ERGOSIGN

WPF UI DEVELOPMENT BEST PRACTICES

David C. Thömmes
Software Engineer
Lead Software Engineering Standards
thoemmes@ergosign.de

Ergosign GmbH

WPF = BESSERE & SCHÖNERE UI'S

Overview | Process CW | Process TBG | Process Alpha | Hydraulic | Oil Tanks | Settings

Param 1	220 c	Length	55mm
Heap:	-99	Tailout	8
ID:	#1235	Slab	13:45
Volt:	12v	Time	NaN

Length 2	567mm
Tailout 2	83
Slab 2	---
Time 2	NaN

#1

Level #1 | Level #2 | Level #3 | Level #4

Stop | Open Setting

Coax --

Function Point

Retriv

Action 1 | Action 2 | Action 3

90 °C

Monday

10/10/2012 - 13:45

Overview Gate 1

Quad	656
Fast	24536.657567
GHF	MODE
Net	12324
Kmu	

Major 1 Gate Overview

Param 1	86	Length	3452	Length 2	976
Heap:	2234	Tailout	8523	Tailout 2	324
ID:	2321	Slab	#	Slab 2	1234
Volt:	534	Time	def.error	Time 2	245

Major

Major #1 | Major #2 | Major #3 | Major #4

THC System

Eloo	55
Running	34
Error	12
Remaining	334
Main	

Temp Surro Gate

Torrent	6
Floated	27
Released	30
Heat	4

Temp Surro Gate

Clean up

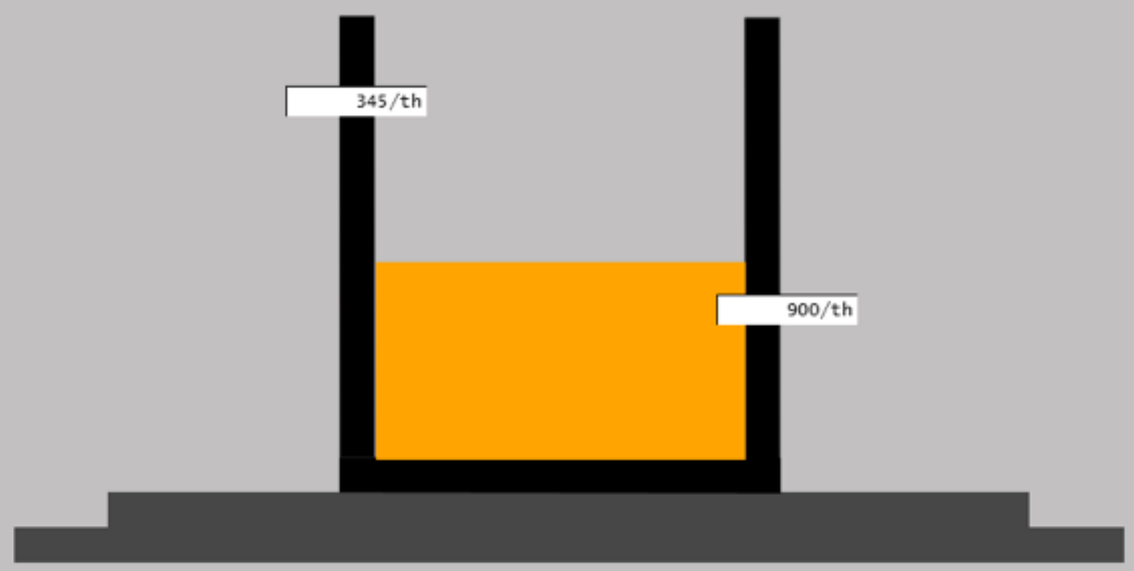
Inform

Help

Rollout

Swap

ABC Test



Param 1	86	Length	3452	Length 2	976
Heap:	2234	Tailout	8523	Tailout 2	324
ID:	2321	Slab	#	Slab 2	1234
Volt:	534	Time	def.error	Time 2	245

Major

Major #1 | Major #2 | Major #3 | Major #4

Weight Gater

Netto	1.5t
Brutto	1.7t
Burst	0.001mg
Swap	-

State

- Safe is ok
- Error Container #1
- Alert is ok
- Maincontainer
- Route is ok
- Hydro is ok
- Giga is ok
- Mega is ok
- Herz is neutral
- CNN is ok
- BBCx is ok
- Mount error

```
10/10/2012 - 13:45 :: Shift Log :: Performance Filter - Down Time up to 6c net frame fps:80 87 64 ach gx                    ACK
10/10/2012 - 13:45 :: Shift Log :: Performance Filter - Down Time up to 6c net frame fps:80 87 64 ach gx                    ACK
10/10/2012 - 13:45 :: Shift Log :: Performance Filter - Down Time up to 6c net frame fps:80 87 64 ach gx                    ACK
10/10/2012 - 13:45 :: Shift Log :: Performance Filter - Down Time up to 6c net frame fps:80 87 64 ach gx                    ACK
10/10/2012 - 13:45 :: Shift Log :: Performance Filter - Down Time up to 6c net frame fps:80 87 64 ach gx                    ACK
10/10/2012 - 13:45 :: Shift Log :: Performance Filter - Down Time up to 6c net frame fps:80 87 64 ach gx                    ACK
10/10/2012 - 13:45 :: Shift Log :: Performance Filter - Down Time up to 6c net frame fps:80 87 64 ach gx                    ACK
10/10/2012 - 13:45 :: Shift Log :: Performance Filter - Down Time up to 6c net frame fps:80 87 64 ach gx                    ACK
```

Quit | Condor Mode | FCK free | UX Mode | Torrent Car

Overview | Process CW | Process TBG | Process Alpha | Hydraulic | Oil Tanks | Settings

Param 1	220 c	Length	55mm
Heap:	-99	Tailout	8
ID:	#1235	Slab	13:45
Volt:	12v	Time	NaN

Length 2	567mm		
Tailout 2	83		
Slab 2	---		
Time 2	NaN		

#1

Level #1 | Level #2 | Level #3 | Level #4 | Stop

Open Setting

Coax --

Function Point

Retriv

Action 1 | Action 2 | Action 3

90 °C

Monday

10/10/2012 - 13:45

Overview Gate 1

Quad	656
Fast	24536.657567
GHF	MODE
Net	12324
Kmu	

Major 1 Gate Overview

Param 1	86	Length	3452	Length 2	976
Heap:	2234	Tailout	8523	Tailout 2	324
ID:	2321	Slab	#	Slab 2	1234
Volt:	534	Time	def.error	Time 2	245

Major

Major #1 | Major #2 | Major #3 | Major #4

THC System

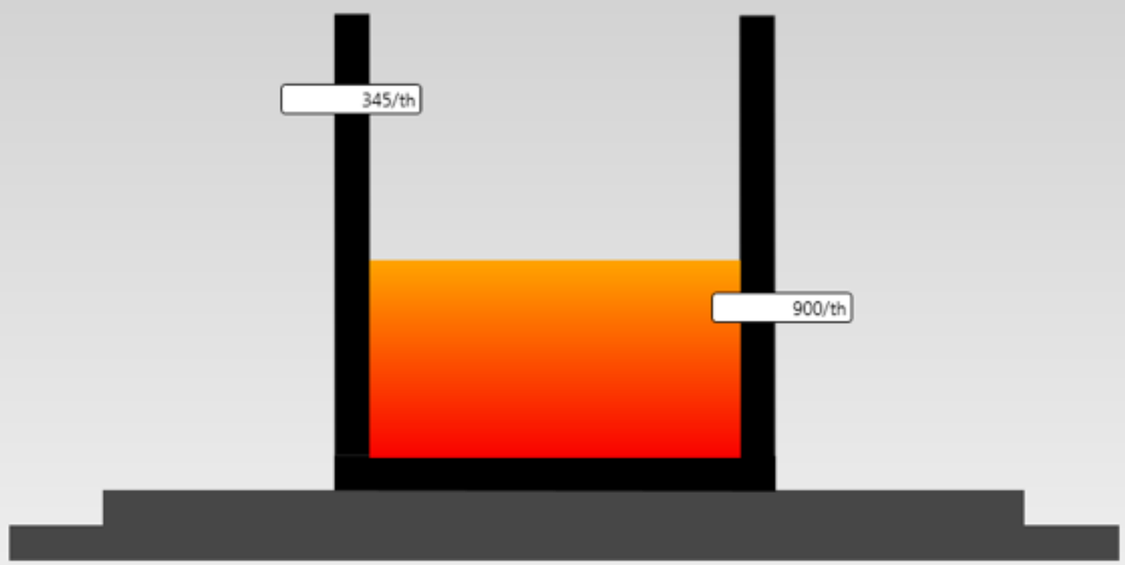
Eloo	55
Running	34
Error	12
Remaining	334
Main	

Temp Surro Gate

Torrent	6
Floated	27
Released	30
Heat	4

Weight Gater

Netto	1.5t
Brutto	1.7t
Burst	0.001mg
Swap	-



Temp Surro Gate

- Clean up
- Inform
- Help
- Rollout
- Swap
- ABC Test

State

- Safe is ok
- Error Container #1
- Alert is ok
- Maincontainer
- Route is ok
- Hydro is ok
- Giga is ok
- Mega is ok
- Herz is neutral
- CNN is ok
- BBCx is ok
- Mount error

Param 1	86	Length	3452	Length 2	976
Heap:	2234	Tailout	8523	Tailout 2	324
ID:	2321	Slab	#	Slab 2	1234
Volt:	534	Time	def.error	Time 2	245

Major

Major #1 | Major #2 | Major #3 | Major #4

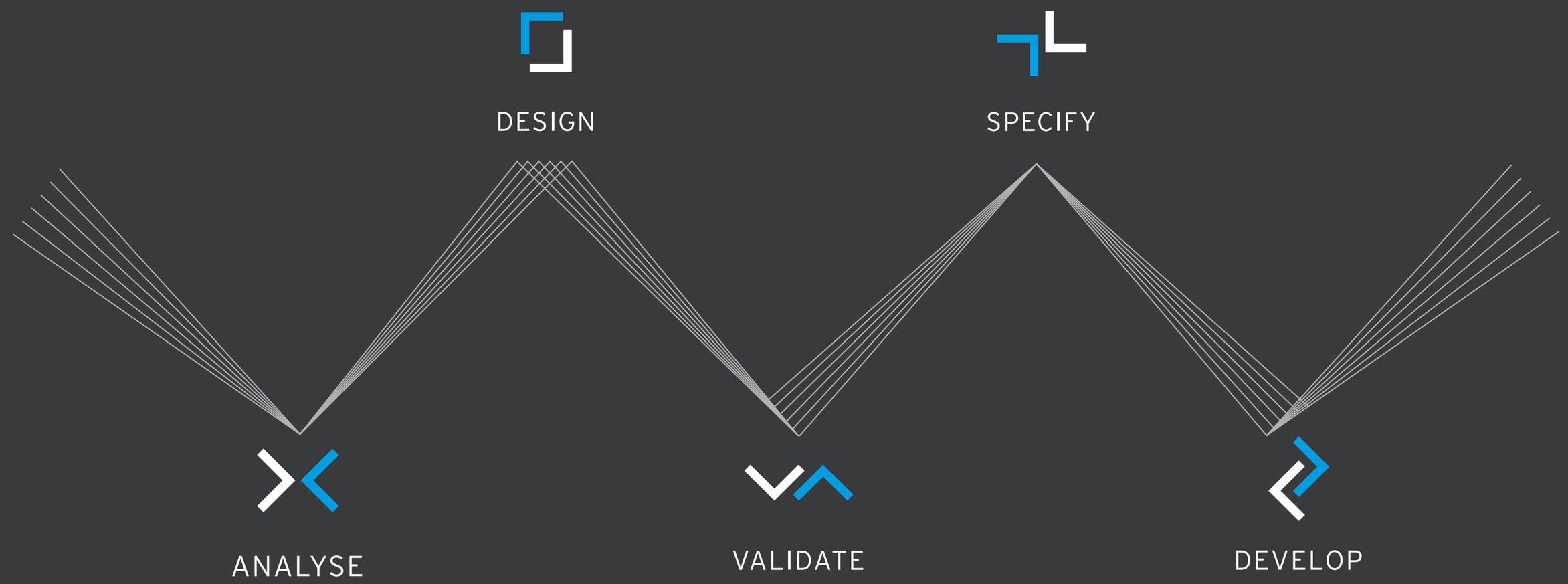

```
10/10/2012 - 13:45 :: Shift Log :: Performance Filter - Down Time up to 6c net frame fps:80 87 64 ach gx    ACK
10/10/2012 - 13:45 :: Shift Log :: Performance Filter - Down Time up to 6c net frame fps:80 87 64 ach gx    ACK
10/10/2012 - 13:45 :: Shift Log :: Performance Filter - Down Time up to 6c net frame fps:80 87 64 ach gx    ACK
10/10/2012 - 13:45 :: Shift Log :: Performance Filter - Down Time up to 6c net frame fps:80 87 64 ach gx    ACK
10/10/2012 - 13:45 :: Shift Log :: Performance Filter - Down Time up to 6c net frame fps:80 87 64 ach gx    ACK
10/10/2012 - 13:45 :: Shift Log :: Performance Filter - Down Time up to 6c net frame fps:80 87 64 ach gx    ACK
10/10/2012 - 13:45 :: Shift Log :: Performance Filter - Down Time up to 6c net frame fps:80 87 64 ach gx    ACK
```

Quit | Condor Mode | FCK free | UX Mode | Torrent Car

WPF = BESSERE & SCHÖNERE UI'S

WPF = BESSERE & SCHÖNERE UI'S

PROZESSE


DESIGN
SPECIFY
ANALYSE
VALIDATE
DEVELOP

ERGOSIGN

ANALYSE

ERGOSIGN

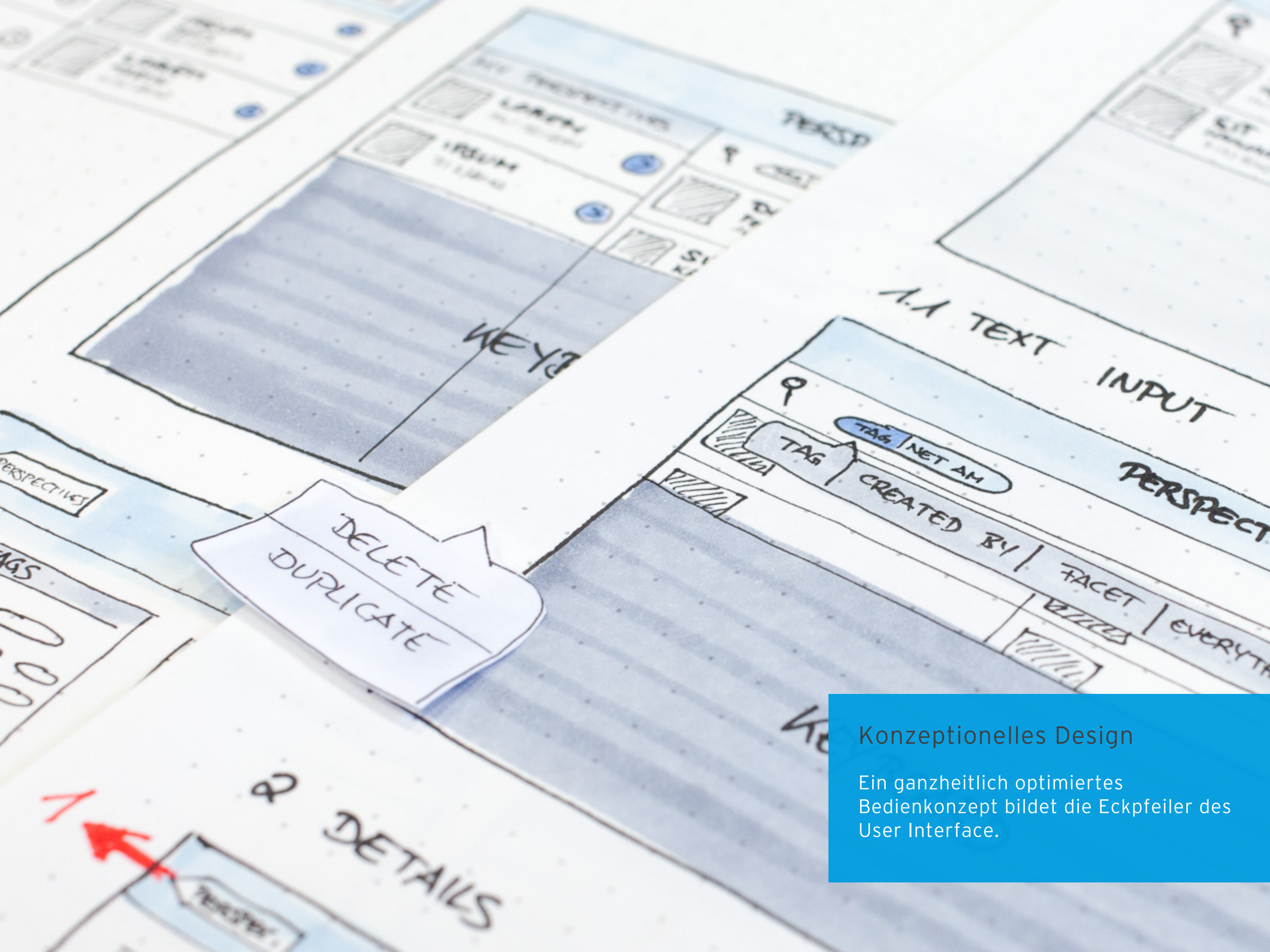
Kontextuelle Analyse

Beobachtungen und semistrukturierte Interviews bei Endanwendern vor Ort.

ERGOSIGN

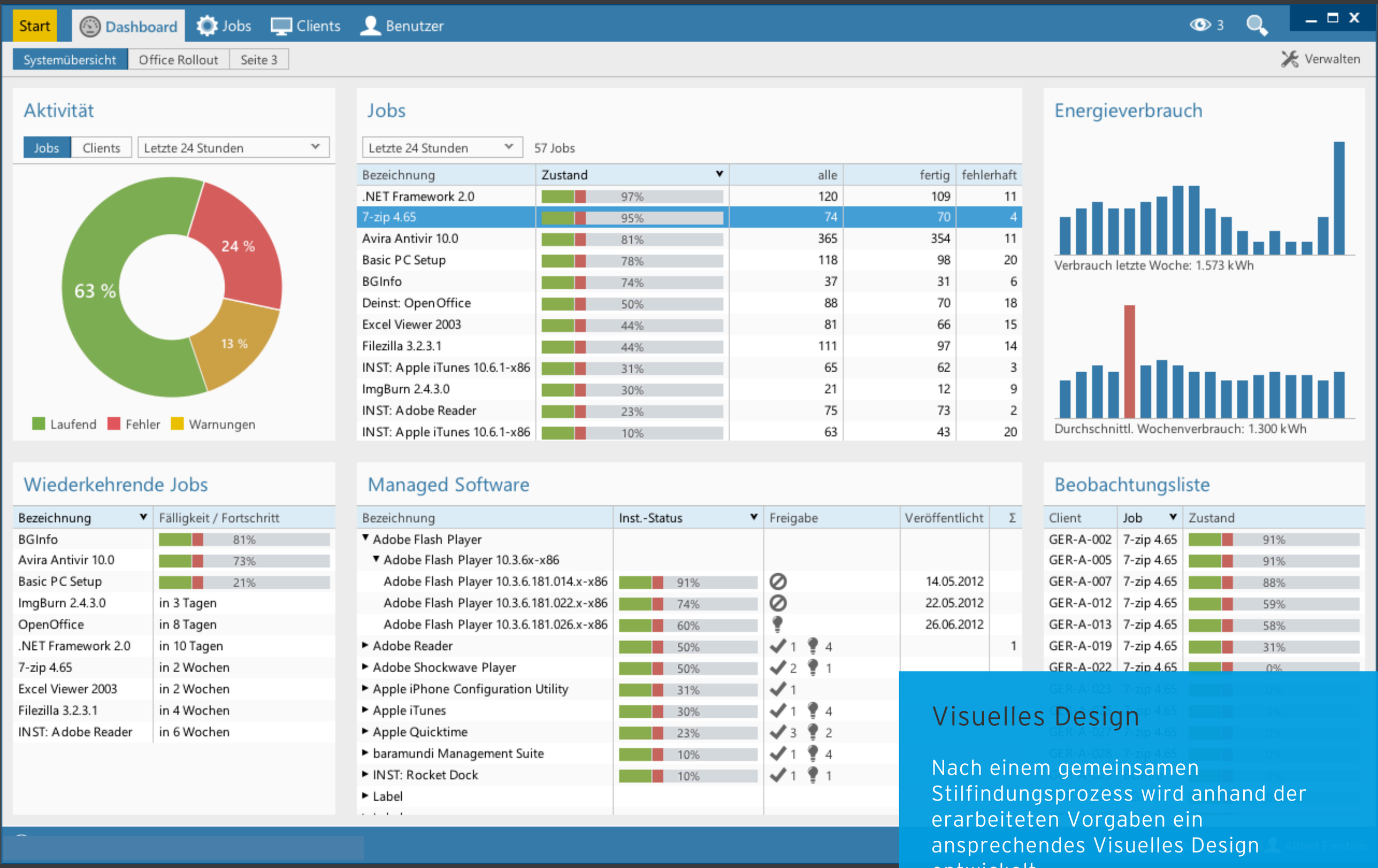
DESIGN

ERGOSIGN



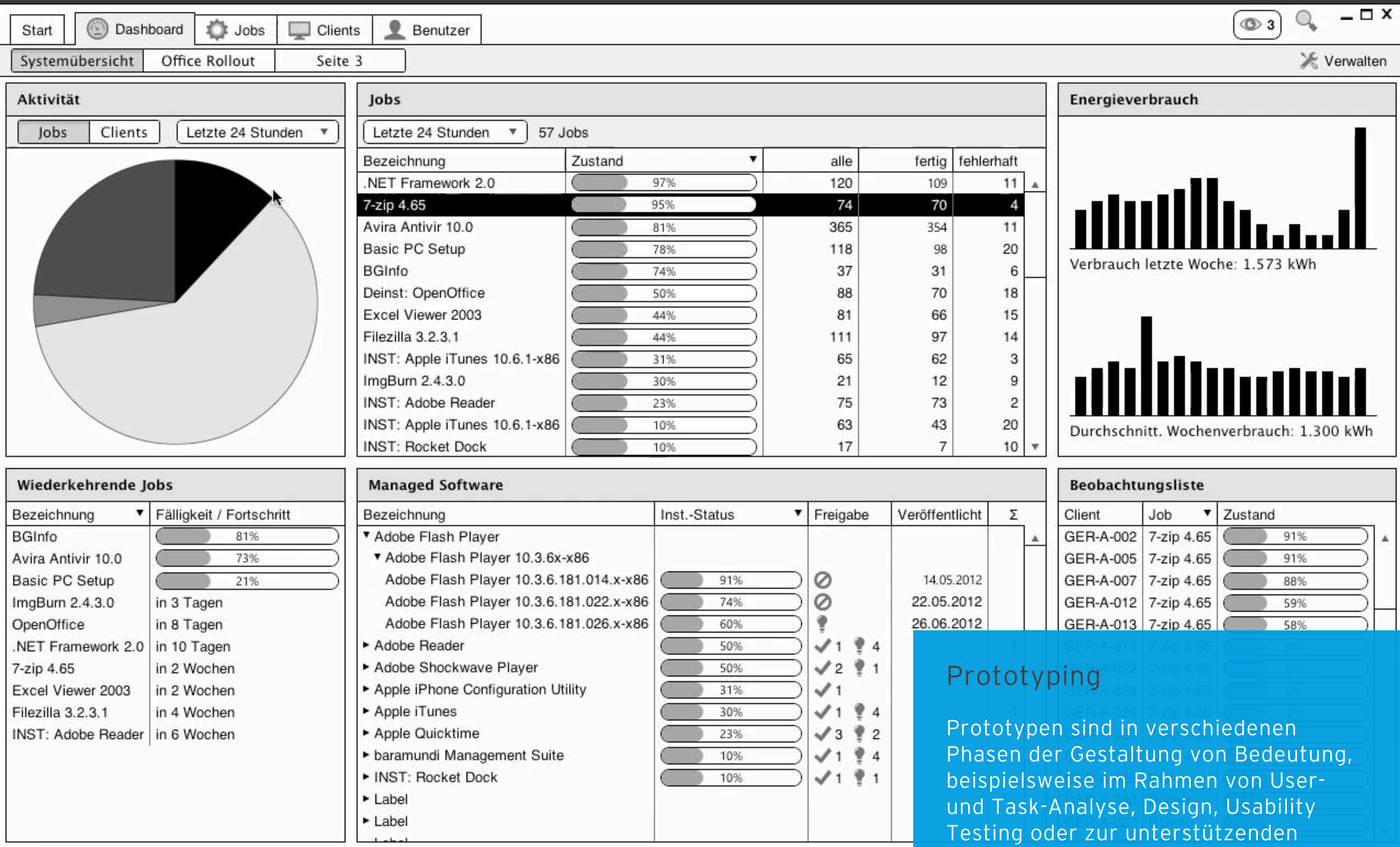
Konzeptionelles Design

Ein ganzheitlich optimiertes Bedienkonzept bildet die Eckpfeiler des User Interface.



Visuelles Design

Nach einem gemeinsamen Stilfindungsprozess wird anhand der erarbeiteten Vorgaben ein ansprechendes Visuelles Design entwickelt.



Prototyping

Prototypen sind in verschiedenen Phasen der Gestaltung von Bedeutung, beispielsweise im Rahmen von User- und Task-Analyse, Design, Usability Testing oder zur unterstützenden Dokumentation bei der Spezifikation.

ERGOSIGN

VALIDATE

ERGOSIGN

Usability Testing

Ein Usability Test mit repräsentativen Endanwendern liefert entscheidende Hinweise zur Optimierung eines User Interface.

ERGOSIGN

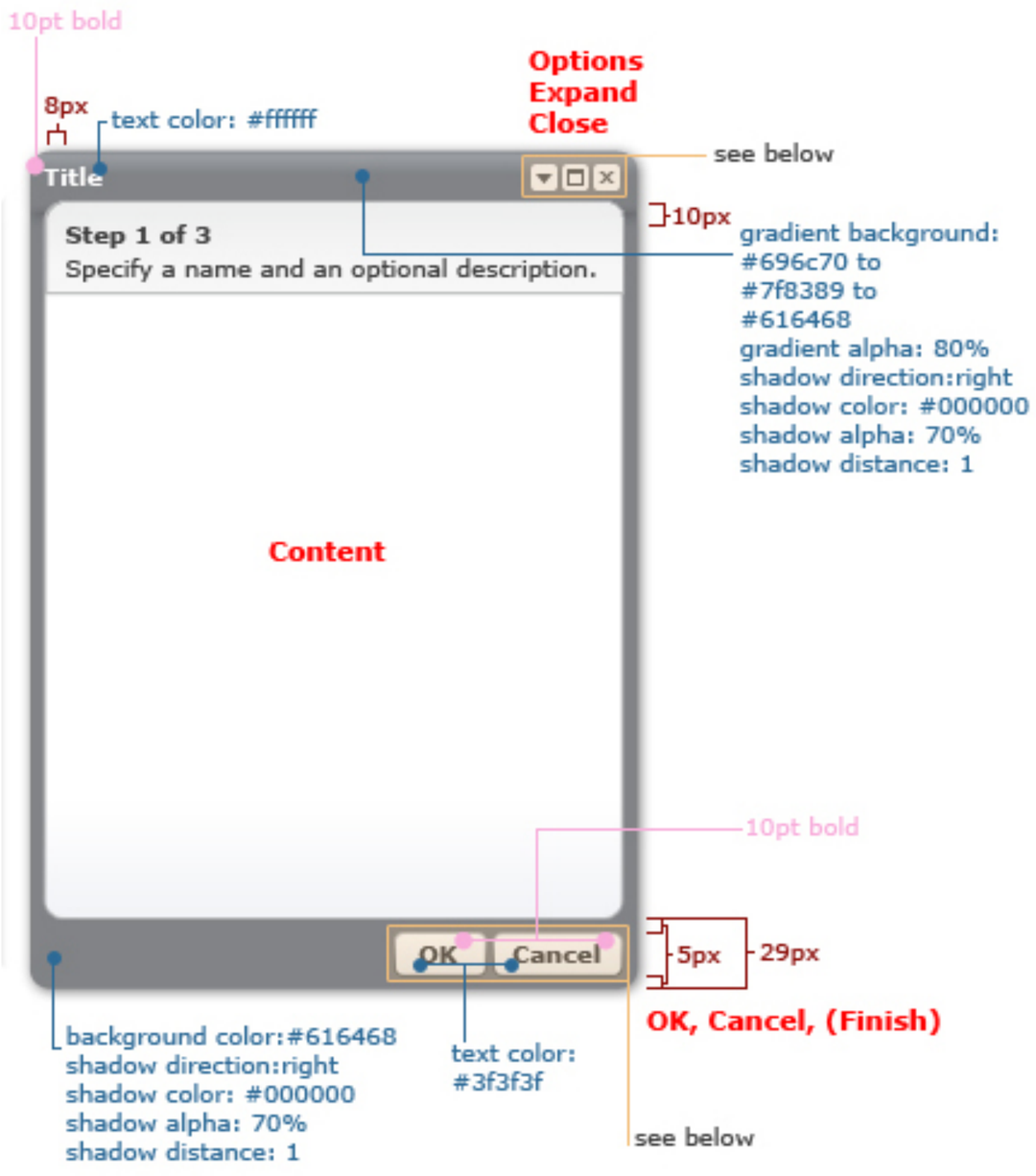
SPECIFY

ERGOSIGN

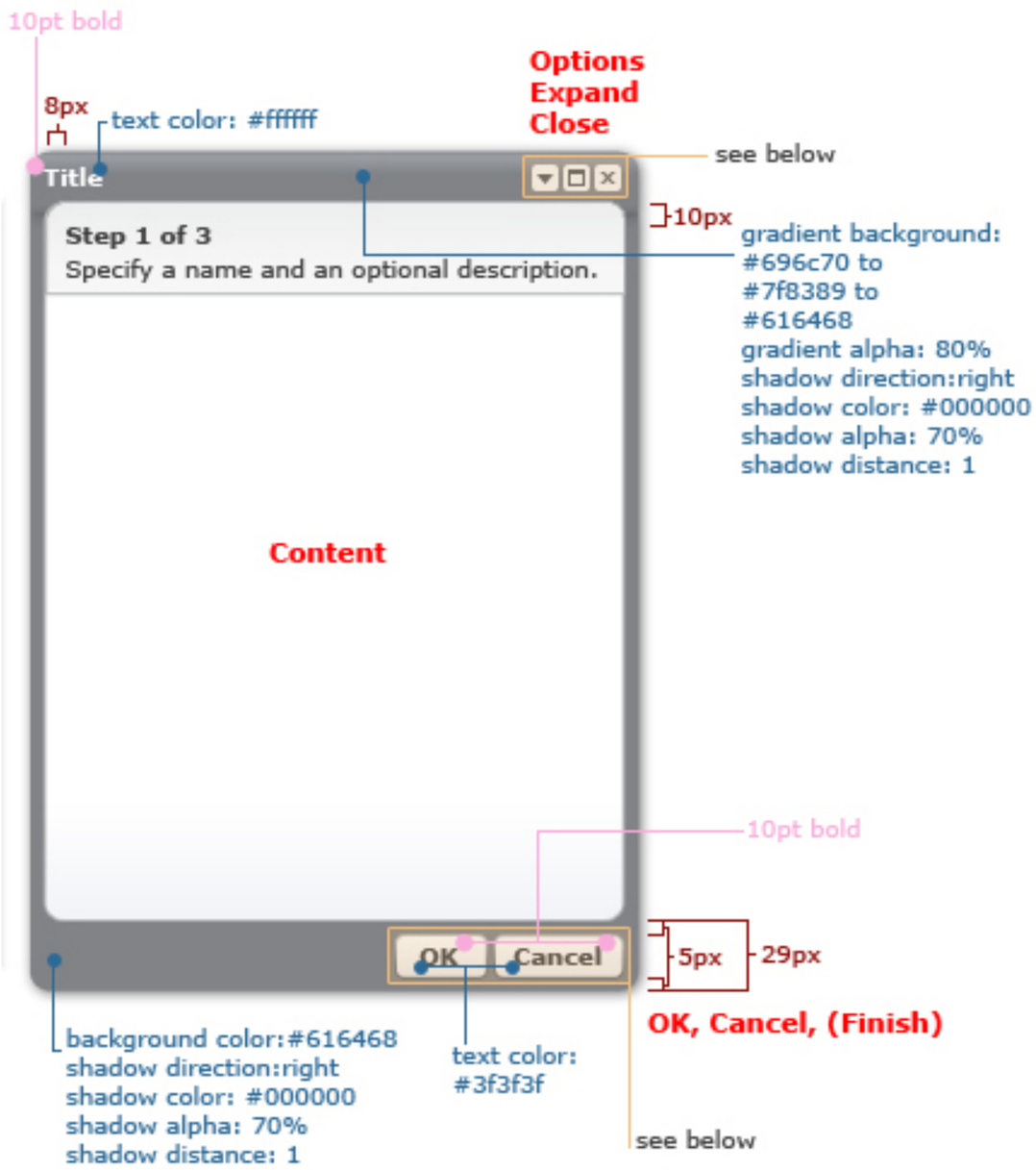
Design Manual

Ein Design Manual fasst die wichtigsten Design-Entscheidungen zusammen und verzichtet auf die Spezifikationen aller einzelnen Kontrollelemente.

