978-3-86645-520-7 Silverlight 3 Crashkurs - Listings

# Kapitel 1

## Keine Listings

# Kapitel 2

## Listing 2.1 Die Klasse *Person*, für die ein Eingabeformular erstellt werden soll

public class Person

{

public string Nachname { get; set; }

public string Vorname { get; set; }

public DateTime Geboren { get; set; }

public bool IstFrau { get; set; }

}

## Listing 2.2 XAML-Code des Personenformulars

<UserControl

xmlns:controls="clr-namespace:System.Windows.Controls;assembly=System.Windows.Controls"

x:Class="SecureServices.PersonenControl"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Width="400" Height="300">

<Grid x:Name="LayoutRoot" Background="White" Margin="20">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition Width="Auto"/>

</Grid.ColumnDefinitions>

<Border Grid.Row="0" Grid.ColumnSpan="2"

Margin="5"

BorderBrush="Red" BorderThickness="1">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition Height="Auto" />

<RowDefinition Height="Auto" />

<RowDefinition Height="Auto" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition Width="Auto" />

</Grid.ColumnDefinitions>

<TextBlock Grid.Column="0" Grid.Row="0" Text="Nachname" />

<TextBlock Grid.Column="0" Grid.Row="1" Text="Vorname" />

<TextBlock Grid.Column="0" Grid.Row="2" Text="Geboren" />

<TextBox Grid.Column="1" Grid.Row="0" Width="180" Margin="2"

Text="{Binding Path=Nachname, Mode=TwoWay, NotifyOnValidationError=True,

ValidatesOnExceptions=True}"

HorizontalAlignment="Left" VerticalAlignment="Top" />

<TextBox Grid.Column="1" Grid.Row="1" Width="180" Margin="2"

Text="{Binding Path=Vorname,

Mode=TwoWay, NotifyOnValidationError=True, ValidatesOnExceptions=True}"

HorizontalAlignment="Left" VerticalAlignment="Top" />

<controls:DatePicker Grid.Column="1" Grid.Row="2"

Width="180" Margin="5"

SelectedDate="{Binding Path=Geboren, Mode=TwoWay}"

SelectedDateFormat="Short" />

<CheckBox Grid.Column="1" Grid.Row="3" Content="Frau"

IsChecked="{Binding Path=IstFrau, Mode=TwoWay, NotifyOnValidationError=True,

ValidatesOnExceptions=True}"

Margin="5"/>

</Grid>

</Border>

<Button Grid.Column="0" Grid.Row="1"

Width="100" Height="26"

Content="Abbruch" Click="OnCancel" /> <Button Grid.Column="1" Grid.Row="1" Width="100" Height="26" Content="Ok" Click="OnOk" /> </Grid> </UserControl>

## Listing 2.3 Automatisch generierte Codedatei für die Klasse

1 public partial class PersonenControl : System.Windows.Controls.UserControl {

2

3 internal System.Windows.Controls.Grid LayoutRoot;

4

5 private bool \_contentLoaded;

6

7 /// <summary>

8 /// InitializeComponent

9 /// </summary>

10 [System.Diagnostics.DebuggerNonUserCodeAttribute()]

11 public void InitializeComponent() {

12 if (\_contentLoaded) {

13 return;

14 }

15 \_contentLoaded = true;

16 System.Windows.Application.LoadComponent(this,

new System.Uri("/SilverlightApplication5;component/MainPage.xaml",

System.UriKind.Relative));

17 this.LayoutRoot = ((System.Windows.Controls.Grid)(this.FindName("LayoutRoot")));

18 }

19 }

## Listing 2.4 Personenformular mit C#-Code erstellt

1 Grid layoutRoot = new Grid();

2 layoutRoot.Name = "LayoutRoot";

3 layoutRoot.Background = new SolidColorBrush(Colors.White);

4 layoutRoot.Margin = new Thickness(20);

5 for (int i = 0; i < 2; i++)

6 {

7 RowDefinition rd = new RowDefinition();

8 rd.Height = new GridLength(0, GridUnitType.Auto);

9 layoutRoot.RowDefinitions.Add(rd);

10 }

11 for (int i = 0; i < 2; i++)

12 {

13 ColumnDefinition cd = new ColumnDefinition();

14 cd.Width = new GridLength(0, GridUnitType.Auto);

15 layoutRoot.ColumnDefinitions.Add(cd);

16 }

17 Border border = new Border();

18 border.BorderBrush = new SolidColorBrush(Colors.Red);

19 border.BorderThickness = new Thickness(1.0);

20 border.SetValue(Grid.ColumnSpanProperty, 2);

21 layoutRoot.Children.Add(border);

22 Grid grid = new Grid();

23 border.Child = grid;

24 for (int i = 0; i < 4; i++)

25 {

26 RowDefinition rd = new RowDefinition();

27 rd.Height = new GridLength(0, GridUnitType.Auto);

28 grid.RowDefinitions.Add(rd);

29 }

30 for (int i = 0; i < 2; i++)

31 {

32 ColumnDefinition cd = new ColumnDefinition();

33 cd.Width = new GridLength(0, GridUnitType.Auto);

34 grid.ColumnDefinitions.Add(cd);

35 }

36

37 TextBlock tb = new TextBlock();

38 tb.Text = "Nachname";

39 tb.SetValue (Grid.ColumnProperty,0);

40 tb.SetValue (Grid.RowProperty,0);

41 grid.Children.Add(tb);

42

43 tb = new TextBlock();

44 tb.Text = "Vorname";

45 tb.SetValue(Grid.ColumnProperty, 0);

46 tb.SetValue(Grid.RowProperty, 1);

47 grid.Children.Add(tb);

48

49 tb = new TextBlock();

50 tb.Text = "Geboren";

51 tb.SetValue(Grid.ColumnProperty, 0);

52 tb.SetValue(Grid.RowProperty, 2);

53 grid.Children.Add(tb);

54

55 TextBox tbX = new TextBox();

56 tbX.SetValue(Grid.ColumnProperty, 1);

57 tbX.SetValue(Grid.RowProperty, 0);

58 Binding binding = new Binding("Nachname");

59 binding.Mode = BindingMode.TwoWay;

60 tbX.SetBinding(TextBox.TextProperty, binding);

61 tbX.Margin = new Thickness(2);

62 tbX.Width = 180;

63 grid.Children.Add(tbX);

64

65 tbX = new TextBox();

66 tbX.SetValue(Grid.ColumnProperty, 1);

67 tbX.SetValue(Grid.RowProperty, 1);

68 binding = new Binding("Vorname");

69 binding.Mode = BindingMode.TwoWay;

70 tbX.SetBinding(TextBox.TextProperty, binding);

71 tbX.Margin = new Thickness(2);

72 tbX.Width = 180;

73 grid.Children.Add(tbX);

74

75 DatePicker datePicker = new DatePicker();

76 datePicker.SetValue(Grid.ColumnProperty, 1);

77 datePicker.SetValue(Grid.RowProperty, 2);

78 binding = new Binding("Geboren");

79 binding.Mode = BindingMode.TwoWay;

80 datePicker.SetBinding(DatePicker.SelectedDateProperty, binding);

81 tbX.Margin = new Thickness(2);

82 tbX.Width = 180;

83 grid.Children.Add(datePicker);

84

85 CheckBox checkBox = new CheckBox();

86 checkBox.SetValue(Grid.ColumnProperty, 1);

87 checkBox.SetValue(Grid.RowProperty, 3);

88 checkBox.Content = "Frau";

89 binding = new Binding("IstFrau");

90 binding.Mode = BindingMode.TwoWay;

91 checkBox.SetBinding(CheckBox.IsCheckedProperty, binding);

92 checkBox.Margin = new Thickness(2);

93 checkBox.Width = 180;

94 grid.Children.Add(checkBox);

95

96

97 Button btn = new Button();

98 btn.SetValue(Grid.ColumnProperty, 0);

99 btn.SetValue(Grid.RowProperty, 1);

100 btn.Width = 100;

101 btn.Height = 26;

102 btn.Content = "Abbruch";

103 btn.Margin = new Thickness(2);

104 btn.Click += new RoutedEventHandler(OnChancel);

105 layoutRoot.Children.Add(btn);

106

107 btn = new Button();

108 btn.SetValue(Grid.ColumnProperty, 1);

109 btn.SetValue(Grid.RowProperty, 1);

110 btn.Width = 100;

111 btn.Height = 26;

112 btn.Content = "Ok";

113 btn.Margin = new Thickness(2);

114 btn.Click += new RoutedEventHandler(OnOk);

115 layoutRoot.Children.Add(btn);

## Listing 2.5 Button mit Inhalt

<Button Grid.Column="1" Grid.Row="1"

Width="100" Height="26"

Click="OnOk" >

<StackPanel Orientation="Horizontal">

<TextBlock Text="Ok" />

<Ellipse Width="20" Height="20" Fill="Red" />

</StackPanel>

</Button>

## Listing 2.6 Grundlegende Property-Element-Syntax

<ObjectName>

<ObjectName.PropertyName>

<PropertyValue>...</PropertyValue>

</ObjectName.PropertyName>

</ObjectName>

## Listing 2.7 XAML für das Eventbubbling-Testprogramm

<UserControl x:Class="SecureServices.EmptyControl"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Width="400" Height="120">

<Grid x:Name="lr" Background="White"

MouseLeftButtonDown="OnGridMouseLeftButtonDown">

<Canvas Margin="10" Background="AliceBlue"

MouseLeftButtonDown="OnCanvasMouseLeftButtonDown">

<Rectangle MouseLeftButtonDown="OnRectangleMouseLeftButtonDown"

Margin="5"

Width="300" Height="100">

<Rectangle.Fill>

<LinearGradientBrush EndPoint="1,1" StartPoint="0,0">

<GradientStop Color="Blue" Offset="0"/>

<GradientStop Color="Red" Offset="1"/>

</LinearGradientBrush>

</Rectangle.Fill>

</Rectangle>

<TextBlock Foreground="Red" Height="40"

MouseLeftButtonDown="OnTextBlockMouseLeftButtonDown"

Text="Hallo Welt" />

<Button Height="20" Width="30"

MouseLeftButtonDown="OnButtonMouseLeftButtonDown"/>

</Canvas>

</Grid>

</UserControl>

## Listing 2.8 *Debug.WriteLine* zur Ausgabe von Informationen zur Laufzeit

private void OnCanvasMouseLeftButtonDown(object sender,

MouseButtonEventArgs e)

{

Debug.WriteLine("Canvas");

}

# Kapitel 3

## Listing 3.1 Verwendung von *FrameworkElement.FindName*

private void OnSucheControlClick(object sender,

RoutedEventArgs e)

{

var o = this.FindName(txtBxNamen.Text);

if (o != null)

{

MessageBox.Show("Element "

+ txtBxNamen.Text

+ " gefunden");

}

else

{

MessageBox.Show("Element "

+ txtBxNamen.Text

+ " nicht gefunden");

}

}

## Listing 3.2 Verschiedene Margin-Einstellungen

<Grid x:Name="LayoutRoot" ShowGridLines="True">

<Grid.ColumnDefinitions >

<ColumnDefinition />

<ColumnDefinition />

<ColumnDefinition />

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Button Grid.Column="0" Content="Ohne Margin" />

<Button Grid.Column="1" Content="Margin '10'" Margin="10" />

<Button Grid.Column="2" Content="Margin '2 10 5 20'" Margin="2 10 5 30"/>

<Button Grid.Column="3" Content="Margin '10 30'" Margin="10 30" />

</Grid>

## Listing 3.3 *Padding* im Einsatz

<Grid x:Name="LayoutRoot" >

<Border

BorderBrush="Black" BorderThickness="1">

<Border

BorderBrush="Red" BorderThickness="1"

Background="AliceBlue"

Padding="1 20 5 30"

Margin="10 30" >

<Border BorderBrush="Blue" BorderThickness="1"

Background="White">

<TextBlock >

<Run>Margin="10 30" </Run>

<LineBreak />

<Run> Padding="1 20 5 30"</Run>

</TextBlock>

</Border>

</Border>

</Border>

</Grid>

## Listing 3.4 *Canvas* mit Ellipsen

<Canvas Background="AliceBlue" Margin="10">

<Ellipse Width="10" Height="10" Fill="Red"

Canvas.Top="10" Canvas.Left="10"/>

<Ellipse Width="10" Height="10" Fill="Red"

Canvas.Top="150" Canvas.Left="150"/>

<Ellipse Width="10" Height="10" Fill="Red"

Canvas.Top="250" Canvas.Left="250"/>

</Canvas>

## Listing 3.5 *StackPanel*, Orientierung vertical

<Grid x:Name="LayoutRoot" Margin="10">

<StackPanel>

<TextBlock Text="Hallo Welt" />

<Button Content="Klick mich" />

<TextBlock Text="Tu's doch" />

<Button Content="Klick mich auch" />

</StackPanel>

</Grid>

## Listing 3.6 *StackPanel*, Orientierung horizontal

<Grid x:Name="LayoutRoot" Margin="10">

<StackPanel Orientation="Horizontal">

<TextBlock Text="Hallo Welt" />

<Button Content="Klick mich" />

<TextBlock Text="Tu's doch" />

<Button Content="Klick mich auch" />

</StackPanel>

</Grid>

## Listing 3.7 *DockPanel* mit der Einstellung *LastChildFill="true"*

<UserControl

xmlns:cKit="clr-namespace:System.Windows.Controls;assembly=System.Windows.Controls.Toolkit"

x:Class="Button\_Layout.MainPage"

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" d:DesignWidth="640" d:DesignHeight="480">

<Grid x:Name="LayoutRoot">

<cKit:DockPanel>

<Button cKit:DockPanel.Dock="Top" Content="Oben" />

<Button cKit:DockPanel.Dock="Bottom" Content="Unten" />

<Button cKit:DockPanel.Dock="Left" Content="Links" />

<Button Content="Ohne" />

</cKit:DockPanel>

</Grid>

</UserControl>

## Listing 3.8 *DockPanel*, letztes Element hat eine feste Breite

<Grid x:Name="LayoutRoot" Margin="10">

<cKit:DockPanel >

<Button cKit:DockPanel.Dock="Top" Content="Oben" />

<Button cKit:DockPanel.Dock="Bottom" Content="Unten" />

<Button cKit:DockPanel.Dock="Left" Content="Links" />

<Button Content="Links" cKit:DockPanel.Dock="Left"

Width="100" Height="100" />

</cKit:DockPanel>

</Grid>

## Listing 3.9 DockPanel mit HorizontalAlignment="Left"

<Grid x:Name="LayoutRoot" Margin="10">

<cKit:DockPanel HorizontalAlignment="Left">

<Button cKit:DockPanel.Dock="Top" Content="Oben" />

<Button cKit:DockPanel.Dock="Bottom" Content="Unten" />

<Button cKit:DockPanel.Dock="Left" Content="Links" />

<Button Content="Links" cKit:DockPanel.Dock="Left"

Width="100" Height="100" />

</cKit:DockPanel>

</Grid>

## Listing 3.10 Ein einfaches *Grid* mit zwei Spalten und drei Zeilen

<Grid x:Name="LayoutRoot" Margin="10" ShowGridLines="True">

<Grid.RowDefinitions>

<RowDefinition />

<RowDefinition />

<RowDefinition />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition />

<ColumnDefinition />

</Grid.ColumnDefinitions>

<TextBlock Grid.Column="0" Grid.Row="0" Text="Name" />

<TextBlock Grid.Column="0" Grid.Row="1" Text="Vorname" />

<TextBlock Grid.Column="0" Grid.Row="2" Text="Wohnort" />

<TextBlock Grid.Column="1" Grid.Row="0" Text="Donald" />

<TextBlock Grid.Column="1" Grid.Row="1" Text="Duck" />

<TextBlock Grid.Column="1" Grid.Row="2" Text="Entenhausen" />

</Grid>

## Listing 3.11 *Grid* mit zwei Zeilen. Das Verhältnis der Zeilenhöhe beträgt 1:2.

<RowDefinition Height="\*"/>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="2\*"/>

</Grid.RowDefinitions>

<Border Grid.Row="0" Background="AliceBlue"

BorderBrush="Black" BorderThickness="1">

<Border HorizontalAlignment="Center" VerticalAlignment="Center">

<TextBlock Grid.Row="1" Grid.Column="0">

RowDefinition Height="\*"

</TextBlock>

</Border>

</Border>

<Border Grid.Row="2" Background="Red"

BorderBrush="Black" BorderThickness="1">

<Border HorizontalAlignment="Center" VerticalAlignment="Center">

<TextBlock Grid.Row="1" Grid.Column="0">

RowDefinition Height="2\*"

</TextBlock>

</Border>

</Border>

</Grid>

## Listing 3.12 *Grid* mit gemischten Größenangaben

<Grid x:Name="LayoutRoot" Margin="10" ShowGridLines="True">

<Grid.RowDefinitions>

<RowDefinition Height="35" />

<RowDefinition Height="Auto" />

<RowDefinition Height="\*"/>

<RowDefinition Height="2\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition />

<ColumnDefinition />

</Grid.ColumnDefinitions>

<TextBlock Grid.Column="0" Grid.Row="0" Text="Name" />

<TextBlock Grid.Column="0" Grid.Row="1" Text="Vorname" />

<TextBlock Grid.Column="0" Grid.Row="2" Text="Wohnort" />

<TextBlock Grid.Column="0" Grid.Row="3" Text="Geboren" />

<TextBlock Grid.Column="1" Grid.Row="0" Text="Donald" />

<TextBlock Grid.Column="1" Grid.Row="1" Text="Duck" />

<TextBlock Grid.Column="1" Grid.Row="2" Text="Entenhausen" />

<TextBlock Grid.Column="1" Grid.Row="3" Text="ca. 1954" />

</Grid>

## Listing 3.13 *ColumnSpan* erlaubt das Überspannen von mehreren Spalten

<Grid x:Name="LayoutRoot" Margin="10" ShowGridLines="True">

<Grid.RowDefinitions>

<RowDefinition Height="35" />

<RowDefinition Height="Auto" />

<RowDefinition Height="\*"/>

<RowDefinition Height="2\*"/>

<RowDefinition Height="0.5\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition />

