Module 10
WPF Custom Controls

MGR. TOMÁŠ HAVETTA - MCT

Obsah

- Tvorba vlastních Controls
- Možnosti jak vytvořit Custom Control
- Visual States
- WPF a Windows Forms

Blok 1: Tvorba vlastních Controls

- Proč vytvářet nový vlastní control
- Varianty pro tvorbu k dispozici
- Implementace User Controls
- Implementace Custom Controls
- Dědění z FrameworkElement třídy

Proč vytvářet nový vlastní control

- Když nestačí
  - WPF controls
  - Styly
  - Control templates
  - Triggers
  - Data templates
- Typicky při potřebě kombinace základních controls do složitějšího celku

Varianty pro tvorbu k dispozici

- Rodič může být
  - UserControl – triviální vytvoření
  - Control – složitější, ale flexibilní vzhled
  - FrameworkElement – renderování pod kontrolou

UserControl

Control

FrameworkElement

Implementace User Controls

- Tvorba pomocí XAML
- K dispozici vše co WPF umí
- Zvolit v případě
  - Vytváření návrhu vzhledu aplikace
  - Control pro jednorázové použití
  - Nedostatek času nebo programátorů

Implementace Custom Controls

- Chování se programuje v kódu
- Vzhled se vytváří v XAML jako Control Template
- Pro různé Themes se může udělat různý template
- Kdy vytvářet Custom Control
  - Chcete umožnit modifikaci vzhledu prvku v jiných projektech
  - Potřebujete zajistit různý vzhled na jiných platformách
  - Tvoříte control pro další prodej jako vývojová komponenta

Dědění z FrameworkElement třídy

- Podpora přímého renderingu
- Vlastní řízení umístění obsahu
- Kdy zvolit
  - Nemáte co dělat za dlouhých zimních večerů
  - Potřebujete přesně řídit vzhled
  - Potřebujete speciální logiku renderování
  - Výsledek nelze dosáhnout ničím co máte k dispozici ve WPF

Blok 2: Možnosti jak vytvořit Custom Control

- UserControl třída
- Implementace vlastností a událostí
- Vytvoření Custom Control
- Použití Commands
- Definování vzhledu pomocí Themes

UserControl třída

- Přidat do projektu UserControl XAML
- Pomocí standardních controls vytvořit UI
- V codebehind implementovat chování

```
<UserControl x:Class="MyNamespace.NumericUpDown" ...>
    <Grid ...>
        <TextBlock .../>
        <RepeatButton ...>Up</RepeatButton>
```

```
namespace MyNamespace
{
    public class NumericUpDown : UserControl
    {
        ...
```

Implementace vlastností a událostí

- Vlastnosti řešit jako DependencyProperty
- Události řešit pomocí Routed Eventů

```
public static readonly DependencyProperty ValueProperty
  = DependencyProperty.Register("Value", ...);

public decimal Value
{
  get { return (decimal)GetValue(ValueProperty); }
  set { SetValue(ValueProperty, value); }
}
```

```
public static readonly RoutedEvent ValueChangedEvent
  = EventManager.RegisterRoutedEvent("ValueChanged", ...);
```

Vytvoření Custom Control

- Přidat do projektu obyčejný class, zadat dědičnost z Control
- Vzhled definovat pomocí defaultního stylu v Resources

```
namespace MyNamespace
{
    public class NumericUpDown : Control {...}
    ...
```

```
<Application
  xmlns:local="clr-namespace:MyNamespace" ...>
  <Application.Resources>
    ...
    <ControlTemplate
      TargetType="{x:Type local:NumericUpDown}">
      <Grid>
        ...
```

Použití Commands

- Reakci na uživatele nejde řešit pomocí Event handleru
- RoutedCommand nabízí cestu k vytvoření vazby

```
public class NumericUpDown : Control
{
  public static RoutedCommand IncreaseCommand;
  public static RoutedCommand DecreaseCommand;
    ...
```