Spezifikation

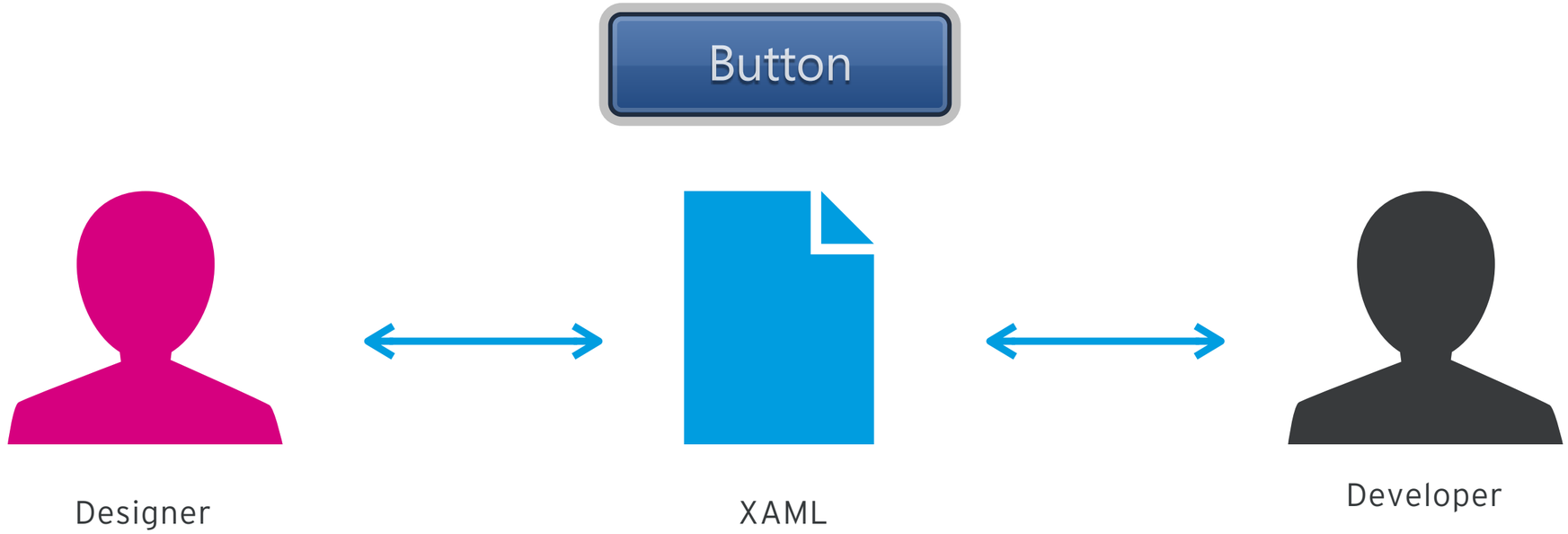


Spezifikation

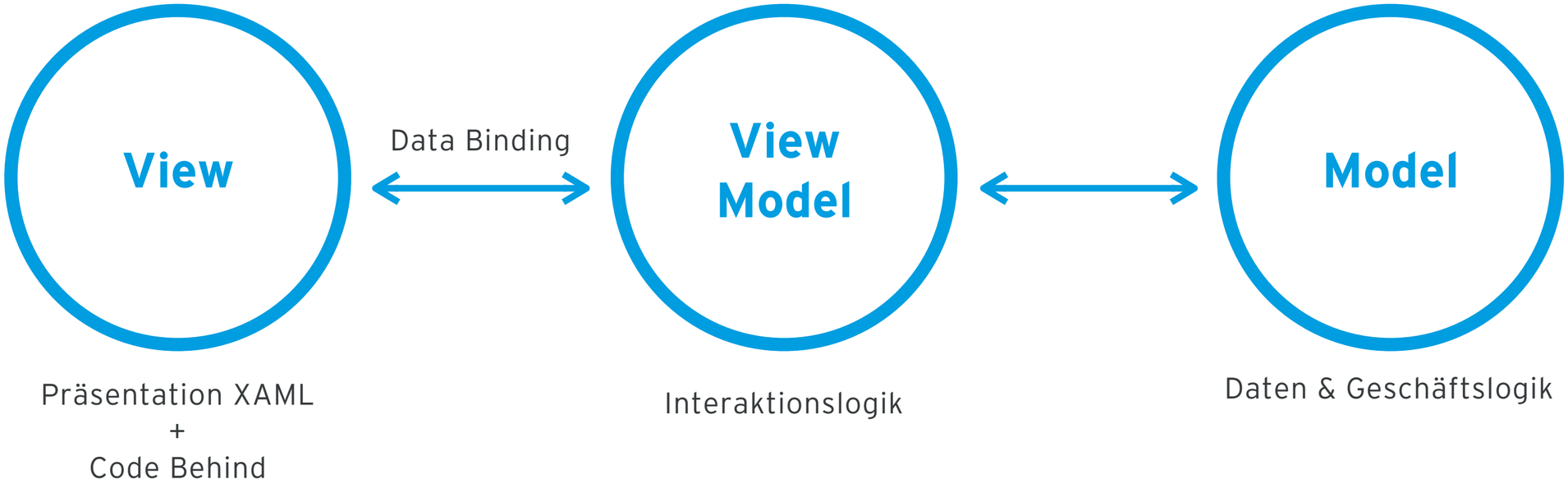


wtf? :-D

Reminder - Deklarative Programmierung mit XAML, Styling, Templates...



Reminder - Model View ViewModel - Klare Trennung Visualisierung & Logik



Probleme

- Komplexität und Umfang der Dokumente (Style Guide, etc.)
- Interpretationsfreiräume des Software Engineers
- Know-how zur Realisierung fehlt
- Allgemein UI Development unterbewertet

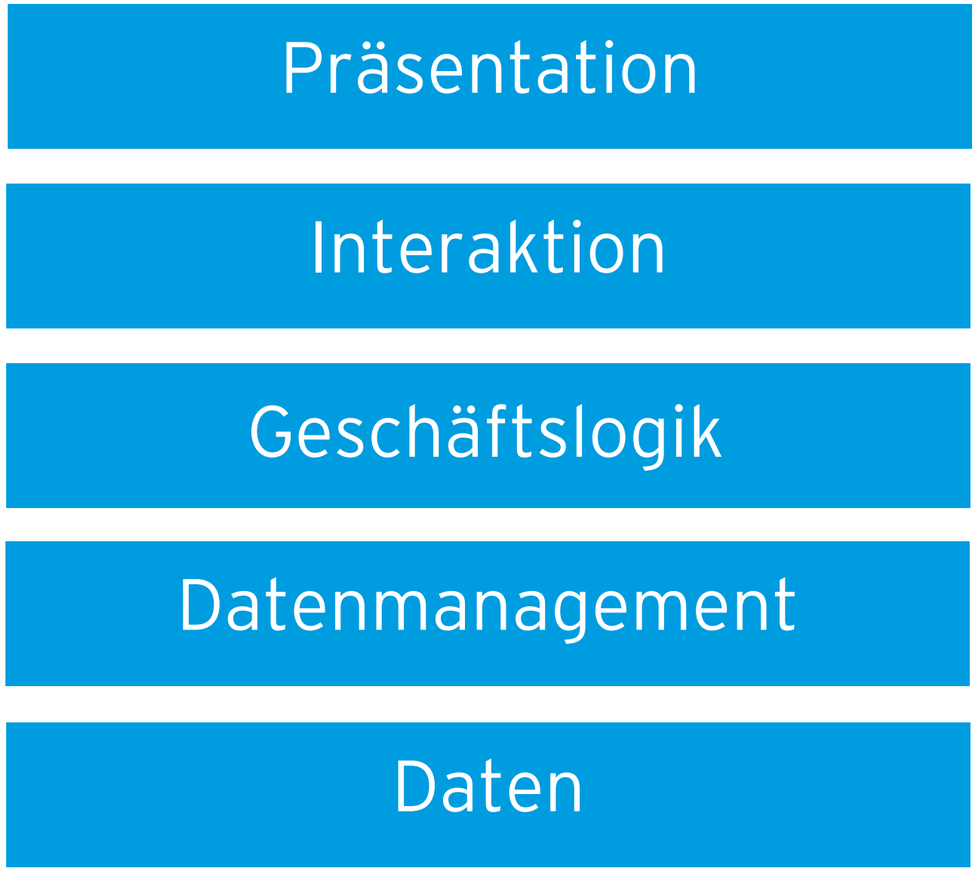
UI Developer als Schlüsselelement

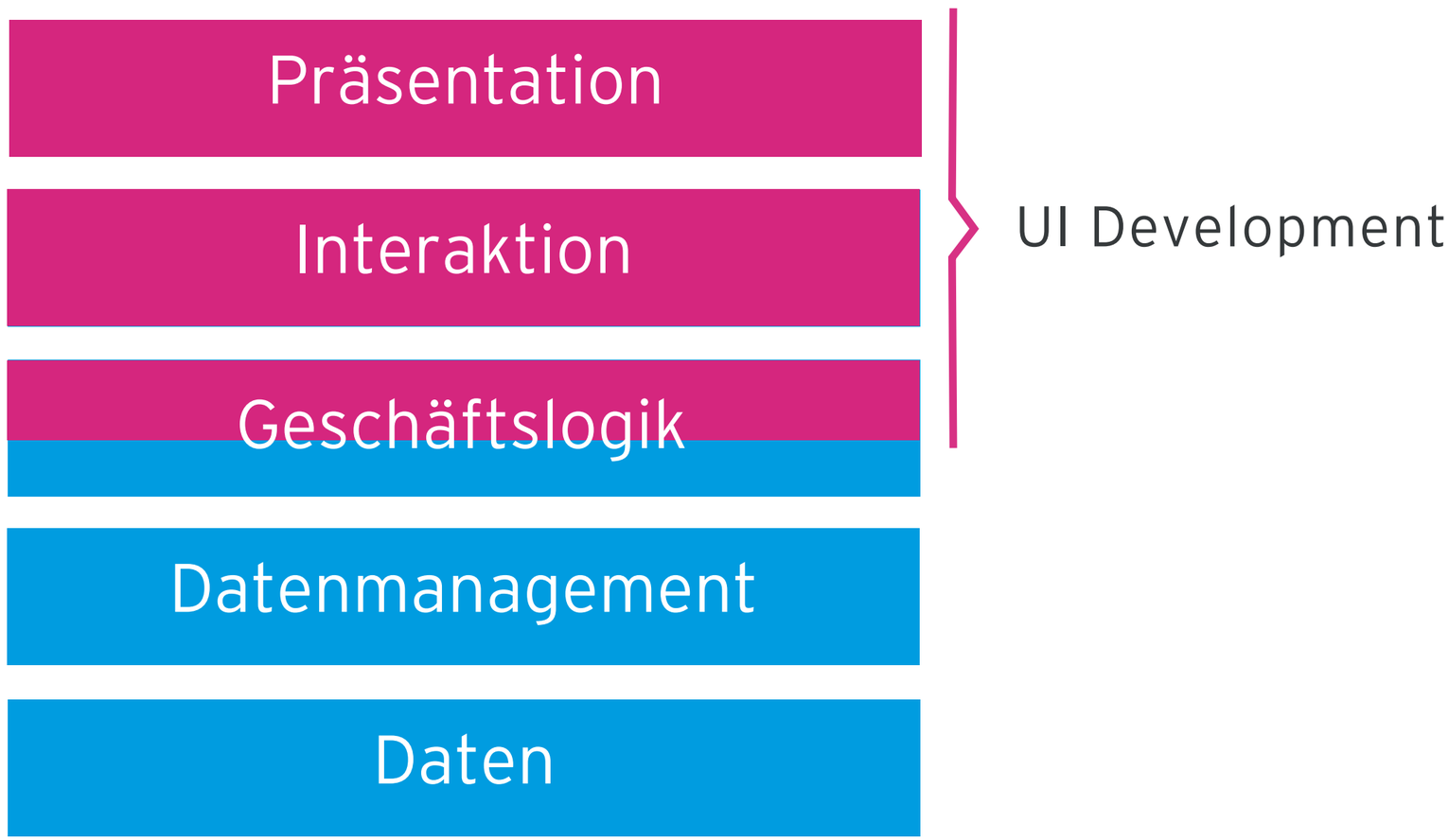
Lösung

- Rolle „UI Developer“ mit eigener Wissensbasis forcieren

Fokus UI Development

- Zwischen UX Designer und Software Engineer aktiv angesiedelt
- Schicht Präsentation und Interaktion
- Styling, Templating, Layouting, Views, ViewModels, Custom Control Development…


Präsentation
Interaktion
Geschäftslogik
Datenmanagement
Daten


Präsentation
Interaktion
Geschäftslogik
Datenmanagement
Daten
UI Development

Zusammenfassung

- UI Development muss als eigene Disziplin innerhalb des Software Engineerings verstanden werden!
- UX Design ist die **Basis** für UI Development
- Ein benutzer-zentrierter Designprozess ist essentiell!

Zusammenfassung

- UI Development muss als eigene Disziplin innerhalb des Software Engineerings verstanden werden!
- UX Design ist die **Basis** für UI Development
- Ein benutzer-zentrierter Designprozess ist essentiell!

User Centered Design
+ Software Engineering
+ UI Development

= Application! ;-)

WPF UI DEVELOPMENT BEST PRACTICES

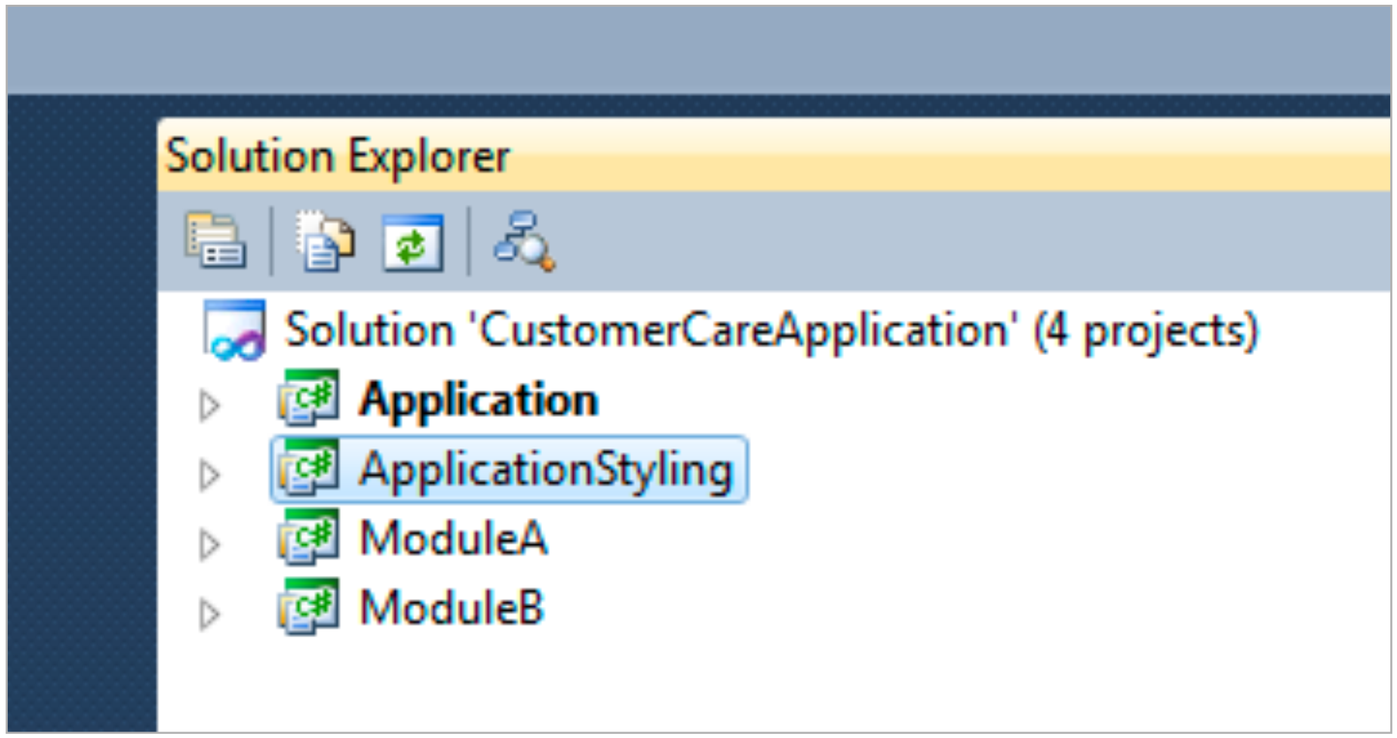
WPF UI DEVELOPMENT BEST PRACTICES
JETZT ABER ;-)

RESOURCE MANAGEMENT

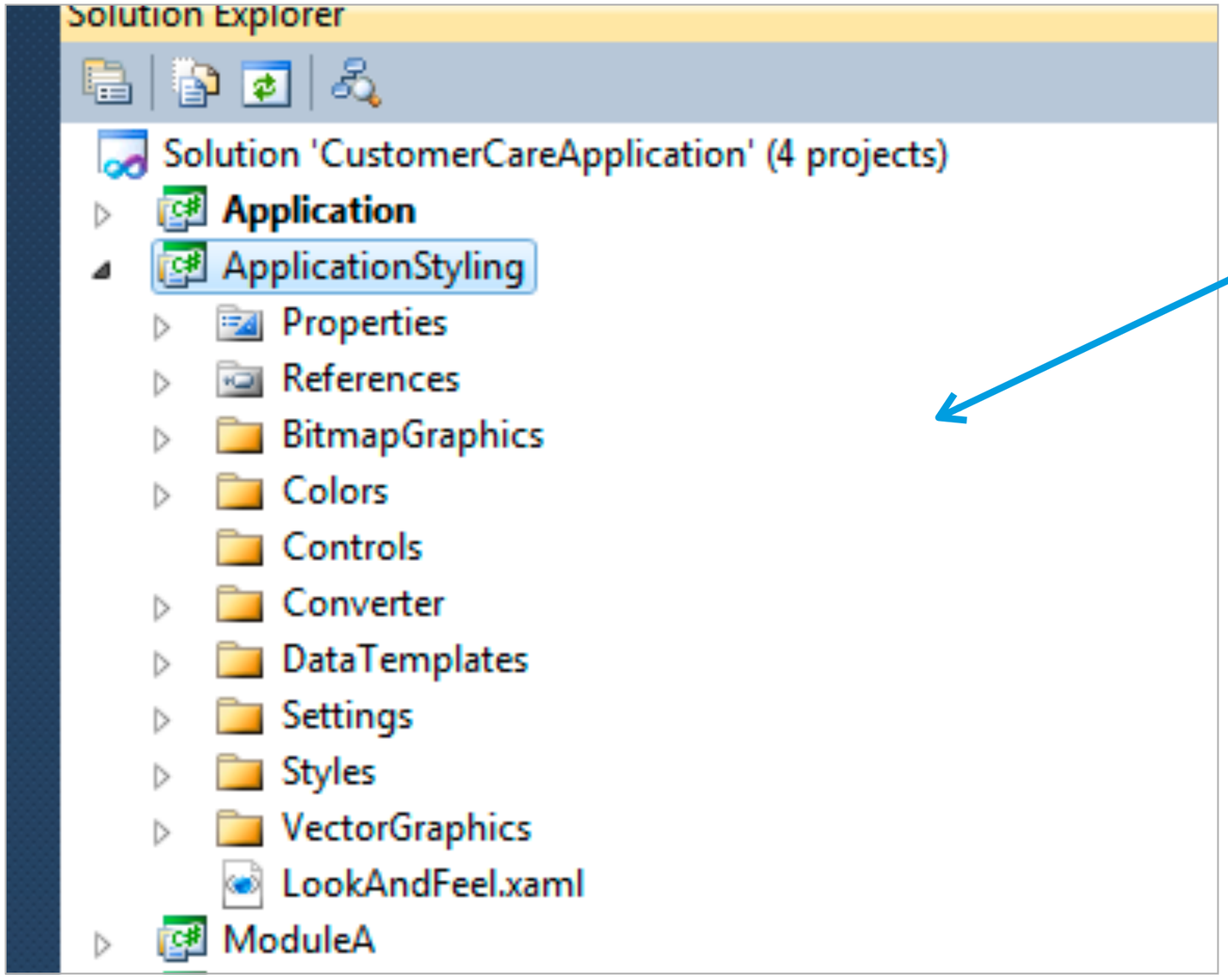
Strategie

- Zentrales „Styling Projekt“ für alle relevanten WPF Ressourcen
- Beinhaltet wohldefinierte Struktur aus Resource Dictionaries
- Pflege, Wartbarkeit und Zusammenarbeit vereinfachen
- Vermeidung unnötiger Inkonsistenzen und Redundanzen

Styling Projekt

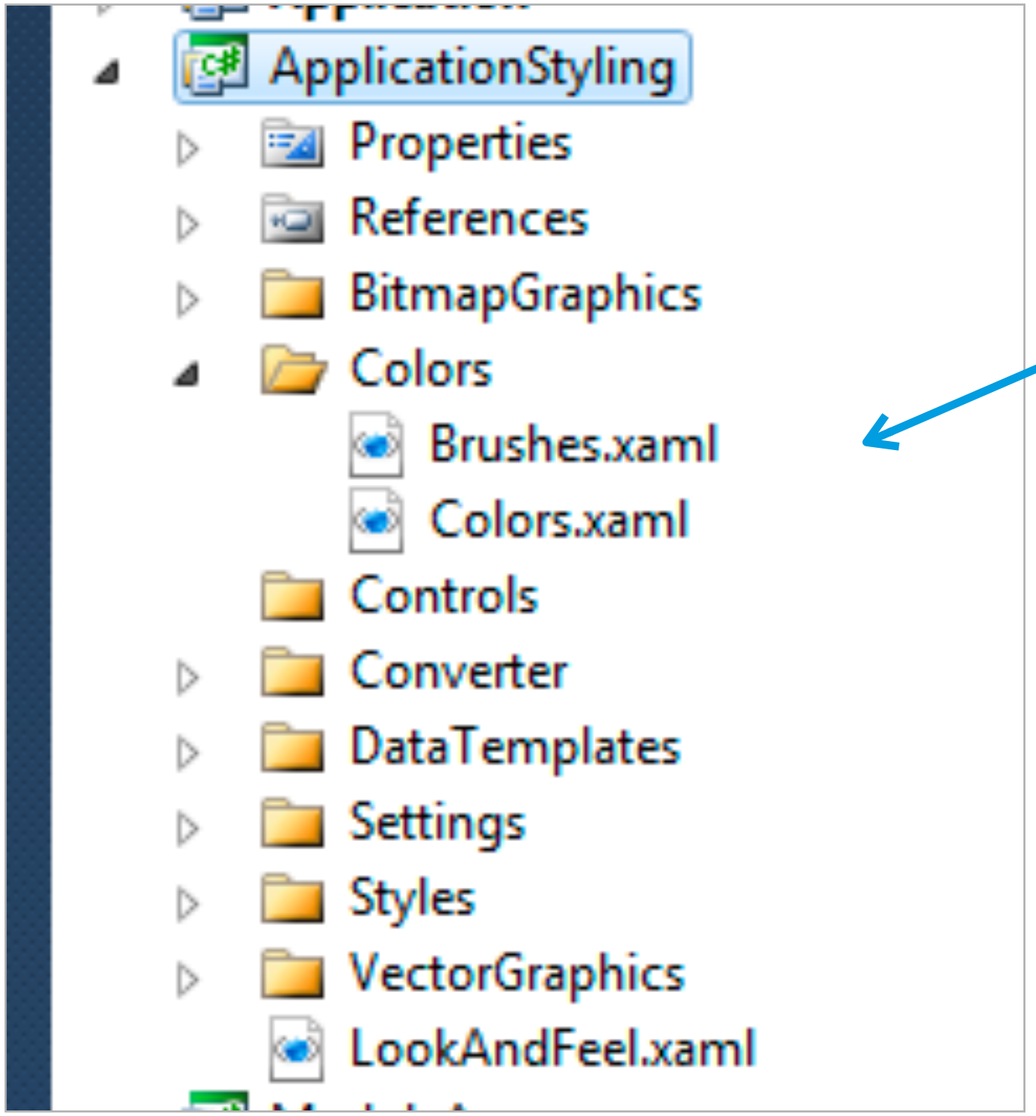


Styling Projekt



Feingranulare Struktur

Colors & Brushes



Auslagern von Colors und Brushes

Colors.xaml

```xml
<!--
This file has been created by ERGOSIGN GmbH - All rights reserved - http://www.ergosign.de
DO NOT ALTER OR REMOVE THIS COPYRIGHT NOTICE OR THIS FILE HEADER.

This file and the code contained in it are subject to the agreed contractual terms and conditions,
in particular with regard to resale and publication.
-->
<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

    <!--Basic Colors//////////////////////////////////////////////////////////////-->
    <!--////////////////////////////////////////////////////////////////////////////-->
    <Color x:Key="WhiteColor">#FFFFFF</Color>
    <Color x:Key="BlackColor">#000000</Color>
    <Color x:Key="TransparentColor">#00FFFFFF</Color>

    <!--Light Colors//////////////////////////////////////////////////////////////-->
    <!--////////////////////////////////////////////////////////////////////////////-->

    <!--HeaderBrd-->
    <Color x:Key="Light01Color">#C3CBD7</Color>

    <!--CheckBoxBackNormal, ListBoxHeader-->
    <Color x:Key="Light02Color">#F7F7F7</Color>

    <Color x:Key="Light03Color">#DDDDDD</Color>

    <!--CheckBoxBackChecked-->
    <Color x:Key="Light04Color">#BBBBBB</Color>

    <!--ButtonBackNormal, PrimaryButtonBackHover, TableDetailPane-->
    <Color x:Key="Light05Color">#EEEEEE</Color>

    <Color x:Key="Light06Color">#CCCCCC</Color>

    <!--PrimaryButtonBackHover-->
    <Color x:Key="Light07Color">#999999</Color>

    <!--WindowBack-->
    <Color x:Key="Light08Color">#E4E4E4</Color>

    <!--LeftSideBarColor-->
    <Color x:Key="Light09Color">#E2E2E2</Color>

    <!--HeaderRowColor-->
    <Color x:Key="Light10Color">#AAAAAA</Color>
```

Farbpalette des UI Designs als Color Objekte anlegen

(Redundanzfrei)

