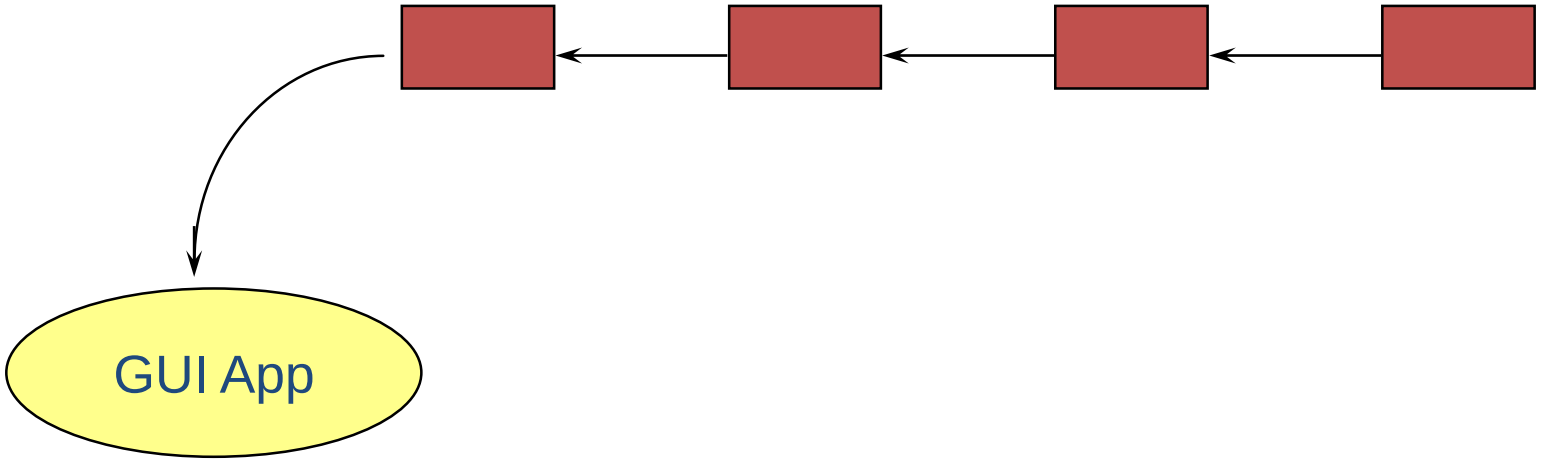
GUI

Graphical User Interface

KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Udalostne riadená aplikácia

- Myšlienka je veľmi jednoduchá
  - Akcie užívateľa v GUI sú do programu prenášané udalosťami
  - Udalosti sú spracovávané postupne



  - Na tomto princípe funguje väčšina GUI

GUI udalosti

- Udalosti z myši
  - Pohyb myši (Mouse move)
  - Klik tlačítka myši (Mouse click)
  - Double-click myši (Mouse double-click)
- Klávesnica
  - Key down
  - Key up
- Udalosti z okna
  - Stlačenie tlačítka v okne (Button click)
  - Výber položky menu (Menu selection)
  - Zmena aktívneho prvku v okne (Change in focus)
  - Okno do popredia – aktivácia (Window activation)
- atď.

GUI

- Native C++
  - MFC – MS native
  - OWL – Borland
  - QT – Multiplatform (pôvod unix)

- .NET (C#, C++ manag., VB, ...)
  - WinForms – rýchly vývoj, MS Windows look&feel
  - WPF – možnosť vlastných návrhov, databinding

Čo sú WinForms?

- GUI aplikácia je založená na tzv. form a control
- Form:
  - Je celé okno (s rámikom) reprezentované triedou Form
  - Obsahuje 0 a viac Controls
- Control:
  - Je ucelený grafický komponent reprezentovaný triedou Control
  - Môže obsahovať ďalšie Controls
  - Vývojár môže vytvoriť potomka a dodefinovať vzhľad a vlastnosti

Ako modifikovať vzhľad a vlastnosti?

- Form a Control obsahuje property
- Upravujú vzhľad a vlastnosti
- Zmeny sa prejavujú okamžite
- Napr. pre Form:
  - `AutoScroll`
  - `BackgroundImage`
  - `ControlBox`
  - `FormBorderStyle` (sizable?)
  - `Icon`
  - `Location`
  - `Size`
  - `StartPosition`
  - `Text` (i.e. window's caption)
  - `WindowState` (minimized, maximized, normal)

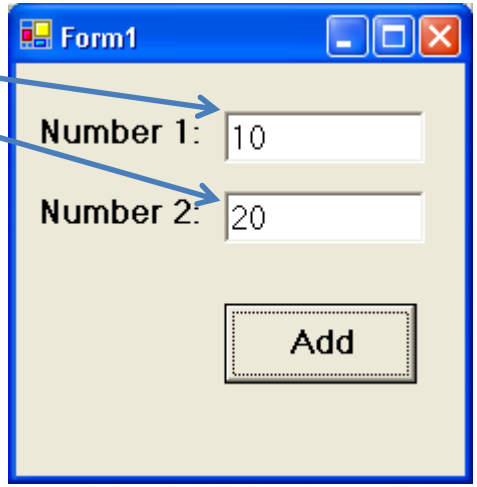
```
Form1  form;
form = new Form1();
form.WindowState = FormWindowState.Maximized;
form.Show();
```

Základné metódy a udalosti Form

- Základné akcie, ktoré môžeme s oknom urobiť
  - `Activate`: aktivácia okna (získa focus)
  - `Close`: zatvorenie a uvoľnenie zdrojov
  - `Hide`: skryť, ale objekt so zdrojmi existuje
  - `Refresh`: prekreslenie (aktualizácia obsahu)
  - `Show`: zobrazenie a aktivácia nemodálne
  - `ShowDialog`: zobrazenie a aktivácia modálne

