# Rádiová tlačítka a databinding

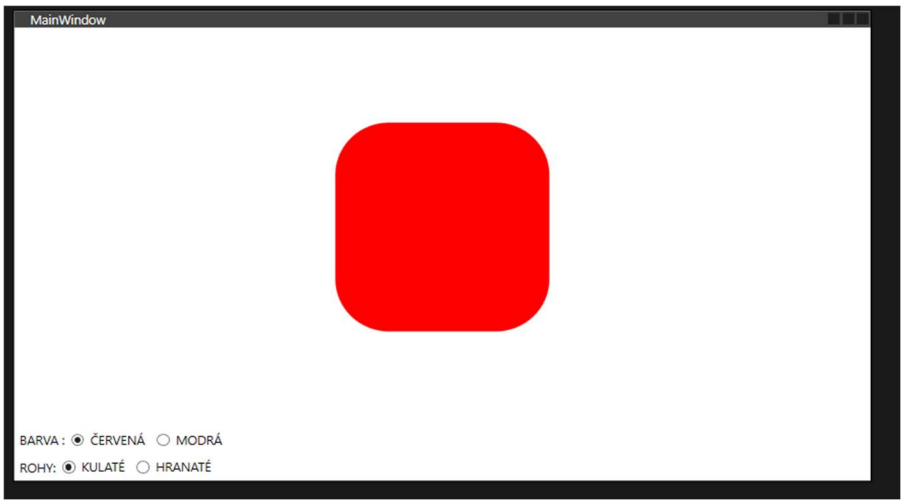
Naposledy jsme si ukazovali, jakým způsobem je možné do naší aplikace vkládat obrázky a jak je nastavovat. Nyní si ukážeme, jakým způsobem pracovat s dalším prvkem, který je možné ve WPF požívat a tím jsou rádiová tlačítka.

Rádiová tlačítka v aplikacích slouží k výběru jedné z většího množství možností, přičemž díky přiřazování do skupin můžeme docílit automatického od označení předchozího výběru. Rádiová tlačítka jsou ve WPF realizována pomocí elementu RadioButton. Po jeho zápisu se v aplikaci objeví malé kolečko. Do elementu RadioButton poté můžeme vložit text, který nám volbu, která bude pomocí tlačítka provedena popíše. Tlačítka můžeme přiřazovat do jednotlivých skupin, a to pomocí vložení do samostatných panelů anebo pomocí atributu GroupName. U tlačítka zároveň můžeme pomocí atributu IsChecked určit, zda je či není označeno. Dejme tomu, že budeme potřebovat vytvořit aplikaci, ve které bude zobrazen čtverec, který budeme moci pomocí rádiových tlačítek nastavovat. Nastavit bude možné to, jaká bude barva čtverce, a to jestli bude čtverec zaoblený. Čtverec můžeme v aplikaci realizovat pomocí elementu Rectangle. Zápis v XAMLu bude vypadat následovně

```
<DockPanel x:Name="MyPanel">

    <WrapPanel Orientation="Horizontal" DockPanel.Dock="Bottom">
        <Label>BARVA :</Label>
        <RadioButton VerticalContentAlignment="Center" Margin="0,0,10,0"
Content="KULATÉ" IsChecked="True"></RadioButton>
        <RadioButton VerticalContentAlignment="Center" Margin="0,0,10,0"
Content="HRANATÉ"></RadioButton>
    </WrapPanel>
    <WrapPanel Orientation="Horizontal" DockPanel.Dock="Bottom">
        <Label>BARVA :</Label>
        <RadioButton  VerticalContentAlignment="Center" Margin="0,0,10,0"
IsChecked="True">ČERVENÁ</RadioButton>
        <RadioButton VerticalContentAlignment="Center"
Margin="0,0,10,0">MODRÁ</RadioButton>
    </WrapPanel>
    <Rectangle x:Name="MySquare" Width="200" Height="200" DockPanel.Dock="Top"
Fill="Red" RadiusX="50" RadiusY="50"></Rectangle>
</DockPanel>
```

a výsledná podoba aplikace bude následující



Nyní, když bude potřeba při změně volby provést patřičné nastavení čtverce, je možné podobně jako u normálního tlačítka využít atribut Click a přiřadit radiovému tlačítku metodu, která bude při jeho stisknutí provedena. Tento způsob řešení je pro takto jednoduchou aplikaci ideální, nicméně v případě aplikací složitějších, kdy bude možné vybírat z velkého množství možností by se nejednalo o zrovna nejlepší realizaci. Problém je v tom, že pro každé tlačítko musíme definovat pomocí C# vlastní metodu – při velkém množství voleb bude velké množství metod. Pokud budeme potřebovat tento problém eliminovat je možné místo atributu Click využít atribut Command a CommandParametr. Problém využití těchto atributů je nutnost používání databindingu, který je už o něco komplexnější než pouhé vytváření metod.

