

Custom Controls Tutorial

Eine Anleitung wie man Custom Controls für Silverlight erstellen kann

Datum: 08.09.2009
Autor: Christopher Skerra
Version: 003

Inhalt

| | |
|---|---|
| Inhalt | 1 |
| Voraussetzungen für ein wiederverwendbares Custom Control | 2 |
| Zugriff von C# aus auf XAML Elemente in generic.xaml | 4 |
| Einfügen von Dependency Properties in C# | 4 |
| Manuelles einbinden der Control Library in XAML | 6 |
| Beispielcode | 6 |

Voraussetzungen für ein wiederverwendbares Custom Control

Um ein wiederverwendbares Custom Control zu programmieren muss zunächst eine Silverlight Klassenbibliothek erstellt werden. In die generierte Class1.cs kann nun das Custom Control geschrieben werden.

Um mehrere Custom Controls in einer DLL zusammenzufassen müssen wir nur eine neue Klasse in das Projekt einfügen.

Unsere Klasse muss von einer vorhandenen Klasse erben zum Beispiel Button, ContentControl oder Panel.

Da wir einen Content Presenter verwenden und keine vorgefertigten Events haben wollen, ist die Klasse ContentControl am besten:

```
public class Slider3D: ContentControl
```

Um die Klasse nun zu initialisieren rufen wir eine Methode auf die auch sofort unsere generic.xaml Datei anspricht und nach einem Standard Template für unser Control sucht.

Diese Methode sieht folgendermaßen aus:

```
public Slider3D ()  
{  
    this.DefaultStyleKey = typeof(Slider3D);  
}
```

Alles weitere wie z.B. die Event Listener werden in die "OnApplyTemplate" Methode geschrieben.

Hier ist ein Beispiel:

```
public override void OnApplyTemplate ()  
{  
    // Hier werden die Elemente aus XAML auf unsere C# Elemente gesetzt  
    xMain = GetTemplateChild("xMain") as Border;  
  
    // Event-Listener für die Maus  
    xInteractiveContainer.MouseLeftButtonDown += new  
    MouseButtonEventHandler(xInteractiveContainer_MouseLeftButtonDown);  
    xInteractiveContainer.MouseLeftButtonUp += new  
    MouseButtonEventHandler(xInteractiveContainer_MouseLeftButtonUp);  
}
```

Die generic.xaml Datei

Wir fügen dem Klassenbibliotheks-Projekt einen Ordner mit dem Namen „themes“ hinzu. In diesem erstellen wir nun eine Text-Datei und nennen diese „generic.xaml“. Die Schreibweise ist wichtig! Dies wird nun unser Resource Dictionary für unsere Controls. Nun stellen wir in den Eigenschaften der generic.xaml Datei die Build Action auf „Resource“ und die Zeile „Custom Tool“ wird gänzlich gelöscht. In die generic.xaml Datei muss nun eine Namespace Deklaration sowie ein Template für unser Custom Control deklariert werden. Eine Muster generic.xaml Datei könnte wie folgt aussehen:

```
<ResourceDictionary
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:custom="clr-
namespace:maximagoControls;assembly=maximagoControls">
  <Style TargetType="custom:Slider3D">
    <Setter Property="Template">
      <Setter.Value>
        <ControlTemplate TargetType="custom:Slider3D">

          <!--ELEMENTE FÜR DAS CONTROL HIER EINFÜGEN-->

        </ControlTemplate>
      </Setter.Value>
    </Setter>
  </Style>
</ResourceDictionary>
```

Um dynamischen Content in das Custom Control aufnehmen zu können wird ein Content Presenter benötigt. Dieser wird wie folgt deklariert:

```
<ContentPresenter x:Name="xContentStack_Main" Content="{TemplateBinding
Content}" />
```

Um die Eigenschaften eines Controls auch in XAML verfügbar zu machen (also z.B. <Button Width="100" />), muss die generic.xaml Datei etwas verändert werden. Über der Zeile „Setter Property = Template“ können die Standard Eigenschaften eines Elements eingegeben werden. Beispielsweise:

```
<Setter Property="Width" Value="200" />
```