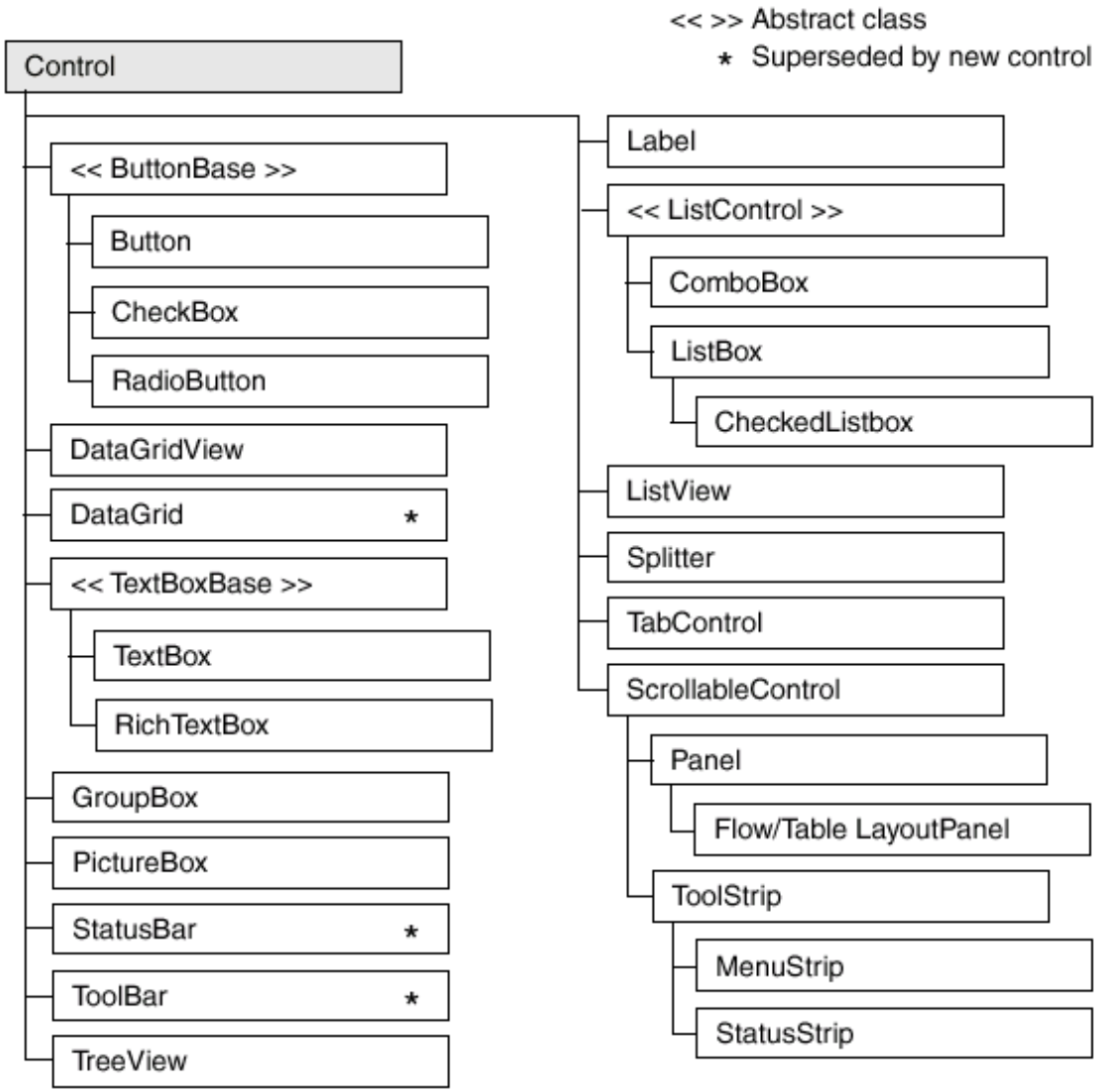
- Základné udalosti okna
  - `Load`: nastane tesne pred prvým zobrazením okna
  - `Closing`: nastane tesne pred zatvorením (dá sa zatvorenie zamietnuť)
  - `Closed`: nastane pri definitívnom zatvorení okna
  - `Resize`: nastane pri zmene veľkosti okna
  - `Click`: nastane pri kliku na pozadie okna (nie na control)
  - `KeyPress`: nastane keď je aktívne okno a užívateľ stlačí klávesu

Najbežnejšie Controls



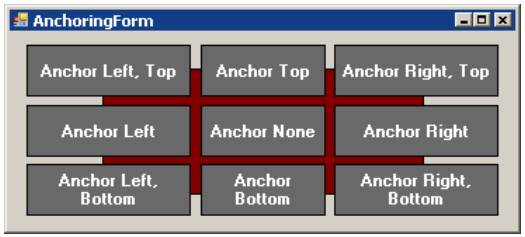
- TextBox – veľmi často používaný
  - Zobrazovanie textov, vkladanie dát do aplikácie
  - Dôležité property
    - `Text`: celý obsah textového poľa (string)
    - `Modified`: bol text v poli modifikovaný?
    - `ReadOnly`: je textové pole iba na čítanie?
  - Property pre viacriadkový text-box
    - `MultiLine`: True dovoľuje do poľa vkladať viac riadkov
    - `Lines`: pole reťazcov (jeden pre každý riadok)
    - `ScrollBars`: none, horizontal, vertical, or both
  - Dôležité udalosti
    - `Enter`, `Leave`: nastane pri aktivácií/deaktivácií (focus)
    - `KeyPress`: nastane pri stlačení ascii znaku
    - `KeyDown`, `KeyUp`: nastane pri stlačení ľubovaľnej kombinácie kláves
    - `TextChanged`: nastane vždy, keď sa text zmení
    - `Validating`, `Validated`: kontrola vstupných hodnôt