<ColumnDefinition />

</Grid.ColumnDefinitions>

<TextBlock Grid.Column="0" Grid.Row="0" Text="Name" />

<TextBlock Grid.Column="0" Grid.Row="1" Text="Vorname" />

<TextBlock Grid.Column="0" Grid.Row="2" Text="Wohnort" />

<TextBlock Grid.Column="0" Grid.Row="3" Text="Geboren" />

<TextBlock Grid.Column="1" Grid.Row="0" Text="Donald" />

<TextBlock Grid.Column="1" Grid.Row="1" Text="Duck" />

<TextBlock Grid.Column="1" Grid.Row="2" Text="Entenhausen" />

<TextBlock Grid.Column="1" Grid.Row="3" Text="ca. 1954" />

<Button Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="4"

Content="Klick mich" />

</Grid>

## Listing 3.14 Verwendung eines *GridSplitter*-Objekts

<Grid x:Name="LayoutRoot" Margin="10" ShowGridLines="True">

<Grid.ColumnDefinitions>

<ColumnDefinition />

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Button Grid.Column="0" Content="Eins" />

<controls:GridSplitter Grid.Column="0" Width="5" />

<Button Grid.Column="1" Content="Zwei" />

</Grid>

## Listing 3.15 *GridSplitter* über zwei Zeilen

<Grid x:Name="LayoutRoot" Margin="10" ShowGridLines="True">

<Grid.ColumnDefinitions>

<ColumnDefinition />

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition />

<RowDefinition />

</Grid.RowDefinitions>

<Button Grid.Column="0" Content="Eins" />

<controls:GridSplitter Grid.Column="0"

Width="5" Background="Red"

Grid.RowSpan="2"/>

<Button Grid.Column="1" Content="Zwei" />

</Grid>

## Listing 3.16 *WrapPanel* mit Inhalt

<Grid x:Name="LayoutRoot" Margin="10" ShowGridLines="True">

<cKit:WrapPanel>

<Button Width="100" Height="100" Content="100\*100" />

<Button Width="100" Height="100" Content="100\*100" />

<Button Width="100" Height="100" Content="100\*100" />

<Button Width="100" Height="100" Content="100\*100" />

<Button Width="100" Height="100" Content="100\*100" />

</cKit:WrapPanel>

</Grid>

## Listing 3.17 *MeasureOverride* des elliptischen Panels

protected override Size MeasureOverride(Size availableSize)

{

foreach (UIElement child in Children)

{

child.Measure(

new Size(double.PositiveInfinity,

double.PositiveInfinity));

}

return base.MeasureOverride(availableSize);

}

## Listing 3.18 *ArrangeOverride* des elliptischen Panels

protected override Size ArrangeOverride(Size finalSize)

{

if (Children.Count == 0) return finalSize;

double step = (360.0 / Children.Count) \* Math.PI / 180;

double maxX = 0;

double maxY =0;

foreach (UIElement child in Children)

{

maxX = Math.Max(child.DesiredSize.Width, maxX);

maxY = Math.Max(child.DesiredSize.Height, maxY);

}

double rX = finalSize.Width / 2.0 - maxX/2;

double rY = finalSize.Height / 2.0 -maxY/2;

Point point = new Point();

double angle = 0.0;

foreach (UIElement child in Children)

{

Point center

= new Point((rX - child.DesiredSize.Width/2) ,

rY- child.DesiredSize.Height/2);

point.X = Math.Cos(angle) \* rX + rX;

point.Y = Math.Sin(angle) \* rY + rY;

child.Arrange(new Rect(point,

new Size(child.DesiredSize.Width, child.DesiredSize.Height)));

angle += step;

}

return finalSize;

}

# Kapitel 4

## Listing 4.1 Ein *StackPanel* mit einer *AutoCompleteBox* in XAML

<StackPanel Background="LightGray">

<TextBlock Text="AutoCompleteBox"

Margin="5"/>

<StackPanel Orientation="Horizontal">

<TextBlock Text="Personen: "

Margin="5"

VerticalAlignment="Center" />

<input:AutoCompleteBox

x:Name="vornamenACB"

Width="100"

/>

</StackPanel>

</StackPanel>

## Listing 4.2 Ein *Border* umrahmt einen *TextBlock*

<Border Margin="10"

BorderBrush="Blue" BorderThickness="1 3 4 10"

CornerRadius="10 2 30 5" Padding="5"

>

<TextBlock TextWrapping="Wrap">

Mit einem Border können Sie andere Objekte einrahmen.

Ein Border gehört zu den Content-Elementen, er kann also nur einen

Inhalt haben – wobei dieser natürlich verschachtelt sein darf.

Sie können einen Border auch dazu verwenden, Elementen ohne

Hintergrund eine Hintergrundfarbe zuzuweisen. TextBlock zum Beispiel

kennt keine Farbe für den Hintergrund des Textes. Betten Sie den

TextBlock in einen Border ein, und weisen Sie dem Border eine Hintergrundfarbe

zu. Als Ergebnis scheint der TextBlock einen (farbigen) Hintergrund zu

besitzen.

Wichtige Eigenschaften sind BorderBrush und BorderThickness

sowie CornerRadius und Padding. Padding ist der innere Abstand

zwischen dem Border und seinem Inhalt. Diese Eigenschaft ist

hilfreich, sobald Sie CornerRadius verwenden. CornerRadius rundet die

Ecken einer Schaltfläche (Button) ab.

</TextBlock>

</Border>

## Listing 4.3 Binden eines Kalenders an einen *TextBlock* über Elementbindung

<StackPanel>

<controls:Calendar x:Name="cal" ></controls:Calendar>

<TextBlock

Text="{Binding SelectedDate,ElementName=cal,

Mode=TwoWay}" Width="100" Height="30"/>

</StackPanel>

## Listing 4.4 *DatePicker* in XAML

<StackPanel>

<controls:DatePicker x:Name="dp" >

</controls:DatePicker>

<TextBlock Text="{Binding SelectedDate, ElementName=dp,

Converter={StaticResource dateConverter}}" />

</StackPanel>

## Listing 4.5 Die Oberfläche des *InkPresenter*-Testprogramms in XAML

<UserControl

x:Class="InkPresenterFromHelp.MainPage"

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" d:DesignWidth="640" d:DesignHeight="480">

<Canvas x:Name="LayoutRoot">

<StackPanel>

<TextBlock Text="InkPresenter" FontWeight="Bold" Margin="10" />

<Grid>

<Rectangle Height="500" Width="500" Margin="10" Stroke="Black" />

<InkPresenter x:Name="inkPresenter" Height="500" Width="500"

Margin="10"

MouseLeftButtonDown="Ink\_MouseLeftButtonDown"

LostMouseCapture="Ink\_LostMouseCapture"

MouseMove="Ink\_MouseMove"

Background="Transparent" Opacity="1" />

</Grid>

</StackPanel>

</Canvas>

</UserControl>

## Listing 4.6 Verwendung des *InkPresenter*-Objekts in Code

1 public partial class MainPage : UserControl

2 {

3 Stroke stroke;

4 public MainPage()

5 {

6 InitializeComponent();

7 SetBoundary();

8 }

9

10 private void Ink\_MouseLeftButtonDown(object sender, MouseEventArgs e)

11 {

12 inkPresenter.CaptureMouse();

13 StylusPointCollection stylusPointCollection

14 = new StylusPointCollection();

15 stylusPointCollection.Add(e.StylusDevice.GetStylusPoints(inkPresenter));

16 stroke = new Stroke(stylusPointCollection);

17 inkPresenter.Strokes.Add(stroke);

18 }

19

20 private void Ink\_MouseMove(object sender, MouseEventArgs e)

21 {

22 if (stroke != null)

23 stroke.StylusPoints.Add(e.StylusDevice.GetStylusPoints(inkPresenter));

24 }

25

26 private void Ink\_LostMouseCapture(object sender, MouseEventArgs e)

27 {

28 stroke = null;

29

30 }

31

32 private void SetBoundary()

33 {

34 RectangleGeometry rectangleGeometry = new RectangleGeometry();

35 rectangleGeometry.Rect = new Rect(0, 0, inkPresenter.ActualWidth, inkPresenter.ActualHeight);

36 inkPresenter.Clip = rectangleGeometry;

37 }

38 }

## Listing 4.7 *ListBox* deklarativ erstellt

<ListBox Margin="10" SelectionMode="Single">

<ListBoxItem>

<TextBlock Text="Hallo Welt" />

</ListBoxItem>

<ListBoxItem>

<StackPanel Orientation="Horizontal">

<TextBlock Text="Eine Schulung" />

<Image Source="Images/CBR003877\_LoRes.jpg" Width="200" />

</StackPanel>

</ListBoxItem>

<ListBoxItem>

<TextBlock Text="Noch ein Text" />

</ListBoxItem>

</ListBox>

## Listing 4.8 XAML-Code für den Aufruf von *OpenFileDialog*

<Grid x:Name="LayoutRoot" Background="White">

<Button Content="Datei öffnen" x:Name="btn"

Height="30" Width="80" Margin="10"

HorizontalAlignment="Left" VerticalAlignment="Top"

Click="OpenFileDialog\_Click" />

<TextBox x:Name="tbResult" Text="... Dateiname ..."

Height="130" Width="300" Margin="10,50"

HorizontalAlignment="Left" VerticalAlignment="Top" />

</Grid>

## Listing 4.9 Öffnen eines *OpenFileDialogs* nach Anwenderinteraktion

private void OpenFileDialog\_Click(object sender, RoutedEventArgs e)

{

OpenFileDialog ofd = new OpenFileDialog();

ofd.Filter = "Text Files (.txt)|\*.txt|All Files (\*.\*)|\*.\*";

ofd.FilterIndex = 1;

ofd.Multiselect = true;

if (ofd.ShowDialog() == true)

{

System.IO.Stream fileStream = ofd.File.OpenRead();

using (System.IO.StreamReader reader

= new System.IO.StreamReader(fileStream))

{

tbResult.Text = reader.ReadLine();

}

fileStream.Close();

}

}

## Listing 4.10 XAML-Code für ein *SaveFileDialog*-Beispiel

<Grid x:Name="LayoutRoot">

<Grid.RowDefinitions>

<RowDefinition />

<RowDefinition />

</Grid.RowDefinitions>

<StackPanel Orientation="Horizontal">

<TextBox x:Name="tb" />

<Button x:Name="btnSaveFile" Width="100" Height="20"

Content="Save File" Click="btnSaveFile\_Click" />

</StackPanel>

<TextBlock x:Name="tblError" Grid.Row="1" />

</Grid>

## Listing 4.11 *SaveFileDialog* in C# verwenden

public partial class MainPage : UserControl

{

SaveFileDialog saveDialog;

public MainPage()

{

InitializeComponent();

Loaded += new RoutedEventHandler(MainPage\_Loaded);

}

void MainPage\_Loaded(object sender, RoutedEventArgs e)

{

this.saveDialog = new SaveFileDialog();

try

{

this.saveDialog.DefaultExt = ".txt";

this.saveDialog.Filter

= "Text Dateien|\*.txt|Log Dateien|\*.log|Alle Dateien|\*.\*";

this.saveDialog.FilterIndex = 2;

}

catch (Exception ex)

{

this.tblError.Text = "SaveFileDialog liefert einen Fehler: "

+ ex.Message;

}

}

private void btnSaveFile\_Click(object sender, RoutedEventArgs e)

{

bool? dialogResult = this.saveDialog.ShowDialog();

if (dialogResult == true)

{

using (Stream stream = saveDialog.OpenFile())

{

// schreiben ...

byte[] info = (new UTF8Encoding(true)).GetBytes(tb.Text);

stream.Write(info, 0, info.Length);

stream.Close();

};

}

}

}

## Listing 4.12 Einen *StreamWriter* statt des Streams aus Listing 4.11 verwenden

if (dialogResult == true)

{

using (Stream stream = saveDialog.OpenFile())

{

StreamWriter writer = new StreamWriter(stream);

writer.Write("Hallo Welt");

stream.Close();

};

}

## Listing 4.13 *TabControl* mit unterschiedlichen *TabItems*

<cntrl:TabControl>

<cntrl:TabItem Header="Header #1">

<Canvas Background="Red" />

</cntrl:TabItem>

<cntrl:TabItem >

<cntrl:TabItem.Header>

<StackPanel Orientation="Horizontal">

<Rectangle Width="10" Height="10"

Margin="2" Fill="Blue" />

<TextBlock Text="Header #2" />

</StackPanel>

</cntrl:TabItem.Header>

<Canvas Background="Red" />

</cntrl:TabItem>

</cntrl:TabControl>

## Listing 4.14 *ToolTipService* in XAML

<StackPanel HorizontalAlignment="Left">

<Button Content="Klick mich" Width="200" Margin="10">

<ToolTipService.ToolTip>

<TextBlock>Text unmittelbar im TextBlock</TextBlock>

</ToolTipService.ToolTip>

</Button>

<Button Content="Klick mich" Width="200" Margin="10">

<ToolTipService.ToolTip>

<StackPanel Orientation="Horizontal">

<Image Source="ucumari\_g.png" Width="100" />

<TextBlock TextWrapping="Wrap" Margin="3" >

Ein StackPanel als Inhalt des Quickinfos.

<LineBreak />

ToolTipService.ToolTip bietet die Möglichkeit, multimedialen Content

<LineBreak />

und Hilfetexte zur Verfügung zu stellen.

</TextBlock>

</StackPanel>

</ToolTipService.ToolTip>

</Button>

</StackPanel>

## Listing 4.15 *TreeView* in XAML

<cntrl:TreeView>

<cntrl:TreeView.Items>

<cntrl:TreeViewItem Header="Die Ducks">

<cntrl:TreeViewItem Header="Donald">

<cntrl:TreeViewItem Header="Tick" />

<cntrl:TreeViewItem Header="Trick" />

<cntrl:TreeViewItem Header="Track" />

</cntrl:TreeViewItem>

<cntrl:TreeViewItem Header="Dagobert" />

<cntrl:TreeViewItem Header="Dasy" />

</cntrl:TreeViewItem>

</cntrl:TreeView.Items>

</cntrl:TreeView>

## Listing 4.16 Textgestaltung mit *TextBlock* und *Inlines*

<TextBlock.Inlines>

<Run FontSize="12" Text="Die drei Spatzen" />

<LineBreak /><LineBreak />

<Run Text="In einem leeren Haselstrauch," />

<LineBreak />

<Run Text="da sitzen drei Spatzen, Bauch an Bauch." />

<LineBreak />

<Run Text="Der Erich rechts und links der Franz" />

<LineBreak />

<Run Text="und mittendrin der freche "/>

<Run FontWeight="Bold" >Hans.</Run>

<LineBreak />

...

<LineBreak />

<Run FontStyle="Italic" Text="Christian Morgenstern" />

</TextBlock.Inlines>

# Kapitel 5

## Listing 5.1 Rechteck mit *LineSegment*

<Path Stroke="Blue" StrokeThickness="2">

<Path.Data>

<PathGeometry>

<PathFigure StartPoint="10,10">

<PathFigure.Segments>

<LineSegment Point="10,100" />

<LineSegment Point="100,100" />

<LineSegment Point="100, 10" />

<LineSegment Point="10,10" />

</PathFigure.Segments>

</PathFigure>

</PathGeometry>

</Path.Data>

</Path>

## Listing 5.2 Das Bogensegment aus Abbildung 5.15 in XAML

<Path Stroke="Red" StrokeThickness="3">

<Path.Data>

<PathGeometry>

<PathFigure StartPoint="150,100">

<ArcSegment Point="300,100" Size="1,0.5" />

</PathFigure>

</PathGeometry>

</Path.Data>

</Path>

## Listing 5.3 Bezierkurve in XAML

<Path Stroke="Blue" StrokeThickness="2">

<Path.Data>

<PathGeometry>

<PathFigureCollection>

<PathFigure StartPoint="10,10">

<PathSegmentCollection>

<BezierSegment

Point1="100,10"

Point2="10,100"

Point3="100,50" />

</PathSegmentCollection>

</PathFigure>

</PathFigureCollection>

</PathGeometry>

</Path.Data>

</Path>

<Path Stroke="Red" StrokeThickness="2">

<Path.Data>

<GeometryGroup>

<LineGeometry StartPoint="10,10" EndPoint="100,10"/>

<EllipseGeometry Center="100,10"

RadiusX="4" RadiusY="5"/>

</GeometryGroup>

</Path.Data>

</Path>

<Path Stroke="Red" StrokeThickness="2">

<Path.Data>

<GeometryGroup>

<LineGeometry StartPoint="100,50"

EndPoint="10,100"/>

<EllipseGeometry Center="10,100" RadiusX="4"

RadiusY="5"/>

</GeometryGroup>

</Path.Data>

</Path>

## Listing 5.4 QuadraticBezierSegment

<Canvas x:Name="LayoutRoot">

<Path Stroke="Blue" StrokeThickness="5"

Canvas.Left="20" Canvas.Top="20">

<Path.Data>

<PathGeometry>

<PathFigure >

<QuadraticBezierSegment Point1="200,200"

Point2="300,100"/>

</PathFigure>

</PathGeometry>

</Path.Data>

</Path>

<Path Stroke="Green" StrokeThickness="2"

StrokeDashArray="5 2" Canvas.Top="20"

Canvas.Left="20">

<Path.Data>

<GeometryGroup>

<LineGeometry StartPoint="200,200"

