Blend für Nicht-Designer

23.06.2010 – Stefan Lange

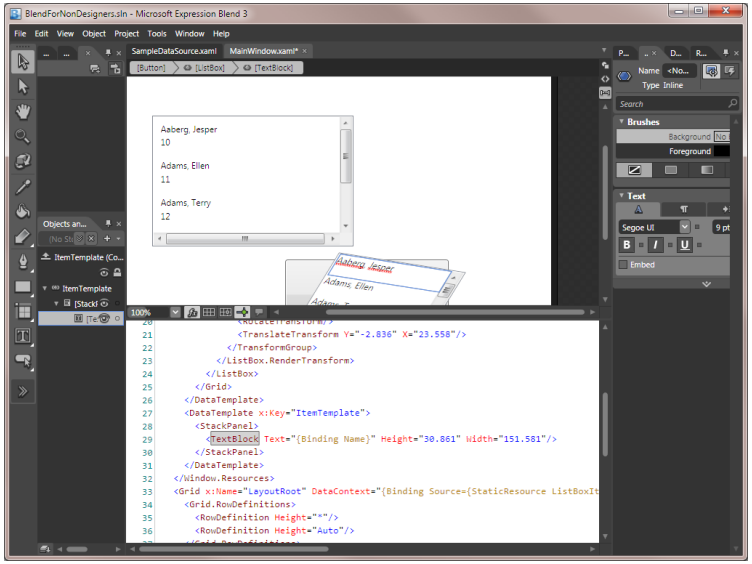
EMail: Stefan.Lange@empira.de
Firma: empira Software GmbH
Blog: www.st-lange.net

Agenda

- XAML
- Styles
- Data Templates
- Control Templates
- Custom Controls
- Visual States
- Behaviors, Trigger und Actions
- SDKs
- Fragen
- …

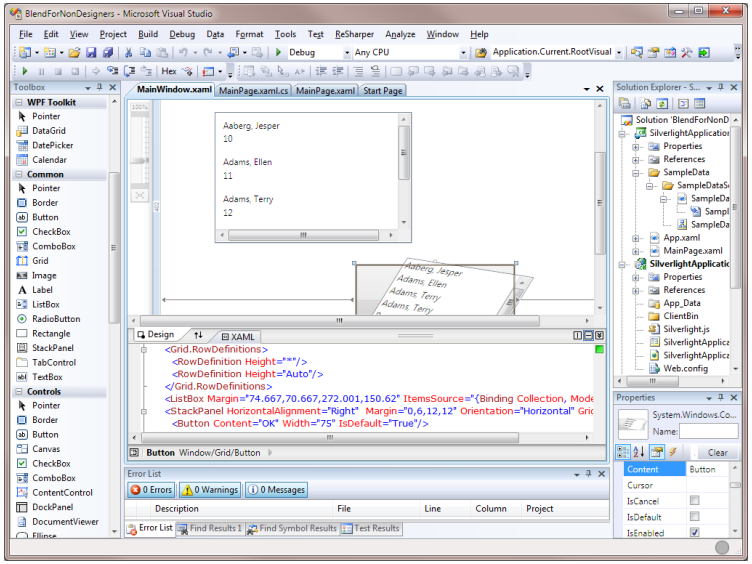
Theoretische Trennung

Expression Blend



Designer

Visual Studio



Developer

- Das hat Microsoft (leider) mal so verkündet…

Fokus dieses Vortrags

- „Es geht um Aufgaben, die (typischerweise) Entwickler tun, die aber mit Visual Studio praktisch nicht möglich sind“
- Beschränkung auf
  - Visual Studio 2010 und Blend 4
  - Nur Silverlight 4, kein WPF

Visual Studio 2010

- Verwendet jetzt ähnlichen Editor wie Blend
- Neuer Property Explorer
- Interaktive Vorschau
- Blend geht jedoch sehr viel weiter