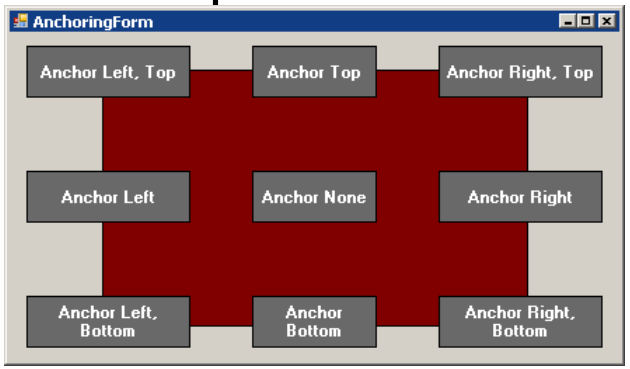
Windows Forms control hierarchy



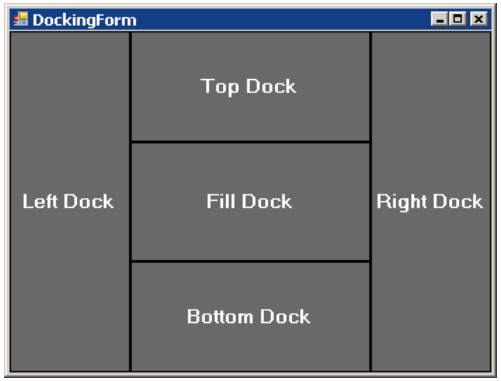
Docking/Anchoring

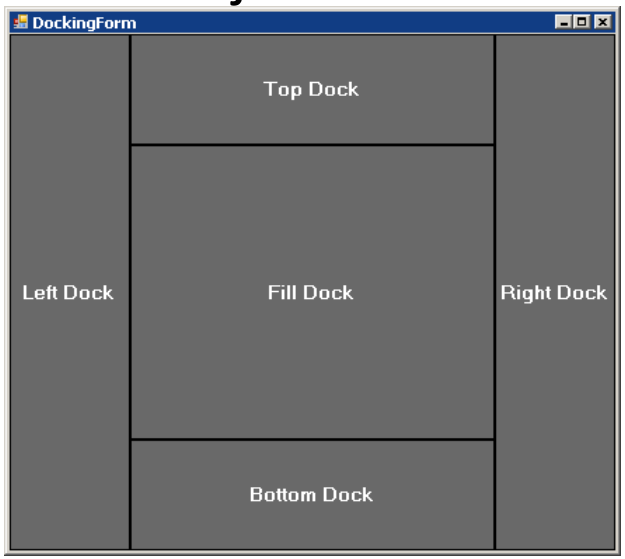
- Ako sa bude chovať Control pri zmene veľkosti okna
- Anchoring – zakotvenie hrany na relatívnu pozíciu v okne





- Docking – prilepenie Control ku danému okraju okna





Pokročilé objektové technológie

WPF

Windows Presentation Foundation

KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Ako tvoriť GUI

- Aplikačná logika priamo v GUI
  - Rýchly vývoj
  - Výmena dát: metódy, udalosti (events)
  - Vhodné pre malé projekty
- Aplikačná logika oddelená od GUI
  - Vyššia počiatočná réžia
  - Výmena dát: binding, udalosti
  - Pre veľké projekty prehľadnejšie
  - Podpora unit testov
  - Znovupoužiteľnosť aplikačnej logiky (klient, server, ...)

Čo je WPF

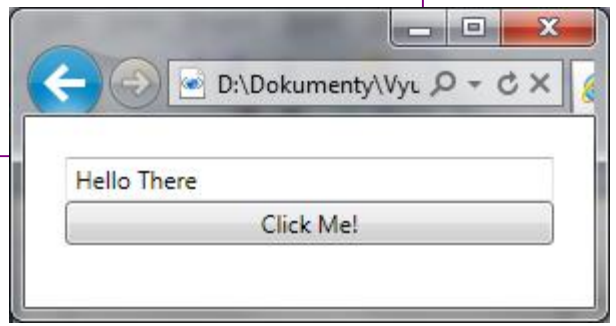
- Pôvodné (CodeName) označenie „Avalon“
- Nový spôsob tvorby GUI
- Hardvérovo renderované GUI
- Založený na vektorovej grafike
- Tvorba GUI ako WWW stránku
  - XAML – značkovací jazyk
- Oddelenie programátora od dizajnéra
  - Každý má iné vývojové prostredie
  - Podpora návrhových vzorov MVC MVVM
- Prepracovaný Databinding (predchodca vo WinForm)

Extensible Application Markup Language

- XML Elementy predstavujú prvky (Controls)
  - <Canvas /> objekt triedy Canvas
  - <Window /> objekt triedy Window
  - <Button /> objekt triedy Button
  - <TextBox /> objekt triedy TextBox


```
<Canvas xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
  <Button>Click Me!</Button>
</Canvas>
```

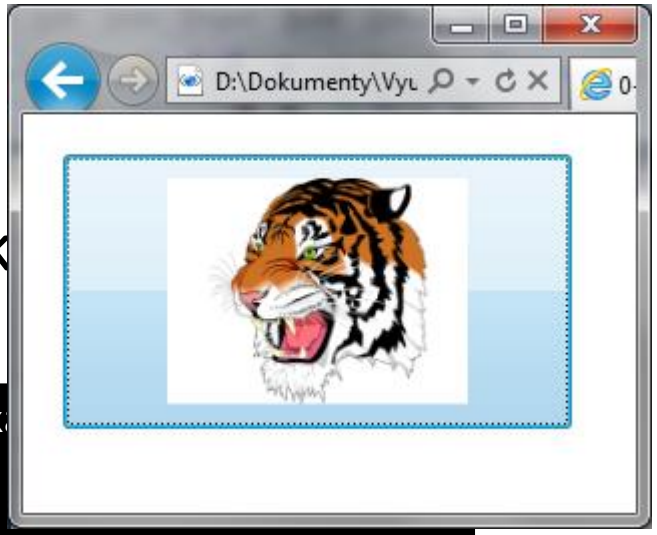
```
<StackPanel xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
  <TextBox>Hello There</TextBox>
  <Button>Click Me!</Button>
</StackPanel>
```