Brushes.xaml | Colors.xaml

```xml
<!--
 This file has been created by ERGOSIGN GmbH - All rights reserved - http://www.ergosign.de
 DO NOT ALTER OR REMOVE THIS COPYRIGHT NOTICE OR THIS FILE HEADER.

 This file and the code contained in it are subject to the agreed contractual terms and conditions,
 in particular with regard to resale and publication.
 -->
<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

    <ResourceDictionary.MergedDictionaries>
        <ResourceDictionary Source="/ApplicationStyling;component/Colors/Colors.xaml"/>
    </ResourceDictionary.MergedDictionaries>

    <!--Basic Brushes/////////////////////////////////////////////////////////////////-->
    <!--///////////////////////////////////////////////////////////////////////////////-->
    <SolidColorBrush x:Key="WhiteBrush" Color="{StaticResource WhiteColor}"/>
    <SolidColorBrush x:Key="BlackBrush" Color="{StaticResource BlackColor}"/>

    <!--Light Brushes/////////////////////////////////////////////////////////////////-->
    <!--///////////////////////////////////////////////////////////////////////////////-->

    <!--HeaderBrd-->
    <SolidColorBrush x:Key="Light01Brush" Color="{StaticResource Light01Color}"/>

    <!--CheckBoxBackNormal, ListBoxHeader-->
    <LinearGradientBrush x:Key="Light03Brush" StartPoint="0.5,0" EndPoint="0.5,1">
        <GradientStop Color="{StaticResource Light02Color}" Offset="0"/>
        <GradientStop Color="{StaticResource Light03Color}" Offset="1"/>
    </LinearGradientBrush>

    <!--PrimaryButtonBackNormal, RadioBtnBackNormal-->
    <LinearGradientBrush x:Key="Light04Brush" StartPoint="0.5,0" EndPoint="0.5,1">
        <GradientStop Color="{StaticResource Light05Color}" Offset="0"/>
        <GradientStop Color="{StaticResource Light06Color}" Offset="1"/>
    </LinearGradientBrush>

    <!--PrimaryButtonBackHover-->
    <LinearGradientBrush x:Key="Light05Brush" StartPoint="0.5,0" EndPoint="0.5,1">
        <GradientStop Color="{StaticResource Light05Color}" Offset="0"/>
        <GradientStop Color="{StaticResource Light07Color}" Offset="1"/>
    </LinearGradientBrush>

    <!--WindowBack-->
    <SolidColorBrush x:Key="Light06Brush" Color="{StaticResource Light08Color}"/>

    <!--WizSteps-->
    <LinearGradientBrush x:Key="Light07Brush" StartPoint="0.5,0" EndPoint="0.5,1">
```

100 %

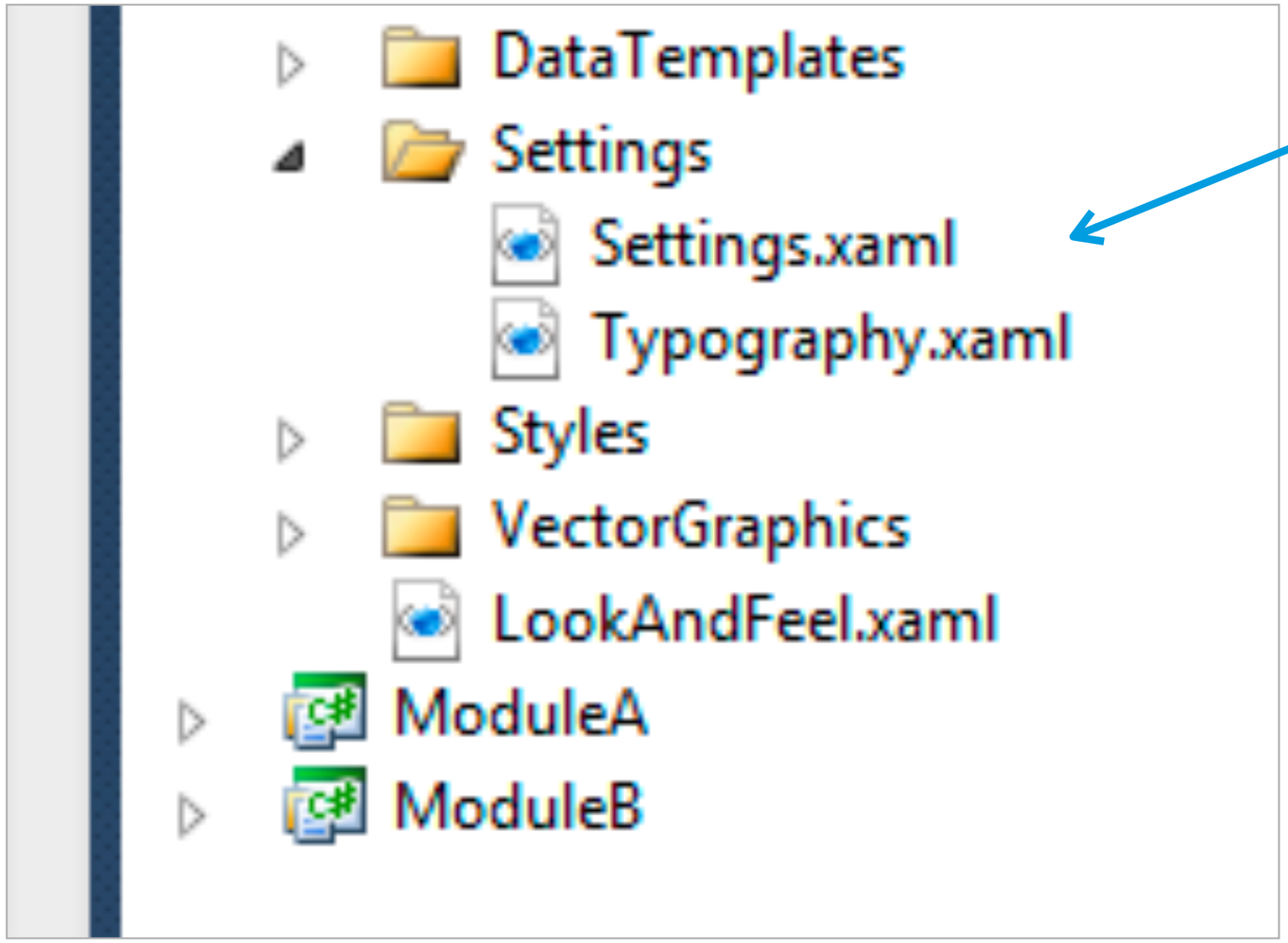
XAML | Design | ResourceDictionary ResourceDictionary

Error List

0 Errors | 24 Warnings | 0 Messages

Aus Colors Brushes erstellen

Settings & Typo



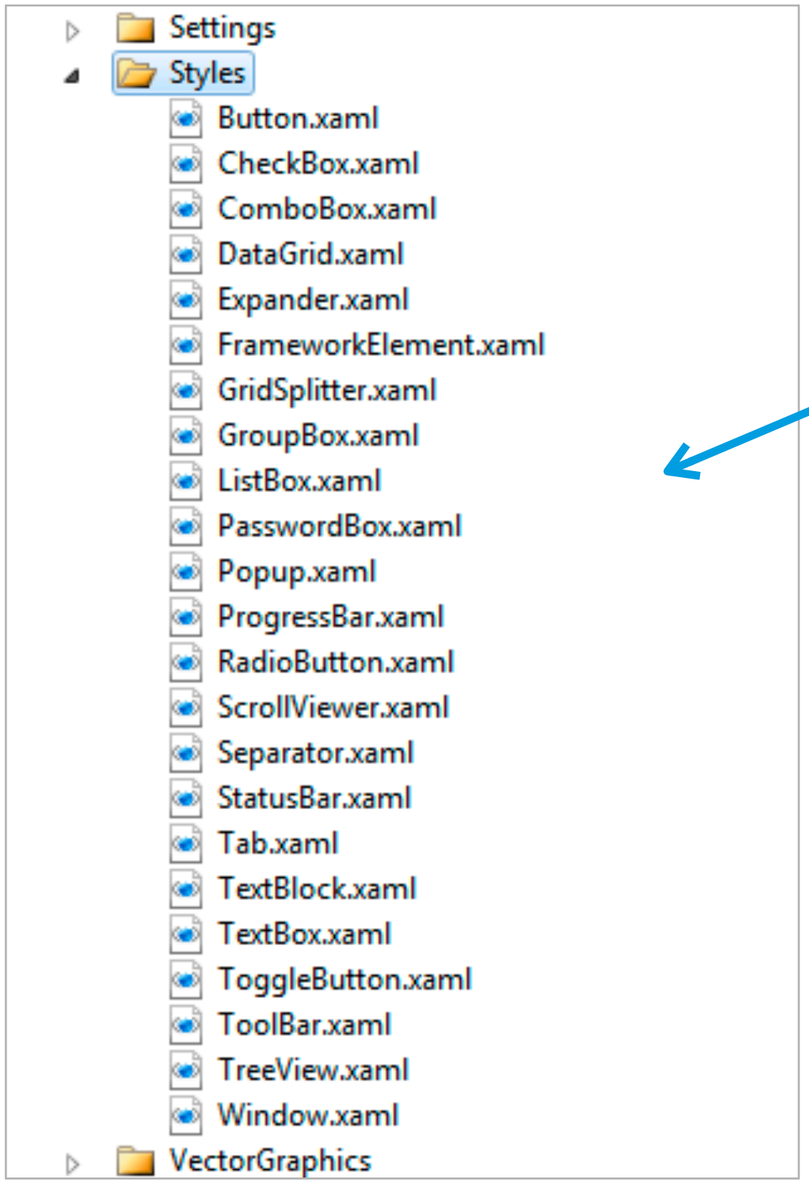
Konstanten, Pfade, Schriftgrößen, Fonts ...

Settings & Typo

```
7   -->
8   <ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
9                       xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
10                      xmlns:sys="clr-namespace:System;assembly=mscorlib">
11
12      <FontFamily x:Key="PrimaryFontFamily">Segoe UI</FontFamily>
13
14      <FontWeight x:Key="NormalFontWeight">Normal</FontWeight>
15      <FontWeight x:Key="BoldFontWeight">Bold</FontWeight>
16
17      <sys:Double x:Key="Layer01Layer">36</sys:Double>
18      <sys:Double x:Key="Layer02Layer">24</sys:Double>
19      <sys:Double x:Key="Layer03Layer">13</sys:Double>
20      <sys:Double x:Key="Layer04Layer">12</sys:Double>
21      <sys:Double x:Key="Layer05Layer">11</sys:Double>
22
23  </ResourceDictionary>
```

Definition der Schrift-Layer (Größen)

Styles



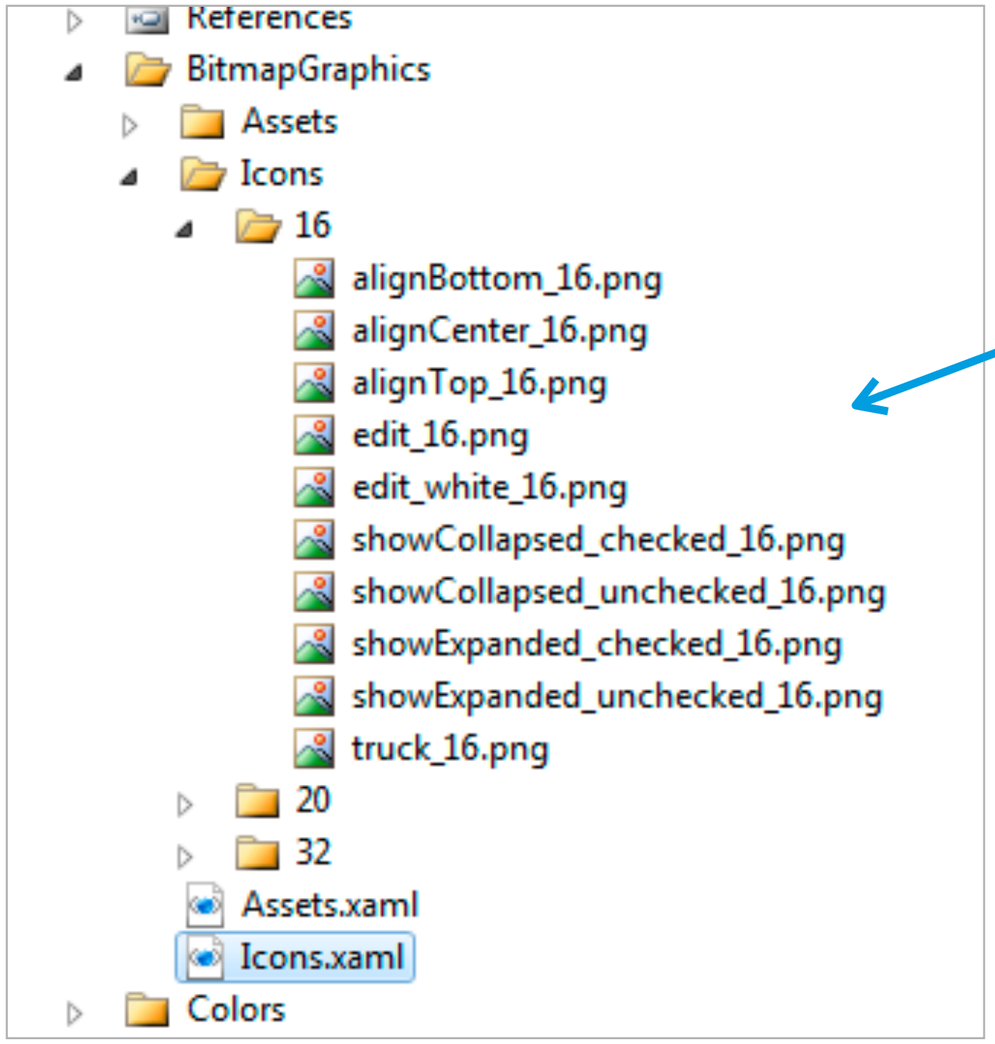
Pro Control-Typ eigenes Resource Dictionary

```xml
17          <ResourceDictionary Source="/ApplicationStyling;component/Styles/FrameworkElement.xaml"/>
18      </ResourceDictionary.MergedDictionaries>
19
20       <Style x:Key="SecondaryButtonStyle" TargetType="Button" BasedOn="{StaticResource FrameworkElementStyle}">
21          <Setter Property="Height" Value="23"/>
22          <Setter Property="MinHeight" Value="22"/>
23          <Setter Property="Padding" Value="10 0"/>
24          <Setter Property="Background" Value="{StaticResource Light04Brush}"/>
25          <Setter Property="Foreground" Value="{StaticResource BlackBrush}"/>
26          <Setter Property="FontFamily" Value="{StaticResource PrimaryFontFamily}"/>
27          <Setter Property="FontWeight" Value="{StaticResource NormalFontWeight}"/>
28          <Setter Property="FontSize" Value="{StaticResource Layer04Layer}"/>
29          <Setter Property="BorderThickness" Value="1"/>
30          <Setter Property="BorderBrush" Value="{StaticResource Dark01Brush}"/>
31          <Setter Property="HorizontalContentAlignment" Value="Center"/>
32          <Setter Property="VerticalContentAlignment" Value="Center"/>
33          <Setter Property="Template">
34              <Setter.Value>
35                  <ControlTemplate TargetType="Button">
36                      <Grid>
37                          <Border BorderThickness="0 0 0 1"
38                                  BorderBrush="{StaticResource WhiteBrush}"
39                                  CornerRadius="4" />
40                          <Border x:Name="rootBrd"
41                                  Background="{TemplateBinding Background}"
42                                  BorderBrush="{TemplateBinding BorderBrush}"
43                                  BorderThickness="{TemplateBinding BorderThickness}"
44                                  CornerRadius="3"
45                                  Margin="0 0 0 1">
46                              <Grid>
47                                  <Border x:Name="highlightBrd"
48                                          BorderBrush="{StaticResource WhiteBrush}"
49                                          BorderThickness="0 1 0 0"
50                                          CornerRadius="2" />
51                                  <Border x:Name="focusBrd"
52                                          BorderBrush="{StaticResource Transparent05Brush}"
53                                          BorderThickness="1"
54                                          CornerRadius="2"
55                                          Visibility="Collapsed" />
56                                  <ContentPresenter Margin="{TemplateBinding Padding}"
57                                                    HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}"
58                                                    VerticalAlignment="{TemplateBinding VerticalContentAlignment}"
59                                                    RecognizesAccessKey="True"/>
60                              </Grid>
61                          </Border>
62                      </Grid>
63                      <ControlTemplate.Triggers>
64                          <Trigger Property="IsKeyboardFocused" Value="True">
65                              <Setter Property="Visibility" Value="Visible" TargetName="focusBrd"/>
66                          </Trigger>
67                      </ControlTemplate.Triggers>
68                  </ControlTemplate>
69              </Setter.Value>
70          </Setter>
71          <Style.Triggers>
72              <MultiTrigger>
73                  <MultiTrigger.Conditions>
```

Fokus auf Style Setter und Control Template

Kompakter XAML Source-Code

Bitmap Graphics



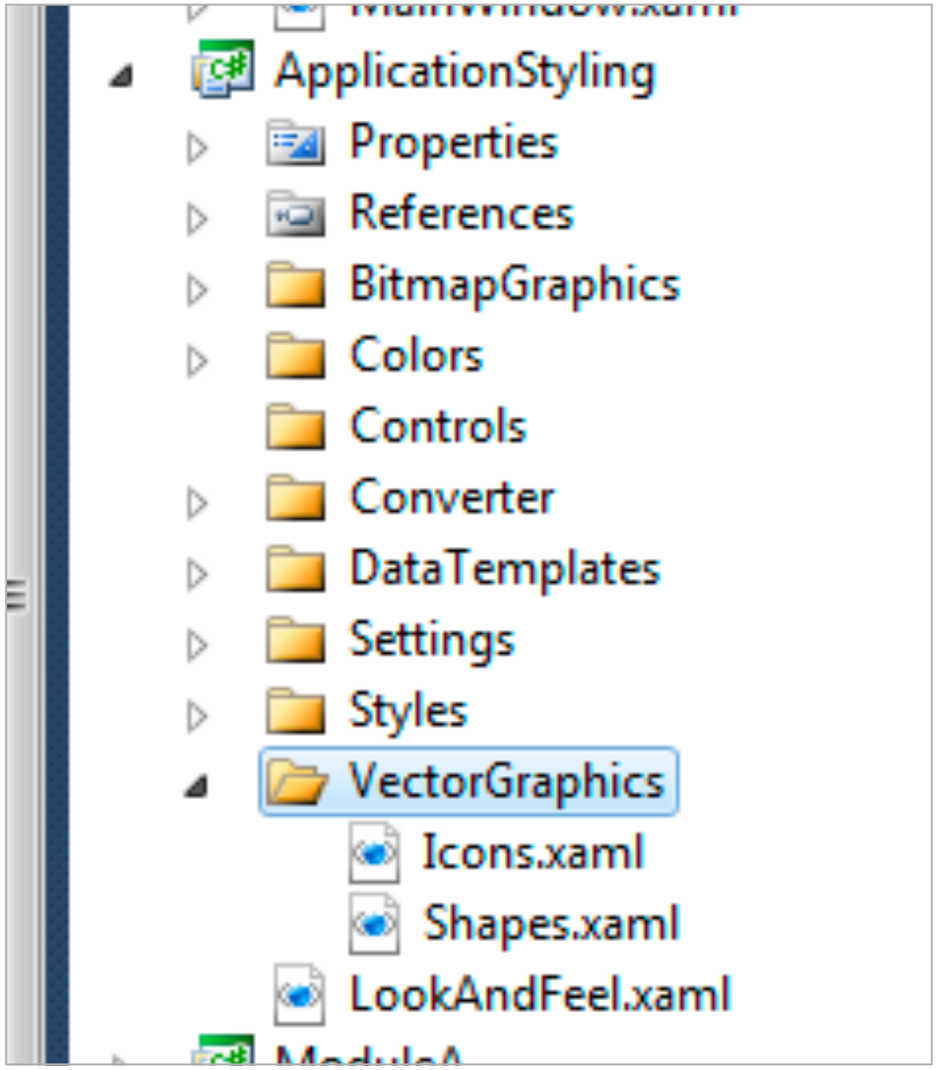
Beinhaltet alle Rastergrafiken (PNG, JPEG, ...)

Bitmap Graphics

```
-->
<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

    <!--16er Icons-->
    <ImageSource x:Key="AlignBottom16Icon">/ApplicationStyling;component/BitmapGraphics/Icons/16/alignBottom_16.png</ImageSource>
    <ImageSource x:Key="AlignCenter16Icon">/ApplicationStyling;component/BitmapGraphics/Icons/16/alignCenter_16.png</ImageSource>
    <ImageSource x:Key="AlignTop16Icon">/ApplicationStyling;component/BitmapGraphics/Icons/16/alignTop_16.png</ImageSource>
    <ImageSource x:Key="ShowCollapsedChecked16Icon">/ApplicationStyling;component/BitmapGraphics/Icons/16/showCollapsed_checked_16.png</ImageSource>
    <ImageSource x:Key="ShowCollapsedUnchecked16Icon">/ApplicationStyling;component/BitmapGraphics/Icons/16/showCollapsed_unchecked_16.png</ImageSource>
    <ImageSource x:Key="ShowExpandedChecked16Icon">/ApplicationStyling;component/BitmapGraphics/Icons/16/showExpanded_checked_16.png</ImageSource>
    <ImageSource x:Key="ShowExpandedUnchecked16Icon">/ApplicationStyling;component/BitmapGraphics/Icons/16/showExpanded_unchecked_16.png</ImageSource>
    <ImageSource x:Key="Truck16Icon">/ApplicationStyling;component/BitmapGraphics/Icons/16/truck_16.png</ImageSource>
    <ImageSource x:Key="Edit16Icon">/ApplicationStyling;component/BitmapGraphics/Icons/16/edit_16.png</ImageSource>
```

Verwaltung als ImageSource

Vector Graphics

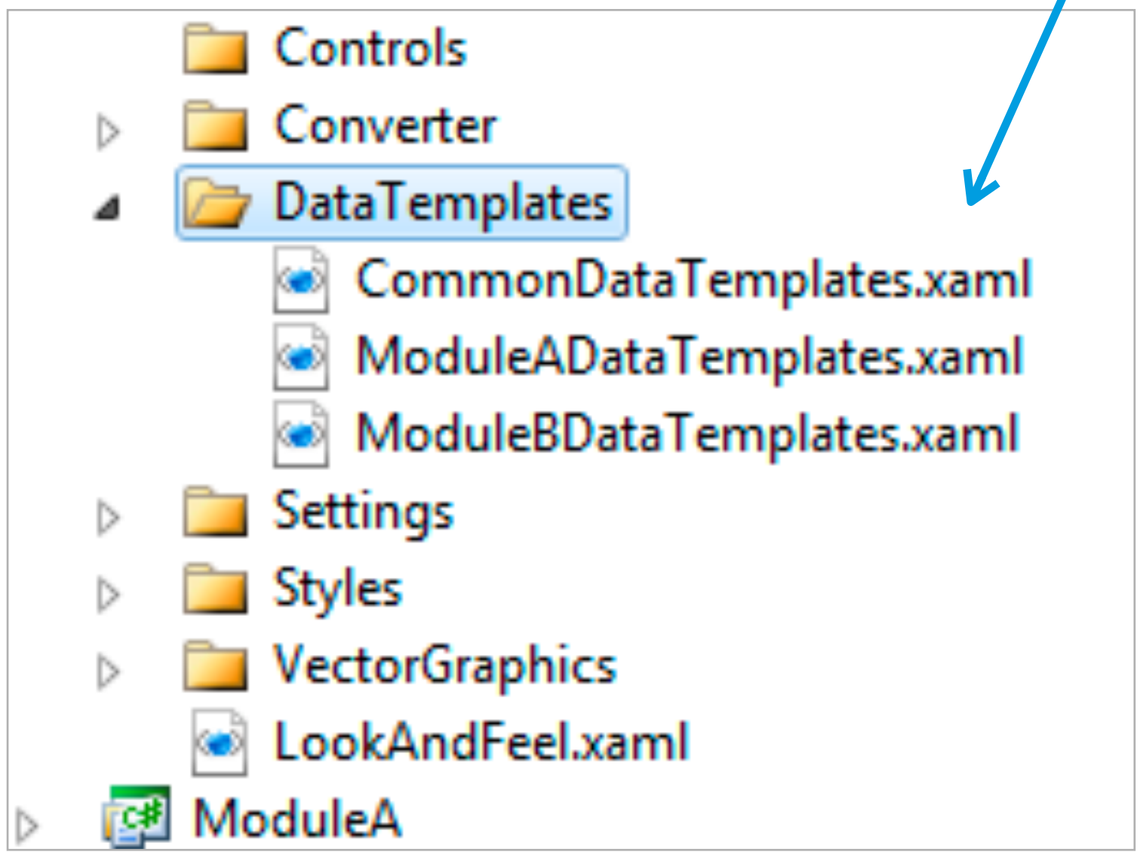


DrawingBrush

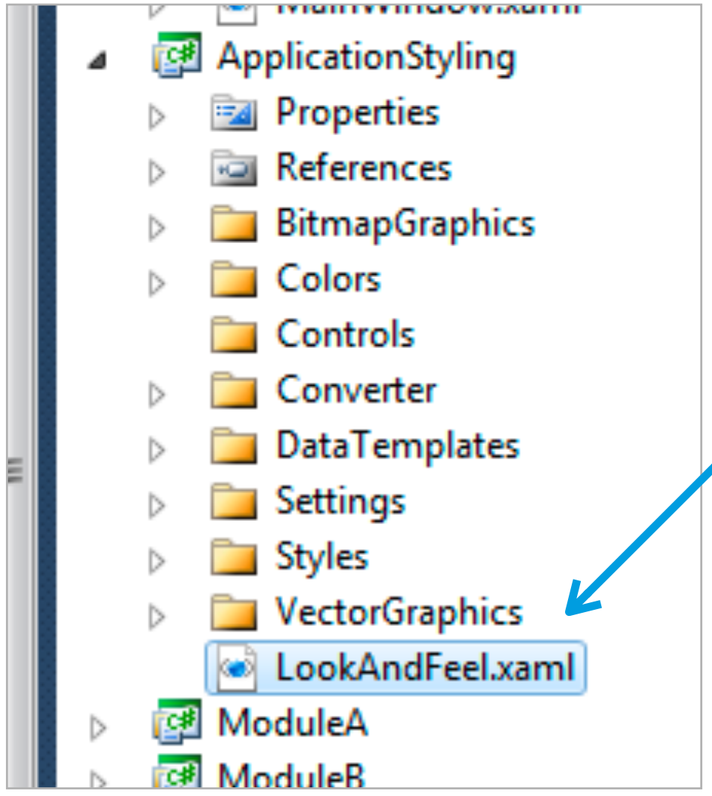
```
-->
<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

    <DrawingBrush x:Key="TankDrawingBrush"
                  Stretch="Uniform">
        <DrawingBrush.Drawing>
            <DrawingGroup>
                <DrawingGroup.Children>
                    <GeometryDrawing Brush="#B31D1D1B"
                                     Geometry="F1 M 306.667,213.333C 306.667,212.599 307.265,212 308,212L 320,212C 320.
                    <GeometryDrawing Geometry="F1 M 300,248L 328,248L 328,245.333L 300,245.333L 300,248 Z ">
                        <GeometryDrawing.Brush>
                            <LinearGradientBrush StartPoint="-1.08991e-006,0.499994"
                                                 EndPoint="0.999999,0.499994">
                                <LinearGradientBrush.GradientStops>
                                    <GradientStop Color="#FFFFFFFF"
                                                  Offset="0" />
                                    <GradientStop Color="#FFBBBBBB"
                                                  Offset="1" />
                                </LinearGradientBrush.GradientStops>
                            </LinearGradientBrush>
                        </GeometryDrawing.Brush>
                    </GeometryDrawing>
                    <GeometryDrawing Geometry="F1 M 300,248L 328,248L 328,245.333L 300,245.333L 300,248 Z ">
                        <GeometryDrawing.Brush>
                            <LinearGradientBrush StartPoint="-1.08991e-006,0.499994"
```

Data Templates

Data Templates ebenfalls auslagern



Look and Feel Resource Dictionary



Ein Resource Dictionary das alle Ressourcen beinhaltet

Kann später in *App.xaml* eingebunden werden

```xml
<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                    xmlns:xControls="clr-namespace:ERGOSIGN.XControls;assembly=XControls"
                    xmlns:xWindow="clr-namespace:ERGOSIGN.XControls.XWindow;assembly=XControls">

    <ResourceDictionary.MergedDictionaries>
        <ResourceDictionary Source="/ApplicationStyling;component/Colors/Colors.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Colors/Brushes.xaml"/>

        <ResourceDictionary Source="/ApplicationStyling;component/BitmapGraphics/Assets.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/BitmapGraphics/Icons.xaml"/>

        <ResourceDictionary Source="/ApplicationStyling;component/Settings/Typography.xaml"/>

        <ResourceDictionary Source="/ApplicationStyling;component/Converter/Converter.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/DataTemplates/DataTemplates.xaml"/>

        <ResourceDictionary Source="/ApplicationStyling;component/Styles/FrameworkElement.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/ScrollViewer.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/Button.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/TextBox.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/CheckBox.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/RadioButton.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/ProgressBar.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/ComboBox.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/Expander.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/Tab.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/ToolBar.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/ToggleButton.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/Separator.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/GroupBox.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/StatusBar.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/ListBox.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/DataGrid.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/TreeView.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/GridSplitter.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/TextBlock.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/PasswordBox.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/Popup.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/Window.xaml"/>
    </ResourceDictionary.MergedDictionaries>

    <!--Default Styles-->
    <Style TargetType="ScrollViewer" BasedOn="{StaticResource ScrollViewerStyle}"/>
    <Style TargetType="Button" BasedOn="{StaticResource SecondaryButtonStyle}"/>
    <Style TargetType="ToggleButton" BasedOn="{StaticResource IconToggleButtonBigStyle}"/>
    <Style TargetType="TextBox" BasedOn="{StaticResource TextBoxStyle}"/>
    <Style TargetType="CheckBox" BasedOn="{StaticResource CheckBoxStyle}"/>
    <Style TargetType="RadioButton" BasedOn="{StaticResource RadioButtonStyle}"/>
    <Style TargetType="ProgressBar" BasedOn="{StaticResource ProgressBarStyle}"/>
    <Style TargetType="ComboBox" BasedOn="{StaticResource ComboBoxLightStyle}"/>
    <Style TargetType="Expander" BasedOn="{StaticResource ExpanderStyle}"/>
    <Style TargetType="TabControl" BasedOn="{StaticResource TabControlLightStyle}"/>
    <Style TargetType="ToolBar" BasedOn="{StaticResource ToolBarStyle}"/>
    <Style TargetType="Separator" BasedOn="{StaticResource SeparatorLightStyle}"/>
```

Reihenfolge beachten!

