**IMultiValueConverter**

xmlns:Converters="clr-namespace:MIR.Media.Coding.Core.Converters"

<UserControl.Resources>

<Converters:FractionConverter x:Key="fractionConverter" />

</UserControl.Resources>

<TextBox.Background>

<MultiBinding Converter="{StaticResource fractionConverter}">

<Binding Path="MediaMessage.AreaNumerator" />

<Binding Path="MediaMessage.AreaDenominator" />

</MultiBinding>

</TextBox.Background>

public class FractionConverter : IMultiValueConverter  
{  
 private readonly SolidColorBrush m\_okBrush = Brushes.White;  
 private readonly SolidColorBrush m\_solidColorBrush = Brushes.Red;  
  
 public object Convert(object[] values, Type targetType, object parameter, CultureInfo culture)  
 {  
 if (values == null || values.Length < 3 || values[0] == DependencyProperty.UnsetValue || values[1] == DependencyProperty.UnsetValue || values[2] == DependencyProperty.UnsetValue)  
 {  
 return m\_okBrush;  
 }  
  
 double numerator, denominator, minimalFraction;  
 try  
 {  
 numerator = System.Convert.ToDouble((short) values[0]);  
 denominator = System.Convert.ToDouble((short) values[1]);  
  
 var fract = (Fraction) values[2];  
  
 if (fract.Numerator == null || fract.Denominator == null)  
 {  
 throw new InvalidOperationException("Unable to set warning about Area because of dependencies missing!");  
 }  
  
 var num = System.Convert.ToDouble((short) fract.Numerator);  
 var den = System.Convert.ToDouble((short) fract.Denominator);  
 minimalFraction = num / den;  
 }  
 catch (Exception)  
 {  
 return m\_okBrush;  
 }  
  
 var fraction = numerator / denominator;  
 return fraction < minimalFraction ? m\_solidColorBrush : m\_okBrush;  
 }  
  
 public object[] ConvertBack(object value, Type[] targetTypes, object parameter, CultureInfo culture)  
 {  
 throw new NotImplementedException();  
 }  
}

**ConvertBack**

IMO, the ConvertBack method is used to convert your visual representation of the data to the specific DataType.

For example: you use a Converter to convert a boolean true to the string "TrueBoolean". This text will be displayed in your TextBox. When you change the value of the TextBox, the ConvertBackmethod will be called as soon as the binding fires again (default OnFocusLost). Now your ConvertBack method will try to convert the new value to the datatype you want it to be. So you will have to implement logic to convert "FalseBoolean" to false.

public class Converter : IValueConverter

{

object IValueConverter.Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

return (bool) value ? "TrueBoolean" : "FalseBoolean";

}

object IValueConverter.ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

var s = (string) value;

if (s.Equals("TrueBoolean",StringComparison.CurrentCultureIgnoreCase))

return true;

if (s.Equals("FalseBoolean", StringComparison.CurrentCultureIgnoreCase))

return false;

throw new Exception(string.Format("Cannot convert, unknown value {0}", value));

}

}

# Proc nemuzeme resolvovat v converteru pomoci Castle

Converter by teoreticky sel zaregistrovat. Kdyz ale zrusime staticky kontejner (container.Current), tak na to abychom dostali tridu od Castlu, potrebujeme Dependecy Injection. Converter ale nikam nedavame jako zavislost ani jako propertu.

Je to samostatna trida, ktere pracuje s view.

# Preneseni property z view do converteru pomoci Bindingu

Visibility="{Binding Path=ButtonsVisibility, Converter={StaticResource VisibleToCollapsedConverter}}" />

Bindovany objekt se dostane do converteru jako object value

# TemplateSelector, aneb vybrani converteru na zaklade nejake bool property z datacontextu:

Viz take <TemplateSelector.docx>