Defined in the template of a Style element

```
<RepeatButton
  Command="{x:Static local:NumericUpDown.IncreaseCommand}"
    ...>Up</RepeatButton>
<RepeatButton
  Command="{x:Static local:NumericUpDown.DecreaseCommand}"
    ...>Down</RepeatButton>
```

Definování vzhledu pomocí Themes

- Vytvořit v projektu složku themes
- Vytvořit ve složce generic.xaml
- Vytvořit ResourceDictionary se Style elementem
- Specifikovat umístění theme pro hostující aplikaci

Defined in generic.xaml

```
<ResourceDictionary ...>
  <Style TargetType="{x:Type local:NumericUpDown}">
    <ControlTemplate TargetType="{x:Type ...}">
      ...
```

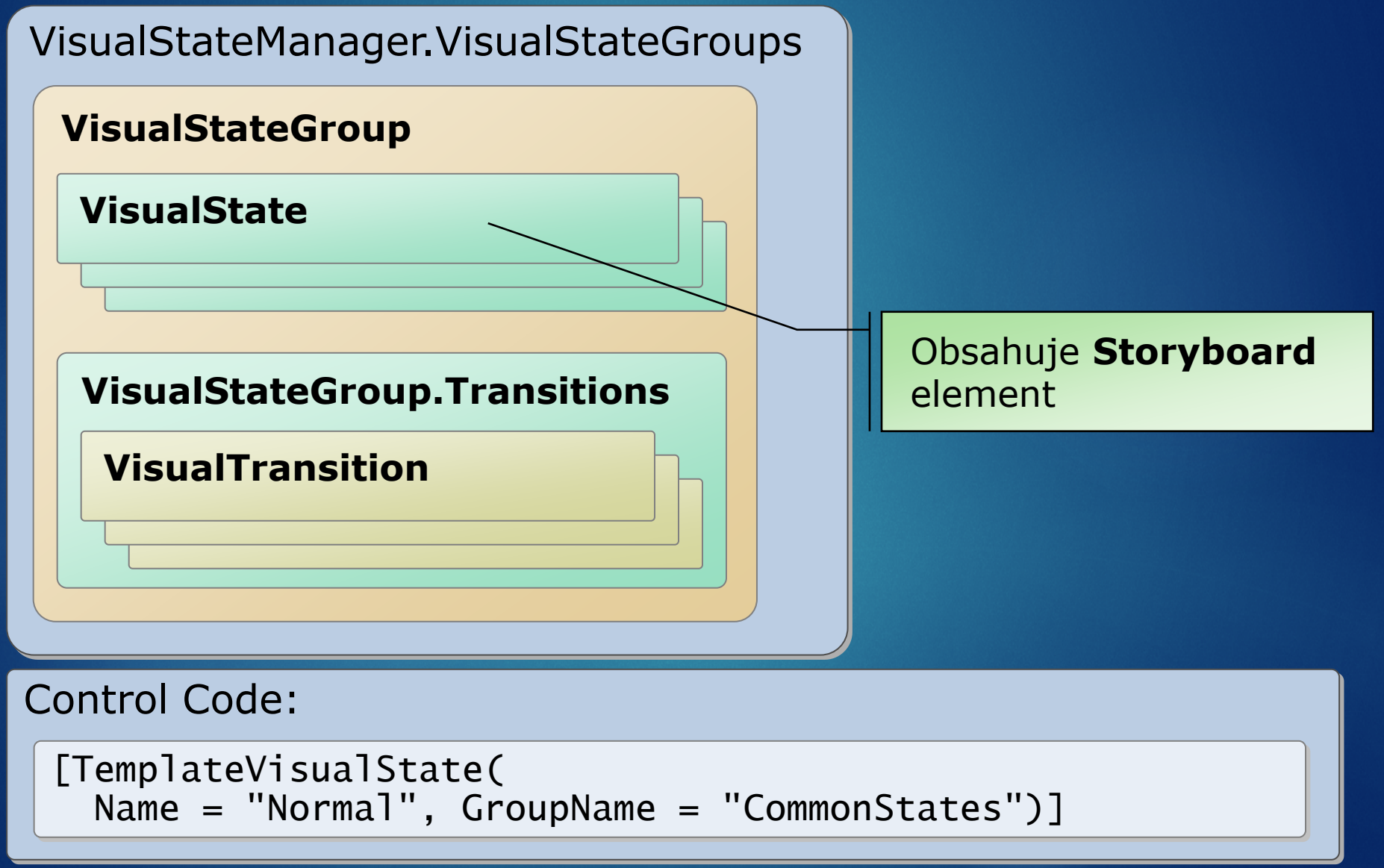
In the hosting application

```
[assembly: ThemeInfo(
  ResourceDictionaryLocation.None,
  ResourceDictionaryLocation.SourceAssembly)]
```