Allgemeines

- Blend und Visual Studio können gleichzeitig an einer Solution arbeiten
- Korrespondierende Versionen:
  - Blend 3 und Visual Studio 2008
  - Blend 4 und Visual Studio 2010
- Die neue Combo (Blend 4 und Visual Studio 2010) kann alle Aufgaben der alten Combo übernehmen

DEMO

- Kurzüberblick Visual Studio / Blend

Stärken von Visual Studio

- Sehr viel besserer Editor für Code (C#, VB)
- Sehr viel besseres Refactoring
- Viel besserer Editor für XAML
- Debugging
- Projekt Verwaltung
  - insb. Solution Folder
- Fast ausschließlich Tastatur bedienbar
- Add-Ons (z.B. Resharper)

Stärken von Blend

- Visuelle Bearbeitung
  - besseres Databinding im Design-Mode
- Bearbeitung von Templates
- Testdaten Generator
- Visual States Editor
- Timeline Editor
- Sketching

Styles

Informelle Definition: Style

- Ein Style ist eine Sammlung von Schlüssel/Wert-Paaren, die den Properties eines UI-Elements zugewiesen werden können

```
<Style x:Key="NameOfThisStyle" TargetType="Button">
 <Setter Property="Background" Value="#FFF7F8FA"/>
 <Setter Property="Padding" Value="3,0,3,0"/>
 <Setter Property="BorderThickness" Value="1"/>
 <Setter Property="BorderBrush">
  <Setter.Value>
   <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
    <GradientStop Color="#FFA3AEB9" Offset="0"/>
    <GradientStop Color="#FF8399A9" Offset="0.375"/>
    <GradientStop Color="#FF718597" Offset="0.375"/>
    <GradientStop Color="#FF617584" Offset="1"/>
   </LinearGradientBrush>
  </Setter.Value>
 </Setter>
</Style>
```

