

## 4. WPF - Animace

---

Díky vektorové grafice a skvělé implementaci animací ve WPF můžeme nyní vytvářet efekty, o kterých se nám ve WinForms mohlo pouze zdát. Vytváření animací je velice jednoduché, dokonce ve většině případů si vystačíme pouze s XAMLe! Zároveň si ale musíme dát pozor, aby jsme to s animacemi v aplikaci nepřehnali a nevytvořili spíše náročné a rušivé uživatelské rozhraní.

Ve WPF pracujeme s animacemi jinak než možná čekáte, nevytváříme časové osy, dokonce ani neanimujeme jednotlivé kontroly .... ve WPF animujeme vlastnosti (*properties*).

### Jak to funguje?

Chceme vytvořit animaci zvětšujícího se tlačítka. Takováto animace je pouze postupné zvyšování hodnot *Width* a *Height* ... a přesně to je animace ve WPF. Animace je v podstatě generátor hodnot, které jsou postupně přiřazovány některé z vlastností kontroly (animovat můžeme pouze vlastnosti typu *DependencyProperty*).

*Příklad:*

```
<Storyboard>
  <DoubleAnimation From="2" To="4"
    Duration="0:0:10"
    Storyboard.TargetProperty="Button.Width" />
</Storyboard>
```

Animace nastavuje hodnoty od 2 do 4 typu *double* v průběhu 10 sekund vlastnosti *Width* tlačítka.

*Tabulka vygenerovaných hodnot animace*

| 0 sec | 2 sec | 4 sec | 6 sec | 8 sec | 10 sec |
|-------|-------|-------|-------|-------|--------|
| 2     | 2.4   | 2.8   | 3.2   | 3.6   | 4      |

## Základní atributy

*From* - počáteční hodnota

*To* - koncová hodnota

*Duration* - délka animace, formát "hh:mm:ss"

- "0:0:5" = 5 sekund
- "0:4:5" = 4 minuty, 5 sekund
- "2:4:15" = 2 hodiny, 4 minuty, 15 sekund
- "0:0:0.5" = 1/2 sekundy

*AutoReverse* - pokud je *true*, provede se animace po dokončení obráceně

*RepeatBehavior* - počet opakování ("3x", "4x", "8x", ..., "Forever")

*Storyboard.TargetProperty* - animovaná vlastnost

*Storyboard.TargetName* - název cílového elementu

*\* Podle typu vlastnosti musíme použít odpovídající animaci, tzn. pokud animujeme vlastnost Width, která je typu double, musíme použít animaci vracející hodnoty typu double*

## Typy animací

Ve WPF máme k dispozici 3 typy animací - **základní**, **frame** a **path** animace. Jednotlivé animace se liší způsobem generování hodnot.

### Základní

Nejjednodušší. Zadáváme pouze počáteční a koncovou hodnotu a délku trvání animace. Jedná se o lineární generátor hodnot, tzn. že vygenerované hodnoty přechází postupně od počáteční po koncovou hodnotu.

- <datový typ>Animation (DoubleAnimation, Int32Animation, ColorAnimation...)
- zadáváme původní a koncovou hodnotu

### Frame animace

Narozdíl od základních animací můžeme u frame animací zadat více než dvě hodnoty - tzv. framy.

- <datový typ>AnimationUsingKeyFrames (DoubleAnimationUsingKeyFrames, ....)
- zadáváme množinu hodnot - framy

K dispozici máme 3 typy framů - *lineární*, *diskrétní* a *spline*. Liší se způsobem přechodu mezi jednotlivými framy.

**Lineární** - hodnoty se generují postupně (stejně jako u základních animací)

*Příklad:*

```
<Storyboard>
  <DoubleAnimationUsingKeyFrames Duration="0:0:10">
    <LinearDoubleKeyFrame Value="0" KeyTime="0:0:0" />
    <LinearDoubleKeyFrame Value="3" KeyTime="0:0:6" />
    <LinearDoubleKeyFrame Value="10" KeyTime="0:0:10" />
  </DoubleAnimationUsingKeyFrames>
</Storyboard>
```

*Tabulka vygenerovaných hodnot*

| 0 sec | 2 sec | 4 sec | 6 sec | 8 sec | 10 sec |
|-------|-------|-------|-------|-------|--------|
| 0     | 1     | 2     | 3     | 6.5   | 10     |