Look and Feel Resource Dictionary

Einfaches einbinden…

App.xaml

```
1  <Application x:Class="Application.App"
2               xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3               xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4               StartupUri="MainWindow.xaml">
5      <Application.Resources>
6          <ResourceDictionary>
7              <ResourceDictionary.MergedDictionaries>
8                  <ResourceDictionary Source="/ApplicationStyling;component/LookAndFeel.xaml"/>
9              </ResourceDictionary.MergedDictionaries>
10         </ResourceDictionary>
11     </Application.Resources>
12 </Application>
13
```

```xml
<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                    xmlns:xControls="clr-namespace:ERGOSIGN.XControls;assembly=XControls"
                    xmlns:xWindow="clr-namespace:ERGOSIGN.XControls.XWindow;assembly=XControls">

    <ResourceDictionary.MergedDictionaries>
        <ResourceDictionary Source="/ApplicationStyling;component/Colors/Colors.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Colors/Brushes.xaml"/>

        <ResourceDictionary Source="/ApplicationStyling;component/BitmapGraphics/Assets.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/BitmapGraphics/Icons.xaml"/>

        <ResourceDictionary Source="/ApplicationStyling;component/Settings/Typography.xaml"/>

        <ResourceDictionary Source="/ApplicationStyling;component/Converter/Converter.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/DataTemplates/DataTemplates.xaml"/>

        <ResourceDictionary Source="/ApplicationStyling;component/Styles/FrameworkElement.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/ScrollViewer.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/Button.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/TextBox.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/CheckBox.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/RadioButton.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/ProgressBar.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/ComboBox.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/Expander.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/Tab.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/ToolBar.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/ToggleButton.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/Separator.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/GroupBox.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/StatusBar.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/ListBox.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/DataGrid.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/TreeView.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/GridSplitter.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/TextBlock.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/PasswordBox.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/Popup.xaml"/>
        <ResourceDictionary Source="/ApplicationStyling;component/Styles/Window.xaml"/>
    </ResourceDictionary.MergedDictionaries>

    <!--Default Styles-->
    <Style TargetType="ScrollViewer" BasedOn="{StaticResource ScrollViewerStyle}"/>
    <Style TargetType="Button" BasedOn="{StaticResource SecondaryButtonStyle}"/>
    <Style TargetType="ToggleButton" BasedOn="{StaticResource IconToggleButtonBigStyle}"/>
    <Style TargetType="TextBox" BasedOn="{StaticResource TextBoxStyle}"/>
    <Style TargetType="CheckBox" BasedOn="{StaticResource CheckBoxStyle}"/>
    <Style TargetType="RadioButton" BasedOn="{StaticResource RadioButtonStyle}"/>
    <Style TargetType="ProgressBar" BasedOn="{StaticResource ProgressBarStyle}"/>
    <Style TargetType="ComboBox" BasedOn="{StaticResource ComboBoxLightStyle}"/>
    <Style TargetType="Expander" BasedOn="{StaticResource ExpanderStyle}"/>
    <Style TargetType="TabControl" BasedOn="{StaticResource TabControlLightStyle}"/>
    <Style TargetType="ToolBar" BasedOn="{StaticResource ToolBarStyle}"/>
    <Style TargetType="Separator" BasedOn="{StaticResource SeparatorLightStyle}"/>
```

Default Styles anbieten!

(Kein x:Key)

SHARED RESOURCE DICTIONARY

Shared Resource Dictionary

Vorsicht

- Bei MergedDictionaries (Speicherfresser)
- Bewirkt mehrfache Allokation der Ressourcen!
- Kritisch bei modularen Projekten mit Ressourcen „Verlinkung“

Lösung

- Shared Resource Dictionary

Shared Resource Dictionary

```
<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
                    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
                    mc:Ignorable="d">

    <ResourceDictionary.MergedDictionaries>
        <ResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Styles/Control.xaml"/>
        <ResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Colors/Brushes.xaml" />
        <ResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Colors/Colorizable/FormsViewFlatAndSimpleBrushes.xaml" />
        <ResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Colors/Effects.xaml"/>
        <ResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Colors/Animations.xaml"/>
        <ResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Settings/Settings.xaml"/>
        <ResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Settings/Typography.xaml"/>
    </ResourceDictionary.MergedDictionaries>

    <Style x:Key="StartButtonStyle" TargetType="{x:Type Button}" BasedOn="{StaticResource BaseControlStyle}">
```

Shared Resource Dictionary

```xml
<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
                    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
                    mc:Ignorable="d">

    <ResourceDictionary.MergedDictionaries>
        <ResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Styles/Control.xaml"/>
        <ResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Colors/Brushes.xaml" />
        <ResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Colors/Colorizable/FormsViewFlatAndSimpleBrushes.xaml" />
        <ResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Colors/Effects.xaml"/>
        <ResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Colors/Animations.xaml"/>
        <ResourceDiction
        <ResourceDiction
    </ResourceDictionary

    <Style x:Key="StartB
```

```xml
<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

    <ResourceDictionary.MergedDictionaries>
        <ResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Styles/Control.xaml"/>
        <ResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Colors/Brushes.xaml" />
        <ResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Colors/Colorizable/FormsViewFlatAndSimpleBrushes.xaml" />
        <ResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Colors/Effects.xaml"/>
        <ResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Colors/Animations.xaml"/>
        <ResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Settings/Settings.xaml"/>
        <ResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Settings/Typography.xaml"/>
        <ResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Styles/ToggleButton.xaml"/>
    </ResourceDictionary.MergedDictionaries>

    <Style x:Key="ModalWindowFlatComboBoxStyle" TargetType="{x:Type ComboBox}">
        <Setter Property="BorderBrush" Value="{StaticResource SimpleButtonBorderBrush}"/>
```

Führt zu Mehrfachanlegung

SharedResourceDictionary.Shared.SharedResourceDictionary | Source

```
19  public class SharedResourceDictionary : ResourceDictionary
20  {
21      /// <summary>
22      /// Dictionary for caching.
23      /// </summary>
24      private static Dictionary<Uri, ResourceDictionary> SharedDictionaries = new Dictionary<Uri, ResourceDictionary>();
25
26      /// <summary>
27      /// Our shared uri.
28      /// </summary>
29      private Uri sharedSourceURI;
30
31      /// <summary>
32      /// Define new implementation of source property.
33      /// </summary>
34      public new Uri Source
35      {
36          get { return sharedSourceURI; }
37
38          set
39          {
40              // Maybe implement lock
41              if (sharedSourceURI == value)
42                  return;
43
44              sharedSourceURI = value;
45
46              if (!SharedDictionaries.ContainsKey(value))
47              {
48                  // Set as source of this rd
49                  base.Source = value;
50                  SharedDictionaries.Add(value, this);
51              }
52              else
53              {
54                  MergedDictionaries.Add(SharedDictionaries[value]);
55              }
56          }
57      }
58  }
59 }
60
```

Shared Resource Dictionary

Shared Resource Dictionary

```
<SharedResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                          xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
                          xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
                          mc:Ignorable="d"
                          >

    <SharedResourceDictionary.MergedDictionaries>
        <SharedResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Styles/Control.xaml"/>
        <SharedResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Colors/Brushes.xaml" />
        <SharedResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Colors/Colorizable/FormsViewFlatAndSimpleBrushes.xaml" />
        <SharedResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Colors/Effects.xaml"/>
        <SharedResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Colors/Animations.xaml"/>
        <SharedResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Settings/Settings.xaml"/>
        <SharedResourceDictionary Source="/ERGOSIGNShowcaseStyling;component/Settings/Typography.xaml"/>
    </SharedResourceDictionary.MergedDictionaries>

    <Style x:Key="StartButtonStyle" TargetType="{x:Type Button}" BasedOn="{StaticResource BaseControlStyle}">
```

Verwendung wie gewohnt, aber…

ndow Help

Parts | LookAndFeel.xaml | AboutView.xaml | DesignTimeResources.xaml | MainWindow.xaml | App.xaml

Nothing selected

Saarbrücken

ERGOSIGN GmbH
Europa-Allee 12
D-66113 Saarbrücken

email: contact@ergosign.de
phone: +49 (0)681 988 412 0
fax: +49 (0)681 988 412 10

München

ERGOSIGN GmbH
Adams-Lehmann-Straße 4
D-80797 Munich

email: muenchen@ergosign.de
phone: +49 (0)89 68 90 60 70
fax: +49 (0)89 68 90 60 710

Hamburg

ERGOSIGN GmbH
Bernhard-Nocht-Str. 109
D-20359 Hamburg

email: hamburg@ergosign.de
phone: +49 (0)40 317 98 68 0
fax: +49 (0)40 317 98 68 10

Basel

ERGOSIGN SWITZERLAND AG
Gartenstrasse 95
CH-4052 Basel

email: contact@ergosign.ch
phone: +41 (0)61 225 66 22
fax: +41 (0)61 225 66 24

100%

Add Design-time Resource Dictionary

The document AboutView.xaml contains at least one resource (HeaderTextBlockStyle) that could not be resolved.

Would you like to select a resource dictionary to use for displaying resources at design-time?

Available dictionaries

Don't show this message again

OK | Cancel

Properties | Resources | Data

- App.xaml
- Design-time Resources
- Flyout.xaml
- Gauge.xaml
- LookAndFeel.xaml
- AboutView.xaml
  - [UserControl]

Results

Errors

Description				
The resource "HeaderTextBlockStyle"				howc
The resource "SimpleFormularTextBlc				howc
The resource "HeaderTextBlockStyle"				howc
The resource "HeaderTextBlockStyle"				howc
The resource "HeaderTextBlockStyle"				howc
The resource "HeaderTextBlockStyle" could not be resolved.	AboutView.xaml	58	27	ERGOSIGNShowc
The resource "SimpleFormularTextBlockStyle" could not be resolved.	AboutView.xaml	60	14	ERGOSIGNShowc
The resource "SimpleFormularTextBlockStyle" could not be resolved.	AboutView.xaml	62	14	ERGOSIGNShowc
The resource "SimpleFormularTextBlockStyle" could not be resolved.	AboutView.xaml	64	14	ERGOSIGNShowc
The resource "SimpleFormularTextBlockStyle" could not be resolved.	AboutView.xaml	66	14	ERGOSIGNShowc
The resource "ERGOSIGNSaarbruecken" could not be resolved.	AboutView.xaml	69	39	ERGOSIGNShowc
The resource "ERGOSIGNMuenchen" could not be resolved.	AboutView.xaml	70	39	ERGOSIGNShowc
The resource "ERGOSIGNHamburg" could not be resolved.	AboutView.xaml	71	39	ERGOSIGNShowc
The resource "ERGOSIGNSwitzerland" could not be resolved.	AboutView.xaml	72	39	ERGOSIGNShowc
The resource "SimpleFormularTextBlockStyle" could not be resolved.	AboutView.xaml	74	14	ERGOSIGNShowc
The resource "SimpleFormularTextBlockStyle" could not be resolved.	AboutView.xaml	76	14	ERGOSIGNShowc
The resource "SimpleFormularTextBlockStyle" could not be resolved.	AboutView.xaml	78	14	ERGOSIGNShowc
The resource "SimpleFormularTextBlockStyle" could not be resolved.	AboutView.xaml	80	14	ERGOSIGNShowc

SharedResourceDictionary.Shared.SharedResourceDictionary | SharedDictionaries

```
1
2   #define GLOBAL_USE_SHAREDRESOURCEDICTIONARY
3
4   using System;
5   using System.Collections.Generic;
6   using System.Linq;
7   using System.Text;
8   using System.Windows;
9   using System.ComponentModel;
10  using System.Windows.Controls;
11  using System.Windows.Media;
12  using System.Windows.Markup;
13
14  namespace SharedResourceDictionary.Shared
15  {
16      /// <summary>
17      /// Shared dictionary, load resources only once :-).
18      /// </summary>
19      public class SharedResourceDictionary : ResourceDictionary
20      {
21          #if GLOBAL_USE_SHAREDRESOURCEDICTIONARY
22          /// <summary> ...
25          private static Dictionary<Uri, ResourceDictionary> SharedDictionaries = new Dictionary<Uri, Res
26
27          /// <summary> ...
30          private Uri sharedSourceURI;
31
32          /// <summary> ...
35          public new Uri Source...
59          #endif
60      }
61  }
62
```

Preprocessor Symbol zum steuern verwenden :-)

LookAndFeel.xaml | AboutView.xaml | DesignTimeResources.xaml | MainWindow.xaml | App.xaml

[TextBlock]

We Shape Interaction

We design, optimize and develop user interfaces for office, industry and medical applications as well as for mobile devices and consumer products. ERGOSIGN develops integrated user experience strategies that help you attract new customers and inspire your existing users.

Saarbrücken	München	Hamburg	Basel
ERGOSIGN GmbH Europa-Al D-66113 S	ERGOSIGN GmbH ann-Straße 4 ich	ERGOSIGN GmbH Bernhard-Nocht-Str. 109 D-20359 Hamburg	ERGOSIGN SWITZERLAND AG Gartenstrasse 95 CH-4052 Basel
email: con phone: +4 0 fax: +49 (	ergosign.de)89 68 90 60 68 90 60 710	email: hamburg@ergosign.de phone: +49 (0)40 317 98 68 0 fax: +49 (0)40 317 98 68 10	email: contact@ergosign.ch phone: +41 (0)61 225 66 22 fax: +41 (0)61 225 66 24

- Cut
- Copy
- Paste
- Delete
- Order
- Align
- Auto Size
- Group Into
- Ungroup
- Pin Active Container
- Set Current Selection
- Path
- Edit Text
- Make Into Control...
- Make Into UserControl...
- Edit Style
  - Edit Current
  - Edit a Copy...
  - Create Empty...
  - Apply Resource
    - BaseControlStyle
    - SimpleFormularTextBlockStyle
    - HeaderTextBlockStyle
- View XAML

100%

Shared Resource Dictionary

In eigener Assembly auslagern und Namespace Mapping in AssemblyInfo.cs definieren ;-)

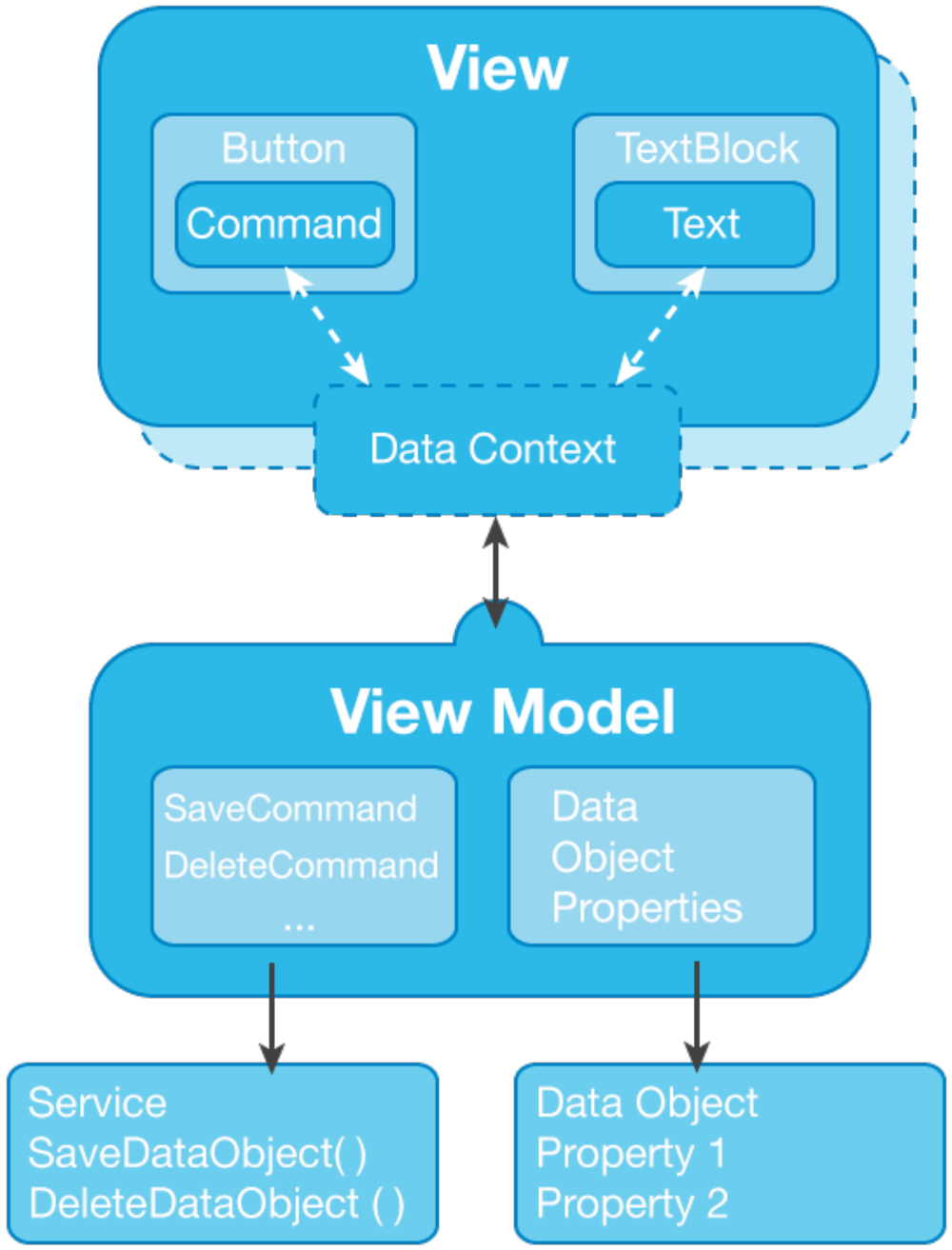
```
31  //      Revision
32  //
33  // You can specify all the values or you can default the Build and Revision Numbers
34  // by using the '*' as shown below:
35  // [assembly: AssemblyVersion("1.0.*")]
36  [assembly: AssemblyVersion("1.0.0.0")]
37  [assembly: AssemblyFileVersion("1.0.0.0")]
38
39  [assembly: XmlnsDefinition("http://schemas.microsoft.com/winfx/2006/xaml/presentation", "SharedResourceDictionary.Shared")]
```

USERCONTROL

UserControl

- Leicht zu erstellen und benutzen
- Struktur = Content
- Element-Referenzierung mit x:Name
- Styling und Templating möglich
- Code-Behind (Blend, Visual Studio)

Model View ViewModel im Überblick



View = UserControl

UserControl

- UserControls dienen der **Zusammenfassung** und **Gruppierung** mehrerer Controls und Layout Panels.

UserControl vs. Custom Control - Gegenüberstellung

UserControl vs. Custom Control - Gegenüberstellung

	USER CONTROL	CUSTOM CONTROL

UserControl vs. Custom Control - Gegenüberstellung

	USER CONTROL	CUSTOM CONTROL
TECHNIK	XAML + Code Behind	Klasse + Style + Template

UserControl vs. Custom Control - Gegenüberstellung

	USER CONTROL	CUSTOM CONTROL
TECHNIK	XAML + Code Behind	Klasse + Style + Template
LOGIK & VISUAL	Harte Verdrahtung	Getrennt

UserControl vs. Custom Control - Gegenüberstellung

	USER CONTROL	CUSTOM CONTROL
TECHNIK	XAML + Code Behind	Klasse + Style + Template
LOGIK & VISUAL	Harte Verdrahtung	Getrennt
STYLING	Hart	Flexibel

UserControl vs. Custom Control - Gegenüberstellung

	USER CONTROL	CUSTOM CONTROL
TECHNIK	XAML + Code Behind	Klasse + Style + Template
LOGIK & VISUAL	Harte Verdrahtung	Getrennt
STYLING	Hart	Flexibel
AUFWAND*	Wenig	Etwas aufwändiger

UserControl vs. Custom Control - Gegenüberstellung

	USER CONTROL	CUSTOM CONTROL
TECHNIK	XAML + Code Behind	Klasse + Style + Template
LOGIK & VISUAL	Harte Verdrahtung	Getrennt
STYLING	Hart	Flexibel
AUFWAND*	Wenig	Etwas aufwändiger
REUSE	Schlecht	Gut

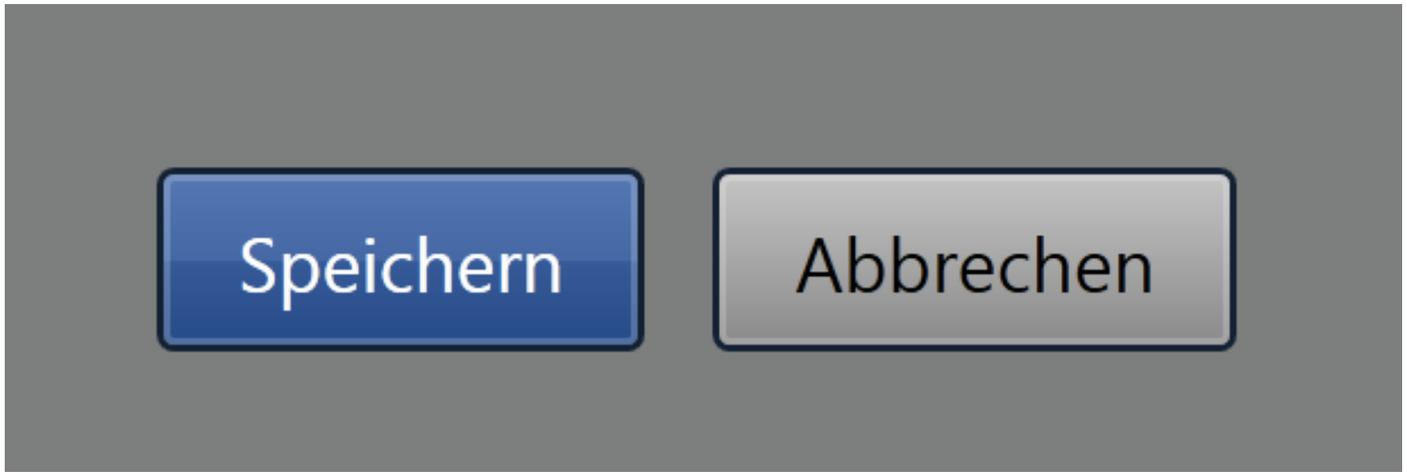
STYLE VERERBUNG

Vererbung

- Vererbung über Property BasedOn des Styles möglich
- Vererbt werden alle per Setter definierten Properties
- Überschreiben eines Setters jeder Zeit möglich

- Style Vererbung sollte aktiv genutzt werden!
  - Tabs, Buttons, DataGrid…

Beispiel an zwei Buttons



Beispiel an zwei Buttons

```
<Style x:Key="PrimaryButtonStyle" TargetType="{x:Type Button}" BasedOn="{StaticResource FrameworkElementStyle}">
    <Setter Property="Background" Value="{StaticResource ButtonBackgroundBrush}"/>
    <Setter Property="BorderBrush" Value="{StaticResource ButtonBorderBrush}"/>
    <Setter Property="Foreground" Value="{StaticResource FontBrush}"/>
    <Setter Property="HorizontalContentAlignment" Value="Center"/>
    <Setter Property="VerticalContentAlignment" Value="Center"/>
    <Setter Property="Padding" Value="11,0,11,0"/>
    <Setter Property="Height" Value="27"/>
    <Setter Property="FontSize" Value="{StaticResource Layer2FontSize}"/>
    <Setter Property="FontFamily" Value="{StaticResource PrimaryFontFamily}"/>
    <Setter Property="BorderThickness" Value="1"/>
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="{x:Type Button}">
                <Border Background="{TemplateBinding Background}"
                        BorderThickness="{TemplateBinding BorderThickness}"
                        Padding="0"
                        CornerRadius="2"
                        BorderBrush="{TemplateBinding BorderBrush}">
                    <Grid>
                        <Border x:Name="highLight"
                                BorderThickness="{TemplateBinding BorderThickness}"
                                CornerRadius="1"
                                BorderBrush="{StaticResource White20Brush}" />
                        <ContentPresenter Margin="{TemplateBinding Padding}"
                                          HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}"
                                          VerticalAlignment="{TemplateBinding VerticalContentAlignment}"
                                          RecognizesAccessKey="True" />
                    </Grid>
                </Border>
                <ControlTemplate.Triggers>
                    <Trigger Property="IsPressed" Value="true">
                        <Setter Property="BorderThickness" Value="0,0,0,1" TargetName="highLight"/>
                    </Trigger>
                </ControlTemplate.Triggers>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
    <Style.Triggers>
        <Trigger Property="IsPressed" Value="true">
            <Setter Property="Background" Value="{StaticResource ButtonPressedBackgroundBrush}"/>
        </Trigger>
    </Style.Triggers>
</Style>

<Style x:Key="SecondaryButtonStyle" TargetType="{x:Type Button}" BasedOn="{StaticResource PrimaryButtonStyle}">
    <Setter Property="Background" Value="{DynamicResource SecondaryButtonBackgroundBrush}"/>
```

Je mehr TemplateBinding desto „sauberer“ und flexibler der Style!

Beispiel an zwei Buttons

```
<Style x:Key="SecondaryButtonStyle" TargetType="{x:Type Button}" BasedOn="{StaticResource PrimaryButtonStyle}">
    <Setter Property="Background" Value="{DynamicResource SecondaryButtonBackgroundBrush}"/>
    <Setter Property="Foreground" Value="{StaticResource BlackBrush}"/>
    <Style.Triggers>
        <Trigger Property="IsPressed" Value="True">
            <Setter Property="Background" Value="{DynamicResource SecondaryButtonPressedBackgroundBrush}"/>
        </Trigger>
    </Style.Triggers>
</Style>
```

Beispiel an zwei Buttons

```
<Style x:Key="SecondaryButtonStyle" TargetType="{x:Type Button}" BasedOn="{StaticResource PrimaryButtonStyle}">
    <Setter Property="Background" Value="{DynamicResource SecondaryButtonBackgroundBrush}"/>
    <Setter Property="Foreground" Value="{StaticResource BlackBrush}"/>
    <Style.Triggers>
        <Trigger Property="IsPressed" Value="True">
            <Setter Property="Background" Value="{DynamicResource SecondaryButtonPressedBackgroundBrush}"/>
        </Trigger>
    </Style.Triggers>
</Style>
```

- SecondaryButtonStyle erbt von PrimaryButtonStyle
- Lediglich überschreiben von Background, Foreground und IsPressed Trigger (Style Ebene)
- Kein Kopieren der Template notwendig, kompakter Source-Code ;-)

```
<Style x:Key="PrimaryButtonStyle" TargetType="{x:Type Button}" BasedOn="{StaticResource FrameworkElementStyle}">
    <Setter Property="Background" Value="{StaticResource ButtonBackgroundBrush}"/>
    <Setter Property="BorderBrush" Value="{StaticResource ButtonBorderBrush}"/>
    <Setter Property="Foreground" Value="{StaticResource FontBrush}"/>
    <Setter Property="HorizontalContentAlignment" Value="Center"/>
    <Setter Property="VerticalContentAlignment" Value="Center"/>
    <Setter Property="Padding" Value="11,0,11,0"/>
    <Setter Property="Height" Value="27"/>
    <Setter Property="FontSize" Value="{StaticResource Layer2FontSize}"/>
    <Setter Property="FontFamily" Value="{StaticResource PrimaryFontFamily}"/>
    <Setter Property="BorderThickness" Value="1"/>
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="{x:Type Button}">
                <Border Background="{TemplateBinding Background}"
                        BorderThickness="{TemplateBinding BorderThickness}"
                        Padding="0"
                        CornerRadius="2"
                        BorderBrush="{TemplateBinding BorderBrush}">
                    <Grid>
                        <Border x:Name="highLight"
                                BorderThickness="{TemplateBinding BorderThickness}"
                                CornerRadius="1"
                                BorderBrush="{StaticResource White20Brush}" />
                        <ContentPresenter Margin="{TemplateBinding Padding}"
                                          HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}"
                                          VerticalAlignment="{TemplateBinding VerticalContentAlignment}"
                                          RecognizesAccessKey="True" />
                    </Grid>
                </Border>
                <ControlTemplate.Triggers>
                    <Trigger Property="IsPressed" Value="true">
                        <Setter Property="BorderThickness" Value="0,0,0,1" TargetName="highLight"/>
                    </Trigger>
                </ControlTemplate.Triggers>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
    <Style.Triggers>
        <Trigger Property="IsPressed" Value="true">
            <Setter Property="Background" Value="{StaticResource ButtonPressedBackgroundBrush}"/>
        </Trigger>
    </Style.Triggers>
</Style>

<Style x:Key="SecondaryButtonStyle" TargetType="{x:Type Button}" BasedOn="{StaticResource PrimaryButtonStyle}">
    <Setter Property="Background" Value="{DynamicResource SecondaryButtonBackgroundBrush}"/>
```

Wann immer mögliche Trigger auf Style Ebene

Design | ResourceDictionary ResourceDictionary

BASIS STYLE

Basis Style

```
<Style x:Key="FrameworkElementStyle" TargetType="FrameworkElement">
    <Setter Property="OverridesDefaultStyle" Value="True"/>
    <Setter Property="SnapsToDevicePixels" Value="True"/>
    <Setter Property="FocusVisualStyle" Value="{x:Null}"/>
    <Style.Triggers>
        <Trigger Property="IsEnabled" Value="False">
            <Setter Property="Opacity" Value=".5"/>
        </Trigger>
    </Style.Triggers>
</Style>
```

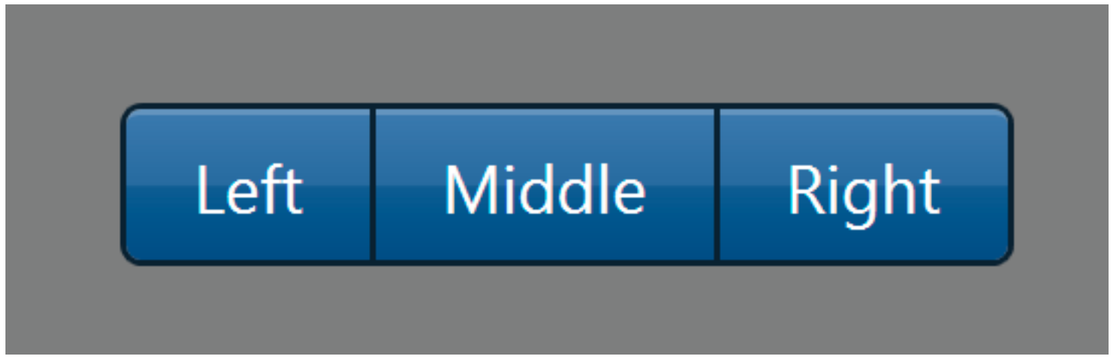
Basis Style

```
<Style x:Key="FrameworkElementStyle" TargetType="FrameworkElement">
    <Setter Property="OverridesDefaultStyle" Value="True"/>
    <Setter Property="SnapsToDevicePixels" Value="True"/>
    <Setter Property="FocusVisualStyle" Value="{x:Null}"/>
    <Style.Triggers>
        <Trigger Property="IsEnabled" Value="False">
            <Setter Property="Opacity" Value=".5"/>
        </Trigger>
    </Style.Triggers>
</Style>
```

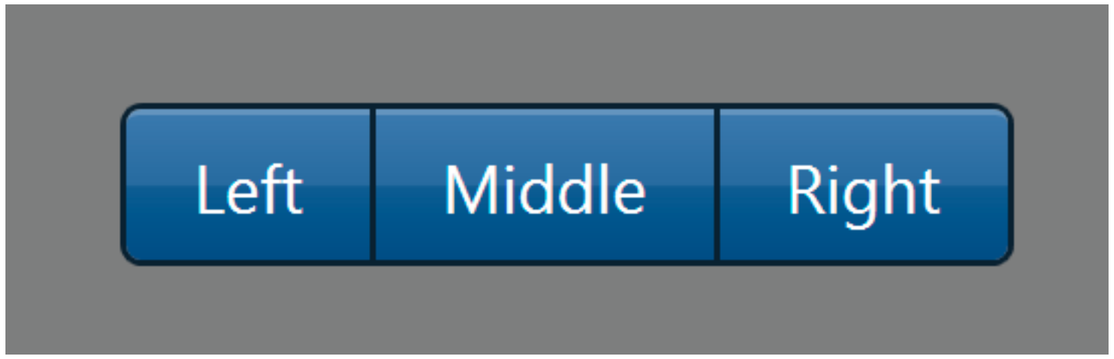
- Bei Styling ist ein Basis Style für alle UI Elemente empfehlenswert
- Wichtige Setter für SnapsToDevicePixels, OverridesDefaults etc.