DEMO

- Simple Styles
- DataGrid Column Styles setzen
- Implicit Styling
- Vererbung

Informelle Definition: Resource

- Eine Resource ist eine Sammlung von benannten wiederverwendbaren Objekten

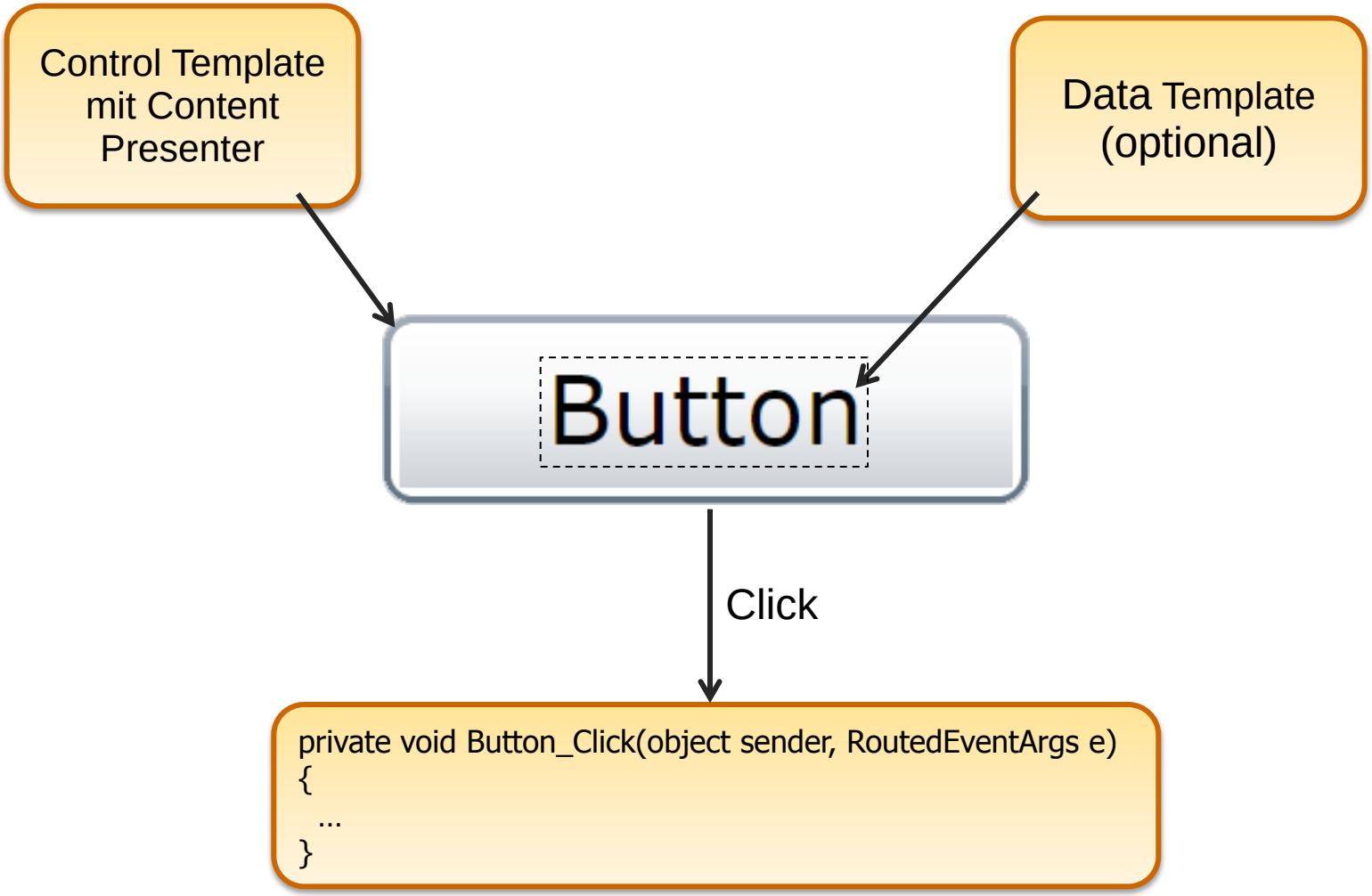
```
<ResourceDictionary
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
 <Style x:Key="ButtonStyle1" TargetType="Button"/>
 <SolidColorBrush x:Key="TitleBrush" Color="ForestGreen"/>
 <DataTemplate x:Key="itemstemplate">
  …
 </DataTemplate>
 <Style x:Key="ListBoxItemStyle1" TargetType="ListBoxItem">
  …
 </Style>
 <ItemsPanelTemplate x:Key="ItemsPanelTemplate1">
  …
 </ItemsPanelTemplate>
 <Style x:Key="ButtonStyle2" TargetType="Button">
  …
 </Style>
</ResourceDictionary>
```

DEMO

- Edit Resources
- Rename Resource

Content Controls

Aspekte eines Content Controls



Anatomie eines Buttons (I)