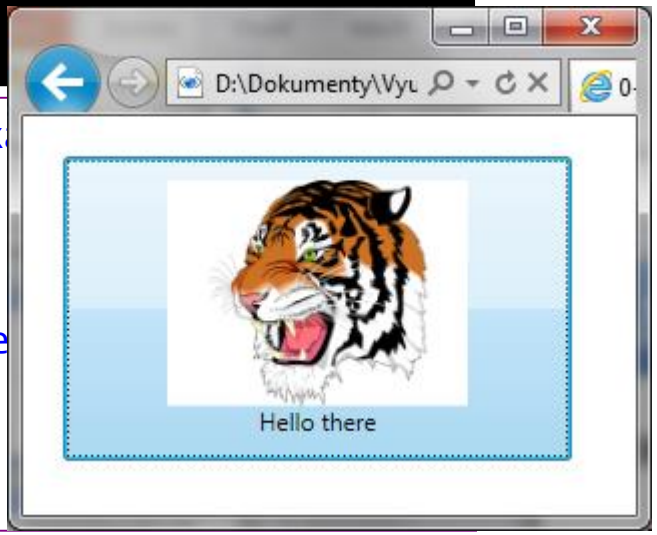
XAML – kompozícia

- Kompozitné prvky
  - Jednoduchý hierarchický model
  - Prvok môže obsahovať ďalšie prvk
    podľa typu prvku)

```
<Canvas xmlns="http://schemas.microsoft.com/winfx/2006/x
    <Button Margin="20" Padding="50 10">
        <Image Source="Images/tiger.png" Width="150" />
    </Button>
</Canvas>
```




```
<Canvas xmlns="http://schemas.microsoft.com/winfx/2006/x
  <Button Margin="20" Padding="50 10">
      <StackPanel>
          <Image Source="Images/tiger.png" Width="150" />
          <TextBlock TextAlignment="Center">Hello there</Te
      </StackPanel>
  </Button>
</Canvas>
```



XAML – kontajnerové prvky

- Kontajnerové prvky
  - Canvas: špeciálne umiestnovanie
  - StackPanel: horizontálne alebo vertikálne ukladanie
  - DockPanel: dokovanie ku hranám
  - Grid: zarovnávanie prvkov ku pomyselným líniám
  - UniformGrid: umiestňovanie do matice
  - TextFlow: zarovnávanie ako tok textu
- Používané ako hlavné prvky, alebo dcérske prvky (ak chceme vložiť viac prvkov)

XAML – nastavovanie property

- Atribútová syntax (jednoduchšie a prehľadnejšie)

```
<TextBlock TextAlignment="Center" Text= "Ahoj"/>
```

- Element syntax (pre zložité atribúty)

```
<TextBlock>
    <TextBlock.TextAlignment>
        Center
    </TextBlock.TextAlignment>
    <TextBlock.Text>
        Ahoj
    </TextBlock.Text>
</TextBlock>
```

Content atribút

- Content atribút (ak prvok obsahuje **ContentProperty**)
  - Property element syntax

```
<Button>
    Click me!
</Button>
```

  - Attribute syntax

```
<Button Content="Click me!">
```

Content atribút - kolekcia

- Implicitná kolekcia (ak je content typu kolekcia)

```
<ListBox>
    <ListBox.Items> <!– ListBox.Items môžeme vynechať -->
        <ListBoxItem Content="aaa" />
        <ListBoxItem Content="bbb" />
    </ListBox.Items>
</ListBox>
```

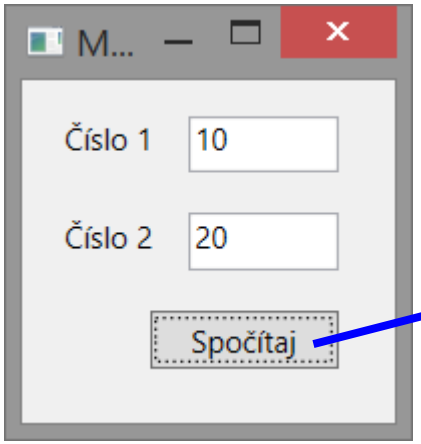
- Bez atribútu kolekcie

```
<ListBox>
    <ListBoxItem Content="aaa" />
    <ListBoxItem Content="bbb" />
</ListBox>
```

Code-behind

- Pre každú udalosť môžeme definovať obslužný algoritmus – kód na pozadí (code behind)
- Práca dizajnéra – navrhovať okná
- Práca programátora – tvoriť kód na pozadí

```
<Window x:Class="POT2016_WPF_code_behind.MainWindow" xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Background="{x:Static SystemColors.ControlBrush}" Title="MainWindow" Height="174" Width="163">
    <Canvas>
        <Label Content="Číslo 1" Height="28" Canvas.Left="12" Canvas.Top="10" />
        <Label Content="Číslo 2" Height="28" Canvas.Left="12" Canvas.Top="49" />
        <TextBox Height="23" Name="textBox1" VerticalAlignment="Top" Width="60" Canvas.Left="66" Canvas.Top="14" Text="0"/>
        <TextBox Height="23" Name="textBox2" VerticalAlignment="Top" Width="60" Canvas.Left="66" Canvas.Top="53" Text="0"/>
        <Button Content="Spočítaj" Height="23" Width="75" Click="button1_Click" Canvas.Left="51" Canvas.Top="92" />
    </Canvas>
</Window>
```