Blok 3: Visual States

- VisualStateManager třída
- Implementace vizuálních stavů
- Změna stavu

VisualStateManager třída



Control Code:

```
[TemplateVisualState(
  Name = "Normal", GroupName = "CommonStates")]
```

Implementace vizuálních stavů

```
<VisualStateManager.VisualStateGroups>
  <VisualStateGroup Name="CommonStates">
    <VisualState Name="Normal" />
    <VisualState Name="MouseOver">
      <Storyboard>
        <ColorAnimation To="Green"
                        Storyboard.TargetName="rectBrush"
                        Storyboard.TargetProperty="Color"/>
      </Storyboard>
    </VisualState>
    <VisualStateGroup.Transitions>
      <VisualTransition To="Normal"
                        GeneratedDuration="00:00:00"/>
      <VisualTransition To="MouseOver"
                        GeneratedDuration="00:00:00.5">
        <VisualTransition.GeneratedEasingFunction>
          <ExponentialEase EasingMode="EaseOut" Exponent="10"/>
        </VisualTransition.GeneratedEasingFunction>
      </VisualTransition>
    </VisualStateGroup.Transitions>
  </VisualStateGroup>
</VisualStateManager.VisualStateGroups>
```

Změna stavu

- V XAML se definují přechody mezi stavy
- Vlastnosti From a To
  - Obě nastavené – přechod mezi konkrétními stavy
  - Jen From – přechod z daného stavu na jakýkoliv
  - Jen To – přechod z jakéhokoliv stavu na daný stav
  - Ani From ani To – Defaultní pro nedefinované přechody
- Změna z kódu:

```
VisualStateManager.GoToState(this, "MouseOver", true);
```

Blok 4: WPF a Windows Forms

- WPF a Windows Forms integrace
- Hostování WF prvku ve WPF
- Hostování WPF controlu ve WF

WPF a Windows Forms integrace

- Proč je míchat
  - WF nemá dobrou podporu pro grafiku
  - Existují WF controls, které neexistují ve WPF
  - Umožňuje to postupnou migraci z WF do WPF

Hostování WF prvku ve WPF

System.Windows.Forms and **WindowsFormsIntegration** assemblies

```
<Window ... >
  ...
    <WindowsFormsHost x:Name="wfh">
      <wf:MaskedTextBox x:Name="mtb" Mask="00/00/000" />
    </WindowsFormsHost>
  ...
</Window>
```

WindowsFormsHost element

- Child property
- Cast to relevant type
- Attach event handlers
- Manipulate properties

```
(this.wfh.Child as MaskedTextBox).ValueChanged +=
  new EventHandler(this.MaskedTextBox_ValueChanged);
```

Hostování WPF controlu ve WF

WindowsFormsIntegration assembly

```
private void MyForm_Load(object sender, EventArgs e)
{
  elemHost = new ElementHost();
  ...

  System.Windows.Controls.Button wpfButton =
    new System.Windows.Controls.Button();
  wpfButton.Content = "Windows Presentation Foundation Button";
  elemHost.Child = wpfButton;

  // Map the Margin property.
  this.AddMarginMapping();
  // Remove the mapping for the Cursor property.
  this.RemoveCursorMapping();
  // Cause the OnMarginChange delegate to be called.
  elemHost.Margin = new Padding(23, 23, 23, 23);
}
```

Lab: Custom Control