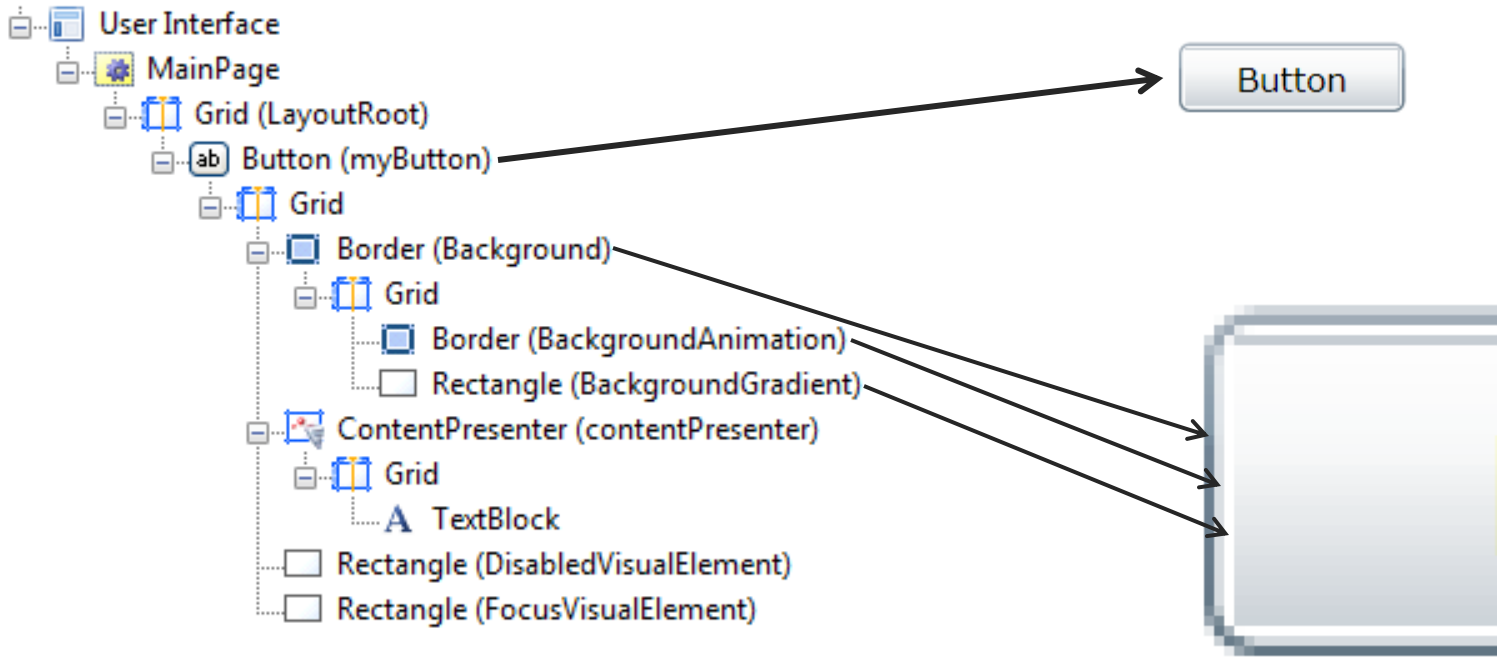
- Button-Definition in XAML

```
<Grid x:Name="LayoutRoot">
   <Button x:Name="myButton" Content="Button" Height="23" Width="75" />
</Grid>
```

- Aussehen im Browser

Anatomie eines Buttons (II)

- Visual Tree des Buttons (mittels Silverlight Spy)



Anatomie eines Buttons (III)

