlignment="Center"
                Width="200" Height="100">
            <StackPanel Orientation="Horizontal">
                <Image Width ="70" Height="70" Source="logo.gif" />
                <TextBlock FontSize="16" Margin="10,0,0,0" Foreground="White">
                    <Run Foreground="Yellow">Uniwersytet</Run><LineBreak/>
                    Mikołaja<LineBreak/>
                    Kopernika
                </TextBlock>
            </StackPanel>
        </Button>
```

```
        <Button Width="100" Height="50" Margin="0,0,0,10" />
        <Button Width="100" Height="50" Margin="0,0,0,10" />
        <Button Width="100" Height="50" Margin="0,0,0,10" />
    </StackPanel>
</Window>
```

Usuńmy z pliku *App.xaml* element określający styl przycisku, czyli element zaczynający się od znacznika `<Style TargetType="Button">`. W jego miejsce wstawimy element opisujący szablon kontrolki. Podsumowuje on wiele z poznanych w drugiej części książki wiadomości dotyczących XAML. Podczas projektowania szablonu programista może przejąć całkowitą kontrolę nad wyglądem i zachowaniem kontrolki, włączając w to reakcje na działania użytkownika i animacje wyzwalane zdarzeniami lub zmianą stanu kontrolki. Listing 13.2 pokazuje pełen kod pliku *App.xaml*, w którym został zdefiniowany szablon o nazwie `SzablonNiebieski` przeznaczony dla przycisków. Definiuje on zaokrąglony prostokąt, który będzie pełnił funkcję tła przycisku. Do określenia jego kolorów korzystamy z gradientów `NiebieskiGradient` i `FioletowyGradient` obecnych już w tym pliku. Ponadto z najechaniem kursorem myszy na kontrolkę związana została animowana zmiana koloru tła, zmiana stopnia jasności oraz obrót przycisku.

Listing 13.2. Kod z pliku App.xaml z wyróżnionym kodem szablonu

```
<Application x:Class="XamlWpf.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:local="clr-namespace:XamlWpf"
             StartupUri="MainWindow.xaml">
    <Application.Resources>
        <LinearGradientBrush x:Key="NiebieskiGradient"
                             StartPoint="0,0.5" EndPoint="1,0.5">
            <GradientStop Color="Blue" Offset="0.0" />
            <GradientStop Color="Navy" Offset="1.0" />
        </LinearGradientBrush>
        <LinearGradientBrush x:Key="FioletowyGradient"
                             StartPoint="0,0.5" EndPoint="1,0.5">
            <GradientStop Color="Blue" Offset="0.0" />
            <GradientStop Color="BlueViolet" Offset="1.0" />
        </LinearGradientBrush>
        <ControlTemplate x:Key="SzablonNiebieski" TargetType="{x:Type Button}">
            <Grid>
                <Rectangle x:Name="tło" RadiusX="15" RadiusY="15"
                           Fill="{StaticResource NiebieskiGradient}">
                    <Rectangle.BitmapEffect>
                        <OuterGlowBitmapEffect GlowColor="black" GlowSize="5"/>
                    </Rectangle.BitmapEffect>
                    <Rectangle.LayoutTransform>
                        <RotateTransform Angle="0" x:Name="obrót" />
                    </Rectangle.LayoutTransform>
                </Rectangle>
                <Viewbox>
                    <ContentPresenter Margin="10,10,10,10" />
                </Viewbox>
            </Grid>
            <ControlTemplate.Triggers>
```