Ve chvíli, když budeme nastavovat atribut Command, musíme nejprve vytvořit vlastní třídu implementující rozhraní ICommand. Třída nebude nijak složitá a může vypadat například takto (pokud by logika vykonávání příkazů musela být složitější je samozřejmě možné třídu upravit takovým způsobem, aby odpovídala potřebám)

```
public class MyCommand : ICommand
{
    public event EventHandler? CanExecuteChanged;
    public Action<object?> _action;

    public MyCommand(Action<object?> action)
    {
        _action = action;
    }

    public bool CanExecute(object? parameter)
    {
        return true;
    }

    public void Execute(object? parameter)
    {
        _action(parameter);
    }
}
```

Nyní můžeme ve třídě MainWindow vytvořit vlastnost s datovým typem MyCommand (naše třída implementující rozhraní ICommand) na kterou se v XAMLu odkážeme. Vlastnost ve třídě MainWindow bude zatím vypadat následovně.

```
public MyCommand ColorChangeCommand {

get;

set;

}
```

Nyní se již můžeme v XAMLu na vlastnost odkázat. Problém je, že byť vlastnost v hlavním okně existuje, není zaregistrována, a i když by aplikace nehlásila chybu při spuštění, k propojení mezi vlastností ColorChangeCommand a patřičnými tlačítky by nedošlo. K zaregistrování vlastnosti můžeme použít statickou metodu třídy DependencyProperty s názvem Register, jejíž výstup si uložíme do vhodně nazvaného statického atributu. Zaregistrování vlastnosti bude vypadat následovně

```
public static DependencyProperty ColorCommandProperty =
DependencyProperty.Register(nameof(ColorChangeCommand), typeof(MyCommand),
typeof(MainWindow));
```

Metoda Register přebírá v tomto případě tři argumenty – název registrované vlastnosti, datový typ vlastnosti a datový typ vlastníka vlastnosti – třída, která vlastnost obsahuje (v tomto případě je vlastníkem vlastnosti třída MainWindow).

Po zaregistrování vlastnosti budeme muset vlastnost upravit takovým způsobem, aby při nastavování a získávání její hodnoty došlo k odkazu na zaregistrovanou hodnotu. Vlastnost po úpravě bude vypadat následovně

```
public MyCommand ColorChangeCommand
{
      get => (MyCommand)GetValue(ColorCommandProperty);
      set => SetValue(ColorCommandProperty, value);
}
```

Poslední věc, kterou bude nutné pomocí C# vytvořit, bude definice metody, která bude při nastavování barvy pomocí tlačítka vykonána a zároveň inicializace vlastnosti ColorChangeCommand. Metodu vytvoříme jako privátní metodu třídy MainWindow a inicializaci provedeme v konstruktoru třídy MainWindow. Metoda nastavující barvu může vypadat takto

```
private void ChangeColor(object? color)
{
    if (color == null) return;
    SolidColorBrush fill = (SolidColorBrush)color;
    MySquare.Fill = fill;
}
```

a inicializace vlastnosti v konstruktoru třídy MainWindow takto

```
ColorChangeCommand = new MyCommand(ChangeColor);
```

Nyní již můžeme vlastnost použít uvnitř XAMLu. Když bude tlačítko s vlastností propojovat budeme muset správně nastavit jeho atribut Command a zároveň atribut CommandParametr. Command v sobě ponese odkaz na vlastnost ColorChangeCommand a CommandParameter bude obsahovat hodnotu parametru se kterou bude příkaz vykonán. Než tyto atributy nastavíme bude ještě nutné přiřadit název oknu naší aplikace, ve kterém se vlastnost ColorChangeCommand nachází. XAML aplikace s přiřazeným názvem, příkazem a parametrem bude vypadat následovně

```
<Window x:Class="MyDesktopApp.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:MyDesktopApp"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800" Name="MyWindow">
    <DockPanel x:Name="MyPanel">
        <WrapPanel Orientation="Horizontal" DockPanel.Dock="Bottom">
            <Label>BARVA :</Label>
            <RadioButton VerticalContentAlignment="Center" Margin="0,0,10,0"
Content="KULATÉ" IsChecked="True"></RadioButton>
            <RadioButton VerticalContentAlignment="Center" Margin="0,0,10,0"
Content="HRANATÉ"></RadioButton>
        </WrapPanel>

        <WrapPanel Orientation="Horizontal" DockPanel.Dock="Bottom">
```

```
            <Label>BARVA :</Label>
            <RadioButton IsChecked="True" VerticalContentAlignment="Center"
Margin="0,0,10,0" Command="{Binding ElementName=MyWindow,
Path=ColorChangeCommand}" CommandParameter="{x:Static
Brushes.Red}">ČERVENÁ</RadioButton>
            <RadioButton VerticalContentAlignment="Center" Margin="0,0,10,0"
Command="{Binding ElementName=MyWindow, Path=ColorChangeCommand}"
CommandParameter="{x:Static Brushes.Blue}">MODRÁ</RadioButton>
        </WrapPanel>
        <Rectangle x:Name="MySquare" Width="200" Height="200"
DockPanel.Dock="Top" Fill="Red" RadiusX="50" RadiusY="50"></Rectangle>
    </DockPanel>
</Window>
```