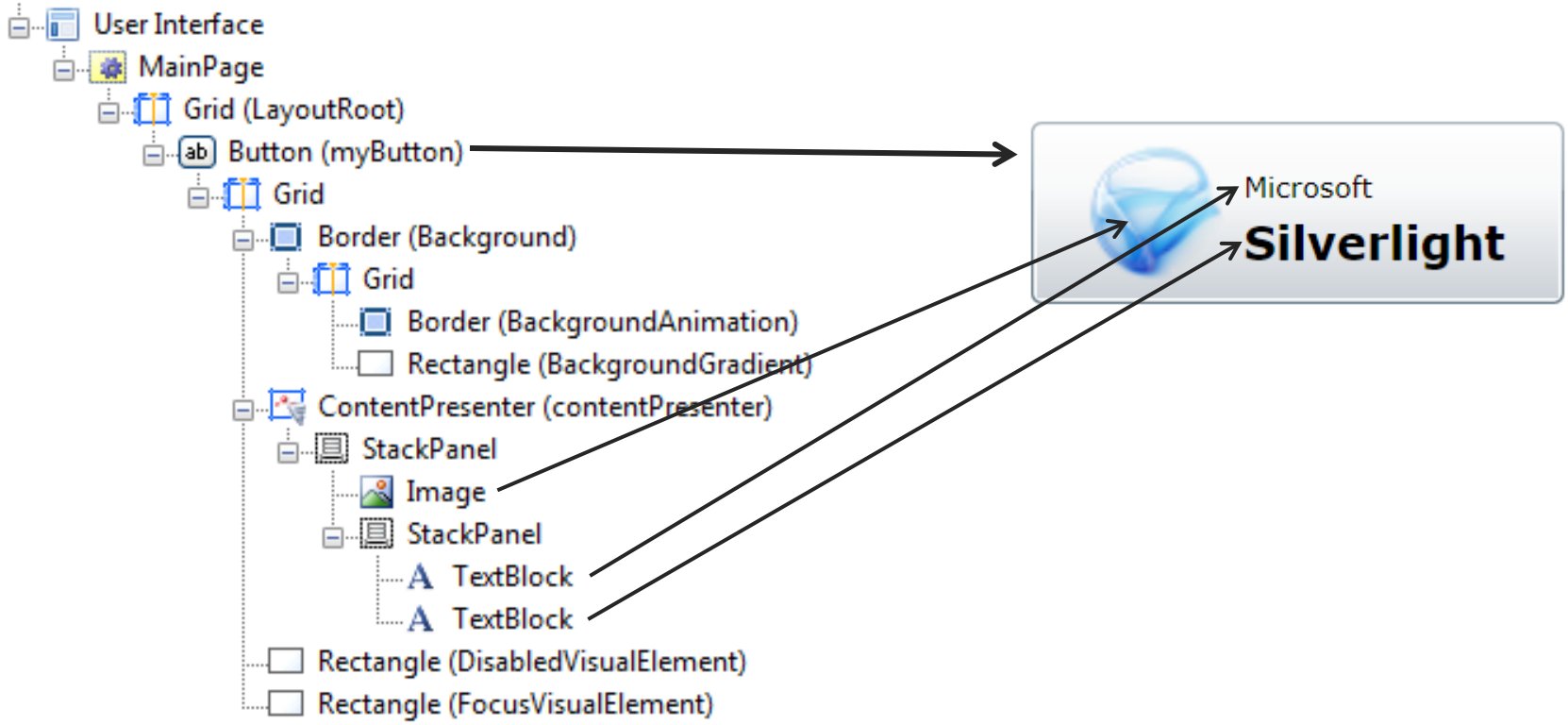
- Button-Definition in XAML

```
<Grid x:Name="LayoutRoot">
 <Button x:Name="myButton" Padding="15 10">
  <StackPanel Orientation="Horizontal">
   <Image Source="/assets/Silverlight-Logo.png“
    Height="46" Width="50" Margin="0 0 5 0"/>
   <StackPanel VerticalAlignment="Bottom">
    <TextBlock FontSize="10">Microsoft</TextBlock>
    <TextBlock FontSize="16" FontWeight="Bold">Silverlight</TextBlock>
   </StackPanel>
  </StackPanel>
 </Button>
</Grid>
```

- Aussehen im Browser

Anatomie eines Buttons (IV)

- Visual Tree des Buttons (mittels Silverlight Spy)



Informelle Definition: Template

- Ein Template ist eine Baumstruktur aus UI-Elementen, die das Aussehen und Layout von Controls festlegt

Unterschied Style und Template

- Ein Style ändert die Properties vorhandener UI-Elemente
- Ein Template definiert oder ersetzt einen Teilbaum von UI-Elementen
- Mit einem Style kann man natürlich auch Templates setzen