EndPoint="300,100"></LineGeometry>

</GeometryGroup>

</Path.Data>

</Path>

<Path Fill="Red" Stroke="Red" StrokeThickness="8"

Canvas.Top="20" Canvas.Left="20">

<Path.Data>

<GeometryGroup>

<EllipseGeometry

Center="200,200">

</EllipseGeometry>

<EllipseGeometry

Center="300,100">

</EllipseGeometry>

</GeometryGroup>

</Path.Data>

</Path>

</Canvas>

## Listing 5.5 Ein Dreieck in C#

Path createPath(Point[] punkte, bool isClosed)

{

Path path = new Path();

PathGeometry geometry = new PathGeometry();

PathFigure figure = new PathFigure();

figure.IsClosed = isClosed;

figure.StartPoint = punkte[0];

PathSegmentCollection psColl

= new PathSegmentCollection();

for (int i = 0; i < punkte.Length; i++)

{

LineSegment segment = new LineSegment();

segment.Point = punkte[i];

psColl.Add(segment);

}

figure.Segments = psColl;

geometry.Figures.Add(figure);

path.Data = geometry;

path.Fill = new SolidColorBrush(Colors.Blue);

return path;

}

## Listing 5.6 Ein über die Pfadgeometrie in C# erzeugtes Dreieck

public partial class Page : UserControl

{

public Page()

{

InitializeComponent();

Loaded += new RoutedEventHandler(Page\_Loaded);

}

void Page\_Loaded(object sender, RoutedEventArgs e)

{

LayoutRoot.Children.Add(createPath(

new Point[]

{

new Point(0,100),

new Point(100,100),

new Point(100,0)

},

true));

}

Path createPath(Point[] punkte, bool isClosed)

{

Path path = new Path();

PathGeometry geometry = new PathGeometry();

PathFigure figure = new PathFigure();

figure.IsClosed = isClosed;

figure.StartPoint = punkte[0];

PathSegmentCollection psColl

= new PathSegmentCollection();

for (int i = 0; i < punkte.Length; i++)

{

LineSegment segment = new LineSegment();

segment.Point = punkte[i];

psColl.Add(segment);

}

figure.Segments = psColl;

geometry.Figures.Add(figure);

path.Data = geometry;

path.Fill = new SolidColorBrush(Colors.Blue);

return path;

}

}

## Listing 5.7 Ein über die Pfadgeometrie in XAML erzeugtes Dreieck

<Path Fill="Red" >

<Path.Data>

<PathGeometry>

<PathGeometry.Figures >

<PathFigure IsClosed="True" StartPoint="0,100">

<PathFigure.Segments>

<LineSegment >

<LineSegment.Point >

0,100

</LineSegment.Point>

</LineSegment>

<LineSegment >

<LineSegment.Point >

100,100

</LineSegment.Point>

</LineSegment>

<LineSegment >

<LineSegment.Point >

100,0

</LineSegment.Point>

</LineSegment>

</PathFigure.Segments>

</PathFigure>

</PathGeometry.Figures>

</PathGeometry>

</Path.Data>

</Path>

## Listing 5.8 Verwendung eines *ImageBrush*-Objekts als Ressource und Verwendung der Ressource

<Grid x:Name="LayoutRoot" Background="Bisque">

<Grid.Resources>

<ImageBrush x:Key="a" Stretch="None"

AlignmentX="Left"

ImageSource="/Content/KleinA.jpg" />

<ImageBrush x:Key="aUniToFill" Stretch="UniformToFill"

AlignmentX="Left"

ImageSource="/Content/KleinA.jpg" />

<ImageBrush x:Key="aUni" Stretch="Uniform"

AlignmentX="Left"

ImageSource="/Content/KleinA.jpg" />

</Grid.Resources>

<controls:WrapPanel>

<StackPanel Margin="3">

<TextBlock>

AlignmentX="Left"<LineBreak/>

Stretch="UniformToFill"

</TextBlock>

<Rectangle Width="100" Height="100"

Fill="{StaticResource a}"

Stroke="Red" />

</StackPanel>

<StackPanel Margin="3">

<TextBlock>

AlignmentX="Left"<LineBreak/>

Stretch="UniformToFill"

</TextBlock>

<Rectangle Width="100" Height="100"

Fill="{StaticResource aUniToFill}"

Stroke="Red" />

</StackPanel>

<StackPanel Margin="3">

<TextBlock>

AlignmentX="Left"<LineBreak/>

Stretch="Uniform"

</TextBlock>

<Rectangle Width="100" Height="100" Fill="{StaticResource aUni}"

Stroke="Red" />

</StackPanel>

</controls:WrapPanel>

</Grid>

## Listing 5.9 XAML-Code für einen horizontalen *LinearGradientBrush*

<Rectangle Width="300" Height="200">

<Rectangle.Fill>

<LinearGradientBrush StartPoint="0.0, 0.5"

EndPoint="1.0, 0.5">

<LinearGradientBrush.GradientStops>

<GradientStop Color="Wheat" Offset="0.0" />

<GradientStop Color="Blue" Offset="1.0" />

</LinearGradientBrush.GradientStops>

</LinearGradientBrush>

</Rectangle.Fill>

</Rectangle>

## Listing 5.10 XAML-Code eines linaren *GradientBrush*-Objekts mit zur Mitte verschobenem *GradientStop*

<Canvas x:Name="LayoutRoot" Background="White">

<Rectangle Canvas.Left="50" Canvas.Top="50"

Width="300" Height="200">

<Rectangle.Fill>

<LinearGradientBrush StartPoint="1.0, 0.0" EndPoint="0.0, 1.0">

<LinearGradientBrush.GradientStops>

<GradientStop Color="Wheat" Offset="0.25" />

<GradientStop Color="Blue" Offset="0.75" />

</LinearGradientBrush.GradientStops>

</LinearGradientBrush>

</Rectangle.Fill>

</Rectangle>

<TextBlock Canvas.Left="30" Canvas.Top="30" Text="0.0, 0.0" Foreground="Black" />

<TextBlock Canvas.Left="5" Canvas.Top="130" Text="0.0, 0.5" Foreground="Black" />

<TextBlock Canvas.Left="30" Canvas.Top="250" Text="0.0, 0.1" Foreground="Black" />

<TextBlock Canvas.Left="330" Canvas.Top="30" Text="0.0, 0.0" Foreground="Black" />

<TextBlock Canvas.Left="355" Canvas.Top="130" Text="0.0, 0.5" Foreground="Black" />

<TextBlock Canvas.Left="330" Canvas.Top="250" Text="0.0, 0.1" Foreground="Black" />

<Ellipse Canvas.Left="47.5" Canvas.Top="47.5" Width="5" Height="5" Fill="Red" />

<Ellipse Canvas.Left="47.5" Canvas.Top="147.5" Width="5" Height="5" Fill="Red" />

<Ellipse Canvas.Left="47.5" Canvas.Top="247.5" Width="5" Height="5" Fill="Red" />

<Ellipse Canvas.Left="347.5" Canvas.Top="47.5" Width="5" Height="5" Fill="Red" />

<Ellipse Canvas.Left="347.5" Canvas.Top="147.5" Width="5" Height="5" Fill="Red" />

<Ellipse Canvas.Left="347.5" Canvas.Top="247.5" Width="5" Height="5" Fill="Red" />

<Line X1="50" Y1="250" X2="350" Y2="50" Stroke="Violet" StrokeThickness="2"/>

</Canvas>

## Listing 5.11 XAML-Code für einen *RadialGradientBrush*

<Canvas Grid.Column="0" Grid.Row="0"

Width="200" Height="200">

<Rectangle Width="150" Height="150"

Canvas.Left="25" Canvas.Top="25" >

<Rectangle.Fill>

<RadialGradientBrush Center="0.5, 0.5"

RadiusX="0.5" RadiusY="0.5">

<GradientStop Color="Wheat" Offset="0" />

<GradientStop Color="Blue" Offset="1"/>

</RadialGradientBrush>

</Rectangle.Fill>

</Rectangle>

</Canvas>

## Listing 5.12 Die Klasse *WriteableBitmap*

public sealed class WriteableBitmap : BitmapSource

{

public WriteableBitmap(BitmapSource source);

public WriteableBitmap(int pixelWidth, int pixelHeight);

public WriteableBitmap(UIElement element, Transform transform);

public int[] Pixels { get; }

[SecuritySafeCritical]

public void Invalidate();

[SecuritySafeCritical]

public void Render(UIElement element, Transform transform);

}

## Listing 5.13 Verblassende Sterne in C#

public partial class MainPage : UserControl

{

Rectangle rectangleBlack;

WriteableBitmap writeableBitmap;

double width, height;

Point mousePosition;

Image imageBlueStar;

Image imageBlur;

public MainPage()

{

InitializeComponent();

width = LayoutRoot.Width;

height = LayoutRoot.Height;

Loaded += new RoutedEventHandler(MainPage\_Loaded);

LayoutRoot.MouseMove

+= new MouseEventHandler(LayoutRoot\_MouseMove);

}

void LayoutRoot\_MouseMove(object sender, MouseEventArgs e)

{

mousePosition = e.GetPosition(LayoutRoot);

Debug.WriteLine(mousePosition.X);

}

void MainPage\_Loaded(object sender, RoutedEventArgs e)

{

BlurEffect blur = new BlurEffect();

blur.Radius = 3;

imageBlur = new Image();

imageBlur.Effect = blur;

imageBlueStar = new Image();

BitmapImage bitmapImage

= new BitmapImage(

new Uri("BlueStar.jpg", UriKind.Relative));

imageBlueStar.Source = bitmapImage;

imageBlueStar.Width = 30;

imageBlueStar.Height = 30;

imageBlueStar.Visibility = Visibility.Collapsed;

LayoutRoot.Children.Add(imageBlueStar);

LayoutRoot.Children.Add(imageBlur);

width = LayoutRoot.Width;

height = LayoutRoot.Height;

rectangleBlack = new Rectangle();

rectangleBlack.Width = width;

rectangleBlack.Height = height;

rectangleBlack.Fill

= new SolidColorBrush(Color.FromArgb(10, 0, 0, 0));

writeableBitmap

= new WriteableBitmap((int)width, (int)height);

imageBlur.Source = writeableBitmap;

CompositionTarget.Rendering

+= new EventHandler(CompositionTarget\_Rendering);

}

void CompositionTarget\_Rendering(object sender, EventArgs e)

{

writeableBitmap.Invalidate();

writeableBitmap.Render(imageBlur,

new MatrixTransform());

TranslateTransform translateTransform

= new TranslateTransform();

translateTransform.X = mousePosition.X;

translateTransform.Y = mousePosition.Y;

writeableBitmap.Render(imageBlueStar,

translateTransform);

writeableBitmap.Render(rectangleBlack,

new MatrixTransform());

}

private void outerFrame\_MouseMove(object sender,

MouseEventArgs e)

{

mousePosition = e.GetPosition(LayoutRoot);

}

}

## Listing 5.14 Komplexe Zahl als *struct ComplexNumber*

public struct ComplexNumber

{

private readonly double real;

private readonly double imaginary;

public double Real

{

get { return real; }

}

public double Imaginary

{

get { return imaginary; }

}

public ComplexNumber(double real, double imaginary)

{

this.real = real;

this.imaginary = imaginary;

}

public static ComplexNumber operator +(ComplexNumber summand1, ComplexNumber summand2)

{

return new ComplexNumber(summand1.real + summand2.real, summand1.imaginary +

summand2.imaginary);

}

public static ComplexNumber operator -(ComplexNumber summand1, ComplexNumber summand2)

{

return new ComplexNumber(summand1.real - summand2.real, summand1.imaginary -

summand2.imaginary);

}

public static ComplexNumber operator \*(ComplexNumber s1, ComplexNumber s2)

{

return new ComplexNumber(

s1.real \* s2.real - s1.imaginary\*s2.imaginary,

s1.real \* s2.imaginary + s1.imaginary \* s2.real

);

}

public static bool operator ==(ComplexNumber op1, ComplexNumber op2)

{

return (op1.real == op2.real

&& op1.imaginary == op2.imaginary);

}

public static bool operator !=(ComplexNumber op1, ComplexNumber op2)

{

return !(op1 == op2);

}

public static implicit operator ComplexNumber(ComplexPoint complexPoint)

{

return new ComplexNumber(complexPoint.Real, complexPoint.Img);

}

public override bool Equals(object obj)

{

if (obj == null) return false;

if (ReferenceEquals(this, obj)) return true;

if (obj != typeof(ComplexNumber)) return false;

if (obj.GetType() == typeof(ComplexNumber)) return false;

return Equals((ComplexNumber) obj);

}

public bool Equals(ComplexNumber number)

{

return (this.real == number.Real && this.imaginary == number.imaginary);

}

public override int GetHashCode()

{

return base.GetHashCode()\*this.imaginary.GetHashCode();

}

public override string ToString()

{

return string.Format("{0} +i{1}",

this.real, this.imaginary);

}

public double Absolute()

{

return Math.Sqrt(this.real\*this.real + this.imaginary\*this.imaginary);

}

}

## Listing 5.15 *ComplexPane*

public struct ComplexPane

{

ComplexNumber topLeft, bottomRight;

public ComplexNumber TopLeft

{

set {topLeft = value;}

get {return topLeft;}

}

public ComplexNumber BottomRight

{

set { bottomRight = value; }

get { return bottomRight; }

}

public ComplexPane(ComplexNumber topLeft,

ComplexNumber bottomRight)

{

this.topLeft = topLeft;

this.bottomRight = bottomRight;

}

}

## Listing 5.16 ComplexPoint implementiert INotifyPropertyChanged

public class ComplexPoint:INotifyPropertyChanged

{

double real;

public double Real

{

get { return real; }

set { real = value;

OnPropertyChanged("Real");

}

}

double img;

public double Img

{

get { return img; }

set { img = value;

OnPropertyChanged("Img");

}

}

public ComplexPoint(double real, double img)

{

this.real = real;

this.img = img;

}

public event PropertyChangedEventHandler PropertyChanged;

void OnPropertyChanged(string property)

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(property));

}

}

## Listing 5.17 Mandelbrot-*Generator*

public class Generator

{

int maxInterations;

ComplexPane complexPane;

int[] pixels;

public int[] Pixels

{

get { return pixels; }

set { pixels = value; }

}

public Generator(int width, int height )

: this(width, height

,new ComplexPane(new ComplexNumber(-2.5,1.5), new ComplexNumber(1.5,-1.5))

, 20)

{

}

public Generator(int width, int height, ComplexPane complexPane)

: this(width, height, complexPane, 20)

{

}

public Generator(int width, int height, ComplexPane complexPane, int maxInterations)

{

this.maxInterations = maxInterations;

pixels = new int[width \* height];

this.complexPane = complexPane;

}

public Color GetColorForComplexNumber(ComplexNumber c)

{

int iteration;

ComplexNumber z = c;

for (iteration = 0; iteration < maxInterations && z.Absolute <= 2; iteration++)

{

z = z \* z + c;

}

if (iteration >= maxInterations)

{

return Colors.Black;

}

return GetColorForLevel(iteration, maxInterations);

}

private static Color GetColorForLevel(int pLevel,int pMaxIteration)

{

double lRatio = pLevel / (double)pMaxIteration;

int i = -(int)(lRatio \* (int.MaxValue)) ;

return IntToColor(i);

}

/// <summary>

/// Diese Utility-Funktionen könnten auch als Erweiterungsmethoden geschrieben werden.