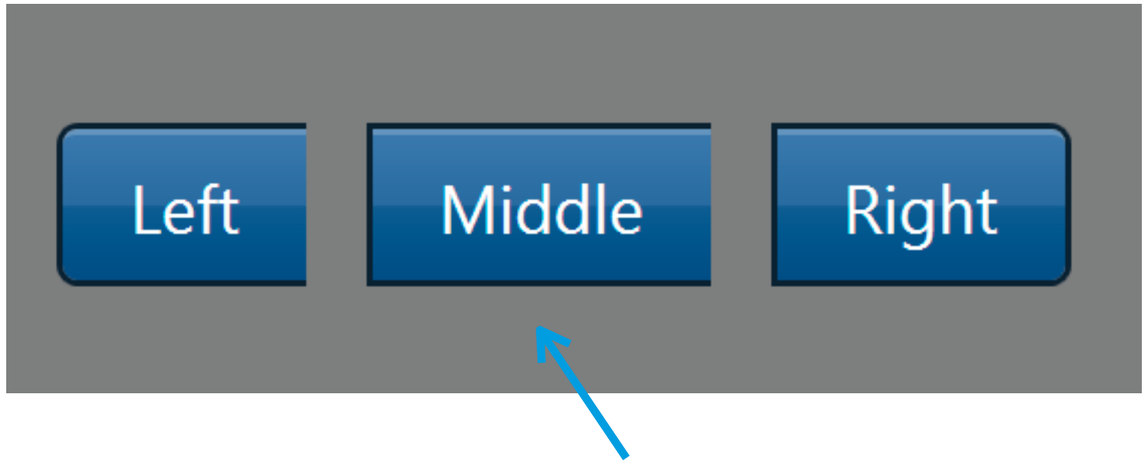
BEISPIEL SEGMENTED RADIOBUTTONS

Beispiel Segmented RadioButtons



Beispiel Segmented RadioButtons





Drei Teile notwendig

Unterschied BorderThickness, CornerRadius

```
<Style x:Key="LeftSegmentedRadionButtonStyle" TargetType="{x:Type RadioButton}" BasedOn="{StaticResource FrameworkElement
    <Setter Property="Background" Value="{StaticResource ButtonBackgroundBrush}"/>
    <Setter Property="BorderBrush" Value="{StaticResource ButtonBorderBrush}"/>
    <Setter Property="Foreground" Value="{StaticResource FontBrush}"/>
    <Setter Property="HorizontalContentAlignment" Value="Center"/>
    <Setter Property="VerticalContentAlignment" Value="Center"/>
    <Setter Property="Padding" Value="11,0,11,0"/>
    <Setter Property="Height" Value="27"/>
    <Setter Property="FontSize" Value="{StaticResource Layer2FontSize}"/>
    <Setter Property="FontFamily" Value="{StaticResource PrimaryFontFamily}"/>
    <Setter Property="BorderThickness" Value="1,1,0,1"/>
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="{x:Type RadioButton}">
                <Border Background="{TemplateBinding Background}"
                        BorderThickness="{TemplateBinding BorderThickness}"
                        CornerRadius="3,0,0,3"
                        BorderBrush="{TemplateBinding BorderBrush}">
                    <Grid>
                        <Border x:Name="highLight"
                                BorderThickness="0,1,0,0"
                                CornerRadius="2,0,0,0"
                                BorderBrush="{StaticResource White20Brush}" />
                        <ContentPresenter Margin="{TemplateBinding Padding}"
                                          HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}"
                                          VerticalAlignment="{TemplateBinding VerticalContentAlignment}"
                                          RecognizesAccessKey="True" />
                    </Grid>
                </Border>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
    <Style.Triggers>
        <Trigger Property="IsChecked" Value="true">
            <Setter Property="Background" Value="{StaticResource ButtonPressedBackgroundBrush}"/>
        </Trigger>
    </Style.Triggers>
</Style>
```

Styling des Left Segmented RadioButton

Beispiel Segmented RadioButtons - Middle

```
<Style x:Key="MiddleSegmentedRadionButtonStyle" TargetType="{x:Type RadioButton}" BasedOn="{StaticResource LeftSegmentedRadionButtonStyle}">
    <!-- Corner radius? :-(, we needed to copy the template to adjust the corner radius ;-/ -->
     <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="{x:Type RadioButton}">
                <Border Background="{TemplateBinding Background}"
                        BorderThickness="{TemplateBinding BorderThickness}"
                        CornerRadius="0"
                        BorderBrush="{TemplateBinding BorderBrush}">
                    <Grid>
                        <Border x:Name="highLight"
                                BorderThickness="0,1,0,0"
                                CornerRadius="0"
                                BorderBrush="{StaticResource White20Brush}" />
                        <ContentPresenter Margin="{TemplateBinding Padding}"
                                          HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}"
                                          VerticalAlignment="{TemplateBinding VerticalContentAlignment}"
                                          RecognizesAccessKey="True" />
                    </Grid>
                </Border>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>
```

Beispiel Segmented RadioButtons - Middle

```
<Style x:Key="MiddleSegmentedRadionButtonStyle" TargetType="{x:Type RadioButton}" BasedOn="{StaticResource LeftSegmentedRadionButtonStyle}">
    <!-- Corner radius? :-(, we needed to copy the template to adjust the corner radius ;-/ -->
     <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="{x:Type RadioButton}">
                <Border Background="{TemplateBinding Background}"
                        BorderThickness="{TemplateBinding BorderThickness}"
                        CornerRadius="0"
                        BorderBrush="{TemplateBinding BorderBrush}">
                    <Grid>
                        <Border x:Name="highLight"
                                BorderThickness="0,1,0,0"
                                CornerRadius="0"
                                BorderBrush="{StaticResource White20Brush}" />
                        <ContentPresenter Margin="{TemplateBinding Padding}"
                                          HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}"
                                          VerticalAlignment="{TemplateBinding VerticalContentAlignment}"
                                          RecognizesAccessKey="True" />
                    </Grid>
                </Border>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>
```

- Trotz Vererbung müssen wir die Control Template anpassen & kopieren
- CornerRadius bei Border und Lichtkante :-(

Beispiel Segmented RadioButtons - Right

```
<Style x:Key="RightSegmentedRadionButtonStyle" TargetType="{x:Type RadioButton}" BasedOn="{StaticResource LeftSegmentedRadionButtonStyle}">
    <Setter Property="BorderThickness" Value="1"/>
    <!-- Corner radius? :-(, we needed to copy the template to adjust the corner radius ;-/ -->
     <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="{x:Type RadioButton}">
                <Border Background="{TemplateBinding Background}"
                        BorderThickness="{TemplateBinding BorderThickness}"
                        CornerRadius="0,3,3,0"
                        BorderBrush="{TemplateBinding BorderBrush}">
                    <Grid>
                        <Border x:Name="highLight"
                                BorderThickness="0,1,0,0"
                                CornerRadius="0,2,0,0"
                                BorderBrush="{StaticResource White20Brush}" />
                        <ContentPresenter Margin="{TemplateBinding Padding}"
                                          HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}"
                                          VerticalAlignment="{TemplateBinding VerticalContentAlignment}"
                                          RecognizesAccessKey="True" />
                    </Grid>
                </Border>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>
```

Beispiel Segmented RadioButtons...

```
<StackPanel Orientation="Horizontal" HorizontalAlignment="Center" VerticalAlignment="Center" Margin="0,30,0,0">
    <RadioButton x:Name="button3" Content="Left" Style="{DynamicResource LeftSegmentedRadionButtonStyle}" />
    <RadioButton x:Name="button4" Content="Middle" Style="{DynamicResource MiddleSegmentedRadionButtonStyle}" />
    <RadioButton x:Name="button5" Content="Right" Style="{DynamicResource RightSegmentedRadionButtonStyle}" />
</StackPanel>
```

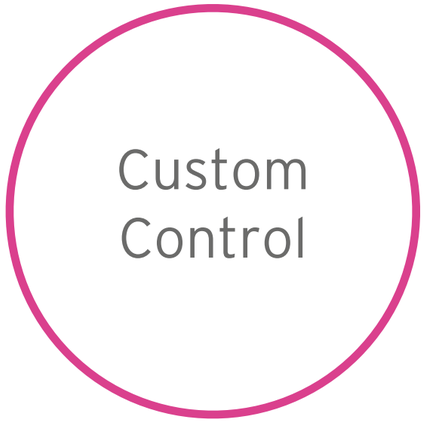
KEINE ANGST VOR CUSTOM CONTROLS!

Oh oh Custom Controls

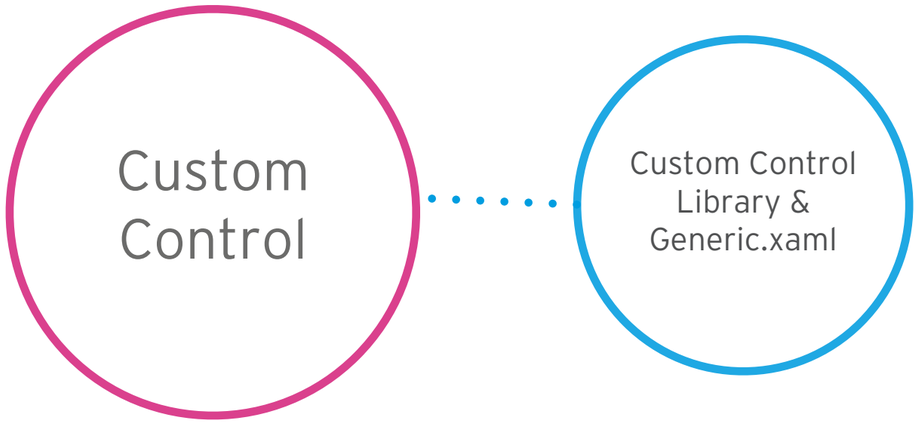
- Ableitung von einer konkreten **Klasse != UserControl**
- Styling- und Templating-Möglichkeiten
- Visueller Aufbau im ControlTemplate
- Zusammenfassung in einer Control Library möglich

Oh oh Custom Controls - Handwerkzeugs...

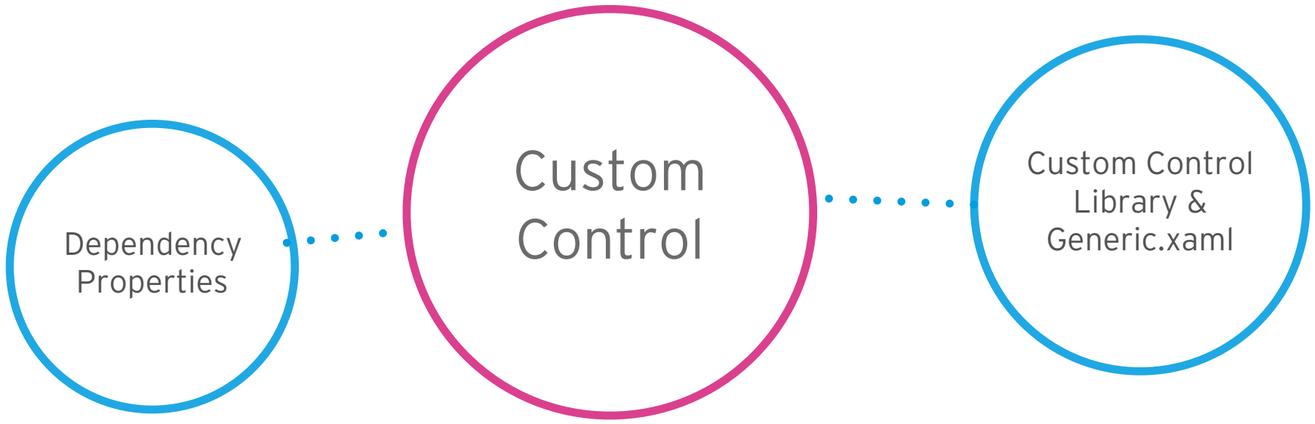
Oh oh Custom Controls - Handwerkzeugs...



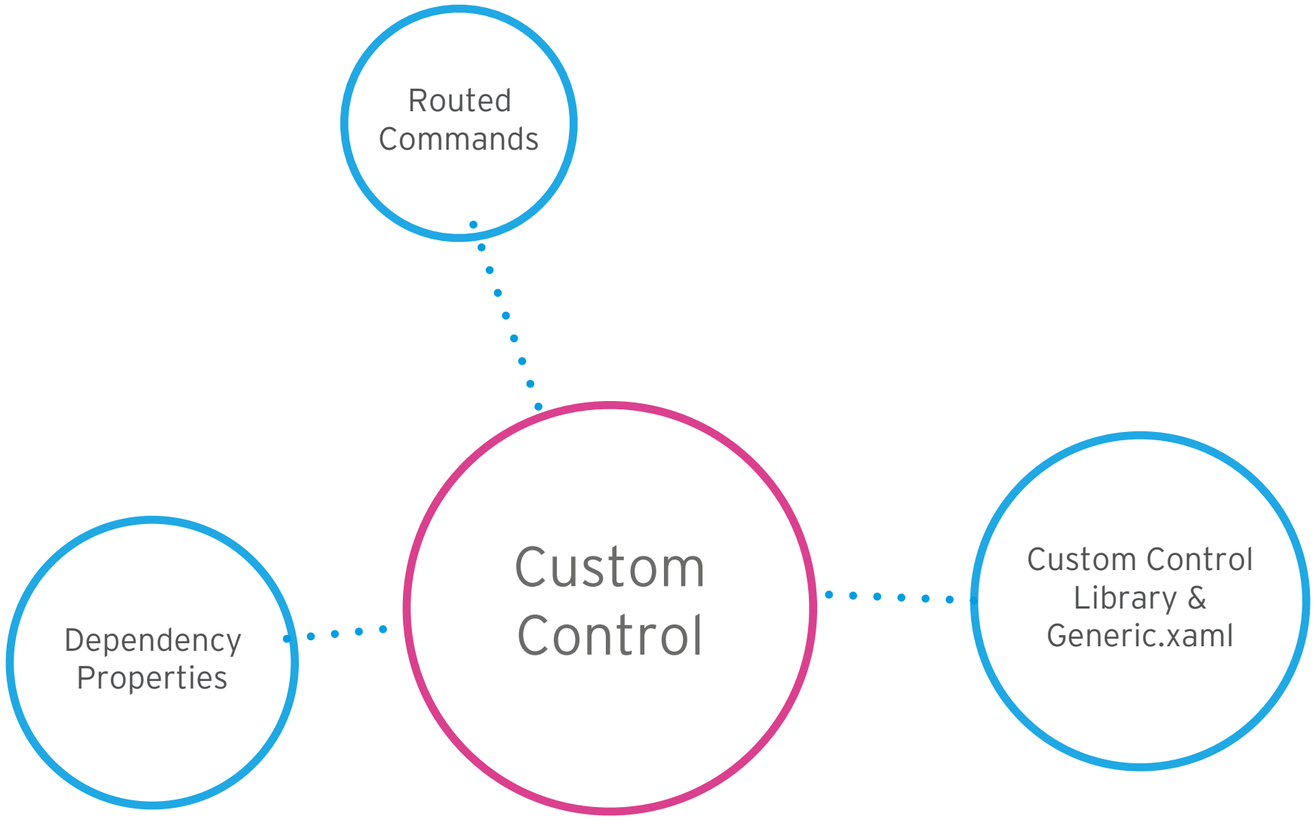
Oh oh Custom Controls - Handwerkzeugs...



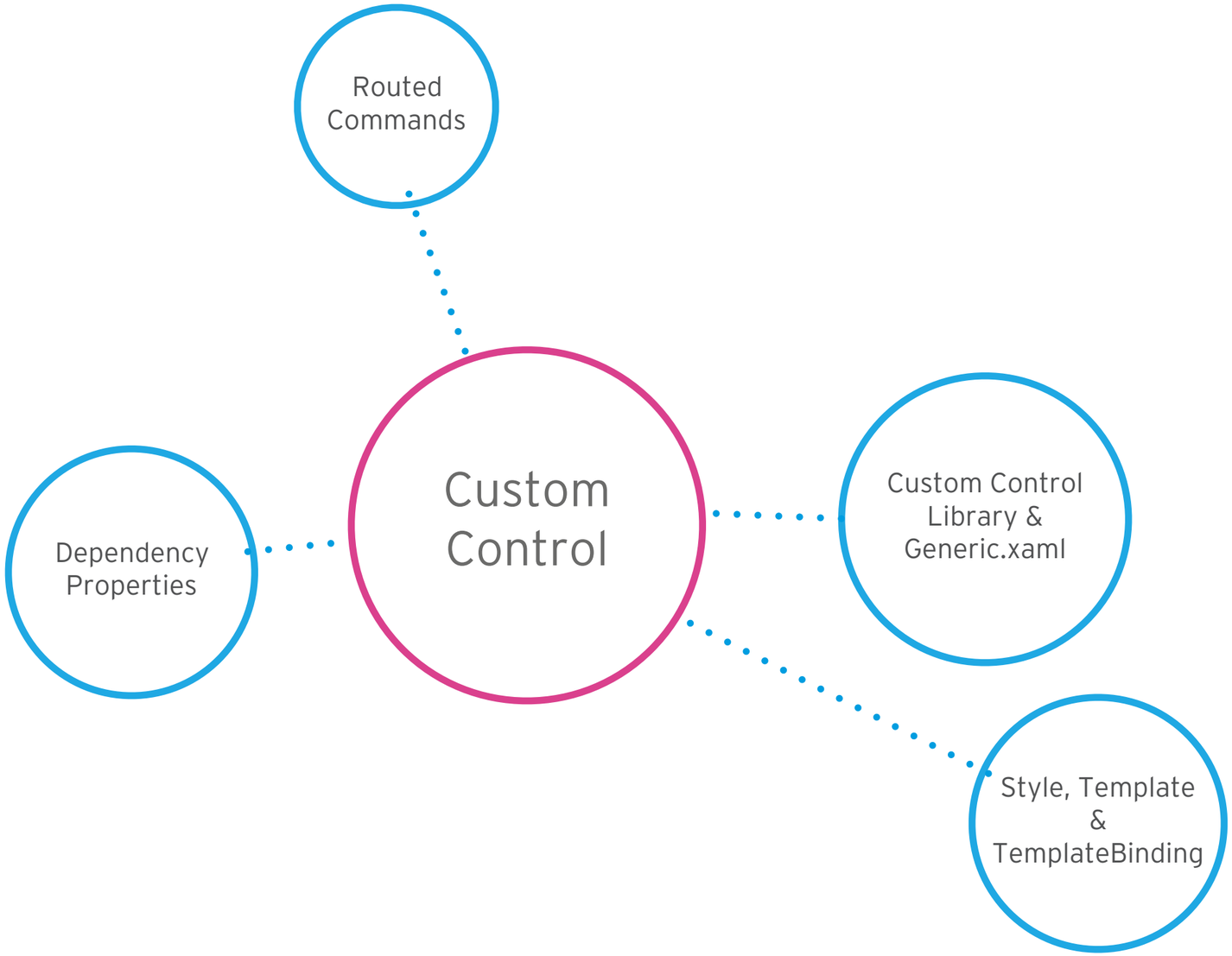
Oh oh Custom Controls - Handwerkzeugs...



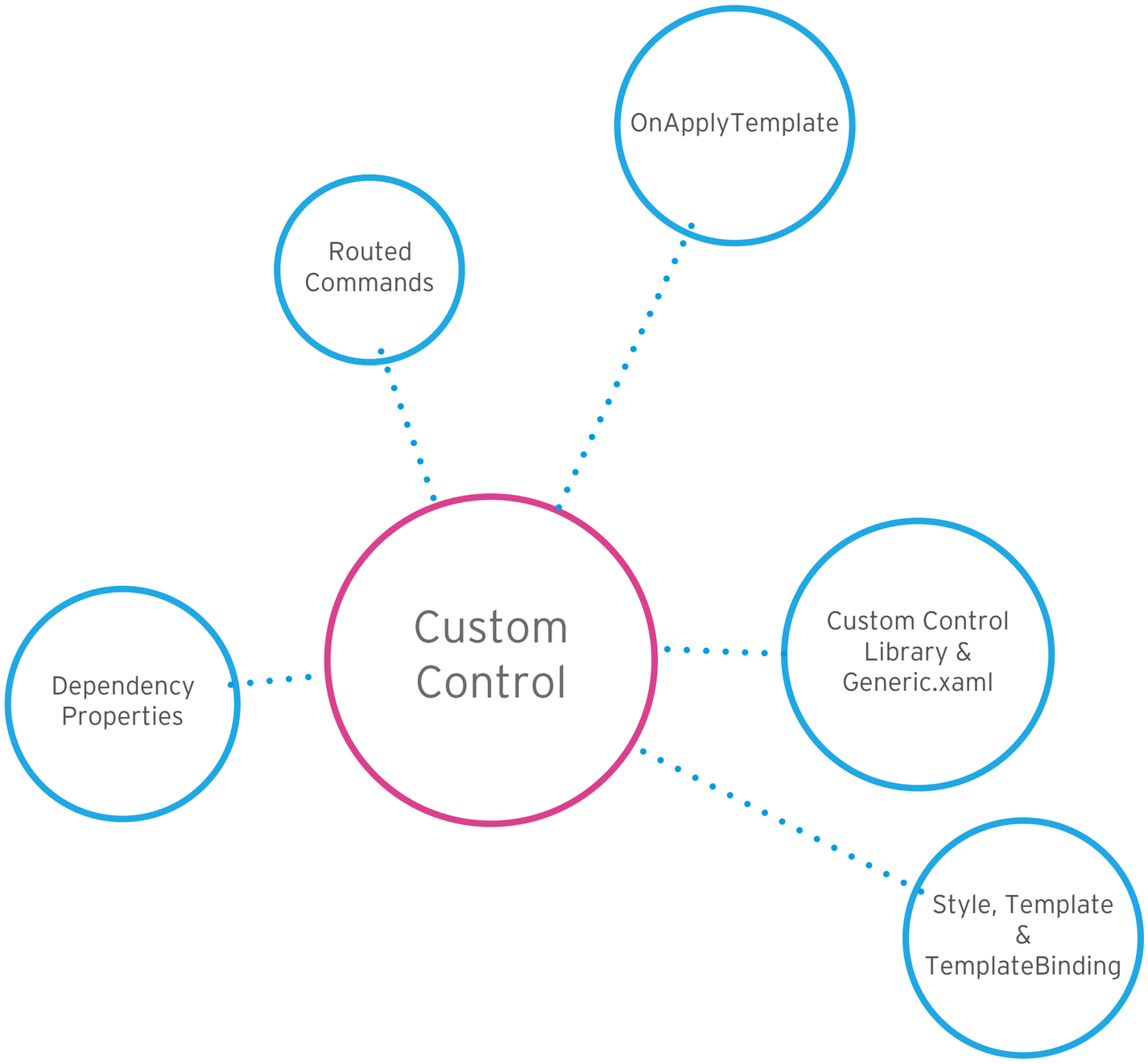
Oh oh Custom Controls - Handwerkzeugs...



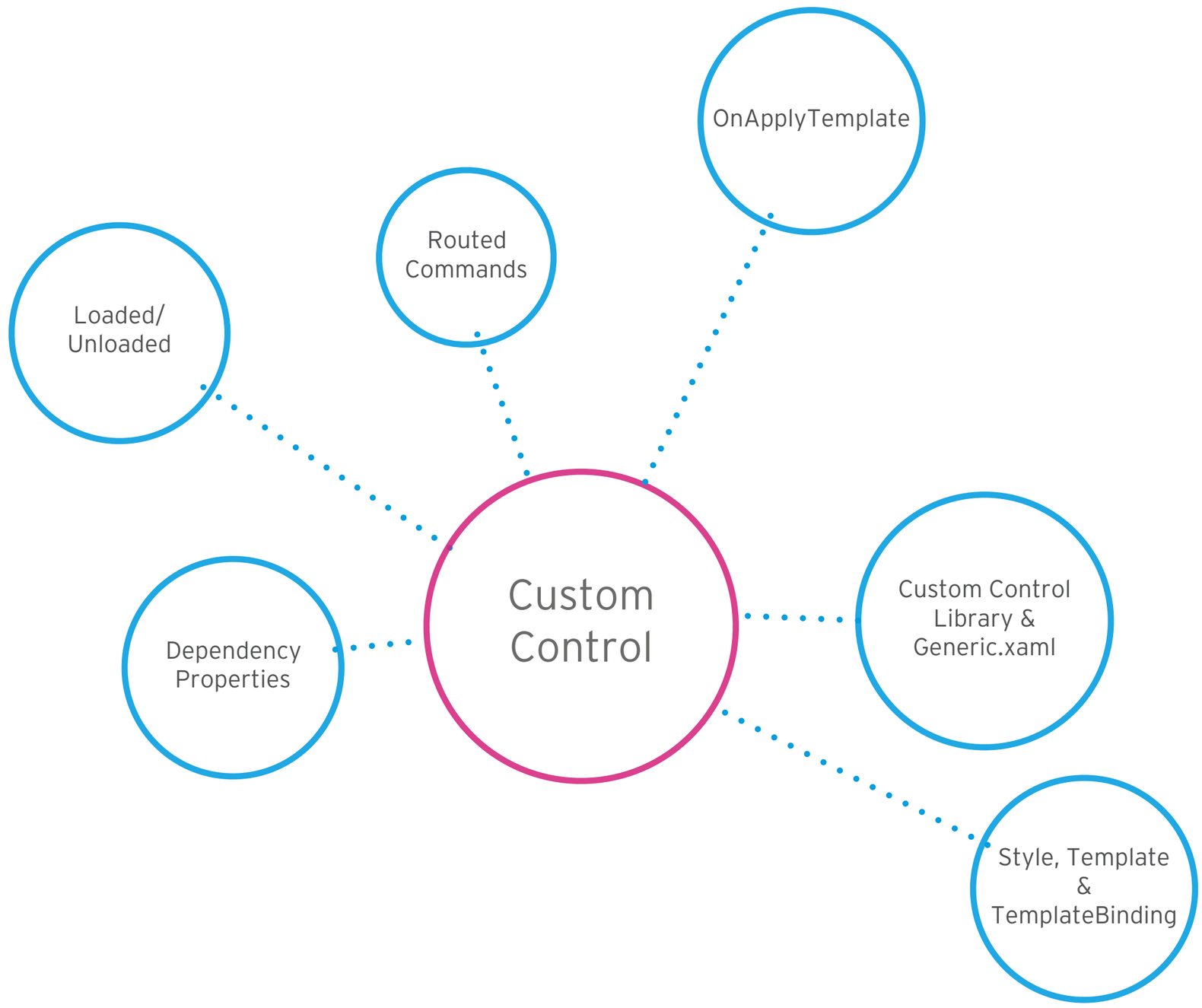
Oh oh Custom Controls - Handwerkzeugs...