**Diskrétní** - nedochází k postupnému přechodu mezi vygenerovanými hodnotami

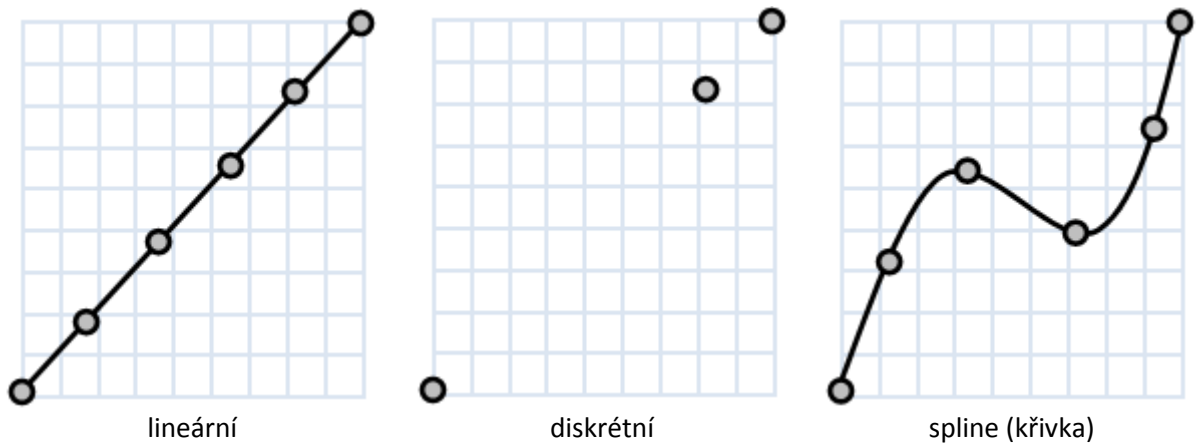
```
<Storyboard>
  <DoubleAnimationUsingKeyFrames Duration="0:0:10">
    <DiscreteDoubleKeyFrame Value="0" KeyTime="0:0:0" />
    <DiscreteDoubleKeyFrame Value="4" KeyTime="0:0:8" />
    <DiscreteDoubleKeyFrame Value="5" KeyTime="0:0:10" />
  </DoubleAnimationUsingKeyFrames>
</Storyboard>
```

*Tabulka vygenerovaných hodnot*

| 0 sec | 2 sec | 4 sec | 6 sec | 8 sec | 10 sec |
|-------|-------|-------|-------|-------|--------|
| 0     | 0     | 0     | 0     | 4     | 5      |

**Spline** - hodnoty jsou generovány podle křivky

## Grafy hodnot



## Path animace

Jedná se o speciální a zřídka používaný typ animace. Generuje hodnoty na základě zadané trajektorie.

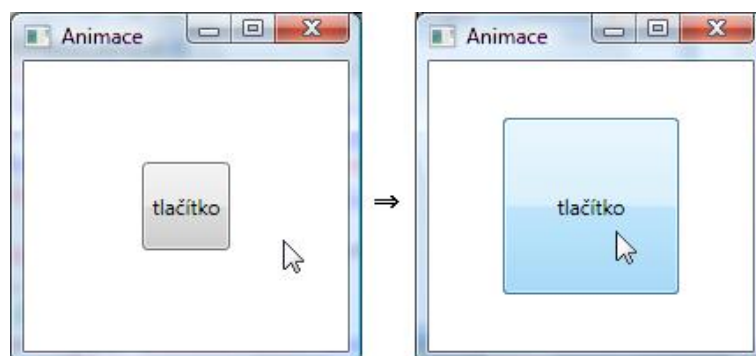
*\* Ve většině případů si vystačíme se základními nebo lineárními frame animacemi*

## Použití

Nejčastější způsob spuštění animace je použití trigru.

*Příklad:*

Po najetí myši nad tlačítko se tlačítko zvětší na dvojnásobnou velikost. Změna velikosti trvá 2 sekundy.



```

<Button Content="tlačítko" Width="50" Height="50">
  <Button.Triggers>
    <EventTrigger RoutedEvent="Button.MouseEnter">
      <EventTrigger.Actions>
        <BeginStoryboard>
          <Storyboard>
            <DoubleAnimation Storyboard.TargetProperty="Width"
                              To="100" Duration="0:0:2" />
            <DoubleAnimation Storyboard.TargetProperty="Height"
                              To="100" Duration="0:0:2" />
          </Storyboard>
        </BeginStoryboard>
      </EventTrigger.Actions>
    </EventTrigger>
  </Button.Triggers>
</Button>