```
                <Trigger Property="IsMouseOver" Value="True">
                    <Setter TargetName="tło" Property="Fill"
                            Value="{StaticResource FioletowyGradient}" />
                    <Setter TargetName="tło" Property="BitmapEffect">
                        <Setter.Value>
                            <OuterGlowBitmapEffect GlowColor="Blue" GlowSize="5"/>
                        </Setter.Value>
                    </Setter>
                    <Trigger.EnterActions>
                        <BeginStoryboard>
                            <Storyboard>
                                <DoubleAnimation
                                    Storyboard.TargetName="obrót"
                                    Storyboard.TargetProperty = "Angle"
                                    Duration="0:0:2"
                                    From="0" To="90" />
                            </Storyboard>
                        </BeginStoryboard>
                    </Trigger.EnterActions>
                </Trigger>
            </ControlTemplate.Triggers>
        </ControlTemplate>
    </Application.Resources>
</Application>
```

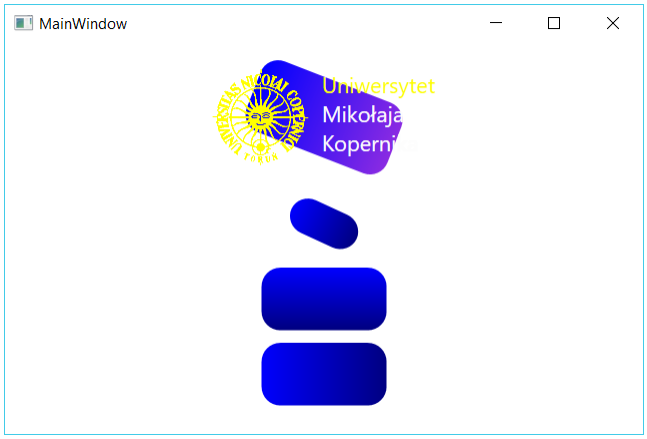
Aby użyć szablonu, należy do elementów przycisków dodać atrybut `Template` wskazujący na `SzablonNiebieski`, np.:

```
<Button Width="100" Height="50" Margin="0,0,0,10"
        Template="{StaticResource SzablonNiebieski}" />
```

Warto zaznaczyć różnice między szablonem kontrolki a stylem. Styl służy do ustalenia własności kontrolek i ich zmiany w pewnych sytuacjach (służą do tego wyzwalacze), natomiast szablon umożliwia zbudowanie kontrolki określonego typu całkowicie od nowa, a nawet określa sposób, w jaki jest rysowana. Zwróć uwagę, że szablon może być jedną z własności ustalanych w stylu.

Jak wspomniałem wyżej, szablon pozwala określić, jak ma wyglądać kontrolka, tzn. jak ma być narysowana. Z kolei tworzenie własnej kontrolki (typu *user control*) polega raczej na budowaniu nowej złożonej kontrolki z gotowych klocków (istniejących kontrolek), aby ułatwić jej wielokrotne używanie. Zatem jeżeli chcemy, żeby nasza kontrolka wyglądała nietypowo, to myślimy o szablonie kontrolki, a jeżeli chcemy, żeby kontrolka spełniała nietypowe zadania, budujemy własną kontrolkę. Oprócz omówionej już kontrolki użytkownika (ang. *user control*) w WPF można tworzyć kontrolki niestandardowe (ang. *custom control*). Poznasz je w kolejnym rozdziale. Są one w istocie standardowymi kontrolkami używającymi niestandardowych szablonów definiowanych przez ich twórcę.

Zastosowany w powyższym szablonie obrót samego prostokąta pełniącego funkcję tła (bez obracania zawartości przycisku, rysunek 13.1) jest dość efektowną, ale raczej niezbyt typową reakcją przycisku. Jeżeli chcielibyśmy obracać cały przycisk, należałoby przypisać transformację obrotu nie do prostokąta, lecz do elementu `Grid`, który obejmuje wszystkie elementy szablonu, i być może użyć transformacji renderowania zamiast transformacji kompozycji. Odpowiednie zmiany w kodzie szablonu pokazuje listing 13.3.



Rysunek 13.1. Obracające się tło przycisków

Listing 13.3. Obrót całego przycisku

```
<ControlTemplate x:Key="SzablonNiebieski" TargetType="{x:Type Button}">
   <Grid>
      <Grid.RenderTransform>
         <RotateTransform Angle="0" x:Name="obrót" />
      </Grid.RenderTransform>
...
```