Collection of Value converters for Windows Phone apps

This article gives some examples for value converters that can be used in different scenarios in Windows Phone apps

Introduction

Value converters (IValueConverter Interface) are a versatile instrument helping in data binding scenarios on the Windows Phone (and Windows 8 Modern UI or WPF) platforms. The spectrum includes

- data type conversions, e.g. `BooleanToVisibilityConverter`
- operations on bound data, e.g. `BooleanNegationConverter`
- individual text conversions, e.g. `TimeToStringConverter`
- special effects, like drawing dynamic images, e.g. `TimeToShapeConverter`
- debugging purposes, e.g. `DatabindingDebugConverter` see How to debug data binding problems on Windows Phone

An introduction into the necessary steps to use converters is given in How to debug data binding problems on Windows Phone, too.

Another very interesting idea is a value converter whose conversion functions can be generated by script: The Last IValueConverter.

Here's a screenshot of how the sample app looks like:

Note: The IValueConverter interface is defined with different function parameters in Win 8 Modern UI aka "Metro" aka Windows 8 Store Apps. That's the reason I couldn't use a Portable Class Library to contain the converter classes.
This also means that by changing the last two function parameters in the Convert and ConvertBack functions these converters can be used for Windows 8, too.

BooleanToVisibilityConverter

This converter can translate a boolean value to the value type needed by the Visibility property of most XAML element. It's mostly used in two cases:

- when a Checkbox or ToggleButton element interactively controls the visibility of another element
- or when a background data value does the same.

For the first case the data binding syntax for the Visibility property refers to the IsChecked property of the controlling xaml element and includes the converter:

```
<toolkit:ToggleSwitch Name="tgVis" Content="Clock Visible" IsChecked="True"
  Checked="ToggleSwitch_Checked" Unchecked="ToggleSwitch_Checked"/>
<Path Data="{Binding CurrentTime, Converter={StaticResource TimeToShapeConverter}}"
  Visibility="{Binding IsChecked, ElementName=tgVis,Converter={StaticResource BooleanToVisibilityConverter}}"
  Stroke="Gray" StrokeThickness="12" StrokeEndLineCap="Round" StrokeStartLineCap="Round"
  Stretch="Uniform" Margin="6"/>
```


(C) Copyright Nokia 2013. All rights reserved.

In the second case the usage is pretty standard data binding.

```
<Textblock Text="Make me invisible"
  Visibility="{Binding VisibilityProperty,Converter={StaticResource BooleanToVisibilityConverter}}"/>
```

Here's the simple code for the converter:

```
/// <summary>
/// Value converter that translates true to <see cref="Visibility.Visible"/> and false to
/// <see cref="Visibility.Collapsed"/>.
/// </summary>
public sealed class BooleanToVisibilityConverter : IValueConverter
   {
   public object Convert(object value, Type targetType, object parameter, string language)
      {
      return (value is bool && (bool)value) ? Visibility.Visible : Visibility.Collapsed;
      }

   public object ConvertBack(object value, Type targetType, object parameter, string language)
      {
      return value is Visibility && (Visibility)value == Visibility.Visible;
      }
   }
```

BooleanNegationConverter

This one can be used to invert a boolean value, e.g. if a bool value and its negative are both needed.

```
/// <summary>
/// Value converter that translates true to false and vice versa.
/// </summary>
public sealed class BooleanNegationConverter : IValueConverter
   {
   public object Convert(object value, Type targetType, object parameter, string language)
      {
      return !(value is bool && (bool)value);
      }

   public object ConvertBack(object value, Type targetType, object parameter, string language)
      {
      return !(value is bool && (bool)value);
      }
   }
```

In this code fragment the converter is used to show the result of toggling the switch:

```
  <StackPanel Orientation="Horizontal">
     <TextBlock Text="Toggle to switch to "/>
     <TextBlock Text="{Binding IsChecked, ElementName=tgVis,Converter={StaticResource BooleanNegationConverter}}"/>
  </StackPanel>
</code
== TimeToStringConverter  ==
Display a date or time from a DateTime instance to displayable string using the TimeToStringConverter.
<code csharp>
/// <summary>
/// Value converter that converts a DateTime to a text and vice versa.
/// </summary>
public sealed class TimeToStringConverter : IValueConverter
   {
   public object Convert(object value, Type targetType, object parameter, string language)
      {
      DateTime dt = (DateTime)value;
      return dt.ToString();
      }

   public object ConvertBack(object value, Type targetType, object parameter, string language)
      {
      DateTime dt;
      DateTime.TryParse(value.ToString(), out dt);
      return dt;
      }
   }
```

TimeToShapeConverter

Converts the time to a clock image using geometric shape objects. This a very rudimentary implementation but I think you get the idea what you can do...

```
/// <summary>
/// Value converter that creates a clock image from a time value.
/// Reference the System.Windows.Media namespace.
/// You can bind the result to a Path element.
/// </summary>
public sealed class TimeToShapeConverter : IValueConverter
   {
   public object Convert(object value, Type targetType, object parameter, string language)
      {
      DateTime dt = (DateTime)value;
      GeometryGroup coll = new GeometryGroup();
      EllipseGeometry ell = new EllipseGeometry();
      ell.Center = new Point(55, 55);
      ell.RadiusX = ell.RadiusY = 60;
      coll.Children.Add(ell);
      LineGeometry hour = new LineGeometry();
      double deg = (dt.Hour % 12) * Math.PI / 6;
      hour.StartPoint = ell.Center;
      hour.EndPoint = new Point(55 + Math.Sin(deg) * 35, 55 - Math.Cos(deg) * 35);
      coll.Children.Add(hour);
      LineGeometry minute = new LineGeometry();
      minute.StartPoint = ell.Center;
```


(C) Copyright Nokia 2013. All rights reserved.

```
        deg = dt.Minute * Math.PI / 30;
        minute.EndPoint = new Point(55 + Math.Sin(deg) * 50, 55 - Math.Cos(deg) * 50);
        coll.Children.Add(minute);
        return coll;
        }

    public object ConvertBack(object value, Type targetType, object parameter, string language)
        {
        throw new NotImplementedException();
        }
    }
```

Regarding the usage in xaml see the above sample for `BooleanToVisibilityConverter`.

DatabindingDebugConverter

This converter is described in the How to debug data binding problems on Windows Phone article, which explains how to inspect elements before they are due to be displayed by a control.

```
/// <summary>
/// Value Converter that does nothing except giving the chance to set a breakpoint.
/// </summary>
public sealed class DatabindingDebugConverter : IValueConverter
    {
    public object Convert(object value, Type targetType, object parameter, string language)
        {
        return value;
        }

    public object ConvertBack(object value, Type targetType, object parameter, string language)
        {
        return value;
        }
    }
```

Defining the Resources

The following declaration tells the XAML page how to find the converters:

```
    <phone:PhoneApplicationPage.Resources>
        <local:TimeToStringConverter x:Key="TimeToStringConverter"/>
        <local:TimeToShapeConverter x:Key="TimeToShapeConverter"/>
        <local:BooleanToVisibilityConverter x:Key="BooleanToVisibilityConverter"/>
        <local:BooleanNegationConverter x:Key="BooleanNegationConverter"/>
    </phone:PhoneApplicationPage.Resources>
```

Invitation

Please feel free to add further value converter samples!

Sample code

This article is accompagnied by a sample that just uses the two time converters to show the current time as a string and as an analog clock. The `BooleanToVisibilityConverter` is used to control the visibility of the clock with a `ToggleSwitch`. In order to use the `ToggleSwitch` you'll need the Windows Phone Toolkit.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.