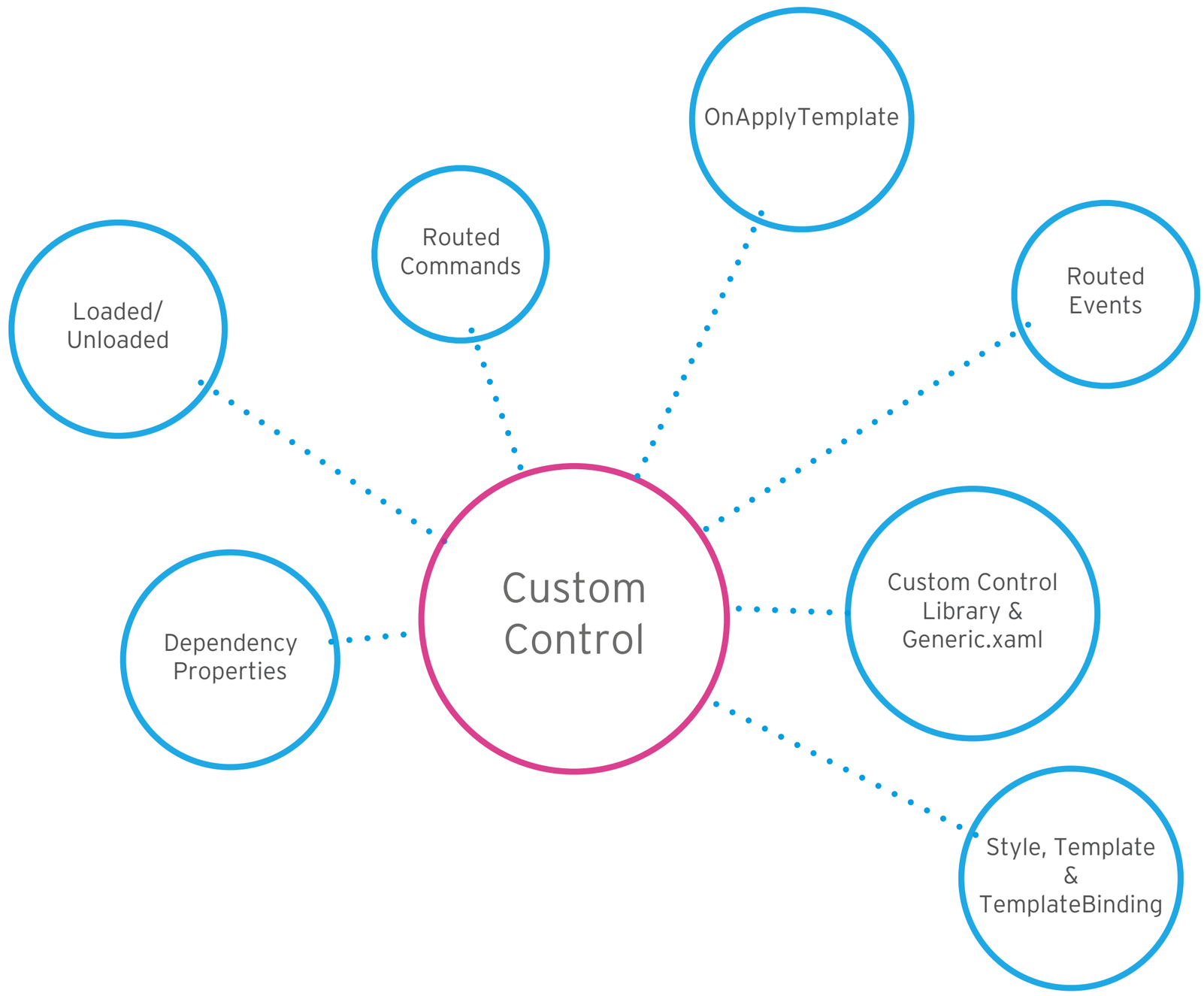
Oh oh Custom Controls - Handwerkzeugs...



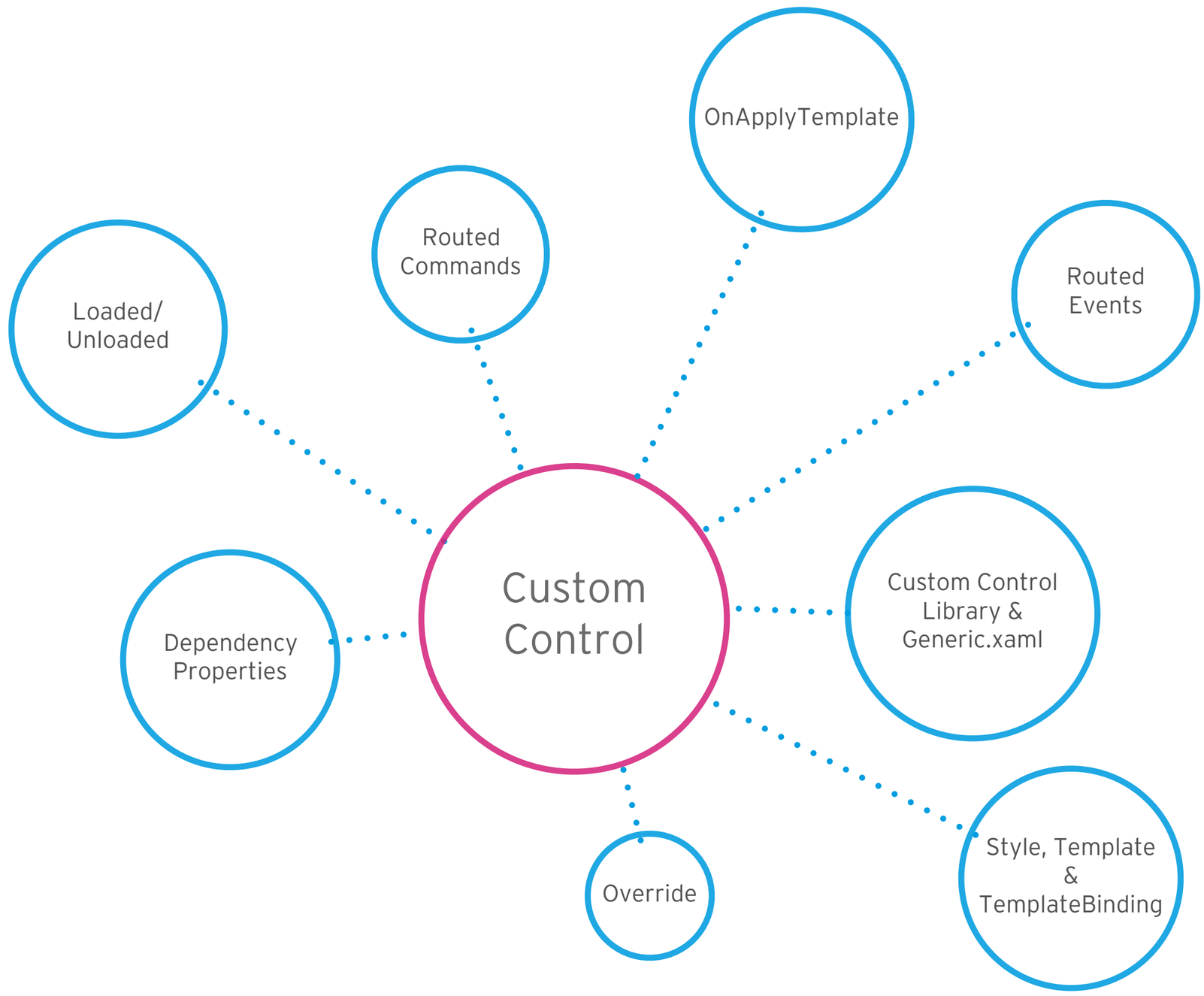
Oh oh Custom Controls - Handwerkzeugs...



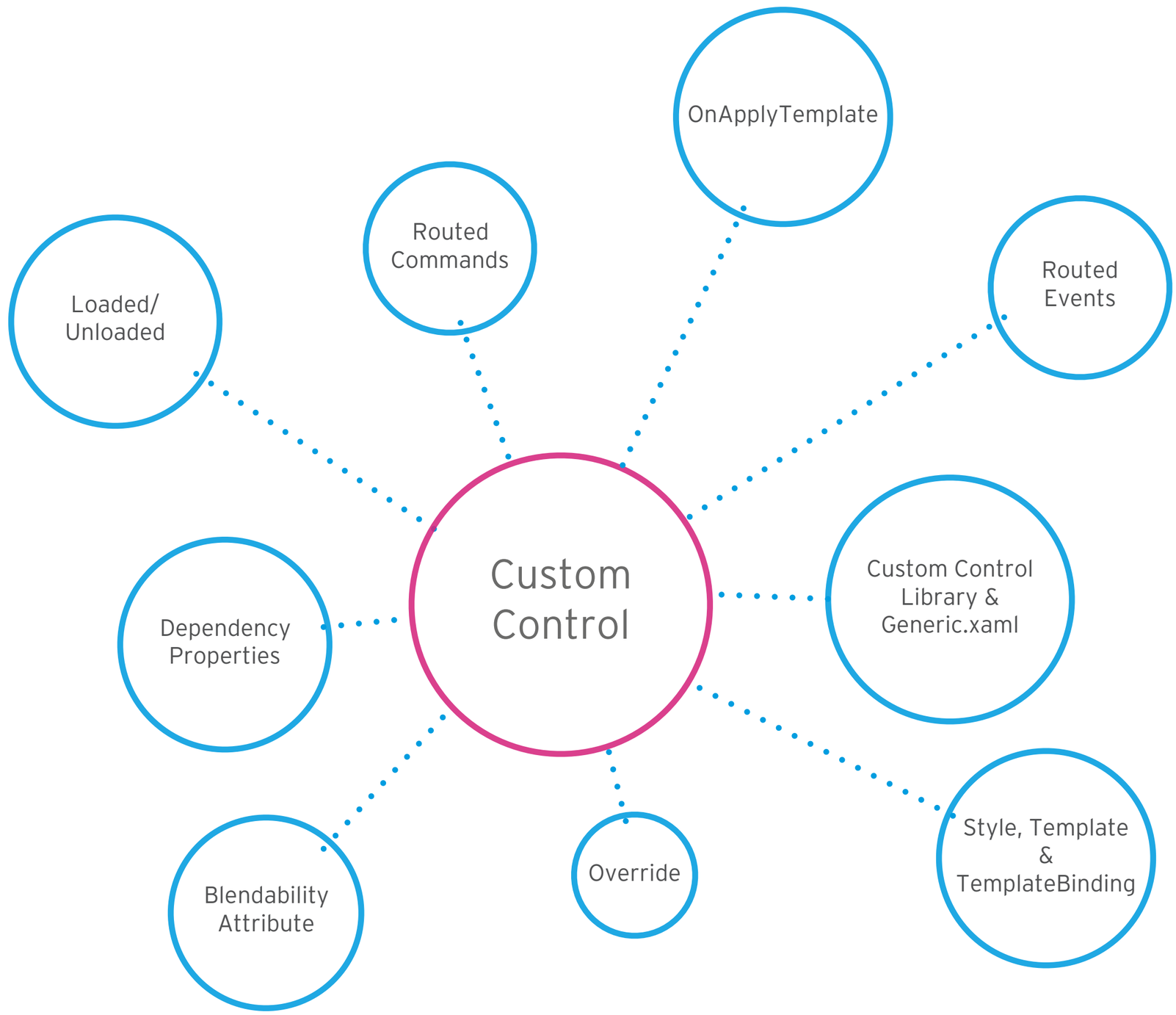
Oh oh Custom Controls - Handwerkzeugs...



Oh oh Custom Controls - Handwerkzeugs...



Oh oh Custom Controls - Handwerkzeugs...



Controls.XRadioButton | CornerRadiusProperty

```
///
/// <para>Author: David C. Thoemmes, André Lanninger</para>
/// <para>Date: 11.05.2011</para>
/// </summary>
public class XRadioButton : RadioButton
{
    /// <summary>
    /// Category description.
    /// </summary>
    private const string PROPERTY_CATEGORY = "ERGOSIGN XRadioButton Properties";

    #region DependencyProperties

    /// <summary>
    /// CornerRadiusProperty - Gets or sets the CornerRadius.
    /// </summary>
    public static readonly DependencyProperty CornerRadiusProperty = DependencyProperty.Register("CornerRadius", typeof(CornerRadius), typeof(XRadioButton), new P

    /// <summary>
    /// HighlightCornerRadiusProperty - Gets or sets the HightligtCornerRadius.
    /// </summary>
    public static readonly DependencyProperty HighlightCornerRadiusProperty = DependencyProperty.Register("HighlightCornerRadius", typeof(CornerRadius), typeof(XR

    /// <summary>
    /// HighlightBorderThicknessProperty - Gets or sets the HighlightBorderThickness.
    /// </summary>
    public static readonly DependencyProperty HighlightBorderThicknessProperty = DependencyProperty.Register("HighlightBorderThickness", typeof(Thickness), typeof

    /// <summary>
    /// HighlightBorderBrushProperty - Gets or sets the HighlightBorderBrushProperty.
    /// </summary>
    public static readonly DependencyProperty HighlightBorderBrushProperty = DependencyProperty.Register("HighlightBorderBrush", typeof(Brush), typeof(XRadioButto

    /// <summary>
    /// TextDropShadowColorProperty - Gets or sets the color of the text drop shadow.
    /// </summary>
    public static readonly DependencyProperty TextDropShadowColorProperty = DependencyProperty.Register("TextDropShadowColor", typeof(Brush), typeof(XRadioButton)

    /// <summary>
    /// FocusColorProperty - Gets or sets the color of the focus border.
    /// </summary>
    public static readonly DependencyProperty FocusColorProperty = DependencyProperty.Register("FocusColor", typeof(Brush), typeof(XRadioButton), new PropertyMeta

    /// <summary>
    /// IconProperty - Gets or sets the IconProperty.
    /// </summary>
    public static readonly DependencyProperty IconProperty = DependencyProperty.Register("Icon", typeof(object), typeof(XRadioButton), new PropertyMetadata(null))
```

Nur Properties hinzufügen :-)

oButton.cs

GOSIGN.XControls.XRadioButton | CornerRadiusProperty

```
40
41  ·</para>
42
43
44
45
46
47
48
49  :GN XRadioButton Properties";
50
51
52
53
54  ıerRadius.
55
56  ·RadiusProperty = DependencyProperty.Register("CornerRadius", typeof(CornerRadius), typeof(XRadioButton), new PropertyMetadata(new CornerRadius(0)));
57
58
59  ; the HightligtCornerRadius.
60
61  ıghtCornerRadiusProperty = DependencyProperty.Register("HighlightCornerRadius", typeof(CornerRadius), typeof(XRadioButton), new PropertyMetadata(new CornerRa
62
63
64  ;ets the HighlightBorderThickness.
65
66  ıghtBorderThicknessProperty = DependencyProperty.Register("HighlightBorderThickness", typeof(Thickness), typeof(XRadioButton), new PropertyMetadata(new Thick
67
68
69   the HighlightBorderBrushProperty.
70
71  ıghtBorderBrushProperty = DependencyProperty.Register("HighlightBorderBrush", typeof(Brush), typeof(XRadioButton), new PropertyMetadata(null));
72
73
74  :he color of the text drop shadow.
75
76  ·opShadowColorProperty = DependencyProperty.Register("TextDropShadowColor", typeof(Brush), typeof(XRadioButton), new PropertyMetadata(null));
77
78
79   of the focus border.
80
81  :olorProperty = DependencyProperty.Register("FocusColor", typeof(Brush), typeof(XRadioButton), new PropertyMetadata(null));
82
83
84  /.
85
86  ·operty = DependencyProperty.Register("Icon", typeof(object), typeof(XRadioButton), new PropertyMetadata(null));
87
```

```
<Style x:Key="XLeftSegmentedRadionButtonStyle" TargetType="{x:Type XRadioButton}" BasedOn="{StaticResource FrameworkElementStyle}"
    <Setter Property="Background" Value="{StaticResource ButtonBackgroundBrush}"/>
    <Setter Property="BorderBrush" Value="{StaticResource ButtonBorderBrush}"/>
    <Setter Property="Foreground" Value="{StaticResource FontBrush}"/>
    <Setter Property="HorizontalContentAlignment" Value="Center"/>
    <Setter Property="VerticalContentAlignment" Value="Center"/>
    <Setter Property="Padding" Value="11,0,11,0"/>
    <Setter Property="Height" Value="27"/>
    <Setter Property="FontSize" Value="{StaticResource Layer2FontSize}"/>
    <Setter Property="FontFamily" Value="{StaticResource PrimaryFontFamily}"/>
    <Setter Property="BorderThickness" Value="1,1,0,1"/>
    <Setter Property="CornerRadius" Value="3,0,0,3"/>
    <Setter Property="HighlightBorderThickness" Value="0,1,0,0"/>
    <Setter Property="HighlightBorderBrush" Value="{StaticResource White20Brush}"/>
    <Setter Property="HighlightCornerRadius" Value="2,0,0,0"/>
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="{x:Type XRadioButton}">
                <Border Background="{TemplateBinding Background}"
                        BorderThickness="{TemplateBinding BorderThickness}"
                        CornerRadius="{TemplateBinding CornerRadius}"
                        BorderBrush="{TemplateBinding BorderBrush}">
                    <Grid>
                        <Border x:Name="highLight"
                                BorderThickness="{TemplateBinding HighlightBorderThickness}"
                                BorderBrush="{TemplateBinding HighlightBorderBrush}"
                                CornerRadius="{TemplateBinding HighlightCornerRadius}" />
                        <ContentPresenter Margin="{TemplateBinding Padding}"
                                          HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}"
                                          VerticalAlignment="{TemplateBinding VerticalContentAlignment}"
                                          RecognizesAccessKey="True" />
                    </Grid>
                </Border>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
    <Style.Triggers>
        <Trigger Property="IsChecked" Value="true">
            <Setter Property="Background" Value="{StaticResource ButtonPressedBackgroundBrush}"/>
        </Trigger>
    </Style.Triggers>
</Style>
```

TemplateBinding an die neuen Properties möglich

Das Beispiel mit den Extended Radio Buttons

```
<Style x:Key="XMiddleSegmentedRadionButtonStyle" TargetType="{x:Type XRadioButton}" BasedOn="{StaticResource XLeftSegmentedRadionButtonStyle}">
    <!-- Wuhu :-) -->
    <Setter Property="CornerRadius" Value="0"/>
    <Setter Property="HighlightBorderThickness" Value="0,1,0,0"/>
    <Setter Property="HighlightCornerRadius" Value="0"/>
</Style>

<Style x:Key="XRightSegmentedRadionButtonStyle" TargetType="{x:Type XRadioButton}" BasedOn="{StaticResource XLeftSegmentedRadionButtonStyle}">
    <Setter Property="BorderThickness" Value="1"/>
    <Setter Property="CornerRadius" Value="0,3,3,0"/>
    <Setter Property="HighlightBorderThickness" Value="0,1,0,0"/>
    <Setter Property="HighlightCornerRadius" Value="0,2,0,0"/>
</Style>
```

Das Beispiel mit den Extended Radio Buttons

```
<Style x:Key="XMiddleSegmentedRadionButtonStyle" TargetType="{x:Type XRadioButton}" BasedOn="{StaticResource XLeftSegmentedRadionButtonStyle}">
    <!-- Wuhu :-) -->
    <Setter Property="CornerRadius" Value="0"/>
    <Setter Property="HighlightBorderThickness" Value="0,1,0,0"/>
    <Setter Property="HighlightCornerRadius" Value="0"/>
</Style>

<Style x:Key="XRightSegmentedRadionButtonStyle" TargetType="{x:Type XRadioButton}" BasedOn="{StaticResource XLeftSegmentedRadionButtonStyle}">
    <Setter Property="BorderThickness" Value="1"/>
    <Setter Property="CornerRadius" Value="0,3,3,0"/>
    <Setter Property="HighlightBorderThickness" Value="0,1,0,0"/>
    <Setter Property="HighlightCornerRadius" Value="0,2,0,0"/>
</Style>
```

- Kein kopieren der Control Template notwendig :-)
- Super kompakter XAML Source-Code ohne Redundanzen

- Nächste Stufe Enum „SegmentPostion“ + Trigger … ;-)

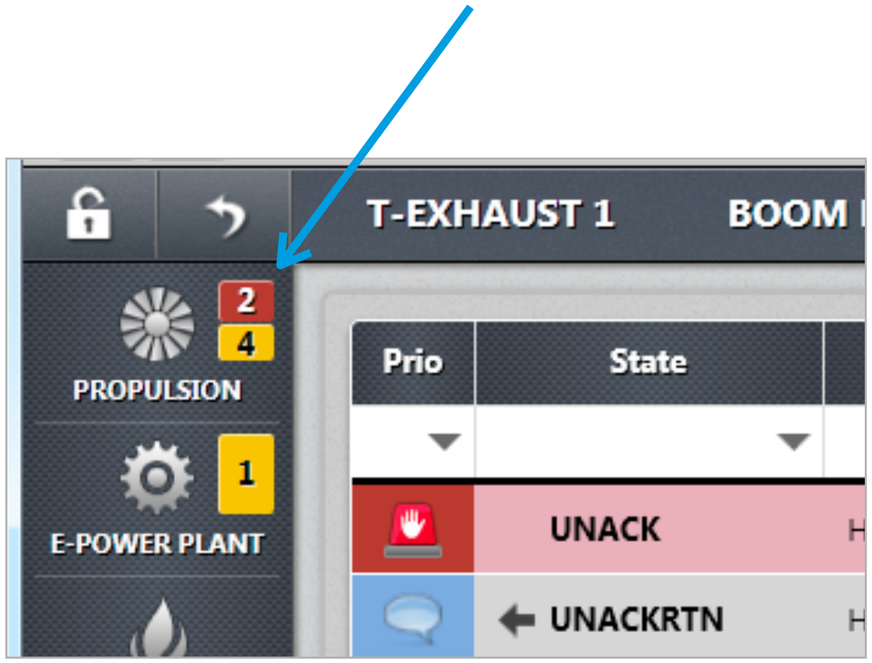
Das Beispiel mit den Extended Radio Buttons

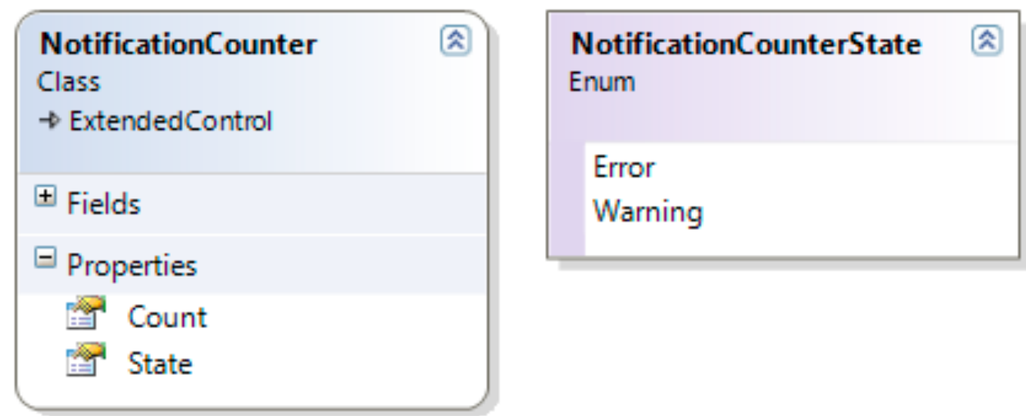
```
<StackPanel Orientation="Horizontal" HorizontalAlignment="Center" VerticalAlignment="Center" Margin="0,30,0,0">
    <XRadioButton x:Name="button6" Content="Left" Style="{DynamicResource XLeftSegmentedRadionButtonStyle}" />
    <XRadioButton x:Name="button7" Content="Middle" Style="{DynamicResource XMiddleSegmentedRadionButtonStyle}" />
    <XRadioButton x:Name="button8" Content="Right" Style="{DynamicResource XRightSegmentedRadionButtonStyle}" />
</StackPanel>
```

LEICHTGEWICHTIG

Leichtgewichtig… :-)

Counter, wird an diversen Stellen im UI verwendet…





CUSTOM LAYOUT CONTAINER

Custom Layout Container

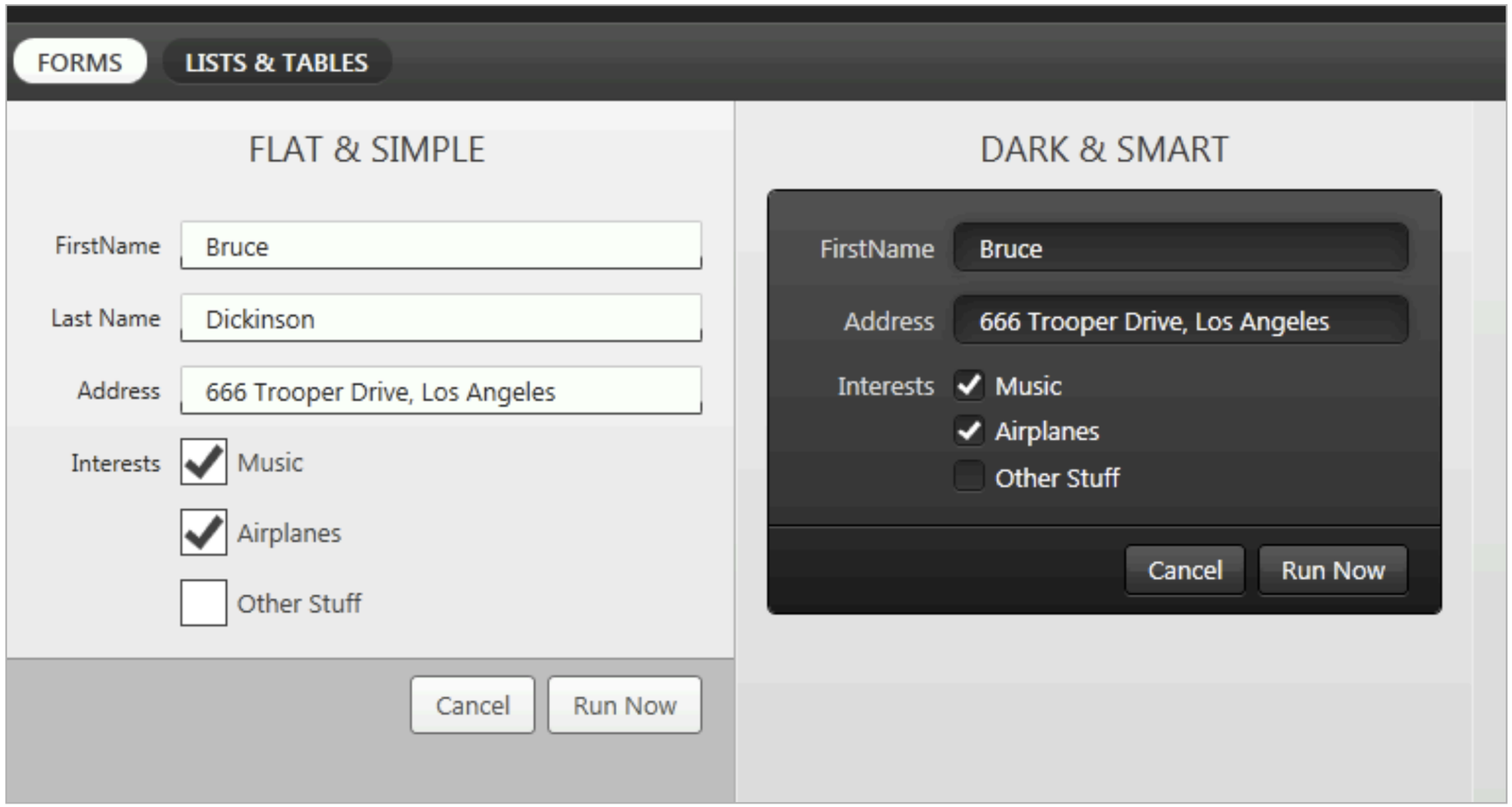
Strategie

- Layout und Design Patterns als Custom Controls abbilden
- Abstraktion und Identifikation der Patterns

Ideen

- Formular
- Dialog oder Window
- Master/Detail
- ...

