

1. Fachkonferenz rund um Grafische Benutzeroberfläche und Design

GUI&Design



WPF-/Silverlight-Controls designen und entwickeln

Thomas Claudius Huber



Veranstalter:  **ppedv**

Über Thomas Claudius Huber

www.thomasclaudiushuber.com

- Senior Consultant
 - Trivadis AG Basel
 - Fokus: .NET, WPF, Silverlight, WinForms, Architecture
- Autor der umfassenden Handbücher zur WPF und Silverlight



WPF-/Silverlight-Controls designen und entwickeln

- Das Aussehen anpassen
- Logik und Aussehen verbinden
- Controls aus Sicht des Entwicklers



WPF/Silverlight-Controls sind „lookless“

- das Aussehen ist von der Logik getrennt
- Das Aussehen eines Controls wird über ein ControlTemplate (XAML) definiert
- Die Logik wird klassisch in C# oder VB.NET implementiert

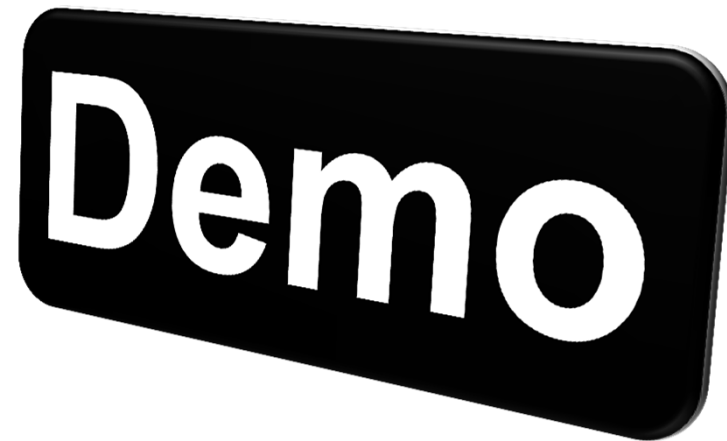


Das Aussehen eines Buttons in XAML anpassen

Demo



Das Aussehen eines Buttons in Expression Blend anpassen



WPF-/Silverlight-Controls designen und entwickeln



- Das Aussehen anpassen
- **Logik und Aussehen verbinden**
- Controls aus Sicht des Entwicklers

Logik und Aussehen verbinden

- Visuelle Zustände via Trigger (nur WPF) oder via Visual State Manager (VSM) einbinden
- Dependency Properties via TemplateBinding im ControlTemplate referenzieren
- PART-Elemente, die vom Control gefordert werden, unterbringen



Visuelle Zustände via Trigger

- Werden nur von der WPF, nicht jedoch von Silverlight unterstützt
- Lassen sich auch einfach direkt in XAML erstellen
- Ein Trigger kann
 - eine Property eines Elements im Template setzen
 - eine Animation auslösen (welche eine Property eines Elements im Template animiert)



Visuelle Zustände via Trigger

Demo



Visuelle Zustände via Visual State Manager

- Der Visual State Manager hat seinen Ursprung in Silverlight
 - wurde in .NET 4.0 auch in die WPF übernommen
- Ist gegenüber Triggern Designer-freundlicher
 - Anstatt mit Events und Properties wird mit tatsächlichen Zuständen gearbeitet
- Lässt sich aufgrund etwas mehr XAML-Code nicht leicht „von Hand“ schreiben



Visuelle Zustände via Visual State Manager

- Ein Visual State beschreibt einen visuellen Zustand eines Controls
- Visual States lassen sich gruppieren
 - Aus jeder Gruppe ist genau ein Visual State aktiv
- Visual States werden auf Klassenebene mit dem `TemplateVisualState`-Attribute deklariert.