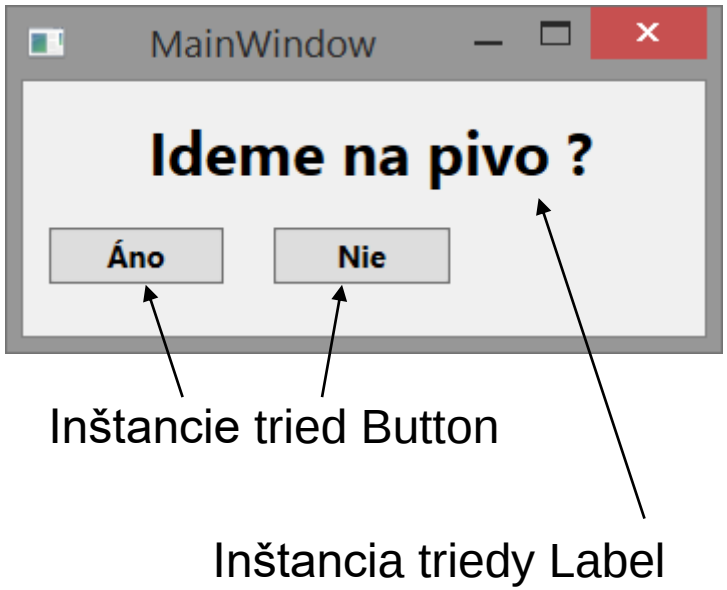



```
public partial class MainWindow : Window
{
  public MainWindow()
  {
     InitializeComponent();
  }

  private void button1_Click(object sender, RoutedEventArgs e)
  {
    int a = Convert.ToInt32(this.textBox1.Text);
    int b = Convert.ToInt32(this.textBox2.Text);
    int c = a + b;
    MessageBox.Show("Sum = " + c);
  }
}
```

Code-behind – dynamika

- Okamžitá zmena vzhľadu programovo

```
<Window x:Class="POT_WPF_code_behind2.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Background="{x:Static SystemColors.ControlBrush}" Title="MainWindow" Height="139" Width="286">
    <Canvas>
        <Label Canvas.Left="45" Canvas.Top="7" Content="Ideme na pivo ?" Name="label1" FontWeight="Bold" FontSize="24" />
        <Button Canvas.Left="10" Canvas.Top="58" Content="Áno" Height="23" Name="button1" Width="70" FontWeight="Bold" />
        <Button Canvas.Left="100" Canvas.Top="58" Content="Nie" Height="23" Name="button2" Width="70" FontWeight="Bold" />
    </Canvas>
</Window>
```

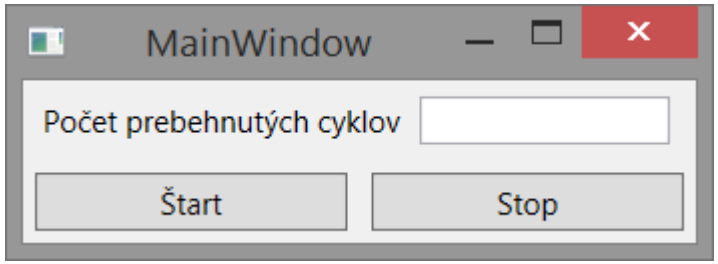



```
public partial class MainWindow : Window
{
  public MainWindow()
  {
    InitializeComponent();
    button1.MouseEnter += new MouseEventHandler(button1_MouseEnter);
    button2.Click += new RoutedEventHandler(button1_Click);
  }

  void button1_MouseEnter(object sender, MouseEventArgs e)
  {
    var x = Canvas.GetLeft(button1);
    Canvas.SetLeft(button1, x < 50 ? 190 : 10);
  }

  void button1_Click(object sender, RoutedEventArgs e)
  {
    MessageBox.Show("Tak seď a dávaj pozor!");
  }
}
```

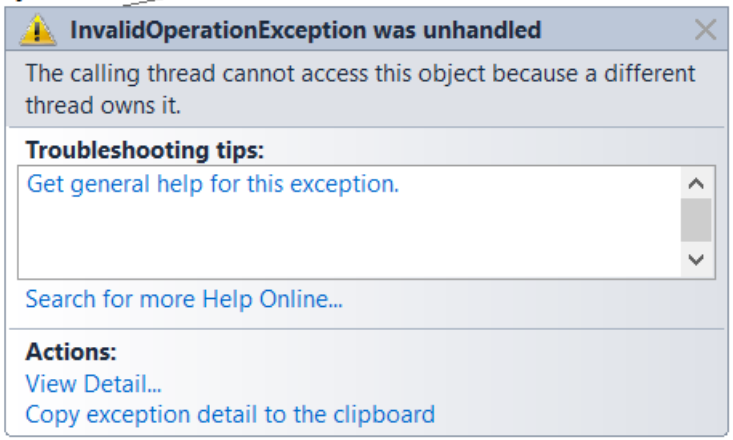
Viacvláknové aplikácie

- GUI beží v jednom vlákne (UIthread)
- GUI nedovolí modifikovať elementy z iných vlákien (končí výnimkou)

```
public partial class MainWindow : Window
{
  public MainWindow()
  {
    InitializeComponent();
  }
  private Thread aThread;

  private void button1_Click(object sender, RoutedEventArgs e)
  {
    if (aThread == null)
    {
      aThread = new Thread(Pocitanie) { IsBackground = true };
      aThread.Start();
    }
  }
  private void button2_Click(object sender, RoutedEventArgs e)
  {
    if (aThread != null)
    {
      aThread.Abort();
      aThread = null;
    }
  }
  private void Pocitanie()
  {
    for (long i = 0; i < long.MaxValue; i++)
    {
      double d = Math.Sin(i);
      if (i % 10 == 0) textBox1.Text = i.ToString();
    }
  }
}
```




```
private void Pocitanie()
{
  for (long i = 0; i < long.MaxValue; i++)
  {
    double d = Math.Sin(i);
    if (i % 10 == 0) textBox1.Text = d.ToString();
  }
}
```