DEMO

- Template eines Buttons austauschen
- RichTextBlock in Tooltip verwenden

Import von externen Tools

- Import aus Photoshop
- Import aus Illustrator

Tipp

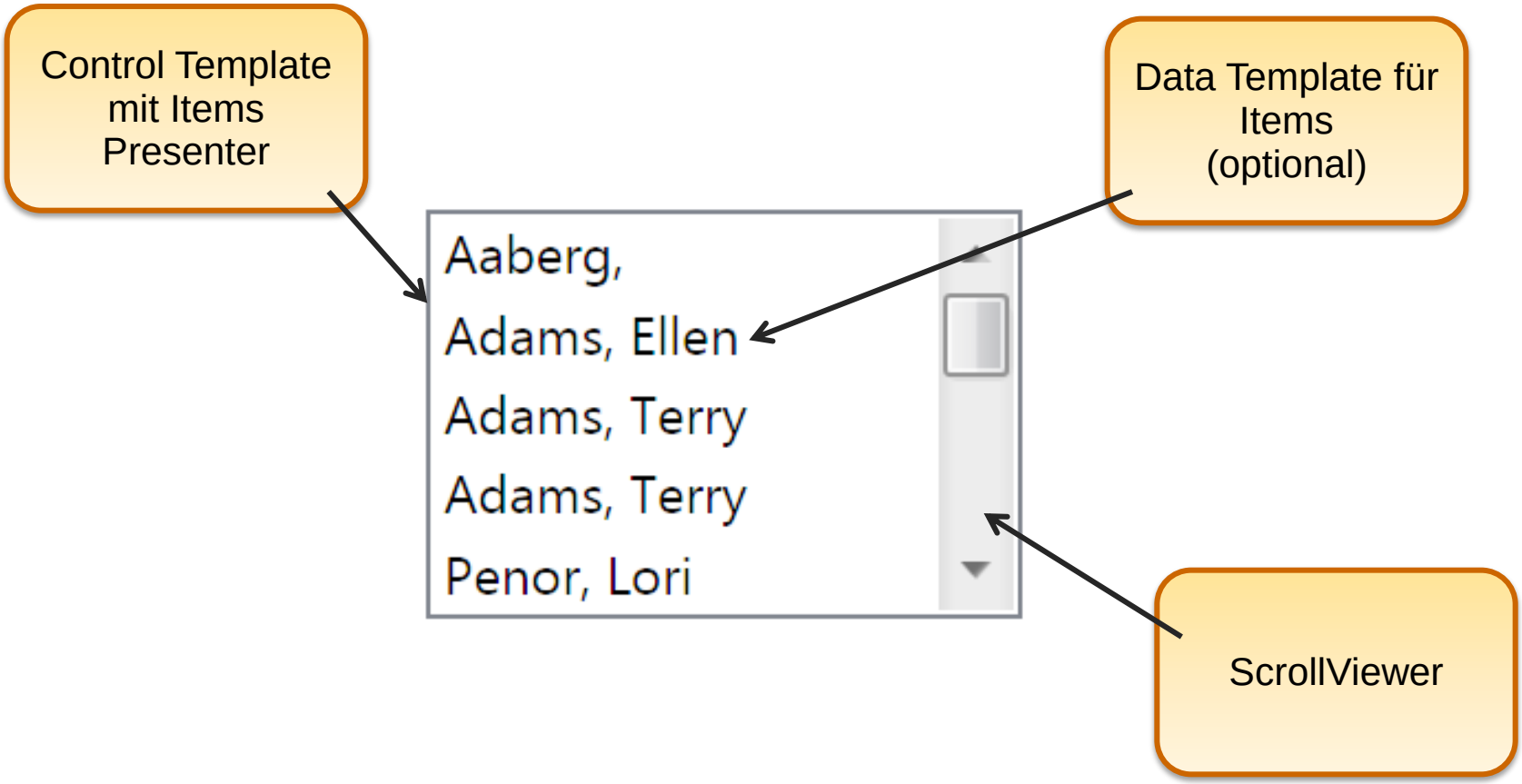
- Datei mit Expression Design vorbereiten