Visuelle Zustände via Visual State Manager

- Expression Blend bietet visuelle Unterstützung für das States-Modell

Demo



Visuelle Zustände via Visual State Manager

- Die Klasse VisualStateManager definiert die Attached-Property VisualStateGroups
- In einer VisualStateGroup werden VisualStates und VisualTransitions untergebracht.
- Ein VisualState definiert eine Animationen für einen Zustand, eine Visual Transition eine Animation für einen Zustandsübergang



Logik und Aussehen verbinden

- Visuelle Zustände via Trigger (nur WPF) oder via Visual State Manager (VSM) einbinden
- Dependency Properties via TemplateBinding im ControlTemplate referenzieren
- PART-Elemente, die vom Control gefordert werden, unterbringen



Dependency-Properties via TemplateBinding referenzieren

```
<Style TargetType="Button">
  ...
  <Setter Property="Template">
    <Setter.Value>
      <ControlTemplate TargetType="Button">
        <Grid>
          <Rectangle Fill="{TemplateBinding Background}" .../>
          <ContentPresenter Content="{TemplateBinding Content}"
            HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}"
            VerticalAlignment="{TemplateBinding VerticalContentAlignment}"/>
        </Grid>
      </ControlTemplate>
    </Setter.Value>
  </Setter>
</Style>
```



PART-Elemente im ControlTemplate unterbringen

- Parts definieren genau benannte Elemente in einem ControlTemplate
 - Eine Control-Klasse sucht im ControlTemplate nach diesen Elementen
- Parts sind auf Klassenebene mit dem TemplatePart-Attribut beschrieben
 - ProgressBar enthält die Parts PART_Track und PART_Indicator
- Im Zusammenhang mit den Visual States spricht man auch vom „Parts & States“-Modell



PART-Elemente im ControlTemplate unterbringen

Demo



WPF-/Silverlight-Controls designen und entwickeln



- Das Aussehen anpassen
- Logik und Aussehen verbinden
- **Controls aus Sicht des Entwicklers**

PART-Elemente nutzen

- OnApplyTemplate-Methode überschreiben
- Elemente in Klassenvariablen speichern und nutzen
- Klasse mit TemplatePart-Attributen ausstatten

Demo



Visual States unterstützen

- Zentrale Methode für einen Statusübergang erstellen
- Klasse mit TemplateVisualState-Attributen ausstatten
- Anfangsstatus bereits in der OnApplyTemplate-Methode setzen

Demo



WPF-/Silverlight-Controls designen und entwickeln



- Das Aussehen anpassen
- Logik und Aussehen verbinden
- Controls aus Sicht des Entwicklers

Conclusion

- ControlTemplates werden über PART-Elemente, TemplateBinding und VisualStates verbunden
- Trigger werden nur von der WPF unterstützt
- Lässt sich das Control in Expression Blend designen, sind die Attribute richtig gesetzt.



Slides und Kontakt

- Slides sind heute Abend verfügbar unter www.thomasclaudiushuber.com/blog
- Kontakt: thomas@thomasclaudiushuber.com



1. Fachkonferenz rund um Grafische Benutzeroberfläche und Design

GUI & Design



FRAGEN?



Veranstalter:  **ppedv**

1. Fachkonferenz rund um Grafische Benutzeroberfläche und Design

GUI & Design



Hat Ihnen mein Vortrag gefallen?
Ich freue mich auf Ihr Feedback!



Veranstalter:  **ppedv**

Wir sehen uns wieder!



building & connecting Know-how

16.-17. Februar 2011 in München

.NET, Visual Studio, SharePoint & more!

www.VSone.de



Trainings und Events der ppedv

Mehr als 90 verschiedene Trainings

auf Microsoft-Technologien spezialisiert

11 Standorte in D & AT

Maßgeschneiderte Trainings

direkt bei Ihnen vor Ort!

www.ppedv.de



GUI&Design

1. Fachkonferenz rund um Grafische Benutzeroberfläche und Design

GUI & Design



Vielen Dank!



Veranstalter:  **ppedv**