Viacvláknové aplikácie - preposlanie

- Riešenie – preposlať udalosť do **UIthread**
- Využiť Dispatcher

```
private void Pocitanie()
{
  for (long i = 0; i < long.MaxValue; i++)
  {
    double d = Math.Sin(i);
    if (i % 10 == 0) AktualizujTextBox1(i.ToString());
  }
}

private void AktualizujTextBox1(string txt)
{
  if (this.CheckAccess()) textBox1.Text = txt;
  else
  {
    this.Dispatcher.Invoke(DispatcherPriority.Background,
              new Action<string>(AktualizujTextBox1), txt);
  }
}
```

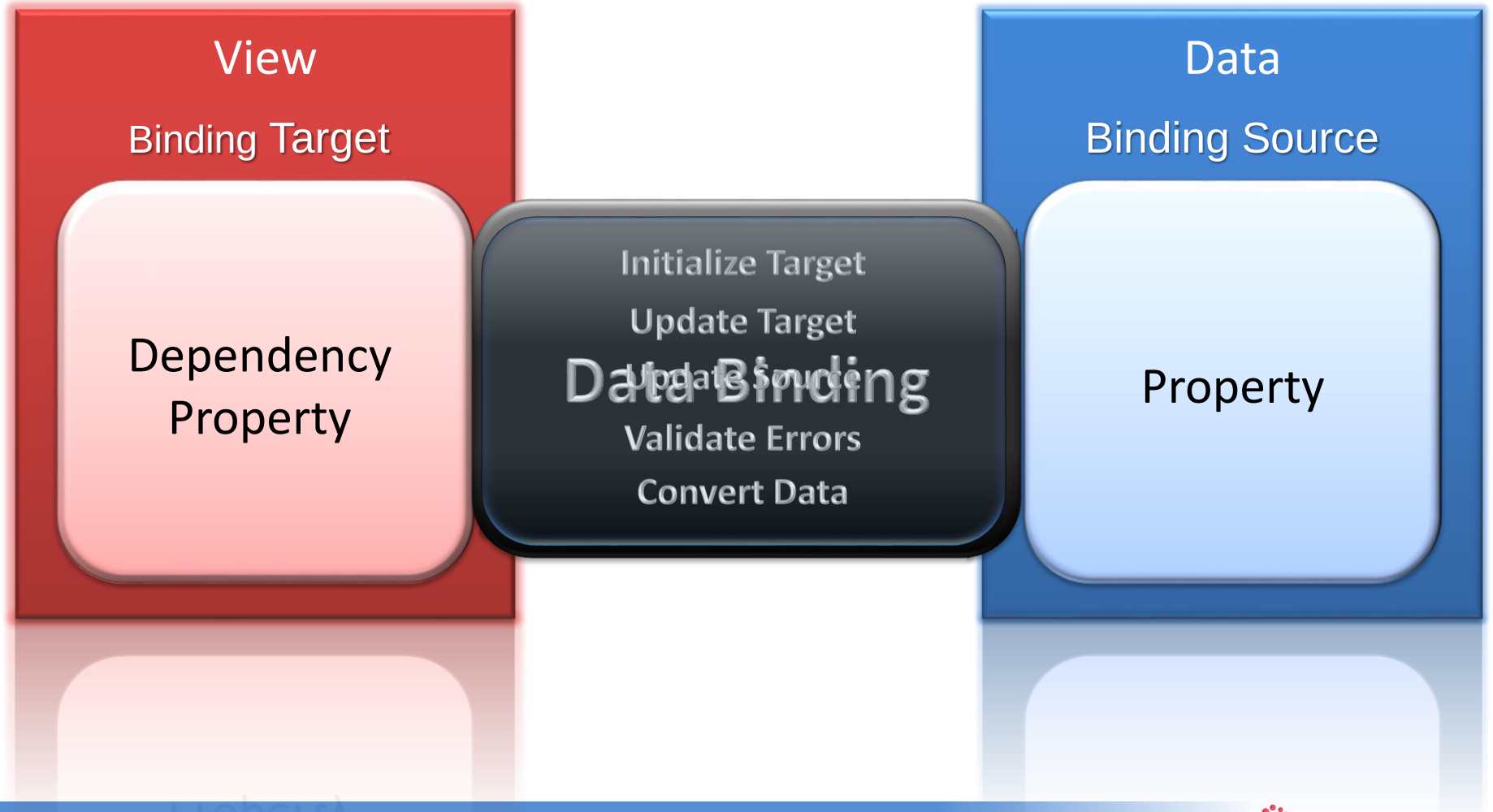
Pokročilé objektové technológie

WPF

Windows Presentation Foundation

KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Data Binding in Action



WPF Data Binding

<TextBox Text="{Binding City}" />

```
class MainWindowViewModel
{
    public string City
    {
        get { … }
        set { … }
    }
    …
}
```

binding

Data binding

- Prenos dát medzi GUI a aplikačnou logikou
  - Target: takmer ľubovoľná property hociakého elementu WPF
  - Source: public property CLR objektu:
    - CLR a XAML property
    - ADO DataColumns (TypeDescriptor)
    - XML Node
  - Dynamika – aktualizácia GUI pri zmene property
    - IPropertyChange
    - DependencyProperty
    - PropertyDescriptor
    - ICollectionChanged – pre zmeny počtu prvkov v kolekcii
  - Typy bindingu: TwoWay, OneWay, ...
  - Value Converter