/// </summary>

static int ColorToInt(Color c)

{

return (c.A << 24) | (c.R << 16) | (c.G << 8) | (c.B);

}

static Color IntToColor(int i)

return Color.FromArgb( (byte)(i >> 24), (byte)(i >> 16), (byte)(i >> 8), (byte)( i));

}

ComplexNumber PointToComplex(Point p, int width, int height)

{

return PointToComplex(p.X, p.Y, width, height);

}

ComplexNumber PointToComplex(double x, double y, int width, int height)

{

double real;

double img;

real = x / width

\* (complexPane.BottomRight.Real - complexPane.TopLeft.Real)

+ complexPane.TopLeft.Real;

img = complexPane.TopLeft.Imaginary

- y / height

\* (complexPane.TopLeft.Imaginary - complexPane.BottomRight.Imaginary);

return new ComplexNumber(real, img);

}

public int[] CalculateBitmap(int width, int height)

{

int[] temp = new int[width\*height];

for (int y = 0; y < width; y++)

{

for (int x = 0; x < height; x++)

{

ComplexNumber number = PointToComplex(x, y, width, height);

Color color = GetColorForComplexNumber(number);

temp[x \* width + y] = ColorToInt(color);

}

return temp;

}

}

## Listing 5.18 Erweiterungsmethoden *ColorToInt* und *IntToColor*

public static class ColorHelper

{

public static int ColorToInt(this Color c)

{

return (c.A << 24) | (c.R << 16) | (c.G << 8) | (c.B);

}

public static Color IntToColor(this int i)

{

return Color.FromArgb((byte)(i >> 24), (byte)(i >> 16), (byte)(i >> 8), (byte)(i));

}

}

## Listing 5.19 Aufruf des *Generators* im Codebehind

public partial class MainPage : UserControl

{

int width = 300;

int height = 300;

WriteableBitmap bitmap;

Generator generator;

public MainPage()

{

InitializeComponent();

Loaded += new RoutedEventHandler(OnLoaded);

}

void OnLoaded(object sender, RoutedEventArgs e)

{

textBlockBussy.Visibility = Visibility.Visible;

int iterationDepth = 100;

bitmap = new WriteableBitmap(width, height);

generator = new Generator(width, height,

new ComplexPane(

new ComplexNumber(-2.5, 1.5),

new ComplexNumber(1.5, -1.5)),

iterationDepth);

int[] temp = new int[width \* height];

temp = generator.CalculateBitmap(width, height);

for (int i = 0; i < width \* height; i++)

{

bitmap.Pixels[i] = temp[i];

}

theImage.Source = bitmap;

}

}

## Listing 5.20 Verwendung des *Generators* mit einem *BackgroundWorker*

public partial class SimpleMainPage : UserControl

{

BackgroundWorker worker;

int width = 300;

int height = 300;

WriteableBitmap bitmap;

Generator generator;

public SimpleMainPage()

{

InitializeComponent();

Loaded += new RoutedEventHandler(OnLoaded);

}

void OnLoaded(object sender, RoutedEventArgs e)

{

textBlockBussy.Visibility = Visibility.Visible;

int iterationDepth = 100;

bitmap = new WriteableBitmap(width, height);

generator = new Generator(width, height,

new ComplexPane(

new ComplexNumber(-2.5, 1.5),

new ComplexNumber(1.5, -1.5)),

iterationDepth);

worker = new BackgroundWorker();

worker.DoWork += new DoWorkEventHandler(worker\_DoWork);

worker.RunWorkerCompleted

+= new RunWorkerCompletedEventHandler(worker\_RunWorkerCompleted);

worker.RunWorkerAsync();

}

void worker\_DoWork(object sender, DoWorkEventArgs e)

{

e.Result =

generator.CalculateBitmap(width, height);

}

void worker\_RunWorkerCompleted(object sender,

RunWorkerCompletedEventArgs e)

{

if (e.Error == null)

{

for (int i = 0; i < width \* height; i++)

{

bitmap.Pixels[i] = (e.Result as int[])[i];

}

theImage.Source = bitmap;

theImage.InvalidateArrange();

theImage.InvalidateMeasure();

textBlockBussy.Visibility = Visibility.Collapsed;

}

else

{

textBlockBussy.Text = e.Error.Message;

}

}

}

## Listing 5.21 Mathematische Farbmuster

void MainPage\_Loaded(object sender, RoutedEventArgs e)

{

int width = 300;

int height = 300;

WriteableBitmap wp = new WriteableBitmap(width, height);

for (int x = 0; x < width; x++)

{

for (int y = 0; y < height; y++)

{

Color c = Color.FromArgb(0, (byte) (x\*y%255), (byte) (y%255), (byte) (x\*y%255));

byte [] bb = new byte[]

{

c.A, c.R, c.G, c.B

};

wp.Pixels[y \* width + x] = BitConverter.ToInt32(bb, 0);

}

}

img.Source = wp;

}

## Listing 5.22 Mathematisches Farbmuster (sin\*cos)

void MainPage\_Loaded(object sender, RoutedEventArgs e)

{

int width = 300;

int height = 300;

WriteableBitmap wp = new WriteableBitmap(width, height);

for (int x = 0; x < width; x++)

{

for (int y = 0; y < height; y++)

{

Color c = Color.FromArgb(

0,

(byte) (x%255),

(byte) (y%255),

(byte) (Math.Sin(x)\*Math.Cos(y) \* 255));

byte [] bb = new byte[]

{

c.A, c.R, c.G, c.B

};

wp.Pixels[y \* width + x]

= BitConverter.ToInt32(bb, 0);

}

}

img.Source = wp;

}

## Listing 5.23 Verlaufendes Gelbgrün: (x > 0.5 ? x : 1.0 - x)

for (int y = 0; y < height; y++)

{

Color c = Color.FromArgb(

0,

(byte)(x % 255),

(byte)(y % 255),

(byte)(x > 0.5 ? x : 1.0 - x)

);

## Listing 5.24 Farbmosaik: ((x & 1) == (y & 1)) ? 0 : 255

Color c = Color.FromArgb(

0,

(byte)(x % 255),

(byte)(y % 255),

(byte)(((x & 1) == (y & 1)) ? 0 : 255)

);

## Listing 5.25 Einschalten der Hardwarebeschleunigung in der HTML-Seite

<object data="data:application/x-silverlight-2," type="application/x-silverlight-2" width="100%"

height="100%">

<param name="source" value="ClientBin/UsingDeepZoom.xap"/>

<param name="onError" value="onSilverlightError" />

<param name="background" value="white" />

<param name="minRuntimeVersion" value="3.0.40818.0" />

<param name="EnableGUPAcceleration" value="true" />

<param name="autoUpgrade" value="true" />

<a href="http://go.microsoft.com/fwlink/?LinkID=149156&v=3.0.40818.0" style="text-decoration:none">

<img src="http://go.microsoft.com/fwlink/?LinkId=108181"

alt="Get Microsoft Silverlight" style="border-style:none"/>

</a>

</object>

## Listing 5.26 Button mit einem Schlagschatten

<Button Content="Klick mich" Width="200" Height="25" Margin="30">

<Button.Effect>

<DropShadowEffect Direction="-45" ShadowDepth="10"

BlurRadius="8"/>

</Button.Effect>

</Button>

</Button>

## Listing 5.27 Schlagschatten und Blur-Effekt

<Border Margin="30">

<Button Content="Klick mich" Width="200" Height="25" >

<Button.Effect>

<DropShadowEffect Direction="-45" ShadowDepth="10" BlurRadius="8"/>

</Button.Effect>

</Button>

<Border.Effect>

<BlurEffect Radius="6" />

</Border.Effect>

</Border>

## Listing 5.28 Einfacher Shader, der das Eingangsbild unverändert zurückliefert

1 sampler2D implicitInput : register(S0);

2

3 float4 main(float2 uv : TEXCOORD) : COLOR

4 {

5 float4 Color;

6 Color = tex2D (implicitInput, uv.xy);

7 return Color;

8 }

## Listing 5.29 IdentityShader.cs

public class IdentityShader : ShaderEffect

{

public IdentityShader()

{

//UsingShaders\Shaders\identity.ps

Uri uri = new Uri(@"/UsingShaders;component/Shaders/identity.ps",

UriKind.Relative);

PixelShader = new PixelShader() { UriSource = uri };

}

}

## Listing 5.30 Einsatz des *IdentityShaders* von Listing 5.28 (HLSL) und Listing 5.29 (C#) in XAML

<UserControl

x:Class="UsingShaders.MainPage"

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"

xmlns:loc="clr-namespace:UsingShaders.Shaders" >

<Grid x:Name="LayoutRoot" ShowGridLines="True" Margin="10" >

<Button Grid.Row="2" Margin="10"

Content="Klick mich" Height="30" Width="150">

<Button.Effect>

<loc:IdentityShader />

</Button.Effect>

</Button>

</Grid>

</UserControl

## Listing 5.31 *FlipUpDown.fx* HLSL-Shader-Quelltext

sampler2D implicitInput : register(S0);

float4 main(float2 uv : TEXCOORD) : COLOR

{

uv.y = 1 - uv.y;

float4 Color;

Color = tex2D (implicitInput, uv.xy);

return Color;

}

## Listing 5.32 C#-Wrapper für einen *FlipUpDownShader*

1 public class FlipUpDownShader : ShaderEffect

2 {

3 static readonly PixelShader pixelShader;

4

5 static FlipUpDownShader()

6 {

7 pixelShader = new PixelShader();

8 pixelShader.UriSource

9 = new Uri( @"/UsingShaders;component/Shaders/flipUpDown.ps",

10 UriKind.Relative);

11 }

12 public FlipUpDownShader()

13 {

14 this.PixelShader = pixelShader;

15 UpdateShaderValue(InputProperty);

16 }

17

18 public static DependencyProperty InputProperty

19 = ShaderEffect.RegisterPixelShaderSamplerProperty(

20 "Input", typeof(FlipUpDownShader),

21 0);

22

23 public Brush Input

24 {

25 get

26 {

27 return (Brush)base.GetValue(InputProperty);

28 }

29 set

30 {

31 base .SetValue(InputProperty,value);

32 }

33 }

34 }

## Listing 5.33 Quelltext des HLSL-Reflection-Shaders

sampler2D implicitInput : register(S0);

float4 main(float2 uv : TEXCOORD) : COLOR

{

if (uv.y > 0.5)

{

uv.y = 1.0 - uv.y;

return tex2D (implicitInput, uv) \* uv.y;

}

return tex2D (implicitInput, uv);

}

## Listing 5.34 C#-Wrapper für den Reflection-Shader

1 public class ReflectionShader:ShaderEffect

2 {

3 public ReflectionShader()

4 {

5 //UsingShaders\Shaders\reflection.ps

6 Uri uri = new Uri(@"/UsingShaders;component/Shaders/Reflection.ps",

7 UriKind.Relative);

8 PixelShader = new PixelShader() { UriSource = uri };

9 }

10

11 public static readonly DependencyProperty PaddingProperty

12 = DependencyProperty.Register("Padding",

13 typeof(double), typeof(ReflectionShader),

14 new PropertyMetadata(50.0, OnPaddingChanged));

15

16

17 public static void OnPaddingChanged(DependencyObject obj,

18 DependencyPropertyChangedEventArgs args)

19 {

20 (obj as ReflectionShader).PaddingBottom = (double) args.NewValue;

21 }

22

23 /// <summary>

24 /// Eine Alternative zu

25 /// (obj as ReflectionShader).PaddingBottom = (double) args.NewValue;

26 /// wäre diese Funktion.

27 /// </summary>

28 /// <param name="oldValue"></param>

29 /// <param name="newValue"></param>

30 protected virtual void OnPaddingChanged(double oldValue,

31 double newValue)

32 {

33 PaddingBottom = newValue;

34 }

35

36 public double Padding

37 {

38 get

39 {

40 return (double)base.GetValue(PaddingProperty);

41 }

42 set

43 {

44 base.SetValue(PaddingProperty, value);

45 }

46 }

47 }

## Listing 5.35 Verwendung des Reflection-Shaders in XAML

<Button Grid.Row="1" Margin="10"

Content="Hallo Welt" Height="30" Width="150" >

<Button.Effect>

<loc:ReflectionShader Padding="30">

</loc:ReflectionShader>

</Button.Effect>

</Button>

<Image Source="ucumari\_g.png" Height="150"

Grid.Column="1" Grid.Row="1" Margin="10">

<Image.Effect>

<loc:ReflectionShader Padding="150" />

</Image.Effect>

</Image>

## Listing 5.36 HLSL-Grün in gleicher Stärke wie Rot

float4 main(float2 uv : TEXCOORD) : COLOR

{

float4 Color;

Color =tex2D (implicitInput, uv);

Color.rg = Color.r \* 1.5;

return Color;

}

## Listing 5.37 Sharpen (Verschärfen)

sampler2D implicitInput : register(S0);

//--------------------------------------------------------

// Pixel Shader Sharper

//--------------------------------------------------------

float4 main(float2 uv : TEXCOORD) : COLOR

{

float4 Color;

Color =tex2D (implicitInput, uv);

Color += tex2D (implicitInput, uv.xy-0.001)\*10.0f;

Color -= tex2D(implicitInput, uv.xy +0.001)\*10.0f;

return Color;

}

## Listing 5.38 Embross (Stanzen)

sampler2D implicitInput : register(S0);

// embross

float4 main(float2 uv : TEXCOORD) : COLOR

{

float4 Color;

Color.a = 1.0f;

Color.rgb =0.5f;

Color =tex2D (implicitInput, uv);

Color -= tex2D (implicitInput, uv.xy-0.001)\*2.0f;

Color += tex2D(implicitInput, uv.xy +0.001)\*2.0f;

Color.rgb = (Color.r+Color.g+Color.b)/3.0f;

return Color;

}

## Listing 5.39 Farbinvertierung

sampler2D implicitInputSampler : register(S0);

float4 main(float2 uv : TEXCOORD) : COLOR

{

float4 color = tex2D( implicitInputSampler, uv );

float4 inverted\_color = 1 - color;

inverted\_color.a = color.a;

inverted\_color.rgb \*= inverted\_color.a;

return inverted\_color;

}

## Listing 5.40 Erhöhung der Transparenz

sampler2D implicitInputSampler : register(S0);

int alpha = 10;

float4 main(float2 uv : TEXCOORD) : COLOR

{

float4 color = tex2D( implicitInputSampler, uv );

if( color.a != 0 ) {

color.a = color.a - alpha;

}

return color;

}

## Listing 5.41 *TextBox*, die sich um die Y-Achse dreht

<Grid x:Name="LayoutRoot">

<Grid.Resources>

<Storyboard x:Name="rotation">

<DoubleAnimation From="0" To="360"

Storyboard.TargetName="projection"

Storyboard.TargetProperty="RotationY"

RepeatBehavior="Forever" Duration="0:0:5">

</DoubleAnimation>

</Storyboard>

</Grid.Resources>

<TextBox Width="200" Height="25" Text="Hallo Welt">

<TextBox.Projection>

<PlaneProjection x:Name="projection" />

<!--RotationX="0" RotationY="0" RotationZ="0"/>-->

</TextBox.Projection>

</TextBox>

</Grid>

## Listing 5.42 XAML-3D-Beispiel

<Grid x:Name="LayoutRoot">

<Grid.ColumnDefinitions>

<ColumnDefinition />

<ColumnDefinition Width="Auto" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition />

</Grid.RowDefinitions>

<Canvas Grid.RowSpan="2">

<Border Canvas.Left="100" Canvas.Top="100"

Width="300" Height="300" BorderBrush="Red" BorderThickness="1">

<TextBlock Text="1" FontSize="80"

HorizontalAlignment="Center" VerticalAlignment="Center"/>

<Border.Projection >

<PlaneProjection x:Name="x1p"

RotationX="-90"

RotationY="0"

RotationZ="0"

CenterOfRotationZ="150" />

</Border.Projection>

</Border>

<Border Canvas.Left="100" Canvas.Top="100"

Width="300" Height="300" BorderBrush="Blue" BorderThickness="1">

<TextBlock Text="2" FontSize="80"

HorizontalAlignment="Center" VerticalAlignment="Center"/>

<Border.Projection >

<PlaneProjection x:Name="x2p"

RotationX="0"

RotationY="0"

RotationZ="0"

CenterOfRotationZ="150" />

</Border.Projection>

</Border>

</Canvas>

<Border Grid.Column="1" Background="AntiqueWhite" CornerRadius="10"

Padding="10" Margin="10">

<Grid Grid.Column="1" ShowGridLines="True" Width="400">

<Grid.ColumnDefinitions>

<ColumnDefinition />

<ColumnDefinition />

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<TextBlock Text="RotationX" />

<TextBox Grid.Column="1" Grid.Row="0" Width="60"

Text="{Binding Path=Value, ElementName=rotX, Mode=TwoWay}" />

<Slider x:Name="rotX"

Grid.Column="2" Grid.Row="0"

Value="{Binding Path=RotX1, Mode=TwoWay, Source={StaticResource werte}}"

Minimum="-180" Maximum="180" />

<TextBlock Grid.Row="1" Text="RotationY" />

<TextBox Grid.Column="1" Grid.Row="1" Width="60"

Text="{Binding Path=Value, ElementName=rotY}" />

<Slider x:Name="rotY"

Grid.Column="2" Grid.Row="1"

Value="{Binding Path=RotY1, Mode=TwoWay, Source={StaticResource werte}}"

Minimum="-180" Maximum="180" />

</Grid>

</Border>

</Grid>

# Kapitel 6

## Listing 6.1 Drehung eines Rechtecks um sein Zentrum. *CenterX* und *CenterY* werden in Absolutwerten angegeben

<Rectangle

Canvas.Top="20" Canvas.Left="20"

Width="50" Height="40"

Fill="BlanchedAlmond" >

<Rectangle.RenderTransform>

<RotateTransform

CenterX="20" CenterY="20"

Angle="45" />

</Rectangle.RenderTransform>

</Rectangle>

## Listing 6.2 Drehung eines Rechtecks, *RenderTransformOrigin* ermöglicht es, das Rotationszentrum relativ anzugeben

<Rectangle Width="200" Height="60" Fill="Blue" >

<Rectangle.RenderTransformOrigin >

<Point X="0.5" Y="0.5" />

</Rectangle.RenderTransformOrigin>

<Rectangle.RenderTransform >

<RotateTransform Angle="-45" />

</Rectangle.RenderTransform>

</Rectangle>

## Listing 6.3 Skalierung eines Rechtecks

<Rectangle Width="200" Height="60" Fill="BlanchedAlmond" >

<Rectangle.RenderTransform >

<ScaleTransform ScaleX="1.5" ScaleY="1.5" />

</Rectangle.RenderTransform>

</Rectangle>

<Rectangle Width="200" Height="60" Stroke="Black" />

## Listing 6.4 Translation in horizontaler Richtung um 20 Pixel, in vertikaler Richtung um 10 Pixel

<Rectangle Width="200" Height="60" Fill="BlanchedAlmond" >

<Rectangle.RenderTransform >

<TranslateTransform X="20" Y="10" />

</Rectangle.RenderTransform>

</Rectangle>

<Rectangle Width="200" Height="60" Stroke="Black" />

## Listing 6.5 Standardscherung und Scherung mit verändertem Scherungszentrum

<Grid x:Name="LayoutRoot" ShowGridLines="True" Background="White">

<Grid.ColumnDefinitions>

<ColumnDefinition />

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Canvas Grid.Column ="0" Width="300" Height="300" Margin="10">