DEMO

- Import aus Illustrator (< Version 10)
- Make into Control

Items Controls

Aspekte eines Items Controls



DEMO

- Listbox mit DataTemplate
- Test-Daten
- DataBinding
- Panel Template

Visual States

Visual States

- Common States (State Group)
  - Normal (State)
  - MouseOver (State)
  - Pressed (State)
  - Disabled (State)
- Focus States (State)
  - Unfocused (State)
  - Focused (State)

Silverlight

Visual State Manager + GotoState Aufruf in Code

WPF

(Data) Trigger in XML

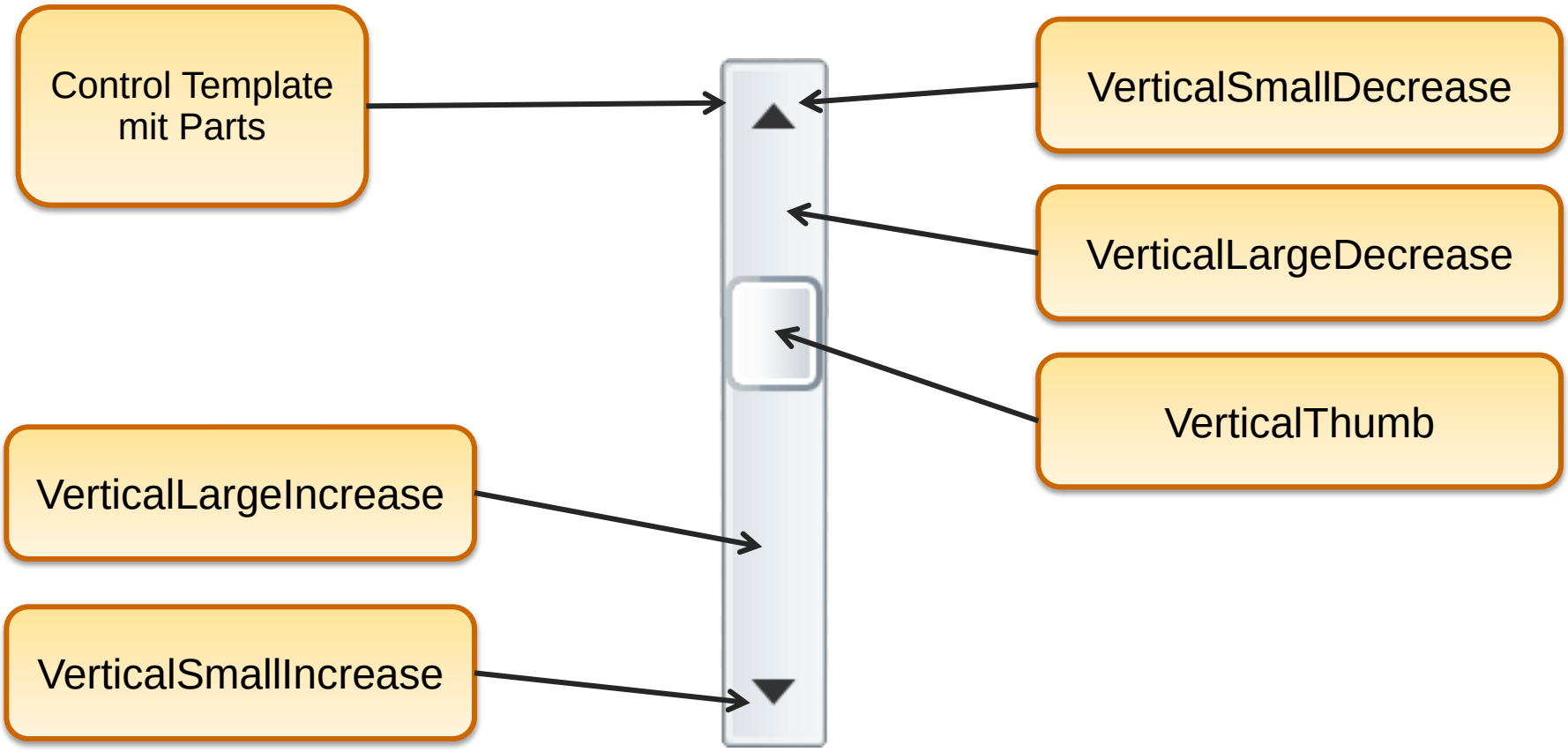
(Visual State Manger ab WPF 4.0)