Data binding

- Jednoduchý zápis binding v XAML

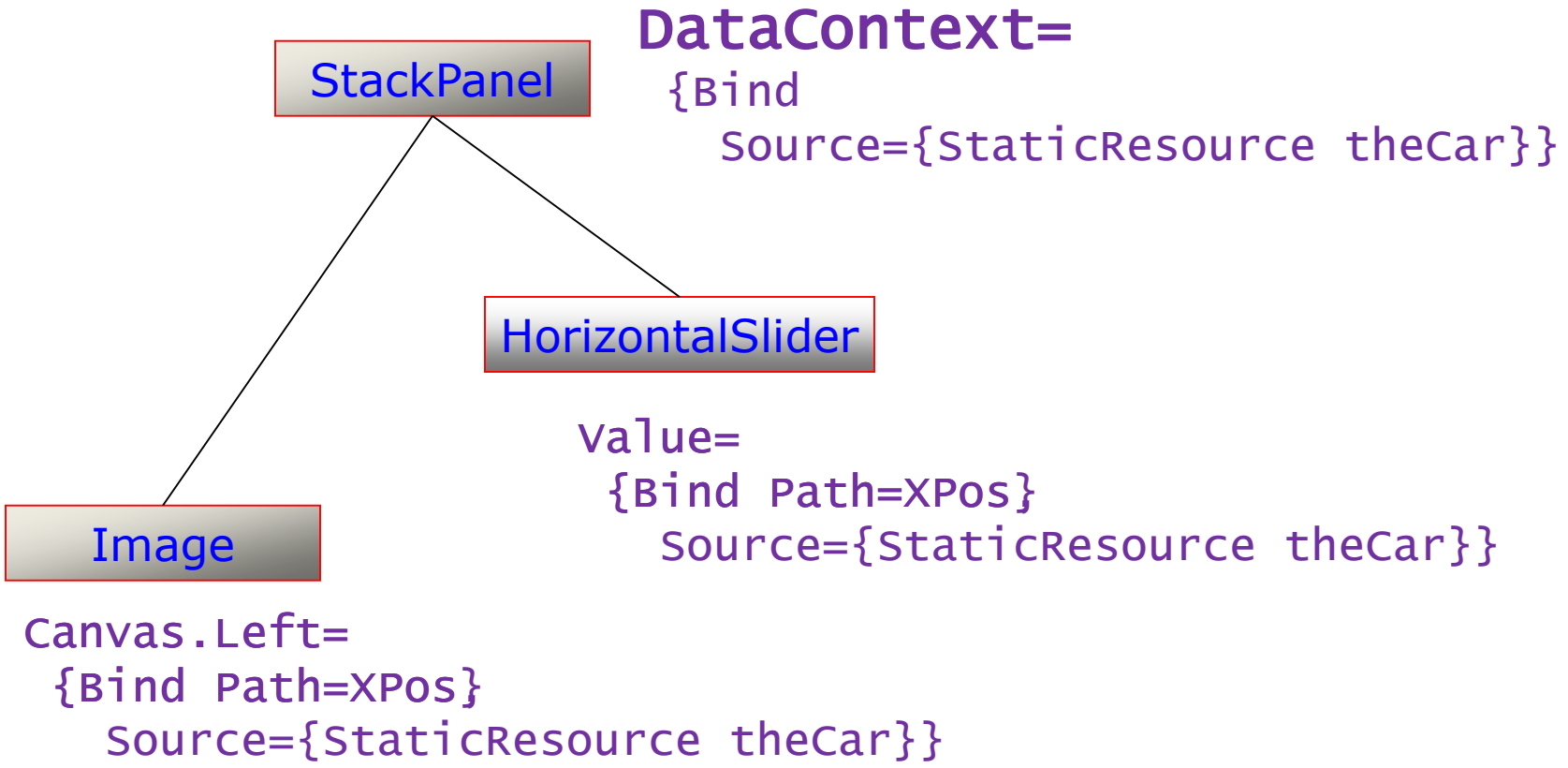
```
<Canvas xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
 <Slider Name="slider" />
 <Label Canvas.Top="50">
  <Label.Content>
   <Binding Path="Value" ElementName="slider" />
  </Label.Content>
 </Label>
</Canvas>
```

```
<Canvas xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" >
 <Slider Name="slider" />
 <Label Canvas.Top="50" Content="{Binding Path=Value, ElementName=slider}" />
</Canvas>
```

- Ukážky (binding2a, binding2b)

Data binding – DataContext

- DataContext – vešeobecný DataSource
- Je prenášaný na prvky vnorené



Data binding – typ väzby

- Možné typy
  - OneWay
  - TwoWay
  - OneTime
  - OneWayToSource
- OneWayToSource a OneTime môže byť použité s ľubovoľným objektom – jednoduché čítanie/zápis property
- OneWay TwoWay využíva rozhranie IPropertyChange
- Kolekcie (napr. ListBox) využíva ICollectionChange

Štýly

- Podobné ako CSS
- Možnosť zmeniť vzhľad celého GUI (themes)

```
<Canvas xmlns="..." xmlns:x="..." >
 <TextBox>
  <TextBox.Style>
   <Style><Setter Property="Control.FontSize" Value="18" /> </Style>
  </TextBox.Style>
 </TextBox>
</Canvas>
```

```
<Canvas xmlns="..." xmlns:x="..." >
 <Canvas.Resources>
  <Style x:Key="MyTextBox">
   <Setter Property="Control.FontSize" Value="18" />
  </Style>
 </Canvas.Resources>
 <TextBox Style="{StaticResource MyTextBox}" /> </Canvas>
```