```

Pro animaci vlastností *Width* a *Height* jsme použili 2 základní animace vracející hodnoty typu *double*

```

<DoubleAnimation Storyboard.TargetProperty="Width"
                  To="100" Duration="0:0:2" />
<DoubleAnimation Storyboard.TargetProperty="Height"
                  To="100" Duration="0:0:2" />

```

Animace trvá 2 sekundy

```
Duration="0:0:2"
```

Koncová šířka i výška tlačítka je 100

```
To="100"
```

Atribut *From* jsme vynechali, jako počáteční hodnota animace je tedy použita aktuální hodnota vlastnosti *Width* nebo *Height*.

Chceme animaci spustit od začátku

```
<BeginStoryboard> ... </BeginStoryboard>
```

Obě animace jsou spuštěny v reakci na událost *Button.MouseEnter*, tzn. po najetí myši nad tlačítko

```

<Button.Triggers>
  <EventTrigger RoutedEvent="Button.MouseEnter">
    <EventTrigger.Actions>
      ...
    </EventTrigger.Actions>
  </EventTrigger>
</Button.Triggers>

```

V tomto příkladu jsme animaci spouštěli, máme ale i jiné možnosti co s animací provést.

*spustit animaci*

```
<BeginStoryboard> ... </BeginStoryboard>
```

*pozastavit běžící animaci*

```
<PauseStoryboard> ... </PauseStoryboard>
```

*pokračovat v pozastavené animaci*

```
<ResumeStoryboard> ... </ResumeStoryboard>
```

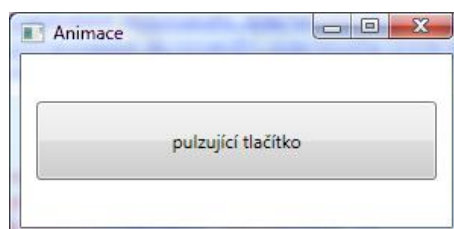
*zastavit animaci*

```
<StopStoryboard> ... </StopStoryboard>
```

## Animace ve stylech

*Příklad:*

Vytvořte "pulzující" tlačítko, tzn. tlačítko, které v krátkých intervalech zvětšuje a zmenšuje svou výšku. Dále po najetí myši nad tlačítko se animace pozastaví a naopak po opuštění tlačítka zase pokračuje.



```

<Grid>
  <Grid.Resources>
    <Style TargetType="Button">
      <Setter Property="Height" Value="25" />
      <Setter Property="Margin" Value="10" />
      <Style.Triggers>
        <!-- Spustit animaci hned po startu -->
        <EventTrigger RoutedEvent="Button.Loaded">
          <EventTrigger.Actions>
            <BeginStoryboard x:Name="ButtonAnimation">
              <Storyboard>
                <DoubleAnimation From="25" To="80"
                                 AutoReverse="True"
                                 RepeatBehavior="Forever"
                                 Storyboard.TargetProperty="Height"
                                 Duration="0:0:1" />
              </Storyboard>
            </BeginStoryboard>
          </EventTrigger.Actions>
        </EventTrigger>
        <!-- Pozastavit animaci -->
        <EventTrigger RoutedEvent="Button.MouseEnter">
          <EventTrigger.Actions>
            <PauseStoryboard BeginStoryboardName="ButtonAnimation" />
          </EventTrigger.Actions>
        </EventTrigger>
        <!-- Pokracovat v animaci -->
        <EventTrigger RoutedEvent="Button.MouseLeave">
          <EventTrigger.Actions>
            <ResumeStoryboard BeginStoryboardName="ButtonAnimation" />
          </EventTrigger.Actions>
        </EventTrigger>
      </Style.Triggers>
    </Style>
  </Grid.Resources>
  <Button Content="pulzující tlačítko" />
</Grid>