DEMO

- Visual States bearbeiten
- Zuweisung von Styles

Control Parts

Controls mit Parts



DEMO

- Control Parts bearbeiten

Themes

Informelle Definition: Theme

- Ein Theme ist eine Sammlung von Styles und Templates für alle Controls (eines Projektes) mit einem (üblicherweise) gemeinsamem Look

DEMO

- 3 neue Themes von Microsoft:
  - Windows 7
  - Cosmopolitan
  - Accent Colors

Animations

DEMO

- Anlegen eines Storyboards

Behaviors & Triggers

Voraussetzungen

- Blend SDK wird benötigt (Teil der Blend Installation)
  - also **nicht** in Visual Studio oder SDK enthalten

- Man benötigt also Blend 4
- Projekte kompilieren in Visual Studio, sofern durch Blend die notwendigen Assemblies installiert wurden

Behaviors und TriggerActions

- Objekte, die Aktionen repräsentieren
- Ersatz für die fehlenden (WPF) Trigger in Silverlight
  - (Silverlight kennt nur EventTrigger)
- Programmierer definieren Standardfunktionalität in Form von Code
- Designer nutzen diese Funktionalität ohne coden zu müssen

Behavior<T>

- Man muss nur zwei Funktionen überschreiben:
  - OnAttached
  - OnDetaching
- Dabei abonniert man ein Event des „attachten“ Objeks
- Klingt auf den ersten Blick ziemlich banal

DEMO

- Einfaches „MakeGreenBehavior“

TriggerAction<T>

- Repräsentiert eine Aktion, die von einem EventTrigger ausgelöst werden kann
- Ähnlich wie Behavior, aber flexibler konfigurierbar
- Man muss zusätzlich noch „Invoke“ überschreiben
- Man kann das die Aktion auslösende Event selber festlegen

TargetedTriggerAction<T>

- Ähnlich TriggerAction
- Führt die Aktion an einem anderen Objekt aus

DEMO

- Einfache „FlipObjectTargetedTriggerAction“

Fluid-Behaviors

- Gibt’s schon seit Silverlight 3
- „Fließender“ Übergang von einem Zustand in den anderen

Layout States

Layout States

- Neues Feature in Silverlight 4
  - Gibt's noch nicht mal in WPF 4 ☺
- Neue Visual State Group mit 3 States:
  - BeforeLoaded
  - AfterLoaded
  - BeforeUnloaded

DEMO

- Listbox mit „fließenden Items“

FluidMoveSetTagBehavior

- Eines von neuen coolen Behaviors in Silverlight 4
- Arbeitet mit FluidMoveBehavior zusammen, indem es… - besser vorführen!

DEMO

- FluidMoveSetTagBahavior
- TransitionEffect
- GoToStateAction

DEMO

- Dynamic Layout Sample (Mix 2010)

Sketching

Blend SDK

Silverlight Smooth Streaming SDK

Generelle Tipps

- Blend zur Analyse von Controls nutzen
- Bestehende Controls erweitern, nicht neu schreiben
- Code oft zwischenspeichern
  (falls Ihr einen Fehler eingebaut habt)

Fazit

- Blend ist ein sehr sehr mächtiges Werkzeug
- Für Developer genauso wichtig wie Visual Studio

Links

- Stefan.Lange@empira.de
- Unterlagen zu dieser Session
  www.st-lange.net