Dependency Property

**Priklad Counteru zalozenem na dep property je v ExplanationSolution. Popsano ve vyjimky exceptions.docx - hledej slovo dispatcher.**

Pouzivame jako backing store pro normalni property. Klavesova zkratka napiseme propdp +TAB a visual studio nam vytvori dep property a normalni propertu k tomu.

* Vytvorime teda dependencyProperty

public int MyProperty

{

get { return (int)GetValue(MyPropertyProperty); }

set { SetValue(MyPropertyProperty, value); }

}

// Using a DependencyProperty as the backing store for MyProperty. This enables animation, styling, binding, etc...

public static readonly DependencyProperty MyPropertyProperty =

DependencyProperty.Register("MyProperty", typeof(int), typeof(ownerclass), new PropertyMetadata(0));

Nastavime ji datovy typ (tady int) muze to byt cokoli => bool , visibility atd.

Nastavime nazev property .

OwnerClass je nazev tridy ve které je depProp.

Do metadat muzeme zadat defaultni hodnotu, pripadne metodu která se vykona když se zmeni properta. DepProp se stara o notifikaci a diky ni jsou na controlech třeba Triggery bez nutnosti psat explicitni kod.

* Když delame custom control mame Xaml a k nemu CodeBehind. DepProp muze byt jenom v codeBehindu. Custom Control pak pouzivame ve View jako normalni control.

Napr ve Frameworku je MultiSelectComboBox (upravit ho muzeme tak, ze otevreme Framework.sln ) Tady je custom control kde jsem v codeBehindu definoval depProp SelectAllButtonVisibility

public Visibility SelectAllButtonVisibility

{

get { return (Visibility)GetValue(SelectAllButtonVisibilityProperty); }

set { SetValue(SelectAllButtonVisibilityProperty, value); }

}

// Using a DependencyProperty as the backing store for SelectAllButtonVisibility. This enables animation, styling, binding, etc...

public static readonly DependencyProperty SelectAllButtonVisibilityProperty =

DependencyProperty.Register("SelectAllButtonVisibility", typeof(Visibility), typeof(MultiSelectComboBox), new PropertyMetadata(default(Visibility)));

* V Xamlu nabindujeme Visibility na buttonu který chceme zneviditelnit na naší novou propertu :

<Button

Width="100"

Height="28"

Background="LightGreen"

Click="SelectAll"

Content="{x:Static resources:Strings.SelectAll}"

Visibility="{Binding Path=SelectAllButtonVisibility, RelativeSource={RelativeSource FindAncestor, AncestorType={x:Type controls:MultiSelectComboBox}}}" />

* V CustomControlu je to vse . Ted budu chtit tento customControl pouzit ve zmenovadle. Muzu ho pouzit protože mam z nugetu Admosphere nainstalovany balicek Framework.Gui

<controls1:MultiSelectComboBox

x:Name="ValueComboBox"

Grid.Row="2"

Grid.Column="1"

Height="25"

Margin="7"

BindableSelectedItems="{Binding Path=BindableList, Mode=OneWayToSource, UpdateSourceTrigger=PropertyChanged}"

DisplayMemberPath="Name"

ItemsSource="{Binding Path=ValueComboItems}"

SelectAllButtonVisibility="Collapsed"

Visibility="{Binding ElementName=DatePicker, Path=Visibility, Converter={StaticResource VisibleToCollapsedVisibilityConverter}}"/>

Nyní mam na MultiSelectComboBoxu propertu SelectAllButtonVisibility kterou si muzu nabindovat kam chci.

Pokud při tvorbe CustomControl vytvorime nejakou propertu která není v codeBehindu , znamena to, ze na tuto propertu muze byt bindovano ve ViewModelu kde customControl pouzivame.

Napr do MultiComba ve Frameworku jsem pridal button Add:

<Button

Width="100"

Height="28"

Margin="2,0,0,0"

Background="LightGray"

Command="{Binding Path=AddCommand}"

Content="{x:Static resources:Strings.Add}"

Do codeBehindu jsem nic nedaval, ale mam tu AddCommand. Když tento Icommand vytvorim ve ViewModelu v jehož xamlu je customControl muzu v metode commandu provadet obsluhu tohoto tlacitka umisteneho v controlu:

Ve FilterPartViewModel.FilterElement :

public ICommand AddCommand { get; }

AddCommand = new RelayCommand(DoAddItemToValueComboItems);

private void DoAddItemToValueComboItems()

{ obsluha }

7.6.2019

DependencyProperty slouzi jako NofifyChanged. Musí byt samozrejme v behindu. Upravoval jsem LightVideoPlayerControl. Potreboval jsem propertu SpeedIncreased typu bool abych mohl z behindu informovat view o zmene rychlosti prehravani.

**Uzivatel stlaci tlacitko, které je napojene na builtIn command IncreaseTreble:**

<Button Width="32" Height="32" Margin="2" Command="MediaCommands.IncreaseTreble"

ToolTip="{x:Static localisation:Localisation.FastForward}">

<Image>

<FrameworkElement.Style>

<Style TargetType="{x:Type Image}">

<Setter Property="Image.Source" Value="../../../Resources/Images/mp\_fast.png"/>

<Setter Property="FrameworkElement.Margin" Value="-1"/>

<Style.Triggers>

<DataTrigger Binding="{Binding SpeedIncreased, ElementName=LightVideoPlayer}" Value="True">

<Setter Property="Image.Source" Value="../../../Resources/Images/next.png"/>

</DataTrigger>

</Style.Triggers>

</Style>

</FrameworkElement.Style>

</Image>

</Button>

**Definice commandu:**

<CommandBinding CanExecute="FastForwardCanExecute" Executed="FastForwardExecuted"

Command="MediaCommands.IncreaseTreble"/>

**Tim se dostaneme sem**:

public void FastForward()

{

this.SpeedIncreased = this.MePlayer.FastForward();

}

**MePlayer je samotny prehravac vlozeny do controlu a je typu MediaPlayer. Jeho metoda nastavi rychlost a vraci bool podle toho jestli je prehravani zrychlene:**

public bool FastForward()

{

this.MePlayer.SpeedRatio = this.MePlayer.SpeedRatio > 1.0 ? 1.0 : 2.0;

return this.MePlayer.SpeedRatio > 1.0;

}

**Control se teda dozvi, ze nastala změna a zmeni se bool SpeedIncreased. Na to aby se mi změna z behindu dostala do view, potrebuju DependencyProperty: Ta se postara o notifikaci:**

public static readonly DependencyProperty **SpeedIncreasedProperty** = DependencyProperty.Register(nameof (SpeedIncreased), typeof (bool), typeof (LightVideoPlayerControl), new PropertyMetadata());

public bool **SpeedIncreased**

{

get

{

return (bool) this.GetValue(LightVideoPlayerControl.SpeedIncreasedProperty);

}

set

{

this.SetValue(LightVideoPlayerControl.SpeedIncreasedProperty, (object) value);

}

}