XML data v XAML

```
<Page.Resources>
<XmlDataProvider x:Key="InventoryData" XPath="Inventory">
  <x:XData>
    <Inventory xmlns="">
      <Books>
       <Book ISBN="0-7356-0562-9" Stock="in" Number="9">
         <Title>XML in Action</Title>
         <Summary>XML Web Technology</Summary>
       </Book>
       <Book ISBN="0-7356-1370-2" Stock="in" Number="8">
         <Title>Programming Microsoft Windows With C#</Title>
         <Summary>C# Programming using the .NET Framework</Summary>
       </Book>
       ...
      </Books>
    </Inventory>
  </x:XData>
</XmlDataProvider>
</Page.Resources>
```

Template

```
<ListBox Name="Zoznam"
  ItemsSource="{Binding Source={StaticResource InventoryData},
                        XPath=Books/*}">
 <ListBox.ItemTemplate>
  <DataTemplate>
   <StackPanel Orientation="Horizontal">
    <Button Content="{Binding XPath=@Number}"
      Padding="10 5" Margin="10 2“ FontSize="20" Foreground="Brown"/>
    <StackPanel>
     <TextBlock Text="{Binding XPath=Title}"
         FontSize="20" Foreground="Brown"/>
     <StackPanel Orientation="Horizontal">
      <TextBlock Text="{Binding XPath=@ISBN}" Width="130"/>
      <TextBlock Text="{Binding XPath=Summary}"/>
     </StackPanel>
    </StackPanel>
   </StackPanel>
  </DataTemplate>
 </ListBox.ItemTemplate>
</ListBox>
```

Template položky

Animácie

- Na animáciu sa používa storyboard
  - Definuje timeline

```
<Page xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
 <Rectangle Name="MyRectangle" Width="100" Height="100" Fill="Blue">
  <Rectangle.Triggers>
   <EventTrigger RoutedEvent="Rectangle.Loaded">
    <BeginStoryboard>
     <Storyboard>
      <DoubleAnimation
       Storyboard.TargetName="MyRectangle"
       Storyboard.TargetProperty="Width" From="100" To="200" Duration="0:0:5"
       AutoReverse="True" RepeatBehavior="Forever" />
     </Storyboard>
    </BeginStoryboard>
   </EventTrigger>
  </Rectangle.Triggers>
 </Rectangle>
</Page>
```

Animácie – komplexnejšie

- Viac storyboard pre ten istý prvok
- Bežia paralelne

```
<Page xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
 <Rectangle Name="MyRectangle" Width="100" Height="100" Fill="Blue">
  <Rectangle.Triggers>
   <EventTrigger RoutedEvent="Rectangle.Loaded">
    <BeginStoryboard>
     <Storyboard>
      <DoubleAnimation
       Storyboard.TargetName="MyRectangle" Storyboard.TargetProperty="Width"
       From="100" To="200" Duration="0:0:2" AutoReverse="True" RepeatBehavior="Forever" />
      <DoubleAnimation
       Storyboard.TargetName="MyRectangle"
       Storyboard.TargetProperty="Height"
       From="100" To="50" Duration="0:0:3"
       AutoReverse="True" RepeatBehavior="Forever" />
     </Storyboard></BeginStoryboard></EventTrigger></Rectangle.Triggers></Rectangle></Page>
```

3D Modely

```
<Page   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      Height="400" Width="500">
  <Viewport3D Name="mainViewport" ClipToBounds="True">
    <Viewport3D.Camera>
      <PerspectiveCamera Position="0,0,4" />
    </Viewport3D.Camera>
    <Viewport2DVisual3D >
      <Viewport2DVisual3D.Transform>
        <RotateTransform3D>
          <RotateTransform3D.Rotation>
            <AxisAngleRotation3D x:Name="uiRotate" Angle="40" Axis="0, 1, 0" />
          </RotateTransform3D.Rotation>
        </RotateTransform3D>
      </Viewport2DVisual3D.Transform>
      <Viewport2DVisual3D.Geometry>
        <MeshGeometry3D Positions="-1,1,0 -1,-1,0 1,-1,0 1,1,0" TextureCoordinates="0,0 0,1 1,1 1,0" TriangleIndices="0 1 2 0 2 3"/>
      </Viewport2DVisual3D.Geometry>
      <Viewport2DVisual3D.Material>
        <DiffuseMaterial Viewport2DVisual3D.IsVisualHostMaterial="True" Brush="White"/>
      </Viewport2DVisual3D.Material>
      <Viewport2DVisual3D.Visual>
        <StackPanel Orientation="Vertical">
          <Button Background="Yellow" >Button1</Button>
          <Button Background="Aqua" >Button2</Button>
          <Button Background="Beige" >Button3</Button>
          <Button Background="Coral" >Button4</Button>
        </StackPanel>
      </Viewport2DVisual3D.Visual>
    </Viewport2DVisual3D>
    <ModelVisual3D>
      <ModelVisual3D.Content>
        <DirectionalLight Color="#FFFFFFFF" Direction="0,0,-1"/>
      </ModelVisual3D.Content>
    </ModelVisual3D>
  </Viewport3D>
</Page>
```

Kamera

Transformácia - otočenie

Kostra

Textúra

Vizuál

Svetlo

Multimedia

- Práca s Audio/Video
  - <MediaElement />
  - Je vlastne wrapper nad Media Player API

```
<Window xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Width="800" WindowState="Maximized">
 <StackPanel>
  <StackPanel Orientation="Horizontal">
   <MediaElement Source="d:\working\wpf\intro.wmv" Width="500" />
   <MediaElement Source="d:\working\wpf\xbox.wmv" Width="500" />
  </StackPanel>
  <StackPanel Orientation="Horizontal">
   <MediaElement Source="d:\working\wpf\numbers.wmv" Width="500" />
   <MediaElement Source="d:\working\wpf\bee.wmv" Width="500" />
  </StackPanel>
 </StackPanel>
</Window>
```

Microsoft Expression Blend