Custom Layout Container



```
<Grid x:Name="lGrid"
      Margin="2,0,0,0">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto" />
        <ColumnDefinition Width="10" />
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="20" />
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="12" />
        <RowDefinition Height="Auto"
                       MinHeight="37" />
        <RowDefinition Height="7" />
        <RowDefinition Height="30" />
        <RowDefinition Height="7" />
        <RowDefinition Height="30" />
        <RowDefinition Height="7" />
        <RowDefinition Height="30" />
        <RowDefinition Height="7" />
        <RowDefinition Height="auto" />
        <RowDefinition Height="17" />
        <RowDefinition Height="*" />
    </Grid.RowDefinitions>

    <Border Background="{DynamicResource FormsViewFlatAndSimpleBackground01Brush}"
            BorderThickness="0,0,1,0"
            Grid.ColumnSpan="4"
            Grid.RowSpan="11"
            BorderBrush="{DynamicResource TransparentBlackBorder01Brush}" />

    <TextBlock Text="Flat And Simple"
               Style="{DynamicResource HeaderTextBlockStyle}"
               HorizontalAlignment="Center"
               Grid.Row="1"
               Grid.ColumnSpan="4" />

    <TextBlock Text="FirstName"
               Grid.Row="3"
               Style="{DynamicResource GridFormularTextBlockStyle}"
               VerticalAlignment="Center"
               HorizontalAlignment="Right" />
    <TextBox Text="Bruce"
             Style="{DynamicResource SimpleTextBoxStyle}"
             Grid.Row="3"
             Grid.Column="2"
             VerticalAlignment="Center" />

    <TextBlock Text="Last Name"
               Grid.Row="5"
               Style="{DynamicResource GridFormularTextBlockStyle}"
```

Problem

Komplexes Layout erhöht die Gefahr von Inkonsistenzen während der UI Entwicklung

Edit View Object Project Tools Window Help

P... Ass... Trig... Sta... Parts

Search

Solution "ERGOSIGNShowcase" (6 projects)

- CrawlTest
- Ergosign.Controls.FormLayoutContainer

Objects and Timeline

(No Storyboard open)

- [UserControl]
- [UserControl]
  - [Grid]
    - IGrid
    - mGrid
      - [Border]
      - [TextBlock] "DARK & SMART"
      - [FormLayoutContainer]
        - Footer
        - Header
        - [FormLayoutElement]
        - [FormLayoutElement]
        - [FormLayoutElement]
        - [FormLayoutElement]
        - [FormLayoutElement]
    - rGrid

FormLayoutContainer.xaml | FormsView.xaml | Button.xaml

[FormLayoutContainer]

DARK & SMART

FirstName Bruce

Address 666 Trooper Drive, Los Angeles

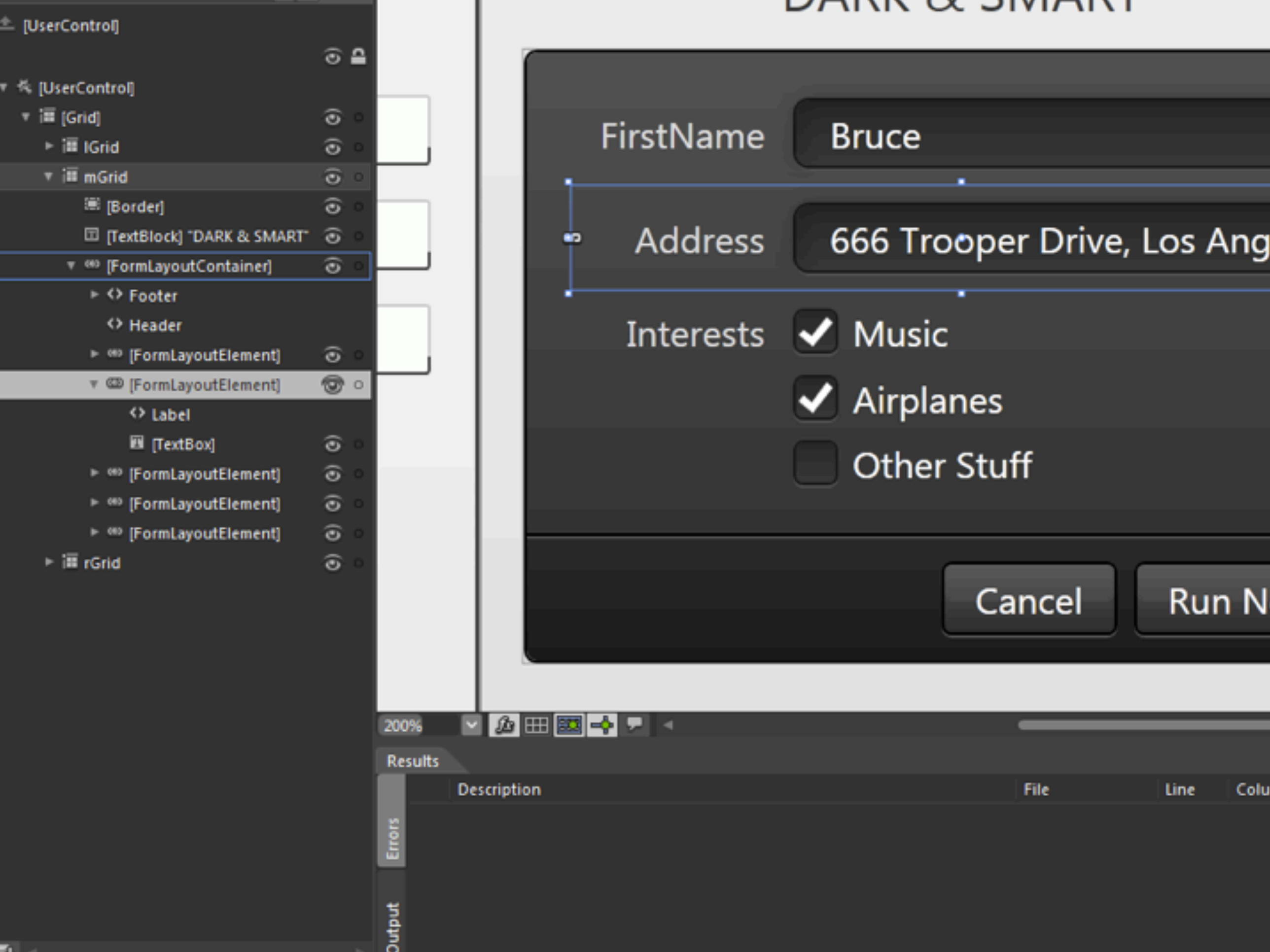
Interests

- [x] Music
- [x] Airplanes
- [ ] Other Stuff

Cancel Run Now

200%

Results


[UserControl]
[UserControl]
[Grid]
IGrid
mGrid
[Border]
[TextBlock] "DARK & SMART"
[FormLayoutContainer]
Footer
Header
[FormLayoutElement]
[FormLayoutElement]
Label
[TextBox]
[FormLayoutElement]
[FormLayoutElement]
[FormLayoutElement]
rGrid
FirstName
Bruce
Address
666 Trooper Drive, Los Ang
Interests
Music
Airplanes
Other Stuff
Cancel
Run N
200%
Results
Errors
Output
Description
File
Line
Colu

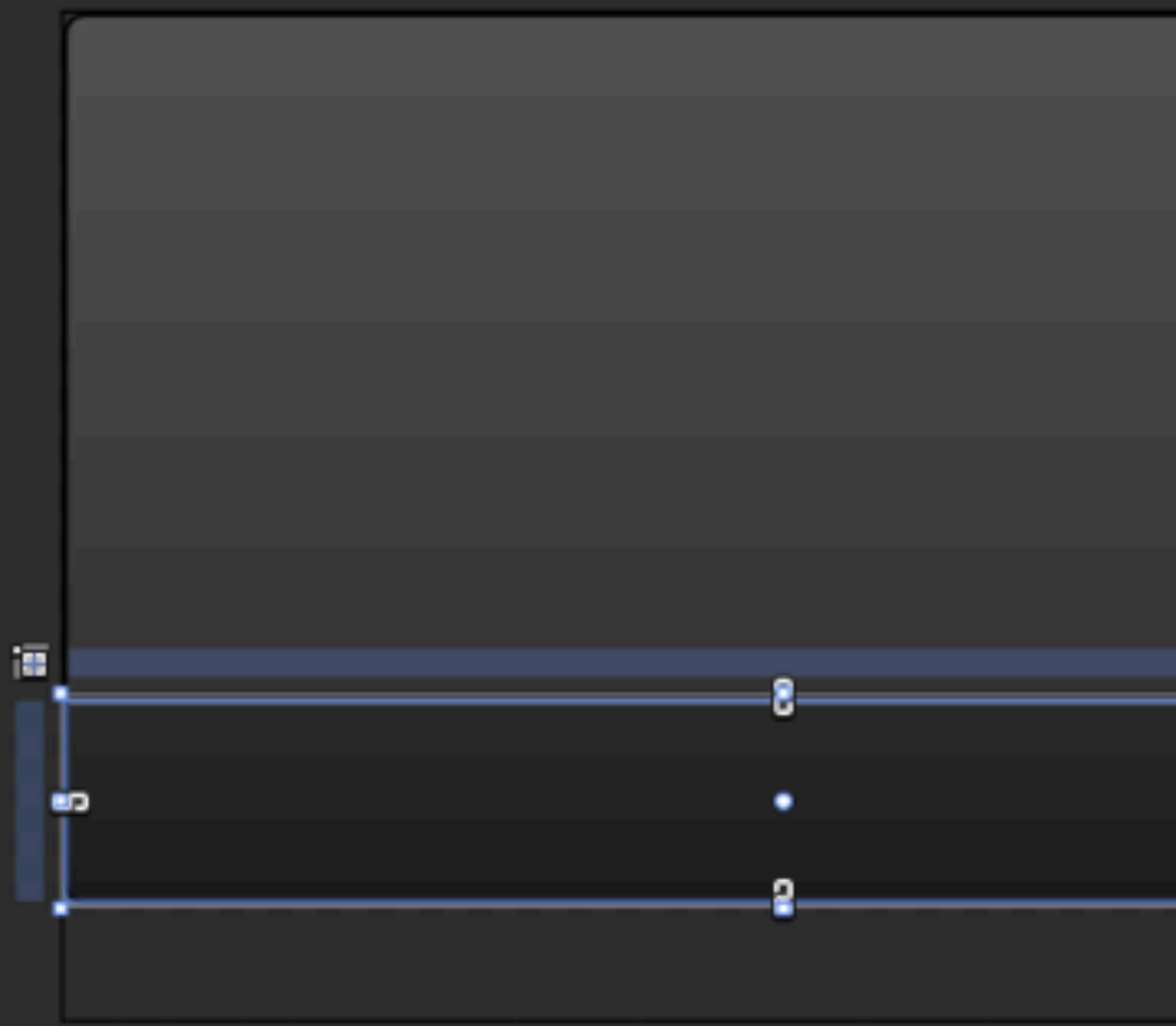
```
<form:FormLayoutContainer
    Style="{StaticResource DarkFormLayoutContainerStyle}"
    Header="FLAT &amp; SIMPLE"
    ItemContainerStyle="{StaticResource VerticalFormLayoutElementStyle}"
    Margin="16,0,16,-4" VerticalAlignment="Top" Grid.Row="1">

    <form:FormLayoutElement Label="FirstName">
        <TextBox Text="Bruce" Style="{StaticResource SmartTextBoxStyle}"/>
    </form:FormLayoutElement>

    <form:FormLayoutElement Label="Address">
        <TextBox Text="666 Trooper Drive, Los Angeles" Style="{StaticResource SmartTextBoxStyle}"/>
    </form:FormLayoutElement>

    <form:FormLayoutElement Label="Interests">
        <CheckBox Content="Music" Style="{StaticResource SmartCheckBoxStyle}" IsChecked="True"/>
    </form:FormLayoutElement>

    <form:FormLayoutElement Label="">
        <CheckBox Content="Airplanes" Style="{StaticResource SmartCheckBoxStyle}" IsChecked="True"/>
    </form:FormLayoutElement>

    <form:FormLayoutElement Label="" >
        <CheckBox Content="Other Stuff" Style="{StaticResource SmartCheckBoxStyle}"/>
    </form:FormLayoutElement>

    <form:FormLayoutContainer.Footer>
        <StackPanel Orientation="Horizontal">
            <Button Content="Cancel"
                Style="{StaticResource SmartButtonStyle}"
                Width="Auto"
                Margin="6,0,0,0" />
            <Button Content="Run Now"
                Style="{StaticResource SmartButtonStyle}" Margin="6,0,0,0"/>
        </StackPanel>
    </form:FormLayoutContainer.Footer>
</form:FormLayoutContainer>
```

Objects and Timeline

(No Storyboard open)

DarkFormLayoutContainerStyle (FormLayoutC...

- Template
  - Root
    - Main
      - [Grid]
        - [TextBlock] "FLAT & SIM
        - [Rectangle]
        - PART_Panel
    - Footer
      - [Grid]
        - [ContentPresenter]
        - [Rectangle]



Objects and Timeline

(No Storyboard open)

VerticalFormLayoutElementStyle (FormLayoutElement...

- Template
  - [Border]
    - [Grid]
      - [Grid]
        - [TextBlock] "FirstName"
        - required
      - [ContentPresenter]

FirstName* FormLayoutEle

300%

Results

Errors

c:\Windows\Microsoft.NET\Framework\v4.0.30319\Csc.exe /noconfig /nowarn:1701,1702 /nostdlib+ /platfor
CrawlTest -> C:\Users\dct\Desktop\ERGOSIGNShowcase\Application\Version0.1\ERGOSIGNShowcase\Crawl
---
Das Projekt "C:\Users\dct\Desktop\ERGOSIGNShowcase\Application\Version0.1\ERGOSIGNShowcase\ERGOS

Custom Layout Container - Überschreiben Methoden ItemsControl

```
/// <summary>
/// Use FormLayoutElement!
/// </summary>
/// <param name="item"></param>
/// <returns></returns>
protected override bool IsItemItsOwnContainerOverride(object item)
{
    return (item is FormLayoutElement);
}

/// <summary>
/// Use FormLayoutElement as container element.
/// </summary>
/// <returns></returns>
protected override DependencyObject GetContainerForItemOverride()
{
    return new FormLayoutElement();
}
```

Eigene Item Wrapper verwenden :-)

Custom Layout Container

Vorteil

- Verhindern, dass Developer UI zerstören ;-)
- XAML Code wird kompakter und einfacher
- Wiederverwendbarkeit und erhöhte Produktivität

Nachteil

- Aufwändigere Implementierung

EFFEKTE

Die Wahrheit über Effekte...

Effekte (Basisklasse Effect)

- Blur und DropShadow
- Performance-hungrig daher sparsam verwenden
- Drittanbieter stellen weitere Effekte bereit

Bitmap-Effekte (Basisklasse BitmapEffect)

- Bevel, Emboss, Outer Glow, ...
- Sollte nicht mehr verwendet werden
- Schlechte Performance

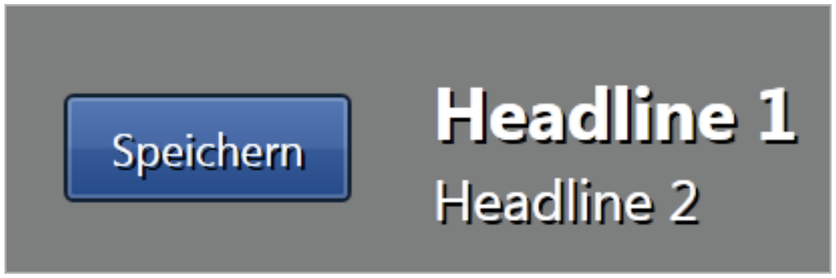
Was tun?

Strategie

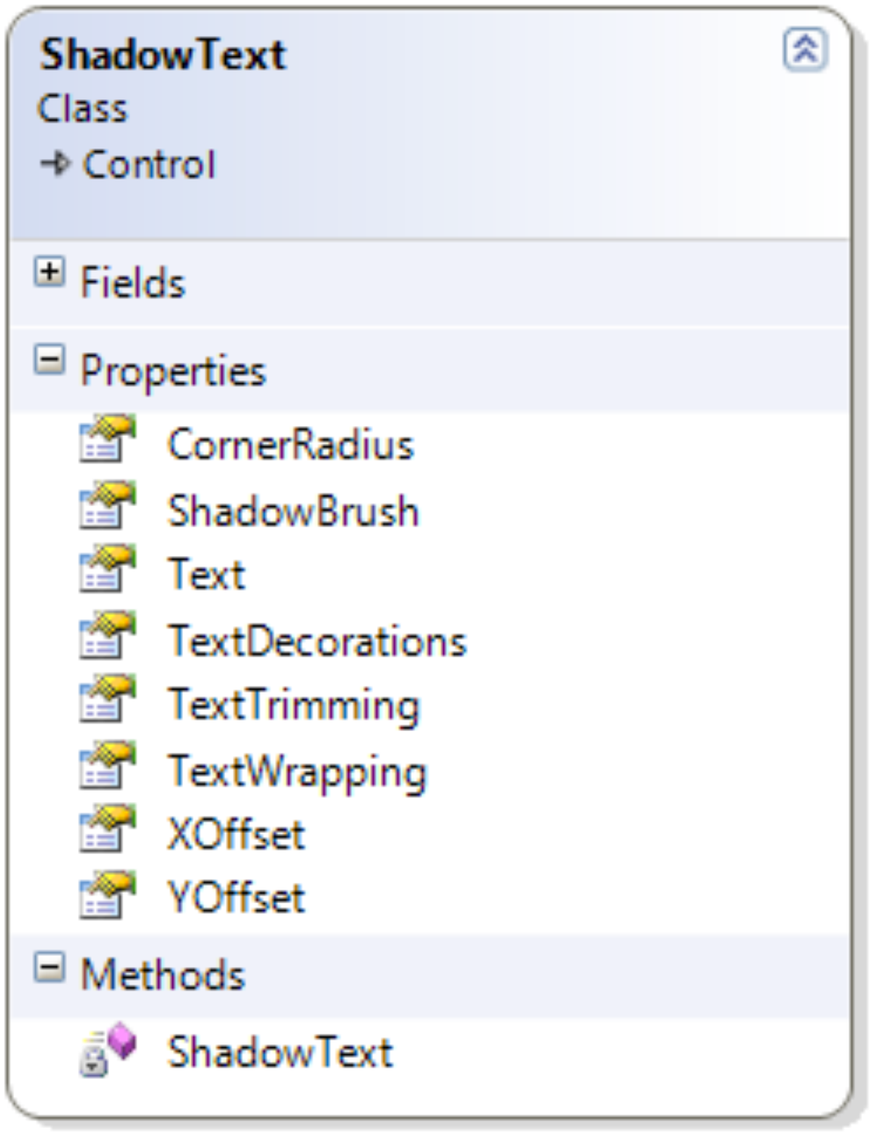
- Effekte mit Custom Controls simulieren
- Rectangle, Border & Co. in Kombination mit Gradient Brush nutzen

EFFEKTE BEISPIEL CUSTOM CONTROL

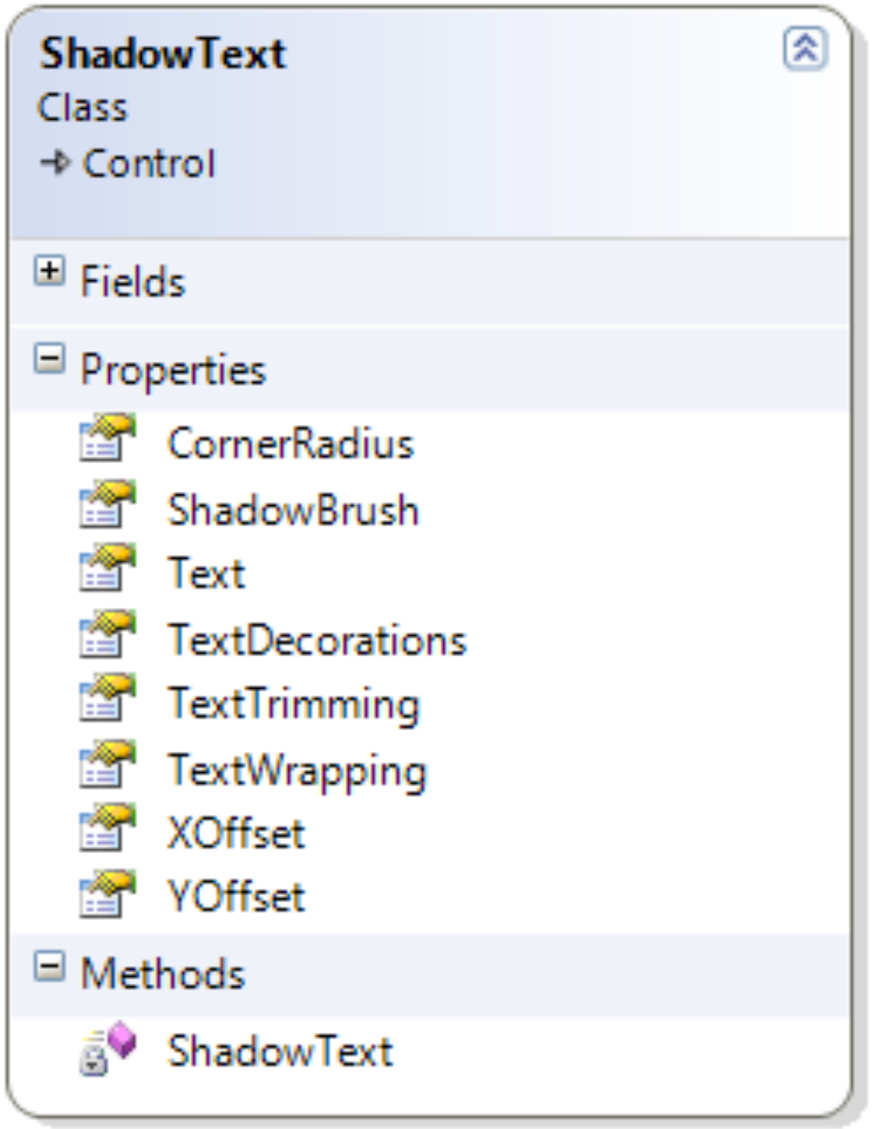
Beispiel Drop Shadow per Custom Control



Beispiel Drop Shadow per Custom Control



Beispiel Drop Shadow per Custom Control



- Leichtgewichtiges Custom Control
- Nur Properties ohne Logik notwendig

Wichtige Properties

- XOffset & YOffset
- ShadowBrush
- Text* (Optionen)

- Idee zwei TextBlöcke in Control Template

ShadowText.cs

ERGOSIGN.XControls.ShadowText.ShadowText | PROPERTY_CATEGORY

```
 * in particular with regard to resale and publication.
 */

namespace ERGOSIGN.XControls.ShadowText
{
    [ContentPropertyAttribute("Text")]
    [TemplatePart(Name = "PART_Translate", Type = typeof(TranslateTransform))]
    [Description("A TextBlock with shadow options.")]
    public class ShadowText : Control
    {
        /// <summary>
        /// Category description.
        /// </summary>
        private const string PROPERTY_CATEGORY = "ERGOSIGN ShadowText Properties";

        /// <summary>
        /// XOffsetProperty - Gets or sets the XOffset.
        /// </summary>
        public static readonly DependencyProperty XOffsetProperty = DependencyProperty.Register("XOffset", typeof(double), typeof(ShadowText)

        /// <summary>
        /// YOffsetProperty - Gets or sets the YOffset.
        /// </summary>
        public static readonly DependencyProperty YOffsetProperty = DependencyProperty.Register("YOffset", typeof(double), typeof(ShadowText)

        /// <summary>
        /// ShadowBrushProperty - Gets or sets the brush of the shadow.
        /// </summary>
        public static readonly DependencyProperty ShadowBrushProperty = DependencyProperty.Register("ShadowBrush", typeof(Brush), typeof(Shad

        /// <summary>
        /// TextProperty - Gets or sets the text.
        /// </summary>
        public static readonly DependencyProperty TextProperty = DependencyProperty.Register("Text", typeof(string), typeof(ShadowText), new

        /// <summary>
        /// CornerRadiusProperty - Gets or sets the CornerRadius.
        /// </summary>
```

100 %

Find Results 1

ShadowText.cs

ERGOSIGN.XControls.ShadowText.ShadowText | PROPERTY_CATEGORY

```
85          /// <summary>
86          /// Gets or sets the XOffset.
87          /// </summary>
88          [Description("Gets or sets the XOffset."), Category(PROPERTY_CATEGORY)]
89          public double XOffset
90          {
91              get { return (double)GetValue(XOffsetProperty); }
92              set { SetValue(XOffsetProperty, value); }
93          }
94
95          /// <summary>
96          /// Gets or sets the YOffset.
97          /// </summary>
98          [Description("Gets or sets the YOffset."), Category(PROPERTY_CATEGORY)]
99          public double YOffset
100         {
101             get { return (double)GetValue(YOffsetProperty); }
102             set { SetValue(YOffsetProperty, value); }
103         }
104
105         /// <summary>
106         /// Gets or sets the brush of the shadow.
107         /// </summary>
108         [Description("Gets or sets the brush of the shadow."), Category(PROPERTY_CATEGORY)]
109         public Brush ShadowBrush
110         {
111             get { return (Brush)GetValue(ShadowBrushProperty); }
112             set { SetValue(ShadowBrushProperty, value); }
113         }
114
115         /// <summary>
116         /// Gets or sets the text.
117         /// </summary>
118         [Description("Gets or sets the text."), Category(PROPERTY_CATEGORY)]
119         public string Text
120         {
121             get { return (string)GetValue(TextProperty); }
122             set { SetValue(TextProperty, value); }
123         }
124
125         /// <summary>
126         /// Gets or sets the CornerRadius.
127         /// </summary>
128         [Description("Gets or sets the CornerRadius."), Category(PROPERTY_CATEGORY)]
```

100 %

```
<Style TargetType="{x:Type xControls:ShadowText}">
    <Setter Property="Background" Value="{x:Null}"/>
    <Setter Property="BorderThickness" Value="0"/>
    <Setter Property="BorderBrush" Value="{x:Null}"/>
    <Setter Property="CornerRadius" Value="0"/>
    <Setter Property="Height" Value="Auto"/>
    <Setter Property="Padding" Value="0"/>
    <Setter Property="HorizontalContentAlignment" Value="Stretch"/>
    <Setter Property="VerticalContentAlignment" Value="Stretch"/>
    <Setter Property="SnapsToDevicePixels" Value="True"/>
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="{x:Type xControls:ShadowText}">
                <Border Background="{TemplateBinding Background}"
                        BorderBrush="{TemplateBinding BorderBrush}"
                        BorderThickness="{TemplateBinding BorderThickness}"
                        Padding="{TemplateBinding Padding}"
                        CornerRadius="{TemplateBinding CornerRadius}">
                    <Grid VerticalAlignment="{TemplateBinding VerticalContentAlignment}"
                          HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}">
                        <TextBlock x:Name="ShadowText"
                                   Text="{TemplateBinding Text}"
                                   RenderTransformOrigin="0.5,0.5"
                                   Foreground="{TemplateBinding ShadowBrush}"
                                   IsHitTestVisible="False"
                                   IsEnabled="False"
                                   TextWrapping="{TemplateBinding TextWrapping}"
                                   TextTrimming="{TemplateBinding TextTrimming}"
                                   TextDecorations="{TemplateBinding TextDecorations}">
                            <TextBlock.RenderTransform>
                                <TranslateTransform x:Name="PART_Translate"
                                                    X="{Binding XOffset, RelativeSource={RelativeSource Mode=FindAncestor, AncestorType={x:Type x
                                                    Y="{Binding YOffset, RelativeSource={RelativeSource Mode=FindAncestor, AncestorType={x:Type x
                            </TextBlock.RenderTransform>
                        </TextBlock>
                        <TextBlock x:Name="NormalText"
                                   Text="{TemplateBinding Text}"
                                   TextWrapping="{TemplateBinding TextWrapping}"
                                   TextTrimming="{TemplateBinding TextTrimming}"
                                   TextDecorations="{TemplateBinding TextDecorations}" />
                    </Grid>
                </Border>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>
```

Grid mit zwei TextBlöcken

Design | ResourceDictionary.ResourceDictionary

Beispiel Drop Shadow per Custom Control

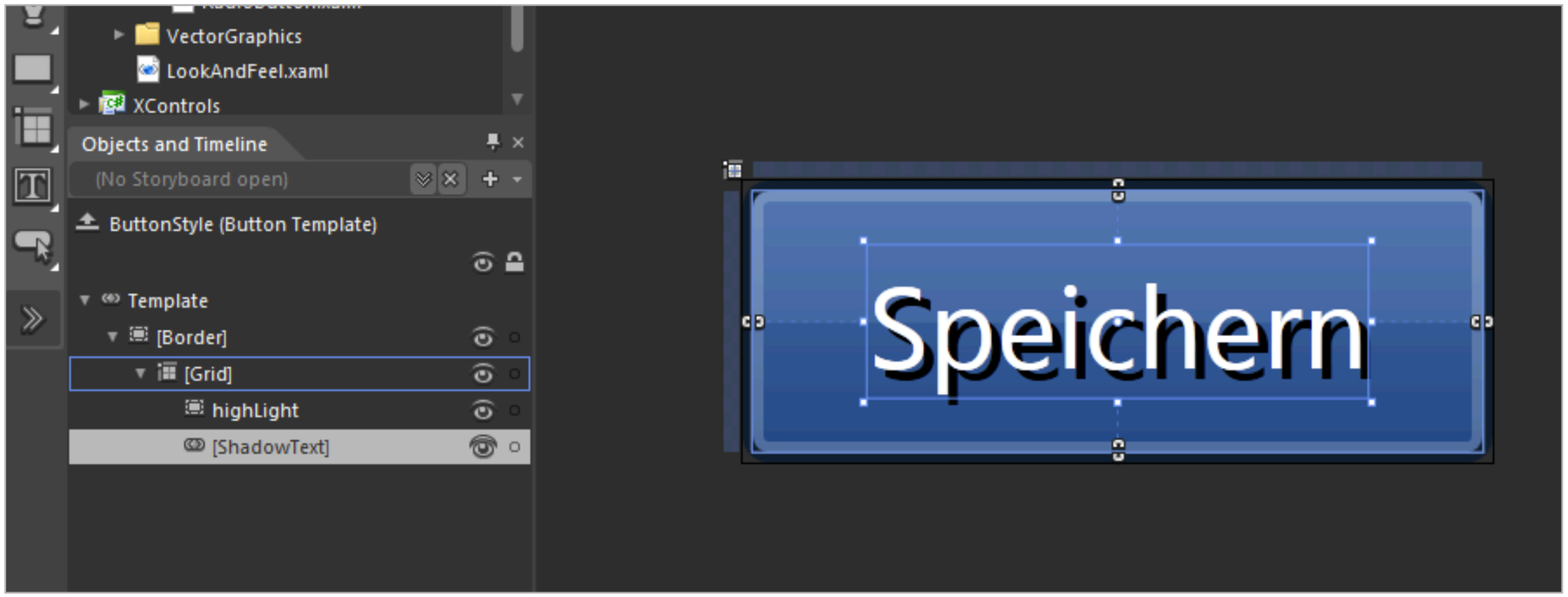
```
  HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}">
<TextBlock x:Name="ShadowText"
           Text="{TemplateBinding Text}"
           RenderTransformOrigin="0.5,0.5"
           Foreground="{TemplateBinding ShadowBrush}"
           IsHitTestVisible="False"
           IsEnabled="False"
           TextWrapping="{TemplateBinding TextWrapping}"
           TextTrimming="{TemplateBinding TextTrimming}"
           TextDecorations="{TemplateBinding TextDecorations}">
    <TextBlock.RenderTransform>
        <TranslateTransform x:Name="PART_Translate"
                            X="{Binding XOffset, RelativeSource={RelativeSource Mode=FindAncestor, AncestorType={x:Type xControls:ShadowText}}}"
                            Y="{Binding YOffset, RelativeSource={RelativeSource Mode=FindAncestor, AncestorType={x:Type xControls:ShadowText}}}" />
    </TextBlock.RenderTransform>
  </TextBlock>
  <TextBlock x:Name="NormalText"
           Text="{TemplateBinding Text}"
           TextWrapping="{TemplateBinding TextWrapping}"
           TextTrimming="{TemplateBinding TextTrimming}"
           TextDecorations="{TemplateBinding TextDecorations}" />
Grid>
ler>
```

- TranslateTransform auf „ShadowText“
- Werte über Properties X & YOffset Property

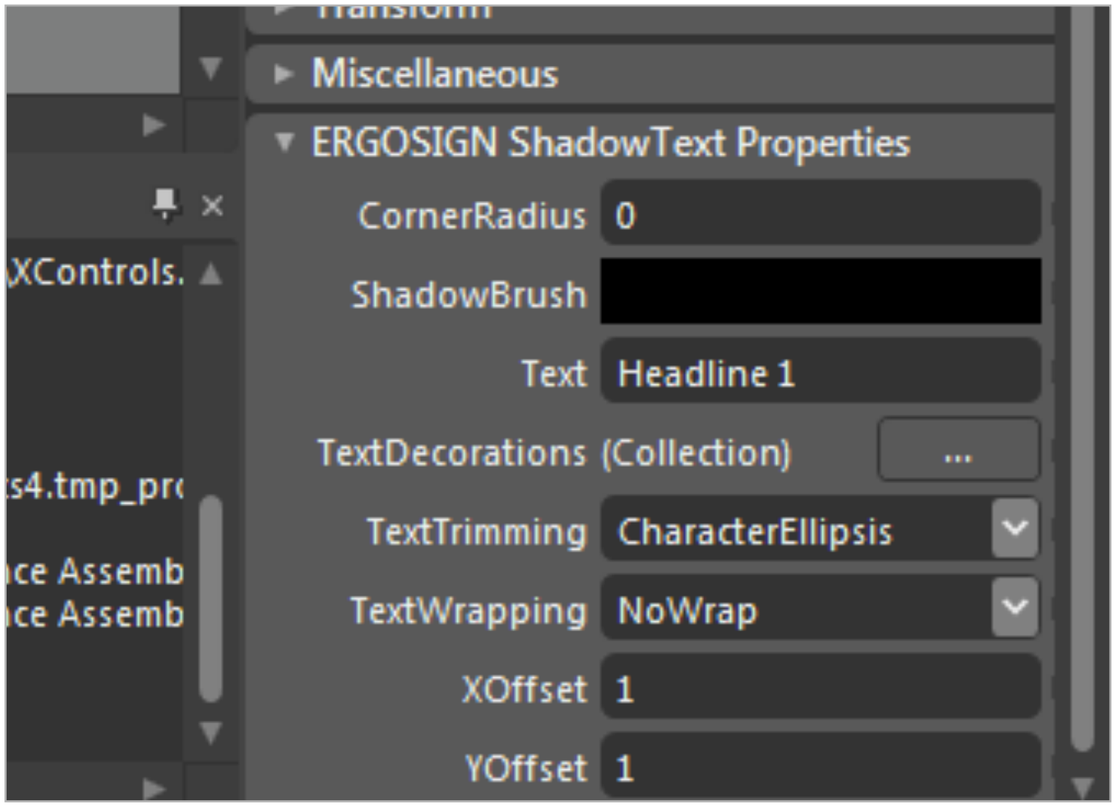
- Per TemplateBinding Controls anbinden z.B. Text