<Rectangle Width="200" Height="60" Fill="BlanchedAlmond">

<Rectangle.RenderTransform >

<SkewTransform AngleX="20" CenterY="-200"/>

</Rectangle.RenderTransform>

</Rectangle>

<Rectangle Width="200" Height="60" Stroke="Black" />

</Canvas>

<Canvas Grid.Column ="1" Width="300" Margin="10" >

<Rectangle Width="200" Height="60" Fill="BlanchedAlmond">

<Rectangle.RenderTransform >

<SkewTransform AngleX="20" CenterY="0"/>

</Rectangle.RenderTransform>

</Rectangle>

<Rectangle Width="200" Height="60" Stroke="Black" />

</Canvas>

</Grid>

## Listing 6.6 Translation über Matrixtransformation mit Referenzrechteck

<Rectangle Width="200" Height="60" Fill="BlanchedAlmond">

<Rectangle.RenderTransform >

<MatrixTransform>

<MatrixTransform.Matrix>

<Matrix OffsetX="30" />

</MatrixTransform.Matrix>

</MatrixTransform>

</Rectangle.RenderTransform>

</Rectangle>

<Rectangle Width="200" Height="60" Stroke="Black" />

## Listing 6.7 »hallo Welt« gespiegelt

<TextBlock Grid.Row="1" Grid.Column="1" Height="28" >

<Run Text="hallo Welt"/>

</TextBlock>

<TextBlock Grid.Column="1" Grid.Row="1"

Height="28"

RenderTransformOrigin="0.5,0.5" Opacity="1" >

<TextBlock.Foreground>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF000000"/>

<GradientStop Color="#00010101" Offset="1"/>

</LinearGradientBrush>

</TextBlock.Foreground>

<TextBlock.RenderTransform>

<TransformGroup>

<ScaleTransform ScaleY="-0.75" ScaleX="1"/>

<SkewTransform AngleX="-52"/>

<RotateTransform/>

<TranslateTransform Y="-3" X="1"/>

</TransformGroup>

</TextBlock.RenderTransform>

<Run Text="hallo Welt"/>

</TextBlock>

## Listing 6.8 XAML-Code für eine Spiegelung einer Grafik

<Canvas x:Name="LayoutRoot" Background="White">

<Image Canvas.Left="200" Canvas.Top="10"

Source="sl buch.png" />

<Image Source="sl buch.png"

Canvas.Left="222.333" Canvas.Top="-7.3"

RenderTransformOrigin="0.5,0.5" >

<Image.OpacityMask>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#01FFFFFF"/>

<GradientStop Color="#FFFFFFFF" Offset="1"/>

</LinearGradientBrush>

</Image.OpacityMask>

<Image.RenderTransform>

<TransformGroup>

<ScaleTransform ScaleY="-0.65"/>

<SkewTransform AngleX="-39" AngleY="0"/>

<RotateTransform/>

<TranslateTransform Y="286" X="-108"/>

</TransformGroup>

</Image.RenderTransform>

</Image>

</Canvas>

## Listing 6.9 *PointAnimation* in C#-Code

1 private void CreateAndStartPointAnimation()

2 {

3 EllipseGeometry ellipseGeometry

4 = new EllipseGeometry();

5 ellipseGeometry.RadiusX = 100;

6 ellipseGeometry.RadiusY = 100;

7 Path path = new Path();

8 path.Fill = new SolidColorBrush( Colors.Blue);

9 path.Data = ellipseGeometry;

10 LayoutRoot.Children.Add(path);

11

12 Duration duration = new Duration(TimeSpan.FromSeconds(6));

13 Storyboard sb = new Storyboard();

14

15 PointAnimation pointAnimation = new PointAnimation();

16 pointAnimation.From = new Point(10, 10);

17 pointAnimation.To = new Point(300, 300);

18 pointAnimation.Duration = duration;

19

20 sb.Children.Add(pointAnimation);

21 Storyboard.SetTarget(pointAnimation, ellipseGeometry);

22 Storyboard.SetTargetProperty(pointAnimation,

23 new PropertyPath("(Center)"));

24

25 LayoutRoot.Resources.Add("theCStoryBoard", sb);

26 sb.Begin();

27 }

## Listing 6.10 *ColorAnimation* in C#-Code

1 private void CreateAndStartColorAnimation()

2 {

3 Rectangle theRectangle = new Rectangle();

4 theRectangle.Width = 200;

5 theRectangle.Height = 200;

6 //Wichtig, das Objekt.Fill muss (!) ein SolidColorBrush-Objekt haben

7 //sonst kann der PropertyPath nicht aufgelöst werden.

8 //Die Farbe dagegen ist egal.

9 theRectangle.Fill = new SolidColorBrush(Colors.Blue);

10 LayoutRoot.Children.Add(theRectangle);

11

12 Duration duration = new Duration(TimeSpan.FromSeconds(2));

13 ColorAnimation colorAnimation = new ColorAnimation();

14 colorAnimation.From = Colors.Blue;

15 colorAnimation.To = Color.FromArgb(55, 12, 44, 88);

16 Storyboard sb = new Storyboard();

17 sb.Duration = duration;

18 sb.Children.Add(colorAnimation);

19 Storyboard.SetTarget(colorAnimation, theRectangle);

20 Storyboard.SetTargetProperty(colorAnimation,

21 new PropertyPath ("(Fill).(SolidColorBrush.Color)"));

22

23 LayoutRoot.Resources.Add("theStoryboard", sb);

24 sb.Begin();

25 }

## Listing 6.11 DoubleAnimation in C#-Code

1 private void CreateAndStartDoubleAnimation()

2 {

3 Rectangle theRectangle = new Rectangle();

4 theRectangle.Width = 100;

5 theRectangle.Height = 100;

6 theRectangle.Fill = new SolidColorBrush(Colors.LightGray);

7 LayoutRoot.Children.Add(theRectangle);

8 theRectangle.SetValue(Canvas.LeftProperty, 10.0);

9 theRectangle.SetValue(Canvas.TopProperty,10.0);

10

11 Duration duration = new Duration(TimeSpan.FromSeconds(4));

12 Storyboard sb = new Storyboard();

13 sb.Duration = duration;

14

15 DoubleAnimation doubleAnimationLeft = new DoubleAnimation();

16 DoubleAnimation doubleAnimationTop = new DoubleAnimation();

17

18 //doubleAnimationLeft.Duration = duration;

19 doubleAnimationTop.Duration = duration + duration;

20

21 sb.Children.Add(doubleAnimationLeft);

22 sb.Children.Add(doubleAnimationTop);

23

24 Storyboard.SetTarget(doubleAnimationLeft, theRectangle);

25 Storyboard.SetTarget(doubleAnimationTop, theRectangle);

26

27 Storyboard.SetTargetProperty(doubleAnimationLeft,

28 new PropertyPath("(Canvas.Left)"));

29 Storyboard.SetTargetProperty(doubleAnimationTop,

30 new PropertyPath("(Canvas.Top)"));

31

32 doubleAnimationLeft.To = 300;

33 doubleAnimationTop.To = 300;

34

35 LayoutRoot.Resources.Add("theStoryBoard", sb);

36 sb.Begin();

37 }

## Listing 6.12 Diskrete Animation mit *DoubleAnimationUsingKeyFrames*

<Canvas>

<Ellipse x:Name="elli"

Fill="AntiqueWhite" Stroke="Blue" StrokeThickness="1" Width="30" Height="30" />

<Path Stroke="Red" Fill="AntiqueWhite">

<Path.Data>

<EllipseGeometry x:Name="rund" RadiusX="30" RadiusY="30" Center="100,100" />

</Path.Data>

</Path>

<Canvas.Triggers>

<EventTrigger RoutedEvent="Canvas.Loaded">

<EventTrigger.Actions>

<BeginStoryboard>

<Storyboard >

<Storyboard.TargetName>rund</Storyboard.TargetName>

<DoubleAnimationUsingKeyFrames Duration="00:00:05"

Storyboard.TargetProperty="RadiusX">

<DiscreteDoubleKeyFrame KeyTime="00:00:00.5" Value="80" />

<DiscreteDoubleKeyFrame KeyTime="00:00:01" Value="120" />

<DiscreteDoubleKeyFrame KeyTime="00:00:02" Value="160" />

<DiscreteDoubleKeyFrame KeyTime="00:00:04" Value="260" />

</DoubleAnimationUsingKeyFrames>

</Storyboard>

</BeginStoryboard>

<!-- -->

<BeginStoryboard>

<Storyboard >

<Storyboard.TargetName>elli</Storyboard.TargetName>

<DoubleAnimationUsingKeyFrames Duration="00:00:05"

Storyboard.TargetProperty="(Canvas.Left)">

<DiscreteDoubleKeyFrame KeyTime="00:00:00.5" Value="80" />

<DiscreteDoubleKeyFrame KeyTime="00:00:01" Value="120" />

<DiscreteDoubleKeyFrame KeyTime="00:00:02" Value="160" />

<DiscreteDoubleKeyFrame KeyTime="00:00:04" Value="260" />

</DoubleAnimationUsingKeyFrames>

</Storyboard>

</BeginStoryboard>

</EventTrigger.Actions>

</EventTrigger>

</Canvas.Triggers>

</Canvas>

## Listing 6.13 Lineare Animation mit *DoubleAnimationUsingKeyFrames*

<!-- Linear Bewegen -->

<BeginStoryboard>

<Storyboard >

<Storyboard.TargetName>elliLiniear</Storyboard.TargetName>

<DoubleAnimationUsingKeyFrames Duration="00:00:05" Storyboard.TargetProperty="(Canvas.Left)">

<LinearDoubleKeyFrame KeyTime="00:00:00.5" Value="80" />

<LinearDoubleKeyFrame KeyTime="00:00:01" Value="120" />

<LinearDoubleKeyFrame KeyTime="00:00:02" Value="160" />

<LinearDoubleKeyFrame KeyTime="00:00:04" Value="260" />

</DoubleAnimationUsingKeyFrames>

</Storyboard>

</BeginStoryboard>

## Listing 6.14 Animiertes Hintergrundbild in einer *TextBox* (C#)

public partial class Hintergrundbild\_in\_TextBox : UserControl

{

bool storyBoardActive = false;

public Hintergrundbild\_in\_TextBox()

{

InitializeComponent();

Loaded += new RoutedEventHandler(OnPageLoaded);

}

void OnPageLoaded(object sender, RoutedEventArgs e)

{

StoryboardDisapear .Completed

+= new EventHandler(StoryboardDisapear\_Completed);

}

private void OnVornameTextChanged(object sender,

TextChangedEventArgs e)

{

TextBox tb = sender as TextBox;

if (tb.Text != "")

{

if (!storyBoardActive)

{

storyBoardActive = true;

StoryboardDisapear.Begin();

}

}

else

{

StoryboardApear.Begin();

}

}

void StoryboardDisapear\_Completed(object sender, EventArgs e)

{

storyBoardActive = false;

}

private void OnVornameTextChanged(object sender,

TextChangedEventArgs e)

{

TextBox tb = sender as TextBox;

if (tb.Text != "")

{

StoryboardDisapear.Begin();

}

else

{

StoryboardApear.Begin();

}

}

}

## Listing 6.15 Animiertes Hintergrundbild in einer *TextBox* (XAML)

<TextBox x:Name="tb" Height="28" Width="80" TextChanged="OnVornameTextChanged">

<TextBox.Background>

<ImageBrush x:Name="image"

ImageSource="LabelVorname.png"

AlignmentX="Left" Stretch="UniformToFill" >

<ImageBrush.Transform>

<TransformGroup>

<ScaleTransform/>

<SkewTransform/>

<RotateTransform/>

<TranslateTransform/>

</TransformGroup>

</ImageBrush.Transform>

</ImageBrush>

</TextBox.Background>

<ToolTipService.ToolTip>

<TextBlock>

Die blaue Schrift ist ein Bild im Hintergrund

<LineBreak/>

Sobald der Anwender seine Eingabe beginnt,

<LineBreak/>

verschwindet der Text.

</TextBlock>

</ToolTipService.ToolTip>

</TextBox>

# Kapitel 7

## Keine Listings

# Kapitel 8

## Listing 8.1 Speichern mit *SaveFileDialog*

private void OnSpeichernClick(object sender, RoutedEventArgs e)

{

SaveFileDialog sfd = new SaveFileDialog();

if ((bool)sfd.ShowDialog())

{

string name = sfd.SafeFileName;

Stream stream = sfd.OpenFile();

StreamWriter streamWriter = new StreamWriter(stream);

streamWriter.WriteLine("Hallo Welt");

streamWriter.Flush();

streamWriter.Close();

}

}

## Listing 8.2 Verwendung des *SaveFileDialog*-Filters

public void OnOpenAllPictureFiles(object sender, RoutedEventArgs e)

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter

= "Alle Bilder ( JPEG,GIF,BMP,PNG,ICO )|\*.jpg;\*.jpeg;\*.gif;\*.bmp;\*.png;\*ico|"

+ "JPEG Dateien ( \*.jpg;\*.jpeg )|\*.jpg;\*.jpeg|GIF Dateien ( \*.gif )|\*.gif|"

+ "BMP Dateien ( \*.bmp )|\*.bmp|PNG Dateien ( \*.png )|\*.png|"

+ "Icons ( \*.ico )|\*.ico";

bool?b = saveFileDialog.ShowDialog();

if (b == true)

{

// ...

}

}

## Listing 8.3 *OpenFileDialog* in der Anwendung

private void OnReadClick(object sender, RoutedEventArgs e)

{

OpenFileDialog ofd = new OpenFileDialog();

if ((bool)ofd.ShowDialog())

{

FileStream fileStream = ofd.File.OpenRead();

StreamReader reader = new StreamReader(fileStream);

string read = string.Empty;

StringBuilder sb = new StringBuilder();

while ((read = reader.ReadLine()) != string.Empty)

{

sb.Append(read + Environment.NewLine);

}

read = sb.ToString();

// ... Text (ver-) bearbeiten ...

}

}

## Listing 8.4 Codebehind-Datei der XAML-Datei aus Listing 8.5

public partial class IsolatedStorageUserControl : UserControl

{

StorageUsage su;

public IsolatedStorageUserControl()

{

InitializeComponent();

su = new StorageUsage();

LayoutRoot.DataContext = su;

BtnIncreaseQuota.Click += su.OnIncreaseQuota;

}

}

## Listing 8.5 XAML-Code des UserControls *IsolatedStorageUserControl*

<Grid x:Name="LayoutRoot" Background="White" Margin="10">

<Grid >

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition Height="Auto" />

<RowDefinition Height="Auto" />

<RowDefinition Height="Auto" />

<RowDefinition Height="Auto" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="2\*" />

</Grid.ColumnDefinitions>

<dataInput:Label Grid.Column="0" Grid.Row="0" Content="Verwendeter Speicher" />

<dataInput:Label Grid.Column="0" Grid.Row="1" Content="Verfügbarer Speicher" />

<dataInput:Label Grid.Column="0" Grid.Row="2" Content="Aktuelle Quota" />

<TextBlock Grid.Column="1" Grid.Row="0" Margin="3"

Text="{Binding Path=SpacedUsed}" />

<TextBlock Grid.Column="1" Grid.Row="1" Margin="3"

Text="{Binding Path=SpaceAvailable}" />

<TextBlock Grid.Column="1" Grid.Row="2" Margin="3"

Text="{Binding Path=CurrentQuota}" />

<Grid Grid.ColumnSpan="2" Grid.Row="3" Margin="3">

<Grid.ColumnDefinitions >

<ColumnDefinition />

<ColumnDefinition />

<ColumnDefinition />

</Grid.ColumnDefinitions>

<dataInput:Label Content="Erhöhen um:" />

<TextBox Grid.Column="1" Width="100" Margin="5 0 5 0"

Text="{Binding Path=SpaceRequest, Mode=TwoWay}"/>

<Button x:Name="BtnIncreaseQuota" Grid.Column="2"

Height="25" Width="100" Content="Quota erhöhen" />

</Grid>

<TextBlock Grid.Row="4" Grid.ColumnSpan="2" TextWrapping="Wrap"

Text="{Binding Path=Message}" />

</Grid>

</Grid>

## Listing 8.6 Ausnahme IncreaseFailedException

public class IncreaseFailedException : Exception

{

}

## Listing 8.7 *StorageUsage*

1 public class StorageUsage:INotifyPropertyChanged

2 {

3

4 long spacedUsed;

5 public long SpacedUsed

6 {

7 get { return spacedUsed; }

8 set { spacedUsed = value;

9 OnPropertyChanged("SpacedUsed");

10 }

11 }

12

13 long spaceAvailable;

14 public long SpaceAvailable

15 {

16 get { return spaceAvailable; }

17 set { spaceAvailable = value;

18 OnPropertyChanged("SpaceAvailable");

19 }

20 }

21

22 long currentQuota;

23 public long CurrentQuota

24 {

25 get { return currentQuota; }

26 set { currentQuota = value;

27 OnPropertyChanged("CurrentQuota");

28 }

29 }

30

31 long spaceRequest;

32

33 public long SpaceRequest

34 {

35 get { return spaceRequest; }

36 set { spaceRequest = value;

37 OnPropertyChanged("SpaceRequest");

38 }

39 }

40 string message;

41

42 public string Message

43 {

44 get { return message; }

45 set { message = value;

46 OnPropertyChanged("Message");

47 }

48 }

49

50 public event PropertyChangedEventHandler PropertyChanged;

51 public void OnPropertyChanged(string propertyName)

52 {

54 PropertyChanged (this,

55 new PropertyChangedEventArgs(propertyName));

56 }

57

58

59 public StorageUsage()

60 {

61 using (IsolatedStorageFile isf

62 = IsolatedStorageFile.GetUserStoreForApplication())

63 {

64 CurrentQuota = isf.Quota;

65 SpaceAvailable = isf.AvailableFreeSpace;

66 SpacedUsed = isf.Quota - isf.AvailableFreeSpace;

67 SpaceRequest = -2;

68 }

69 }

70

71 public void OnIncreaseQuota(object sender, RoutedEventArgs e)

72 {

73 using (IsolatedStorageFile isf

74 = IsolatedStorageFile.GetUserStoreForApplication())

75 {

76 try

77 {

78 long newSpace = SpaceRequest + currentQuota;

79 if (!isf.IncreaseQuotaTo(newSpace))