Danach muss im folgenden XAML Code nur jede Eigenschaft welche dynamisch sein soll folgendermaßen deklariert werden:

```
<Rectangle Width="{TemplateBinding Width}" />
```

Zugriff von C# aus auf XAML Elemente in generic.xaml

Nach der Namespace Deklaration des Custom Controls wird für jedes Element aus XAML (welches benutzt werden möchte) eine solche Zeile Code verfasst:

```
[TemplatePart(Name = "xMain", Type = typeof(Border))]
```

Nun müssen wir im Code ein Objekt erzeugen welches dieses Element auffängt. In diesem Beispiel wäre das:

```
public Border xMain = new Border();
```

Als letztes muss nun noch in der „OnApplyTemplate“ Methode dem Objekt unser Element zugewiesen werden. Hier würde das so aussehen:

```
xMain = GetTemplateChild("xMain") as Border;
```

Nun kann auf unser Border über die Variable „xMain“ zugegriffen werden.

Einfügen von Dependency Properties in C#

Um Dependency Properties einzufügen, auf welche auch in Blend als Eigenschaften zugegriffen werden kann, muss folgender Code eingefügt werden:

```
public static readonly DependencyProperty DoScrollProperty =  
DependencyProperty.Register("DoScroll", typeof(bool), typeof(Slider3D),  
new PropertyMetadata(true));  
  
public bool DoScroll  
{  
    set  
    {  
        this.SetValue(DoScrollProperty, value);  
    }  
    get  
    {  
        return (bool) this.GetValue(DoScrollProperty);  
    }  
}
```

Hier eine kurze Erklärung der einzelnen Elemente:

...Register(Name der DP, Typ der DP, Typ des Elements auf die sich die DP bezieht, Standardwert der DP);

ACHTUNG: Sollte der Datentyp einer Dependency Property „double“ sein, dann muss diese in den Metadaten gecastet werden. Also:

```
public static readonly DependencyProperty DoubleProperty =  
DependencyProperty.Register("Double", typeof(double), typeof(Slider3D), new  
PropertyMetadata((double)150));
```

Manuelles einbinden der Control Library in XAML

Um das Custom Control nun nutzen zu können muss man nur eine Referenz auf unsere erstellte Klassenbibliothek setzen.

Dann kann man entweder in Blend die Elemente einfach per Drag & Drop nutzen oder manuell eine Referenz in XAML erstellen und von dort aus die Elemente einfügen.

Eine Referenz in XAML sieht so aus:

```
xmlns:custom="clr-namespace:maximagoControls;assembly=maximagoControls"
```

Nun kann über den Namespace „custom“ auf unsere Controls zugegriffen werden

```
<custom:Slider3D />
```

Beispielcode

Folgend nun eine Kopie einer vollständigen C# Quellcode Vorlage für Custom Controls:

```
using System;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Ink;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;

namespace maximagoControls
{
    // Uebergabe der XAML Elemente
    [TemplatePart(Name = "xMain", Type = typeof(Border))]

    public class Slider3D : ContentControl
    {
        // Dependency Properties
        public static readonly DependencyProperty FPSProperty =
            DependencyProperty.Register("FPS", typeof(int), typeof(Slider3D), new
            PropertyMetadata(10));
        public int FPS
        {
            set
            {
                this.SetValue(FPSProperty, value);
            }
            get
            {
                return (int)GetValue(FPSProperty);
            }
        }
    }
}
```

```
        {
            return (int) this.GetValue(FPSProperty);
        }
    }

    // Deklarationen fuer Control Elemente
    public Border xMain = new Border();

    public override void OnApplyTemplate()
    {
        // Uebernahme der XAML Elemente in den Code
        xMain = GetTemplateChild("xMain") as Border;
        // Event-Listener für Maus-Klick
        xMain.MouseLeftButtonDown += new
        MouseButtonEventHandler(xMain_MouseLeftButtonDown);
    }

    public Slider3D()
    {
        this.DefaultStyleKey = typeof(Slider3D);
    }

    void xMain_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)
    {
        throw new NotImplementedException();
    }
}
```