Nyní, když budeme přepínat mezi volbami barvy, bude se barva čtverce měnit na základě hodnoty v atributu CommandParameter daného tlačítka. Pro změnu zaoblení tlačítka pomocí příkazů bude postup podobný – vytvoříme vlastnost, vlastnost zaregistrujeme, vytvoříme metodu a vlastnost inicializujeme. Výsledný kód metody MainWindow bude následující

```
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;

namespace MyDesktopApp
{
    public partial class MainWindow : Window
    {
        public static DependencyProperty ColorCommandProperty =
DependencyProperty.Register(nameof(ColorChangeCommand), typeof(MyCommand),
typeof(MainWindow));

        public MyCommand ColorChangeCommand { get =>
(MyCommand)GetValue(ColorCommandProperty); set => SetValue(ColorCommandProperty,
value); }

        public static DependencyProperty CornerCommandProperty =
DependencyProperty.Register(nameof(CornerChangeCommand), typeof(MyCommand),
typeof(MainWindow));

        public MyCommand CornerChangeCommand { get =>
(MyCommand)GetValue(CornerCommandProperty); set =>
SetValue(CornerCommandProperty, value); }

        public MainWindow()
        {
            CornerChangeCommand = new MyCommand(ChangeCorners);
            ColorChangeCommand = new MyCommand(ChangeColor);
            InitializeComponent();
        }

        private void ChangeCorners(object? radius)
        {
            if (radius == null) return;
            MySquare.RadiusX = (double)radius;
            MySquare.RadiusY = (double)radius;
        }
```

```
        private void ChangeColor(object? color)
        {
            if (color == null) return;
            SolidColorBrush fill = (SolidColorBrush)color;
            MySquare.Fill = fill;
        }
    }
}
```

V případě XAMLu bude postup při změně zaoblení rovněž téměř stejný, jako v případě změně barvy. Rozdíl bude v definici hodnoty atributu CommandParametr u kterého nyní budeme muset určit i jeho datový typ – jinak by hodnota byla předána jako string. Výsledný XAML bude vypadat následovně

```
<Window x:Class="MyDesktopApp.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:MyDesktopApp"
        xmlns:sys="clr-namespace:System;assembly=mscorlib"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800" Name="MyWindow">
    <DockPanel x:Name="MyPanel">

        <WrapPanel Orientation="Horizontal" DockPanel.Dock="Bottom">
            <Label>ROHY: </Label>
            <RadioButton IsChecked="True" VerticalContentAlignment="Center"
Margin="0,0,10,0" Command="{Binding ElementName=MyWindow,
Path=CornerChangeCommand}" Content="KULATÉ">
                <RadioButton.CommandParameter>
                    <sys:Double>50</sys:Double>
                </RadioButton.CommandParameter>
            </RadioButton>
            <RadioButton VerticalContentAlignment="Center" Margin="0,0,10,0"
Command="{Binding ElementName=MyWindow, Path=CornerChangeCommand}"
Content="HRANATÉ">
                <RadioButton.CommandParameter>
                    <sys:Double>0</sys:Double>
                </RadioButton.CommandParameter>
            </RadioButton>
        </WrapPanel>
        <WrapPanel Orientation="Horizontal" DockPanel.Dock="Bottom">
            <Label>BARVA :</Label>
            <RadioButton IsChecked="True" VerticalContentAlignment="Center"
Margin="0,0,10,0" Command="{Binding ElementName=MyWindow,
Path=ColorChangeCommand}" CommandParameter="{x:Static
Brushes.Red}">ČERVENÁ</RadioButton>
            <RadioButton VerticalContentAlignment="Center" Margin="0,0,10,0"
Command="{Binding ElementName=MyWindow, Path=ColorChangeCommand}"
CommandParameter="{x:Static Brushes.Blue}">MODRÁ</RadioButton>
        </WrapPanel>
        <Rectangle x:Name="MySquare" Width="200" Height="200"
DockPanel.Dock="Top" Fill="Red" RadiusX="50" RadiusY="50"></Rectangle>
    </DockPanel>
</Window>
```