80 throw new IncreaseFailedException();

81 Message = " Speicher erhöht auf: " + isf.Quota.ToString();

82 CurrentQuota = isf.Quota;

83 SpaceAvailable = isf.AvailableFreeSpace;

84 SpacedUsed = isf.Quota - isf.AvailableFreeSpace;

85 }

86 catch (Exception fehler)

87 {

88 Message = fehler.Message;

89 }

90 }

91 }

92 public void RemoveStorage(object sender, RoutedEventArgs e)

93 {

94 using (IsolatedStorageFile isf

95 = IsolatedStorageFile.GetUserStoreForApplication())

96 {

97 isf.Remove();

98 }

99 using (IsolatedStorageFile isf

100 = IsolatedStorageFile.GetUserStoreForApplication())

101 {

102 CurrentQuota = isf.Quota;

103 SpaceAvailable = isf.AvailableFreeSpace;

104 SpacedUsed = isf.Quota - isf.AvailableFreeSpace;

105 }

106 }

107

108 }

## Listing 8.8 Die Klasse *PageSettings* implementiert *INotifyPropertyChanged*

public class PageSettings:INotifyPropertyChanged

{

double links;

double oben;

Color farbe;

Brush pinsel;

string text;

public double Links

{

get { return links; }

set { links = value;

OnChanged("Links");

}

}

public double Oben

{

get { return oben; }

set { oben = value;

OnChanged("Oben");

}

}

public Color Farbe

{

get { return farbe; }

set { farbe = value;

this.Pinsel = new SolidColorBrush(farbe);

OnChanged("Farbe");

}

}

public Brush Pinsel

{

get { return pinsel; }

set { pinsel = value;

OnChanged("Pinsel");

}

}

public string Text

{

get { return text; }

set { text = value;

OnChanged("Text");

}

}

public event PropertyChangedEventHandler PropertyChanged;

protected void OnChanged(string propertyName)

{

if (PropertyChanged != null)

PropertyChanged(this,

new PropertyChangedEventArgs(propertyName));

}

}

# Kapitel 9

## Listing 9.1 Klasse *Person*

public class Person

{

int personID;

public int PersonID

{

get

{

return personID;

}

}

public string Vorname { get; set; }

public string Nachname { get; set; }

public DateTime Geburtsdatum { get; set; }

public Person() { }

}

## Listing 9.2 Beispiel einer Abhängigkeitseigenschaft

public static readonly DependencyProperty IsSpinningProperty =

DependencyProperty.Register(

"IsSpinning", typeof(Boolean),

typeof(MyCode), null

);

public bool IsSpinning

{

get { return (bool)GetValue(IsSpinningProperty); }

set { SetValue(IsSpinningProperty, value); }

}

## Listing 9.3 XAML-Bindungscode

<TextBlock Grid.Column="1" Grid.Row="0"

x:Name="tbPersonID"

Text="{Binding Path=PersonID}" />

<TextBox Grid.Column="1" Grid.Row="1"

Text="{Binding Path=Vorname, Mode=TwoWay, NotifyOnValidationError=True,

ValidatesOnExceptions=True}"

Width="200" HorizontalAlignment="Left" VerticalAlignment="Top" />

<TextBox Grid.Column="1" Grid.Row="2"

x:Name="tbNachname"

Text="{Binding Path=Nachname, Mode=TwoWay, NotifyOnValidationError=True,

ValidatesOnExceptions=True}"

Width="200" HorizontalAlignment="Left" VerticalAlignment="Top" />

<controls:DatePicker Grid.Column="1" Grid.Row="3" Width="150" HorizontalAlignment="Left"

SelectedDate="{Binding Path=Geburtsdatum, Mode=TwoWay}"

SelectedDateFormat="Short" />

## Listing 9.4 Datenkontext im Codebehind der XAML-Datei setzten

public partial class MainPage : UserControl

{

Person person;

public MainPage()

{

InitializeComponent();

person = Person.GetPerson();

this.DataContext = person;

}

}

## Listing 9.5 Die Eigenschaft *Vorname* der Klasse *Person*

private string vorname;

public string Vorname

{

get { return vorname; }

set

{

vorname = value;

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs("Vorname"));

}

}

## Listing 9.6 Klasse *Person* implementiert *INotifyPropertyChanged*

1 public class Person:INotifyPropertyChanged

2 {

3 int personID;

4 public int PersonID

5 {

6 get

7 {

8 return personID;

9 }

10 }

11

12 string vorname;

13 public string Vorname

14 {

15 get { return vorname; }

16 set

17 {

18 vorname = value;

19 if (PropertyChanged != null)

20 PropertyChanged(this, new PropertyChangedEventArgs("Vorname"));

21 }

22 }

23

24 string nachname;

25 public string Nachname

26 {

27 get { return nachname; }

28 set { nachname = value; }

29 }

30

31 DateTime geburtsdatum;

32 public DateTime Geburtsdatum

33 {

34 get { return geburtsdatum; }

35 set { geburtsdatum = value; }

36 }

37

38 public event PropertyChangedEventHandler PropertyChanged;

39 public void OnPropertyChanged(string propertyName)

40 {

41 if (PropertyChanged != null)

42 PropertyChanged(this, new pertyChangedEventArgs(propertyName));

43 }

44

45 public Person() { }

46

47 public static Person GetPerson()

48 {

49 return new Person()

50 {

51 personID = 1,

52 Vorname = "Hans",

53 Nachname = "Dampf",

54 Geburtsdatum = new DateTime(1972, 1, 24)

55 };

56 }

57

58 public void Save()

59 {

60 }

61 }

## Listing 9.7 Zeitgeber; implementiert *INotifyPropertyChanged*

public class Zeitgeber:INotifyPropertyChanged

{

private DispatcherTimer timer;

private bool isEnabled;

public bool IsEnabled

{

get { return isEnabled; }

set

{

isEnabled = value;

if (isEnabled)

StartTimer();

else

StopTimer();

OnPropertyChanged("IsEnabled");

}

}

public DateTime Time

{

get

{

return DateTime.Now;

}

}

public Zeitgeber()

{

timer = new DispatcherTimer();

timer.Interval = TimeSpan.FromSeconds(1);

timer.Tick += new EventHandler(timer\_Tick);

}

void timer\_Tick(object sender, EventArgs e)

{

OnPropertyChanged("Time");

}

private void StartTimer()

{

timer.Start();

}

private void StopTimer()

{

timer.Stop();

}

public event PropertyChangedEventHandler PropertyChanged;

void OnPropertyChanged(string propertyName)

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

## Listing 9.8 XAML-Oberfläche der Zeitgeberapplikation

<StackPanel Width="300" Margin="10" >

<TextBlock Text="Dauer" />

<TextBlock Text="Zeit" />

<TextBox Text="{Binding Time, Mode=OneWay}"

Width="200" Height="25" />

<CheckBox Grid.Column="1" Grid.Row="2" IsChecked="{Binding IsEnabled, Mode=TwoWay}"

Content="Enabled"/>

</StackPanel>

## Listing 9.9 Datenobjekte als Ressource in XAML

<UserControl

x:Class="Datenbindung.MainPage"

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"

xmlns:loc="clr-namespace:Datenbindung">

<Grid x:Name="LayoutRoot">

<Grid.Resources>

<loc:Person x:Key="person" Nachname="Duck" Vorname="Donald"/>

</Grid.Resources>

<StackPanel>

<TextBlock Text="Vorname" />

<TextBox Text="{Binding Vorname, Source={StaticResource person}}" />

<TextBlock Text="Nachname" />

<TextBox Text="{Binding Nachname, Source={StaticResource person}}" />

</StackPanel>

</Grid>

</UserControl>

## Listing 9.10 Relative Bindung

<Button Content="Klick mich" >

<Button.Template>

<ControlTemplate>

<StackPanel>

<TextBlock

Text="{Binding Path=Content, RelativeSource={RelativeSource TemplatedParent}}" />

</StackPanel>

</ControlTemplate>

</Button.Template>

</Button>

## Listing 9.11 XAML der Element-Element-Bindung

<StackPanel Margin="10">

<Slider x:Name="slider" Minimum="10" Maximum="250" Width="300" Margin="5"

Value="{Binding Path=Text, ElementName=txtBx, Mode=TwoWay}" />

<TextBox x:Name="txtBx"

Text="{Binding Path=Value, Mode=TwoWay, ElementName=slider}" Width="200" />

<Rectangle Width="{Binding Path=Value, ElementName=slider}"

Height="{Binding Path=Value, ElementName=slider}"

Fill="Bisque" Margin="5"/>

<Button Width="{Binding Path=Value, ElementName=slider}"

Height="{Binding Path=Value, ElementName=slider}"

Margin="5"/>

</StackPanel>

## Listing 9.12 BoolToColorConverter implementiert IValueConverter

public class BoolToColorConverter:IValueConverter

{

public object Convert(object value, Type targetType, object parameter,

System.Globalization.CultureInfo culture)

{

if ((value == null) || (bool)value)

return new SolidColorBrush(Colors.Yellow);

else

return new SolidColorBrush(Colors.LightGray);

}

public object ConvertBack(object value, Type targetType,

object parameter, System.Globalization.CultureInfo culture)

{

throw new NotImplementedException();

}

}

## Listing 9.13 Konverter im *ResourceDictionary* von *LayoutRoot*

<Grid x:Name="LayoutRoot">

<Grid.Resources>

<loc:BoolToColorConverter x:Key="boolToColorConverter" />

</Grid.Resources>

...

## Listing 9.14 XAML-Code der *CheckBox* mit Wertumwandlung

<CheckBox x:Name="xk" Margin="10">

<Ellipse Width="100" Height="100"

Fill="{Binding Path=IsChecked, ElementName=xk, Converter={StaticResource

boolToColorConverter}}"

StrokeThickness="1"/>

</CheckBox>

<StackPanel>

<CheckBox x:Name="checker" />

<Ellipse Width="100" Height="100"

Fill="{Binding Path=IsChecked, ElementName=checker, Converter={StaticResource

boolToColorConverter}}"

Stroke="Black" StrokeThickness="1"/>

</StackPanel>

## Listing 9.15 Datenklasse *Person*

public class Person

{

public string Vorname { get; set; }

public string Nachname { get; set; }

public DateTime Geboren { get; set; }

public static List<Person> GetPersonen()

{

return new List<Person>()

{

new Person(){Vorname="Donald", Nachname="Duck", Geboren=new DateTime(1952,1,1)},

new Person(){Vorname="Dagobert", Nachname="Duck", Geboren=new DateTime(1951,2,1)},

new Person(){Vorname="Tick", Nachname="Duck", Geboren=new DateTime(1959,12,1)},

new Person(){Vorname="Trick", Nachname="Duck", Geboren=new DateTime(1959,12,1)},

new Person(){Vorname="Track", Nachname="", Geboren=new DateTime(1959,12,1)},

};

}

}

## Listing 9.16 ListBox mit DisplayMemberPath="Vorname"

<ListBox x:Name="listBox" DisplayMemberPath="Vorname" Margin="10"

Width="200" Height="150" >

</ListBox>

## Listing 9.17 *ListBox* mit eigenem *ItemTemplate*

<ListBox x:Name="listBox" Margin="10" Width="200" Height="250" >

<ListBox.ItemTemplate>

<DataTemplate>

<Border BorderBrush="Red" BorderThickness="2">

<StackPanel >

<Border Background="AntiqueWhite" Margin="3" BorderBrush="Beige"

BorderThickness="3">

<TextBlock Text="{Binding Vorname}" />

</Border>

<TextBlock Text="{Binding Nachname}" Margin="3"/>

<TextBlock Text="{Binding Geboren}" Margin="3"/>

</StackPanel>

</Border>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

## Listing 9.18 *ListBox* mit horizontalem *ItemsPanel*

<ListBox x:Name="listBox" Margin="10" Width="500" Height="120" >

<ListBox.ItemTemplate>

<DataTemplate>

<Border BorderBrush="Red" BorderThickness="2">

<StackPanel >

<Border Background="AntiqueWhite" Margin="3" BorderBrush="Beige"

BorderThickness="3">

<TextBlock Text="{Binding Vorname}" />

</Border>

<TextBlock Text="{Binding Nachname}" Margin="3"/>

<TextBlock Text="{Binding Geboren}" Margin="3"/>

</StackPanel>

</Border>

</DataTemplate>

</ListBox.ItemTemplate>

<!-- ItemsPanel -->

<ListBox.ItemsPanel>

<ItemsPanelTemplate>

<StackPanel Orientation="Horizontal" />

</ItemsPanelTemplate>

</ListBox.ItemsPanel>

</ListBox>

## Listing 9.19 ObservableCollection<Person> GetPersonen()

public static ObservableCollection<Person> GetPersonen()

{

return new ObservableCollection<Person>()

{

new Person(){Vorname="Donald", Nachname="Duck", Geboren=new DateTime(1952,1,1)},

new Person(){Vorname="Dagobert", Nachname="Duck", Geboren=new DateTime(1951,2,1)},

new Person(){Vorname="Tick", Nachname="Duck", Geboren=new DateTime(1959,12,1)},

new Person(){Vorname="Trick", Nachname="Duck", Geboren=new DateTime(1959,12,1)},

new Person(){Vorname="Track", Nachname="Duck", Geboren=new DateTime(1959,12,1)},

};

}

## Listing 9.20 Erweiterungsmethode zum Umwandeln einer *IEnumerable<T>*-Auflistung in eine *ObservableCollection<T>*

public static class CollectionExtensions

{

public static ObservableCollection<T> ToObservableCollection<T>(this IEnumerable<T> coll)

{

var c = new ObservableCollection<T>();

foreach (var e in coll)

c.Add(e);

return c;

}

}

## Listing 9.21 LINQ liefert einen Teil der Daten zurück

public IEnumerable<Person> Neffen()

{

var qry = from person in GetPersonen()

where person.Geboren.Year > 1955

select person;

return qry;

}

## Listing 9.22 IntToBrushConverter, ein IValueConverter

public class IntToBrushConverter : IValueConverter

{

public object Convert(object value,

Type targetType,

object parameter,

System.Globalization.CultureInfo culture)

{

if (value == null) return null;

int alter = (int)value;

if (alter < 18)

return new SolidColorBrush(Colors.Red);

else

return null;

}

public object ConvertBack(object value,

Type targetType,

object parameter,

System.Globalization.CultureInfo culture)

{

throw new NotImplementedException();

}

}

## Listing 9.23 DataGrid mit DataTemplate und IValueConverter

<data:DataGrid x:Name="dataGrid"

AutoGenerateColumns="False">

<data:DataGrid.Columns>

<data:DataGridTextColumn Header="Vorname"

Binding="{Binding Vorname}" />

<data:DataGridTextColumn Header="Nachname"

Binding="{Binding Nachname}" />

<data:DataGridTemplateColumn Header="Alter">

<data:DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Border Background="{Binding Alter,

Converter={StaticResource intToBrushConverter}}">

<TextBlock Text="{Binding Alter}" Margin="2" />

</Border>

</DataTemplate>

</data:DataGridTemplateColumn.CellTemplate>

</data:DataGridTemplateColumn>

</data:DataGrid.Columns>

</data:DataGrid

## Listing 9.24 Klasse *Anwender* mit den Annotationen *Required*, *DataType* und *Display*

public class Anwender : INotifyPropertyChanged

{

string userName;

[Required]

[StringLength(10, ErrorMessage="Bitte nur 10 Zeichen eingeben")]

[Display(Name = "Anwendername",

Description = "Bitte einen Anwendernamen mit weniger als 10 Zeichen")]

public string UserName

{

get { return userName; }

set

{

Validator.ValidateProperty(value,

new ValidationContext(this, null, null)

{ MemberName = "UserName" });

userName = value;

NotifyPropertyChanged("UserName");

}

}

DateTime userSince;

[Required]

[DataType(DataType.DateTime)]

[Display(Name = "Anwender seit...",

Description = "Seit wann ist der User Mitglied")]

public DateTime UserSince

{

get { return userSince; }

set { userSince = value;

NotifyPropertyChanged("UserSince");

}

}

public event PropertyChangedEventHandler PropertyChanged;

public void NotifyPropertyChanged(string property)

{

if (PropertyChanged != null)

{

PropertyChanged(this,

new PropertyChangedEventArgs(property));

}

}

}

## Listing 9.25 *LayoutRoot.DataContext* mit einem *user*-Objekt verknüpfen

public partial class MainPage : UserControl

{

public MainPage()

{

InitializeComponent();

Anwender user

= new Anwender() { UserName = "Bernd24" };

LayoutRoot.DataContext = user;

}

}

## Listing 9.26 XAML-Code zum Einsatz von *ValidationSummary*

<UserControl

xmlns:dataInput="clr-namespace:System.Windows.Controls;assembly=System.Windows.Controls.Data.Input"

x:Class="SilverlightApplication1.MainPage"

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" >

<Grid x:Name="LayoutRoot">

<StackPanel Margin="20">

<StackPanel Orientation="Horizontal">

<dataInput:Label Margin="0 0 10 0"

Target="{Binding ElementName=tbUserName}" />

<TextBox x:Name="tbUserName"

Text="{Binding UserName, Mode=TwoWay,

ValidatesOnExceptions=true,

NotifyOnValidationError=true}"/>

<dataInput:DescriptionViewer

Target="{Binding ElementName=tbUserName}" />

</StackPanel>

<StackPanel Orientation="Horizontal">

<dataInput:Label Margin="0 0 10 0"

Target="{Binding ElementName=tbUserSince}" />

<TextBox x:Name="tbUserSince"