Beispiel Drop Shadow per Custom Control

Beispiel Drop Shadow per Custom Control



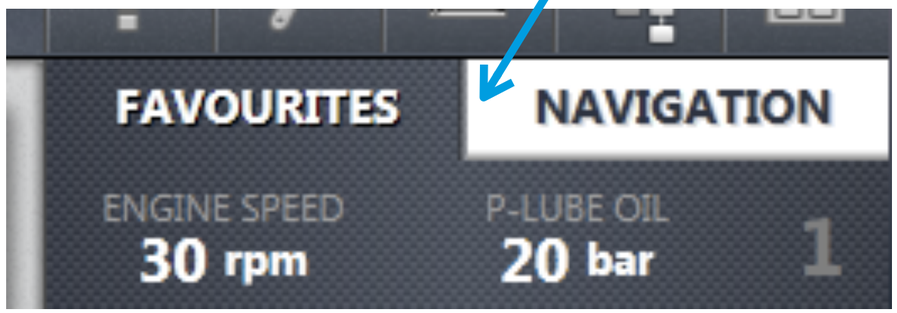
Beispiel Drop Shadow per Custom Control



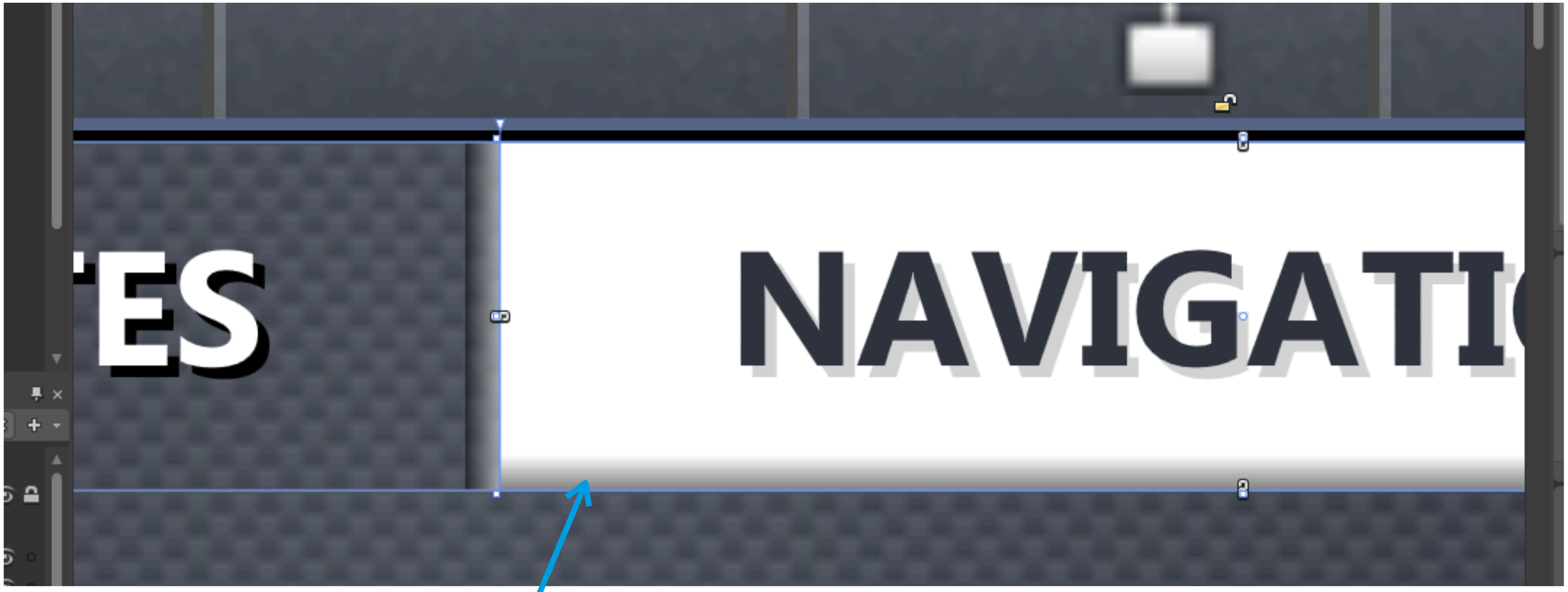
EFFEKTE BEISPIEL RECTANGLE

Beispiel Drop Shadow per Rectangle

Tabs mit Schatten



Beispiel Drop Shadow per Rectangle



Rectangle mit LinearGradientBrush

Beispiel Drop Shadow per Rectangle

Beispiel Drop Shadow per Rectangle

```
<Rectangle x:Name="rectangle"
           Width="Auto"
           HorizontalAlignment="Stretch"
           VerticalAlignment="Bottom"
           Height="3"
           Visibility="Visible">
    <Rectangle.Fill>
        <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
            <GradientStop Color="#7FFFFFFF"/>
            <GradientStop Color="#7F000000" Offset="1"/>
        </LinearGradientBrush>
    </Rectangle.Fill>
</Rectangle>
```

Mit Alpha Wert der Farbe arbeiten

XAML KONVENTIONEN

XAML!

Allgemein

- Jede Stufe der Hierarchie (UI) einrücken
- x:Name oder x:Key Attribute als erstes schreiben
- Attribute nach Zusammengehörigkeit gruppieren
  - Width, Height, VerticalAlignment, HorizontalAlignment..-

ERWEITERTE ATTRIBUT FORMATIERUNG

Erweiterte Attribut Formatierung

- Attribute eines Elementes in jeweils einer separaten Zeile
- Das erste Attribut steht in der gleichen Zeile wie das Element
- Alle nachfolgenden Attribute werden entlang der Vertikalen untereinander ausgerichtet

```
<Style x:Key="PrimaryButtonStyle" TargetType="{x:Type Button}" BasedOn="{StaticResource FrameworkElementStyle}">
    <Setter Property="Background" Value="{StaticResource ButtonBackgroundBrush}"/>
    <Setter Property="BorderBrush" Value="{StaticResource ButtonBorderBrush}"/>
    <Setter Property="Foreground" Value="{StaticResource FontBrush}"/>
    <Setter Property="HorizontalContentAlignment" Value="Center"/>
    <Setter Property="VerticalContentAlignment" Value="Center"/>
    <Setter Property="Padding" Value="11,0,11,0"/>
    <Setter Property="Height" Value="27"/>
    <Setter Property="FontSize" Value="{StaticResource Layer2FontSize}"/>
    <Setter Property="FontFamily" Value="{StaticResource PrimaryFontFamily}"/>
    <Setter Property="BorderThickness" Value="1"/>
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="{x:Type Button}">
                <Border Background="{TemplateBinding Background}"
                        BorderThickness="{TemplateBinding BorderThickness}"
                        BorderBrush="{TemplateBinding BorderBrush}"
                        Padding="0"
                        CornerRadius="2">
                    <Grid>
                        <Border x:Name="highLight"
                                BorderThickness="{TemplateBinding BorderThickness}"
                                BorderBrush="{StaticResource White20Brush}"
                                CornerRadius="1" />
                        <ContentPresenter Margin="{TemplateBinding Padding}"
                                          HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}"
                                          VerticalAlignment="{TemplateBinding VerticalContentAlignment}"
                                          RecognizesAccessKey="True" />
                    </Grid>
                </Border>
                <ControlTemplate.Triggers>
                    <Trigger Property="IsPressed" Value="true">
                        <Setter Property="BorderThickness" Value="0,0,0,1" TargetName="highLight"/>
                    </Trigger>
                </ControlTemplate.Triggers>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
    <Style.Triggers>
        <Trigger Property="IsPressed" Value="true">
            <Setter Property="Background" Value="{StaticResource ButtonPressedBackgroundBrush}"/>
        </Trigger>
    </Style.Triggers>
</Style>
```

Überblick

Leichtes Scannen für Auge

```
<Style x:Key="PrimaryButtonStyle" TargetType="{x:Type Button}" BasedOn="{StaticResource FrameworkElementSty
    <Style.Triggers>
        <Trigger Property="IsPressed" Value="true">
            <Setter Property="Background" Value="{StaticResource ButtonPressedBackgroundBrush}"/>
        </Trigger>
    </Style.Triggers>
    <Setter Property="Background" Value="{StaticResource ButtonBackgroundBrush}"/>
    <Setter Property="BorderBrush" Value="{StaticResource ButtonBorderBrush}"/>
    <Setter Property="Foreground" Value="{StaticResource FontBrush}"/>
    <Setter Property="HorizontalContentAlignment" Value="Center"/>
    <Setter Property="VerticalContentAlignment" Value="Center"/>
    <Setter Property="Padding" Value="11,0,11,0"/>
    <Setter Property="Height" Value="27"/>
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="{x:Type Button}">
                <Border Background="{TemplateBinding Background}" BorderThickness="{TemplateBinding BorderT
                    <Grid>
                        <Border x:Name="highLight" BorderThickness="{TemplateBinding BorderThickness}" Bord
                        <ContentPresenter Margin="{TemplateBinding Padding}" HorizontalAlignment="{Template
                    </Grid>
                </Border>
                <ControlTemplate.Triggers>
                    <Trigger Property="IsPressed" Value="true">
                        <Setter Property="BorderThickness" Value="0,0,0,1" TargetName="highLight"/>
                    </Trigger>
                </ControlTemplate.Triggers>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
    <Setter Property="FontSize" Value="{StaticResource Layer2FontSize}"/>
    <Setter Property="FontFamily" Value="{StaticResource PrimaryFontFamily}"/>
    <Setter Property="BorderThickness" Value="1"/>
</Style>
```

Blend…

Formatierung?

Reihenfolge?

Erweiterte Attribut Formatierung

Anwenden bei

- Elemente bzw. Controls die als Content einer Template definiert sind (Control Template, Data Template, ...)
- Elemente bzw. Controls die als Content eines Controls mit Code-Behind definiert sind (UserControl, Window, Page...)
- Root-Element eines Dokuments und dessen Attribute

Erweiterte Attribut Formatierung

Ausnahmen

- Setter
- Trigger
- Colors
- Brushes...

- Nicht für Konstrukte mit wiederholenden Zeichenketten geeignet
- Nicht für Konstrukte mit wiederholenden Zeichenketten geeignet

```
<Style x:Key="PrimaryButtonStyle"
       TargetType="{x:Type Button}"
       BasedOn="{StaticResource FrameworkElementStyle}">
    <Setter Property="Background"
            Value="{StaticResource ButtonBackgroundBrush}" />
    <Setter Property="BorderBrush"
            Value="{StaticResource ButtonBorderBrush}" />
    <Setter Property="Foreground"
            Value="{StaticResource FontBrush}" />
    <Setter Property="HorizontalContentAlignment"
            Value="Center" />
    <Setter Property="VerticalContentAlignment"
            Value="Center" />
    <Setter Property="Padding"
            Value="11,0,11,0" />
    <Setter Property="Height"
            Value="27" />
    <Setter Property="FontSize"
            Value="{StaticResource Layer2FontSize}" />
    <Setter Property="FontFamily"
            Value="{StaticResource PrimaryFontFamily}" />
    <Setter Property="BorderThickness"
            Value="1" />
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="{x:Type Button}">
                <Border Background="{TemplateBinding Background}"
                        BorderThickness="{TemplateBinding BorderThickness}"
                        BorderBrush="{TemplateBinding BorderBrush}"
                        Padding="0"
                        CornerRadius="2">
                    <Grid>
                        <Border x:Name="highLight"
                                BorderThickness="{TemplateBinding BorderThickness}"
                                BorderBrush="{StaticResource White20Brush}"
                                CornerRadius="1" />
                        <ContentPresenter Margin="{TemplateBinding Padding}"
                                          HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}"
                                          VerticalAlignment="{TemplateBinding VerticalContentAlignment}"
                                          RecognizesAccessKey="True" />
```

Failed O_o

Auge bleibt bei jeder Zeile an dem entsprechenden Value Attribut "hängen"

WIE WERDE ICH EIN GUTER WPF UI DEVELOPER? ;-)

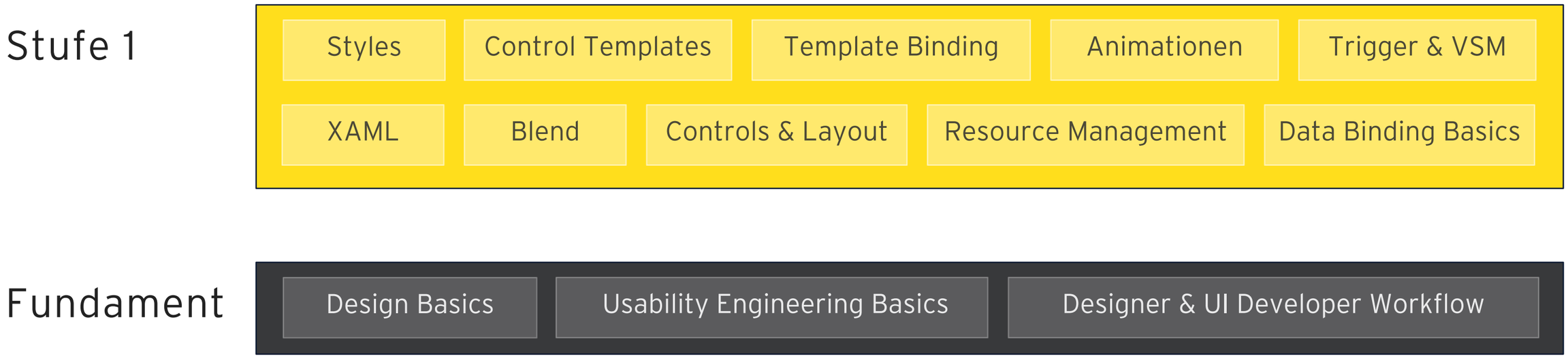
Stufenmodell

Stufenmodell

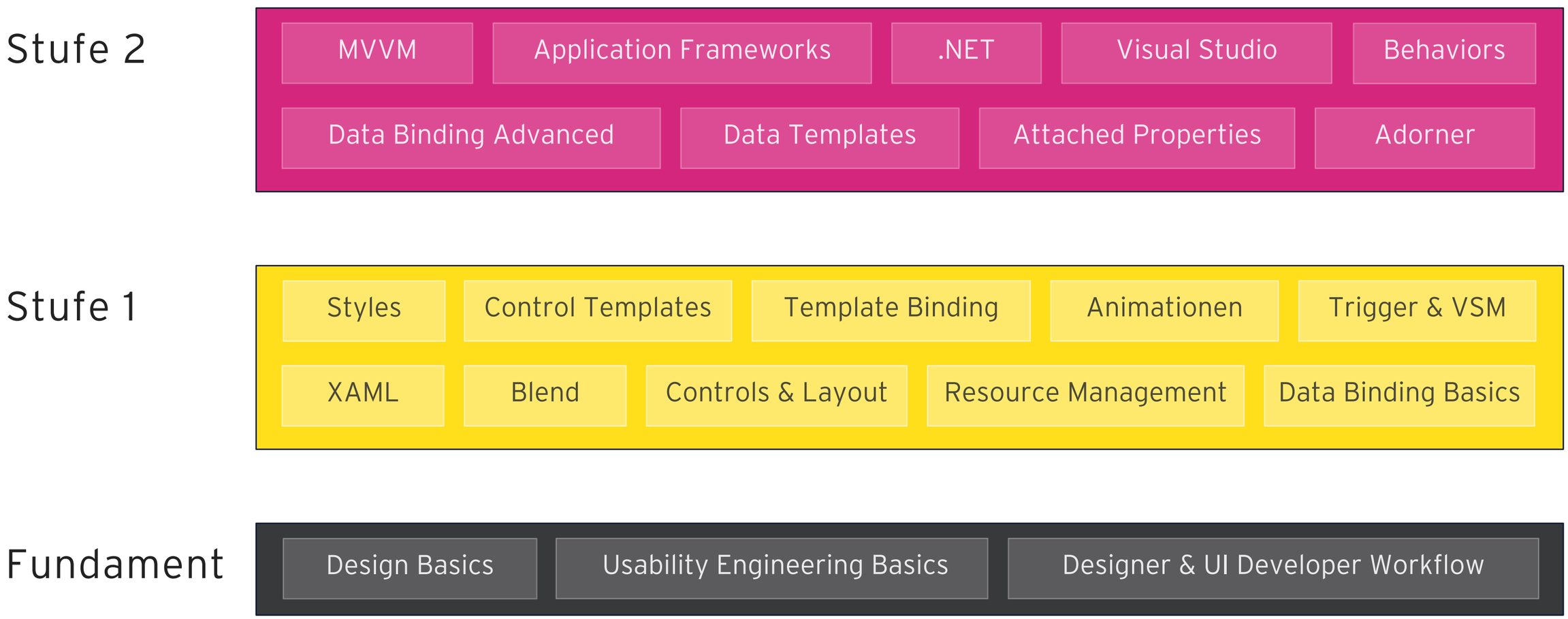
Fundament

Design Basics	Usability Engineering Basics	Designer & UI Developer Workflow

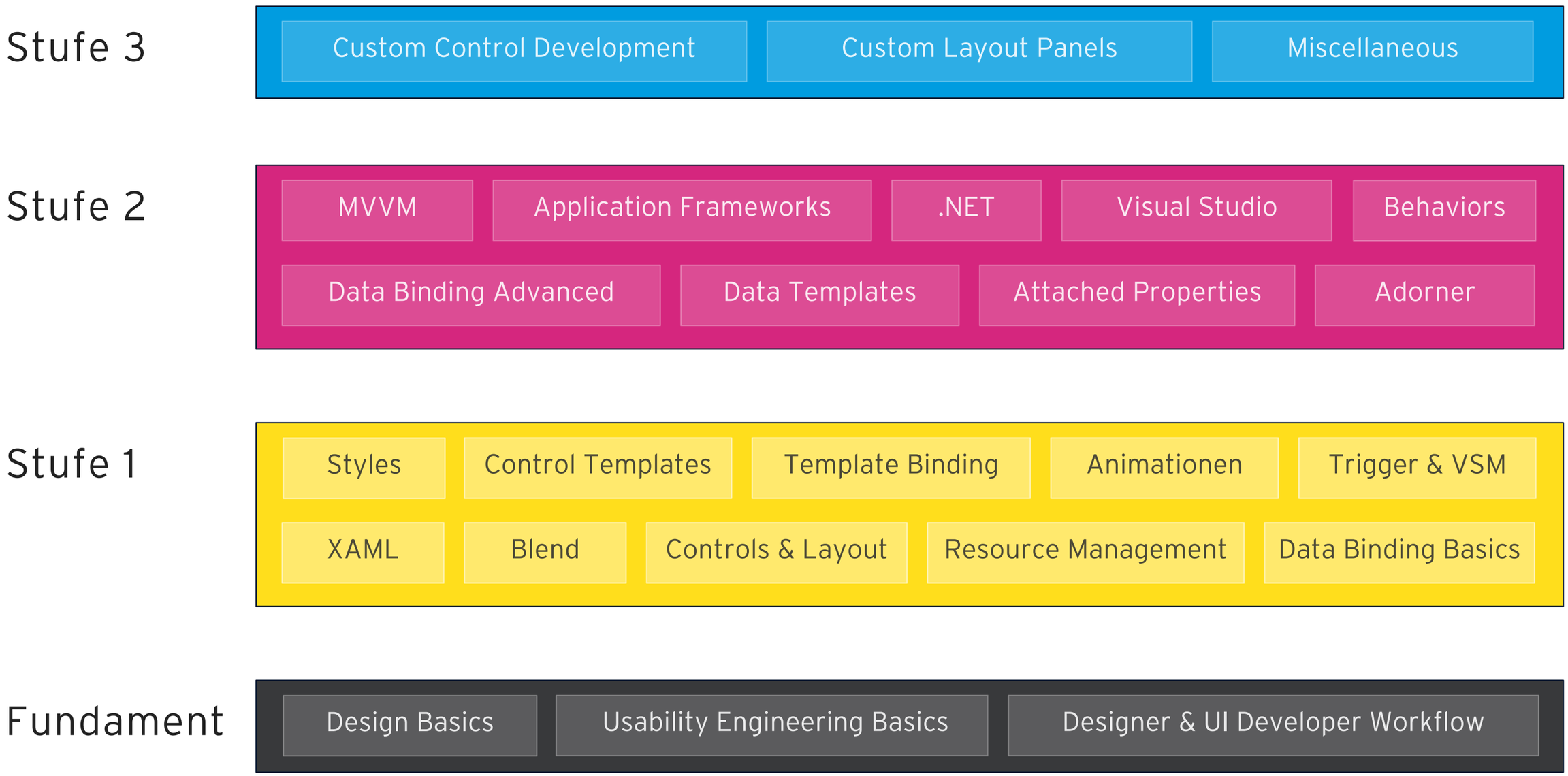
Stufenmodell



Stufenmodell



Stufenmodell



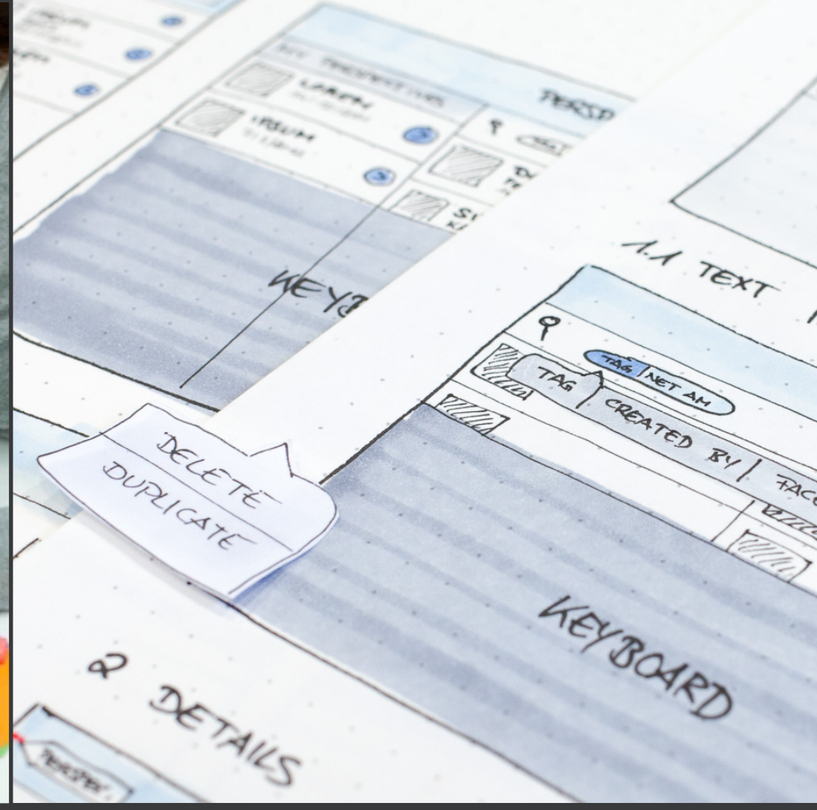
FAZIT

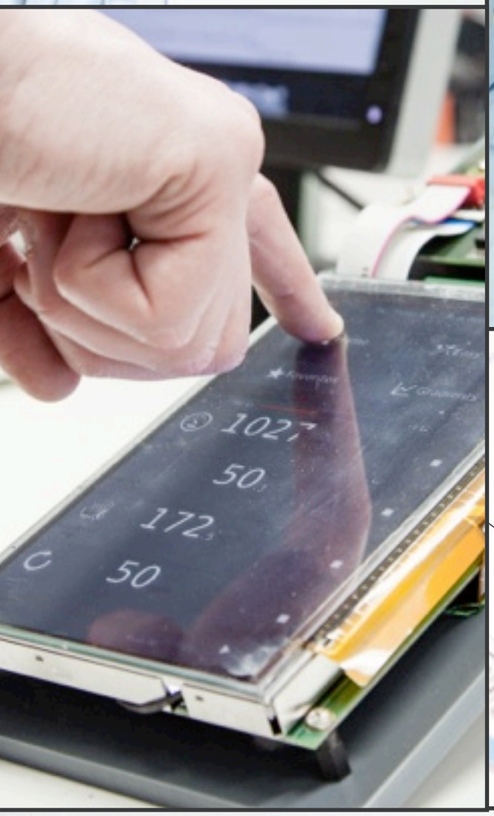
FAZIT

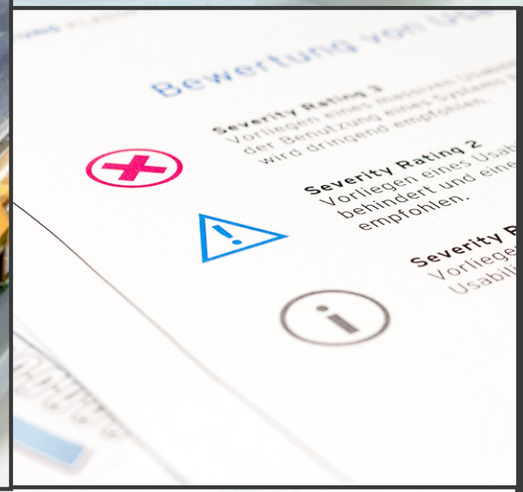
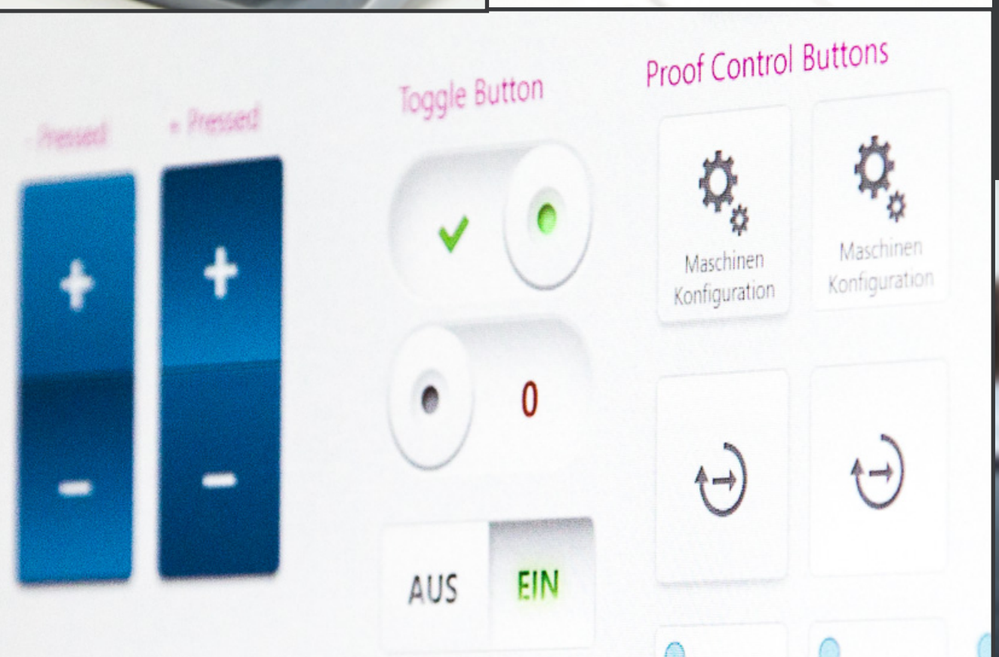
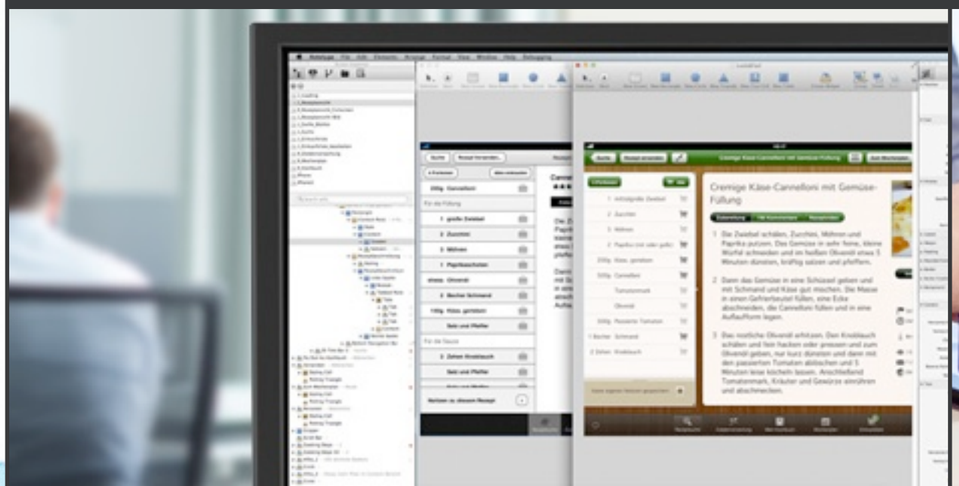
- Denken Sie an einen benutzer-zentrierten Designprozess
- Schätzen Sie gutes UI Development
- Reizen Sie die Möglichkeiten von WPF aus! (Keine halben Sachen)
- Legen Sie Wert auf Struktur und Konsistenz während der Entwicklung

FRAGEN? :-)

VIELEN DANK!







Ein benutzerzentrierter Prozess ist Voraussetzung für das Design einer positiven **User Experience**.