```

spuštění  
animace

pozastavení  
animace

pokračování  
animace

1. vytvořili jsme obecný styl, který je použit na každé tlačítko v *Gridu*
2. nadefinovali jsme dva *Settery* nastavující výšku a okraj tlačítka
3. vytvořili jsme 3 trigry
  - 1. trigr spustí animaci pulzujícího tlačítka hned po vytvoření tlačítka. Pulzující = nastavujeme hodnotu Height vlastnosti (*TargetProperty=Height*) na 100px (*From=25, To=100*), po ukončení se animace provede obráceně (*AutoReverse=True*) a tento postup se neustále opakuje (*RepeatBehavior=Forever*). Abychom s touto animací mohli pracovat i v jiných trigrech, pojmenovali jsme si ji (*x:Name=ButtonAnimation*)
  - 2. trigr reaguje jakmile myš najedeme nad tlačítko (*RoutedEvent=Button.MouseEnter*) a pozastaví animaci (*PauseStoryboard*) se jménem "ButtonAnimation"

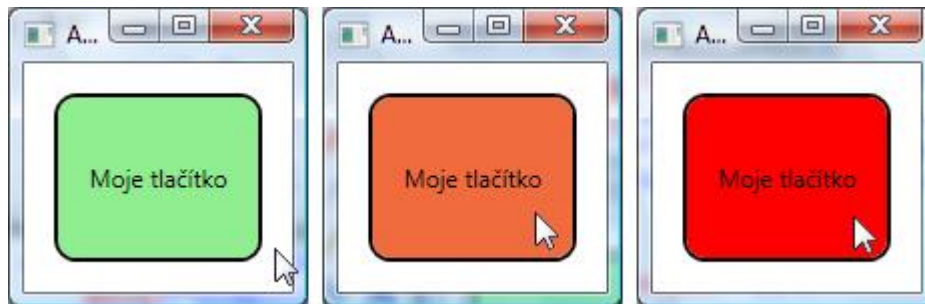
- 3. trigr reaguje jakmile myš opustí prostor tlačítka (*RoutedEvent=Button.MouseLeave*) a pokračuje v animaci (*ResumeStoryboard*) se jménem "ButtonAnimation" od místa kde byla pozastavena

## Animace v šablonách

Nakonec si ukážeme jak můžeme používat animace v šablonách.

*Příklad:*

Přepíšeme šablonu tlačítka a vytvoříme nový efekt - po kliknutí na tlačítko se během 2 sekund změni barva pozadí ze světle zelené na červenou.



1. obrázek: před kliknutím
2. obrázek: 0.5 sekundy po kliknutí
3. obrázek: > 1 sekunda po kliknutí

```
<Button>
  <Button.Template>
    <ControlTemplate>
      <!-- Template -->
      <Grid>
        <Border CornerRadius="10"
          BorderBrush="Black" BorderThickness="2">
          <Border.Background>
            <SolidColorBrush x:Name="myBorderBackground"
              Color="LightGreen" />
          </Border.Background>
          <Label HorizontalAlignment="Center"
            VerticalAlignment="Center" Content="Moje tlačítko" />
        </Border>
      </Grid>
      <!-- Trigger -->
      <ControlTemplate.Triggers>
        <EventTrigger RoutedEvent="Button.Click">
          <EventTrigger.Actions>
            <BeginStoryboard>
              <Storyboard>
                <ColorAnimation To="Red" Duration="0:0:1"
                  Storyboard.TargetName="myBorderBackground"
                  Storyboard.TargetProperty="Color" />
              </Storyboard>
            </BeginStoryboard>
          </EventTrigger.Actions>
        </EventTrigger>
      </ControlTemplate.Triggers>
    </ControlTemplate>
  </Button.Template>
</Button>
```

```
</EventTrigger>
</ControlTemplate.Triggers>
</ControlTemplate>
</Button.Template>
</Button>
```

Nemohli jsme použít *ColorAnimation* přímo na vlastnost *Background*, protože je typu *Brush* a *ColorAnimation* vrací hodnoty typu *Color*. Proto jsme uvnitř *Border.Background* vytvořili a pojmenovali *SolidColorBrush*, jehož vlastnost *Color* už animovat můžeme.

*\* Druhá možnost je animovat vlastnost Background kontroly tímto způsobem přístupu k vlastnosti:  
Storyboard.TargetProperty="(Border.Background).(SolidColorBrush.Color)"*

## Závěr

Díky animacím můžeme v aplikacích vytvářet mnoho zajímavých efektů, měli bychom ale zároveň brát na vědomí, že musíme být v používání animací opatrní, nejen že přehnané množství by mohlo působit rušivě, ale také čím více animací použijeme, o to víc bude naše aplikace náročnější.

Více informací o WPF naleznete na [www.unitedstatesof.net/wpf](http://www.unitedstatesof.net/wpf)

*Aleš Šturala, 30. 1. 2007*

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