Text="{Binding UserSince, Mode=TwoWay,

ValidatesOnExceptions=true,

NotifyOnValidationError=true}"/>

<dataInput:DescriptionViewer

Target="{Binding ElementName=tbUserSince}" />

</StackPanel>

<dataInput:ValidationSummary />

</StackPanel>

</Grid>

</UserControl>

# Kapitel 10

## Listing 10.1 *AppManifest*.*xaml* aus der XAP-Datei im *ClientBin*-Verzeichnis der Webapplikation – ohne Assemblycache

<Deployment xmlns="http://schemas.microsoft.com/client/2007/deployment"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

EntryPointAssembly="Assembly\_Cache\_Demo\_OhneCache" EntryPointType="Assembly\_Cache\_Demo\_OhneCache.App"

RuntimeVersion="3.0.40307.0">

<Deployment.Parts>

<AssemblyPart x:Name="Assembly\_Cache\_Demo\_OhneCache" Source="Assembly\_Cache\_Demo\_OhneCache.dll" />

<AssemblyPart x:Name="System.ComponentModel.DataAnnotations"

Source="System.ComponentModel.DataAnnotations.dll" />

<AssemblyPart x:Name="System.ComponentModel" Source="System.ComponentModel.dll" />

<AssemblyPart x:Name="System.Json" Source="System.Json.dll" />

<AssemblyPart x:Name="System.Windows.Ria" Source="System.Windows.Ria.dll" />

<AssemblyPart x:Name="System.Xml.Linq" Source="System.Xml.Linq.dll" />

<AssemblyPart x:Name="System.Xml.Serialization" Source="System.Xml.Serialization.dll" />

<AssemblyPart x:Name="System.Xml.Utils" Source="System.Xml.Utils.dll" />

</Deployment.Parts>

<!-- ... -->

</Deployment>

## Listing 10.2 *AppManifest.xaml* aus der XAP-Datei im *ClientBin*-Verzeichnis der Webapplikation – mit Assemblycache

<Deployment xmlns="http://schemas.microsoft.com/client/2007/deployment"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" EntryPointAssembly="Assembly\_Cache\_Demo"

EntryPointType="Assembly\_Cache\_Demo\_OhneCache.App" RuntimeVersion="3.0.40307.0">

<Deployment.Parts>

<AssemblyPart x:Name="Assembly\_Cache\_Demo" Source="Assembly\_Cache\_Demo.dll" />

<AssemblyPart x:Name="System.Windows.Ria" Source="System.Windows.Ria.dll" />

</Deployment.Parts>

<Deployment.ExternalParts>

<ExtensionPart Source="http://go.microsoft.com/fwlink/?LinkID=142565" />

<ExtensionPart Source="http://go.microsoft.com/fwlink/?LinkID=141727" />

<ExtensionPart Source="http://go.microsoft.com/fwlink/?LinkId=142567" />

<ExtensionPart Source="http://go.microsoft.com/fwlink/?LinkId=142576" />

<ExtensionPart Source="http://go.microsoft.com/fwlink/?LinkId=142577" />

<ExtensionPart Source="http://go.microsoft.com/fwlink/?LinkId=142578" />

</Deployment.ExternalParts>

<!-- ... -->

</Deployment>

## Listing 10.3 Einrichten eines Webclients zum Download einer Abbildung

1 private void OnLoadImage(object sender, RoutedEventArgs e)

2 {

3 progressBar.Value = 0.0;

4 progressBar.Visibility = Visibility.Visible;

5 WebClient wc = new WebClient();

6 Uri uri

= new Uri("../Images/hs-2005-02-d-full\_tif.jpg",

UriKind.Relative);

7 wc.OpenReadAsync(uri);

8 wc.OpenReadCompleted

9 += new OpenReadCompletedEventHandler(wc\_OpenReadCompleted);

10 wc.DownloadProgressChanged

11 += new DownloadProgressChangedEventHandler(wc\_DownloadProgressChanged);

12 }

## Listing 10.4 Entgegennahme der Daten nach erfolgreichem Download mit einem *WebClient*-Objekt

1 void wc\_OpenReadCompleted(object sender,

OpenReadCompletedEventArgs e)

2 {

3 if (e.Error == null)

4 {

5 BitmapImage bitmap = new BitmapImage();

6 StreamResourceInfo streamRI

= new StreamResourceInfo(e.Result as Stream, null);

7 bitmap.SetSource(streamRI.Stream);

8 Image image = new Image();

9 image.Source = bitmap;

10 image.Stretch = Stretch.Uniform;

11 sp.Children.Add(image);

12 progressBar.Visibility = Visibility.Collapsed;

13 }

14 }

## Listing 10.5 XAML für das Projekt zum Nachladen von Bildern

<UserControl x:Class="Grafik\_Nachladen.MainPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Width="400" Height="300">

<Grid x:Name="LayoutRoot" Background="White">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition/>

</Grid.RowDefinitions>

<StackPanel Orientation="Horizontal">

<Button Content="Image laden"

Margin="5"

Height="28" Width="100"

Click="OnLoadImage" />

<ProgressBar Margin="5" Width="200"

Visibility="Collapsed"

x:Name="progressBar" />

</StackPanel>

<ScrollViewer Grid.Row="1" >

<StackPanel x:Name="sp" />

</ScrollViewer>

</Grid>

</UserControl>

## Listing 10.6 Bild aus einer ZIP-Datei extrahieren

1 void wc\_OpenReadFromZipCompleted(object sender,

OpenReadCompletedEventArgs e)

2 {

3 if (e.Error == null)

4 {

5 BitmapImage bitmap = new BitmapImage();

6

7 StreamResourceInfo streamRI

8 = new StreamResourceInfo(e.Result as Stream, null);

9 StreamResourceInfo appStreamResourceInfo

10 = Application.GetResourceStream(streamRI,

11 new Uri(e.UserState.ToString(), UriKind.Relative));

12

13 bitmap.SetSource(appStreamResourceInfo.Stream);

14 Image image = new Image();

15 image.Source = bitmap;

16 image.Stretch = Stretch.Uniform;

17 double width = image.Width;

18 sp.Children.Add(image);

19

20 progressBar.Visibility = Visibility.Collapsed;

21 }

22 }

## Listing 10.7 Mit einem HTTP-Handler XAML-Code erzeugen

public void ProcessRequest(HttpContext context)

{

context.Response.Expires = -1;

context.Response.ContentType = "text/xaml";

context.Response.Write(

"<Border xmlns= 'http://schemas.microsoft.com/client/2007' "

+

"xmlns:x='http://schemas.microsoft.com/winfx/2006/xaml'>" +

"<TextBlock Foreground='black' Padding='10' FontSize='20'> " +

"<Run>XAML content</Run><LineBreak/>" +

"<Run>[generated " + DateTime.Now.ToLongTimeString() + "]</Run>" +

"</TextBlock></Border>");

}

## Listing 10.8 Delegat OpenReadCompletedEventHandler

void webClient\_OpenReadCompleted(object sender, OpenReadCompletedEventArgs e) {

if (e.Error == null) {

StreamResourceInfo sri = new StreamResourceInfo(e.Result, null);

StreamResourceInfo appSri

= Application.GetResourceStream(sri, new Uri("Zusatz.dll", UriKind.Relative));

AssemblyPart ap = new AssemblyPart();

Assembly a = ap.Load(appSri.Stream);

this.Frame.Content

= a.CreateInstance("Zusatz.MainPage");

}

}

## Listing 10.9 Code nachladen und sofort verwenden

void webClient\_OpenReadCompleted(object sender, OpenReadCompletedEventArgs e)

{

AssemblyPart assemblyPart = new AssemblyPart();

Assembly assembly = assemblyPart.Load(e.Result);

// An dieser Stelle kann Foo nicht instanziiert werden.

// Foo f = new Foo(); führt zu einem Fehler. Aber:

Object f = CreateFoo(); // geht

TextBlock tb = new TextBlock();

tb.Text = SayHelloFoo(tb, f);

LayoutRoot.Children.Add(tb);

}

[MethodImpl(MethodImplOptions.NoInlining)]

private string SayHelloFoo(TextBlock tb, Object f)

{

return ((Foo)f).SayHello();

}

// aus den Namespace System.Runtime.CompilerServices

// verbietet dem Compiler, die Methode Inline zu erzeugen

[MethodImpl(MethodImplOptions.NoInlining)]

static Object CreateFoo()

{

return new Foo();

}

# Kapitel 11

## Listing 11.1 Eine Testklasse

[TestClass]

public class SomeTests:Microsoft.Silverlight.Testing.SilverlightTest

{

[TestMethod]

public void ThisTestFails()

{

Microsoft.VisualStudio.TestTools.UnitTesting.Assert.IsFalse(true);

}

[TestMethod ]

public void ThisTestWorks()

{

Microsoft.VisualStudio.TestTools.UnitTesting.Assert.IsFalse(false);

}

}

## Listing 11.2 Signaturen der auf asynchrone Bearbeitung bezogenen Methoden der Klasse *WorkItemTest*

public virtual void EnqueueCallback(Action testCallbackDelegate);

public void EnqueueCallback(params Action[] actions);

public virtual void EnqueueConditional(Func<bool> conditionalDelegate);

public void EnqueueDelay(double milliseconds);

public virtual void EnqueueDelay(TimeSpan delay);

public virtual void EnqueueTestComplete();

public virtual void EnqueueWorkItem(IWorkItem testTaskObject);

public virtual void TestComplete();

## Listing 11.3 *WaitForEvent*

protected void WaitForEvent<T> (T objectOwner, string eventName)

{

EventInfo eventInfo

= objectOwner.GetType().GetEvent(eventName);

bool eventRaised = false;

if (typeof(RoutedEventHandler).IsAssignableFrom(eventInfo.EventHandlerType))

eventInfo.AddEventHandler(objectOwner,

(RoutedEventHandler)delegate { eventRaised = true; });

else if (typeof(EventHandler).IsAssignableFrom(eventInfo.EventHandlerType))

eventInfo.AddEventHandler(objectOwner,

(EventHandler)delegate { eventRaised = true; });

EnqueueConditional(() => eventRaised);

}

# Kapitel 12

## Listing 12.1 Das Interface der Datenbank-Engine

public interface IDataBaseEngine

{

void Save(Customer customer);

void Save(Order order);

int CountCustomers();

int CountOrders();

}

## Listing 12.2 Objekte der Bibliothek (Ausschnitt)

[Export("Nachricht")]

public class SayHallo

{

public override string ToString()

{

return "Hallo aus der Bibliothek";

}

}

[Export("Nachricht")]

public class XButton : Button

{

public XButton()

{

this.Content = "Hallo Button";

}

}

## Listing 12.3 Dynamisches Nachladen von Paketen

public partial class MainPage : UserControl

{

PackageCatalog catalog;

[ImportMany("Nachricht", AllowRecomposition = true)]

public ObservableCollection<string> Messages {get;set;}

public MainPage()

{

InitializeComponent();

Loaded += new RoutedEventHandler(MainPage\_Loaded);

}

void MainPage\_Loaded(object sender, RoutedEventArgs e)

{

Messages = new ObservableCollection<string>();

catalog = new PackageCatalog();

catalog.AddPackage(Package.Current);

var container = new CompositionContainer(catalog);

container.ComposeParts(this);

LayoutRoot.DataContext = Messages;

Uri uri = new Uri("GreetingsContainer.xap",

UriKind.Relative);

Package.DownloadPackageAsync(uri, downloadCompleted);

}

void downloadCompleted(AsyncCompletedEventArgs e, Package p)

{

catalog.AddPackage(p);

}

}

public class hallo

{

[Export("Nachricht")]

public string Msg = "ich sag Hi";

}

## Listing 12.4 Mit dem Attribut *Export* markierte Elemente können nachgeladen werden

public class Greetings

{

[Export("Nachricht")]

public string Deutsch = "Hallo Welt";

[Export("Nachricht")]

public string Englisch = "Hello World";

[Export("Nachricht")]

public string Spanisch = "Hola Mundo";

[Export("Nachricht")]

public string Frensh = "Bonjour tout le monde";

}

# Kapitel 13

## Listing 13.1 C#-Code, der *BrowserInformation* nutzt

public partial class MainPage : UserControl

{

public MainPage()

{

InitializeComponent();

Loaded += new RoutedEventHandler(MainPage\_Loaded);

}

void MainPage\_Loaded(object sender, RoutedEventArgs e)

{

BrowserInfos.DataContext = HtmlPage.BrowserInformation;

}

}

## Listing 13.2 C#-Code mit Typkonvertierer für das *BrowserInformation*-Projekt

public class JaNeinValueConverter : IValueConverter

{

public object Convert(object value,

Type targetType,

object parameter,

System.Globalization.CultureInfo culture)

{

bool ja;

bool.TryParse(value.ToString(), out ja);

return ja ? "Ja" : "Nein";

}

/// <summary>

/// wird nicht benötigt.

/// </summary>

public object ConvertBack(object value,

Type targetType,

object parameter,

System.Globalization.CultureInfo culture)

{

throw new NotImplementedException();

}

}

## Listing 13.3 XAML-Code zum *BrowserInformation*-Projekt

<UserControl

x:Class="BrowserEigenschaften.MainPage"

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"

xmlns:locCode="clr-namespace:BrowserEigenschaften.Code"

mc:Ignorable="d" d:DesignWidth="640" d:DesignHeight="480">

<Grid x:Name="LayoutRoot">

<StackPanel x:Name="BrowserInfos" Margin="20" Width="300">

<StackPanel.Resources>

<locCode:JaNeinValueConverter x:Key="JaNeinConverter" />

<Style TargetType="TextBlock" x:Key="HeaderStyle">

<Setter Property="FontWeight" Value="Bold" />

<Setter Property="FontStyle" Value="Italic" />

<Setter Property="Margin" Value="-6 0 0 2" />

</Style>

</StackPanel.Resources>

<TextBlock Text="Version: " Style="{StaticResource HeaderStyle}" />

<TextBlock x:Name="Version" Text="{Binding BrowserVersion}" />

<TextBlock Text="Cookies: " Style="{StaticResource HeaderStyle}"/>

<TextBlock x:Name="CookiesEnabled"

Text="{Binding CookiesEnabled,

Converter={StaticResource JaNeinConverter}}" />

<TextBlock Text="Browser: " Style="{StaticResource HeaderStyle}"/>

<TextBlock x:Name="BrowserName" Text="{Binding Name}" />

<TextBlock Text="Platform: " Style="{StaticResource HeaderStyle}" />

<TextBlock x:Name="Platform" Text="{Binding Platform}" />

<TextBlock Text="UserAgent:" Style="{StaticResource HeaderStyle}"/>

<TextBlock x:Name="UserAgent" Text="{Binding UserAgent}"

TextWrapping="Wrap"/>

</StackPanel>

</Grid>

</UserControl>

## Listing 13.4 XAML-Code einer einfachen Bookmark-Navigationslösung

<UserControl

x:Class="UsingBookmarksInSimpleApp.PageWithBookmarks"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Width="400" Height="300">

<Grid x:Name="LayoutRoot" Background="White">

<Grid.ColumnDefinitions >

<ColumnDefinition />

<ColumnDefinition />

</Grid.ColumnDefinitions>

<ListBox x:Name="lb"

ItemsSource="{Binding}">

<ListBox.ItemTemplate>

<DataTemplate>

<StackPanel>

<TextBlock Text="{Binding Name}" />

</StackPanel>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

<Grid Grid.Column="1" x:Name="pageGrid">

<StackPanel Orientation="Vertical">

<TextBlock Text="{Binding Name}" />

<TextBlock Text="{Binding Id}"/>

</StackPanel>

</Grid>

</Grid>

</UserControl>

## Listing 13.5 Codebehind-Datei der Navigationslösung aus Listing 13.4

1 public partial class PageWithBookmarks : UserControl

2 {

3 public PageWithBookmarks()

4 {

5 InitializeComponent();

6 ObservableCollection<Seite> seiten =

7 new ObservableCollection<Seite>() {

8 new Seite() { Id=1, Name = "Seite 1" },

9 new Seite() { Id=2, Name = "Seite 2" },

10 new Seite() { Id=3, Name = "Seite 3"},

11 new Seite() { Id=4, Name = "Seite 4", }

12 };

13 lb.ItemsSource = seiten;

14

15 lb.SelectionChanged += (s, e) =>

16 {

17 pageGrid.DataContext = lb.SelectedItem;

18 Seite aktuelleSeite = (Seite)lb.SelectedItem;

19 HtmlPage.Window.NavigateToBookmark ("Seite="

20 + aktuelleSeite.Id.ToString());

21 };

22 }

23 }

24 public class Seite

25 {

26 public string Name { get; set; }

27 public int Id { get; set; }

28 }

## Listing 13.6 DIV-Größe einstellen: XAML-Code

<UserControl xmlns:inputToolkit="clrnamespace:

System.Windows.Controls;assembly=System.Windows.Controls.Input.Toolkit"

x:Class="AutoSize.MainPage"

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"

xmlns:loc="clr-namespace:AutoSize">

<Grid x:Name="LayoutRoot" Width="250" Height="100">

<Grid.Resources>

<loc:CheckToVisibilityConverter x:Key="checkToVisibilityConverter" />

</Grid.Resources>

<StackPanel Background="AliceBlue" >

<Button Content="Größe anpassen" Width="200" Height="50" Click="OnChangeDivSize" />

<CheckBox x:Name="checkBx" Content="Text (un-)sichtbar"

Checked="checkBx\_Checked" Unchecked="checkBx\_Unchecked"/>

<StackPanel x:Name="spMitText"

Background="AliceBlue" Height="100"

Visibility="{Binding ElementName=checkBx,

Mode=OneWay, Path=IsChecked,

Converter={StaticResource checkToVisibilityConverter}}">

<TextBlock x:Name="textBlock"

FontSize="{Binding ElementName=upDown, Path=Value, Mode=OneWay}"

TextWrapping="Wrap"

Text="Das ist ein Text in veränderlicher Größe" />

<inputToolkit:NumericUpDown x:Name="upDown" Minimum="6" Width="100" Height="50"

FontSize="22" VerticalContentAlignment="Center" />

</StackPanel>

</StackPanel>

</Grid>

</UserControl>

## Listing 13.7 DIV-Größe einstellen: C#-Codebehind-Datei

public partial class MainPage : UserControl

{

public MainPage()

{

InitializeComponent();

}

private void OnChangeDivSize(object sender, RoutedEventArgs e)

{

HtmlDocument document = HtmlPage.Document;

HtmlElement element = document.GetElementById("silverlightControlHost");

element.SetStyleAttribute("width", LayoutRoot.Width.ToString() + "px");

element.SetStyleAttribute("height", LayoutRoot.Height.ToString() + "px");

}

private void checkBx\_Checked(object sender, RoutedEventArgs e)

{

LayoutRoot.Height = 200;

}

private void checkBx\_Unchecked(object sender, RoutedEventArgs e)

{

LayoutRoot.Height = 100;

}

}

## Listing 13.8 CheckToVisibilityConverter

public class CheckToVisibilityConverter:IValueConverter

{

public object Convert(object value, Type targetType, object parameter,

System.Globalization.CultureInfo culture)

{

return (bool)value == true ?

Visibility.Visible

: Visibility.Collapsed;

}

public object ConvertBack(object value, Type targetType, object parameter,

System.Globalization.CultureInfo culture)

{

throw new NotImplementedException();

}

}

## Listing 13.9 XAML des Silverlight-HTML-Popups

<Grid x:Name="LayoutRoot" MouseLeave="LayoutRoot\_MouseLeave" Width="200" Height="200">

<Border BorderBrush="RosyBrown" BorderThickness="4" CornerRadius="15 0 23 0">

<TextBlock FontSize="22" TextWrapping="Wrap">

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt

</TextBlock>

</Border>

</Grid>

## Listing 13.10 Codebehind-Datei des Silverlight-HTML-Popup-Objekts aus Listing 13.9

public partial class MainPage : UserControl

{

public MainPage()

{

InitializeComponent();

Loaded += new RoutedEventHandler(MainPage\_Loaded);

}

void MainPage\_Loaded(object sender, RoutedEventArgs e)

{

HtmlDocument document = HtmlPage.Document;

HtmlElement element = document.GetElementById("target");

HtmlPage.RegisterScriptableObject("elementMouseOver", this);

element.AttachEvent("onmouseover", elementMouseOver);

}

[ScriptableMember()]

public void elementMouseOver(object sender, HtmlEventArgs e)

{

HtmlDocument document = HtmlPage.Document;

HtmlElement targetElement = document.GetElementById("target");

targetElement.SetStyleAttribute("background" ,"green");

double left = double.Parse(targetElement.GetProperty("offsetLeft").ToString()) - 10;

double top = double.Parse(targetElement.GetProperty("offsetTop").ToString()) - 10;

HtmlElement slControlHost = HtmlPage.Document.GetElementById("silverlightControlHost");

slControlHost.SetStyleAttribute("left", left.ToString() + "px");

slControlHost.SetStyleAttribute("top", top.ToString() + "px");

slControlHost.SetStyleAttribute("width", LayoutRoot.Width.ToString() + "px");

slControlHost.SetStyleAttribute("height", LayoutRoot.Height.ToString() + "px");

}

private void LayoutRoot\_MouseLeave(object sender, MouseEventArgs e)

{

HtmlElement slControlHost = HtmlPage.Document.GetElementById("silverlightControlHost");

slControlHost.SetStyleAttribute("width", "0px");

slControlHost.SetStyleAttribute("height", "0px");

}

}

## Listing 13.11 Silverlight-Plug-In mit *createObjectEx* aus XAP-Datei erzeugen

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

3 <html>

4 <head>

5 <title>Silverlight aus XAP erzeugen</title>

6

7 <script src="Silverlight.js" type="text/javascript"></script>

8

9 </head>

10 <body>

11 <center>

12 <div id="SilverlightHostDiv" style="background: #FFFFFF">

13

14 <script type="text/javascript">

15 var parentDIV = document.getElementById("SilverlightHostDiv");

16 </script>

17

18 </div>

19 </center>

20

21 <script type="text/javascript">

22 Silverlight.createObjectEx(

23 { source: 'ClientBin/SimpleHallo.xap',

24 parentElement: parentDIV,

25 id: 'IdSLPlugIn',

26 properties: {

27 width: '300',

28 height: '300',

29 background: '#FFFFFFFF',

30 isWindowless: 'true',

31 framerate: '24', version: '3.0.40624.0'

32 },

33 events: {

34 onError: null, onLoad: null

35 },

36 context: null

37 });

38 </script>

49

40 </body>

41 </html>

## Listing 13.12 XAML-Oberfläche für das JavaScript-Experiment mit Rechteck und *Textblock*

<Canvas xmlns="http://schemas.microsoft.com/client/2007"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

<Rectangle Width="100" Height="200" Fill="Beige"

MouseLeftButtonDown="MouseClicked"/>

<TextBlock FontSize="30" Canvas.Left="10" Canvas.Top="10"

MouseLeftButtonDown="MouseClicked"

Foreground="Red">Hallo Silverlight</TextBlock>

</Canvas>

## Listing 13.13 Silverlight-Objekttag im HTML-Code – Zugriff auf Servervariablen

<object data="data:application/x-silverlight-2," type="application/x-silverlight-2"

width="100%" height="100%">

<param name="source" value="ClientBin/Access\_to\_Server\_Side.xap" />

<param name="onError" value="onSilverlightError" />

<param name="background" value="white" />

<param name="minRuntimeVersion" value="3.0.40624.0" />

<param name="autoUpgrade" value="true" />

<param name="initParams" value='REMOTE\_ADDR=<%=(Request.ServerVariables["REMOTE\_ADDR"])%>,

HTTP\_USER\_AGENT=<%=(Request.ServerVariables["HTTP\_USER\_AGENT"])%>,

HTTP\_ACCEPT\_LANGUAGE=<%=(Request.ServerVariables["HTTP\_ACCEPT\_LANGUAGE"])%>,

HTTP\_COOKIE=<%=(Request.ServerVariables["HTTP\_COOKIE"])%>,

URL=<%=(Request.ServerVariables["URL"])%> ' />

<a href="http://go.microsoft.com/fwlink/?LinkID=149156&v=3.0.40624.0"

style="text-decoration: none">

<img src="http://go.microsoft.com/fwlink/?LinkId=108181"

alt="Get Microsoft Silverlight"

style="border-style: none" />

</a>

</object>

## Listing 13.14 Parameter in die Ressourcenauflistung einfügen

private void Application\_Startup(object sender, StartupEventArgs e)

{

if (e.InitParams != null)

{

foreach (var item in e.InitParams)

{

this.Resources.Add(item.Key, item.Value);

}

}

this.RootVisual = new MainPage();

}

## Listing 13.15 *InitParams* im Silverlight-Objekttag

<object data="data:application/x-silverlight-2," type="application/x-silverlight-2"

width="100%" height="100%">

<param name="source" value="ClientBin/InitialisierungsParameter.xap"/>

<param name="initParams" value="Blog=silverlight.city, Author=Otto, Thema=Silverlight" />

</object>

## Listing 13.16 Speichern der Initialisierungsparameter in *IDictionary<string*, *string>*

internal IDictionary<string, string> InitParams;

private void Application\_Startup(object sender, StartupEventArgs e)

{

this.InitParams = e.InitParams;

this.RootVisual = new MainPage();

}

## Listing 13.17 Initialisierungsparameter verwenden

public MainPage()

{

InitializeComponent();

IDictionary<string, string> initParams = ((App)App.Current).InitParams;

TextBlock text = new TextBlock();

foreach (string key in initParams.Keys)

{

Run run = new Run();

run.Text = key + " = " + initParams[key];

text.Inlines.Add(run);

text.Inlines.Add (new LineBreak ());

}

Thickness margin = new Thickness(10);

text.Margin = margin;

LayoutRoot.Children.Add(text);

}

## Listing 13.18 Verwendung der in der Applikationsressource gespeicherten Initialisierungsparameter

public MainPage()

{

InitializeComponent();

TextBlock text = new TextBlock();

AddParamToText(text, "Blog");

AddParamToText(text, "Author");

AddParamToText(text, "Thema");

Thickness margin = new Thickness(10);

text.Margin = margin;

LayoutRoot.Children.Add(text);

}

private void AddParamToText(TextBlock text, string param)

{

string blog = GetParam(param);

if (blog != string.Empty)

{

Run run = new Run();

run.Text = "Blog" + " = " + blog;

text.Inlines.Add(run);

text.Inlines.Add(new LineBreak());

}

}

private string GetParam(string p)

{

if (App.Current.Resources[p] != null)

return App.Current.Resources[p].ToString();

else

return string.Empty;

}

## Listing 13.19 Zugriff auf *QueryString*

public MainPage()

{

InitializeComponent();

TextBlock tb = new TextBlock();

foreach (var item in HtmlPage.Document.QueryString)

{

tb.Inlines.Add(

new Run()

{

Text = String.Format(

" {0} = {1}",

item.Key,

item.Value)

});

tb.Inlines.Add ( new LineBreak());

}

LayoutRoot.Children.Add(tb);

}

# Kapitel 14

## Listing 14.1 Traditionelle Datenbindung

public partial class MainPage : UserControl

{

ProductContext productContext = new ProductContext();

public MainPage()

{

InitializeComponent();

this.dataGridProducts.ItemsSource = productContext.Products;

productContext.Load(productContext.GetProductsQuery());

}

}

## Listing 14.2 Deklarative Datenbindung mit *DomainDataSource*

<UserControl

xmlns:riaControls="clr-namespace:System.Windows.Controls;assembly=System.Windows.Ria.Controls"

xmlns:data="clr-namespace:System.Windows.Controls;assembly=System.Windows.Controls.Data"

xmlns:loc="clr-namespace:ProductApp.Web"

x:Class="ProductApp.MainPage"

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" d:DesignWidth="640" d:DesignHeight="480">

<Grid x:Name="LayoutRoot">

<StackPanel>

<StackPanel Orientation="Horizontal">

</StackPanel>

<data:DataGrid x:Name="dataGridProducts"

ItemsSource="{Binding Data, ElementName=productDataSource}" />

<riaControls:DomainDataSource x:Name="productDataSource" LoadSize="10"

QueryName="GetProductsQuery" AutoLoad="True">

<riaControls:DomainDataSource.DomainContext>

<loc:ProductContext />

</riaControls:DomainDataSource.DomainContext>

</riaControls:DomainDataSource>

</StackPanel>

</Grid>

</UserControl>

## Listing 14.3 Deklaratives Filtern mit einer *DomainDataSource*

<riaControls:DomainDataSource x:Name="productDataSource" LoadSize="10"

QueryName="GetProductsQuery" AutoLoad="True">

<riaControls:DomainDataSource.DomainContext>

<loc:ProductContext />

</riaControls:DomainDataSource.DomainContext>

<riaControls:DomainDataSource.FilterDescriptors>

<riaData:FilterDescriptorCollection>

<riaData:FilterDescriptor PropertyPath="UnitPrice" Operator="IsGreaterThanOrEqualTo">

<riaData:ControlParameter ControlName="unitPriceTB" PropertyName="Text"

RefreshEventName="TextChanged" />

</riaData:FilterDescriptor>

</riaData:FilterDescriptorCollection>

</riaControls:DomainDataSource.FilterDescriptors>

</riaControls:DomainDataSource>

## Listing 14.4 Ändern der Bindung von *wsHttpBinding* auf *basicHttpBinding*

<service behaviorConfiguration="SecureServices.Web.Service1Behavior"

name="SecureServices.Web.Service1">

<endpoint address="" binding="wsHttpBinding" contract="SecureServices.Web.IService1">

<identity>

<dns value="localhost" />

</identity>

</endpoint>

<endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange" />

</service>

## Listing 14.5 Verwendung des *AspNetCompatibilityRequirements*-Attributs

[AspNetCompatibilityRequirements(RequirementsMode= AspNetCompatibilityRequirementsMode.Required)]

public class Service1 : IService1

{

public void DoWork()

{

}

public string MalZwei(int i)

{

return (2 \* i).ToString();

}

}

## Listing 14.6 Authorisierungsknoten in der *Web.Config*. Mit *<deny users="?" />* sichern Sie die komplette Silverlight-Anwendung, nicht nur

<?xml version="1.0" encoding="utf-8"?>

<configuration>

<system.web>

<authorization>

<deny users="?" />

<allow users="DerUser" />

</authorization>

</system.web>

</configuration>

## Listing 14.7 Ändern Sie die Authentifizierungsmethode in der *Web.Config* auf Forms

<authentication mode="Forms">

</authentication>

## Listing 14.8 ASP.NET-Objekte *Login* und *LoginStatus* in der HTML-Testseite

<body>

<form id="form1" runat="server" style="height: 100%">

<asp:Login ID="Login1" runat="server" />

<asp:LoginStatus runat="server" />

<div id="silverlightControlHost">

<object data="data:application/x-silverlight-2," ...

## Listing 14.9 Servicevereinbarung *IMatheService (IMatheService.cs)*

[ServiceContract]

public interface IMatheService

{

[OperationContract]

void DoWork();

[OperationContract]

string MalZwei(int i);

}

## Listing 14.10 (Codebehind-) Implementierung des MatheService*-*Dienstes *(MatheService.svc.cs)*

[AspNetCompatibilityRequirements(RequirementsMode

= AspNetCompatibilityRequirementsMode.Required)]

public class MatheService : IMatheService

{

public MatheService()

{

Thread.CurrentPrincipal = HttpContext.Current.User;

}

public void DoWork()

{

}

[PrincipalPermission( SecurityAction.Demand, Name = "DerUser" )]

public string MalZwei(int i)

{

return (2 \* i).ToString();

}

}

## Listing 14.11 MatheService-Dienst *(MatheService.svc)*

<%@ ServiceHost Language="C#" Debug="true" Service="SecureServices.Web.MatheService"

CodeBehind="MatheService.svc.cs" %>

## Listing 14.12 *AuthenticationService* als skriptbare Methode

<%@ ServiceHost Language="C#" Debug="true"

Service="System.Web.ApplicationServices.AuthenticationService"

%>

## Listing 14.13 Dienstknoten *AuthenticationService*

<services>

<!-- WCF Authentication Service -->

<service behaviorConfiguration="AuthenticationServiceTypeBehaviors"

name="System.Web.ApplicationServices.AuthenticationService">

<endpoint binding="basicHttpBinding"

bindingConfiguration="httpBindingConfiguration"

bindingNamespace="http://asp.net/ApplicationServices/v200"

contract="System.Web.ApplicationServices.AuthenticationService" />

</service>

</services>

## Listing 14.14 Konfiguration des Dienstverhaltens (*serviceBehaviors*)

<behaviors>

<serviceBehaviors>

<!—Behavior für WCF-Authentifizierungsdienst -->

<behavior name="AuthenticationServiceTypeBehaviors">

<serviceMetadata httpGetEnabled="true" />

<serviceDebug includeExceptionDetailInFaults="false" />

</behavior>

</serviceBehaviors>

</behaviors>

## Listing 14.15 Die Bindung des Dienstverhaltens

<bindings>

<basicHttpBinding>

<!-- die BindingConfiguration für den AuthenticationService -->

<binding name="httpBindingConfiguration">

<!-- Im "Normalbetrieb" sollte hier Transport stehen -->

<security mode="None" />

</binding>

</basicHttpBinding>

</bindings>

## Listing 14.16 Der Knoten *system.web.extension* erlaubt den Skriptzugriff auf den Authentifizierungsdienst

<system.web.extensions>

<scripting>

<webServices>

<authenticationService enabled="true" requireSSL="false"/>

</webServices>

</scripting>

</system.web.extensions>

## Listing 14.17 Hauptformular, von dem aus *LoginDialog* aufgerufen wird

<UserControl x:Class="SecureServices.MainPage"

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"

xmlns:loc="clr-namespace:SecureServices"

xmlns:sys="clr-namespace:System;assembly=mscorlib"

mc:Ignorable="d" d:DesignWidth="640" d:DesignHeight="480"

Width="500" Height="400" Background="AntiqueWhite" >

<Grid x:Name="LayoutRoot" ShowGridLines="True">

<Grid.Resources>

<loc:LoginStatusConverter x:Key="loginStatusConverter" />

<loc:User x:Key="user" />

</Grid.Resources>

<ScrollViewer Grid.Column="1">

<StackPanel Margin="5">

<Button x:Name="loginButton"

Width="250"

Content="{Binding user, Source={StaticResource user},

Path=Username, Mode=TwoWay, Converter={StaticResource loginStatusConverter}}"

Click="OnShowLoginDialog" />

<TextBox x:Name="tbox" />

<TextBlock x:Name="tBlock" />

<Button Content="\* 2" Click="OnClickMultiplication"

Width="100"/>

</StackPanel>

</ScrollViewer>

</Grid>

</UserControl>

## Listing 14.18 Klasse *User* in der *MainPage*

public class User : INotifyPropertyChanged

{

private string username;

public string Username

{

get { return username; }

set

{

username = value;

OnPropertyChanged("Username");

}

}

public event PropertyChangedEventHandler PropertyChanged;

void OnPropertyChanged(string property)

{

if (PropertyChanged != null)

this.PropertyChanged(this,

new PropertyChangedEventArgs(property));

}

}

## Listing 14.19 Codebehind der *MainPage*

1 public partial class MainPage : UserControl

2 {

3 User User;

4 public MainPage()

5 {

6 InitializeComponent();

7 Loaded += new RoutedEventHandler(MainPage\_Loaded);

8 }

9

10 void MainPage\_Loaded(object sender, RoutedEventArgs e)

11 {

12 User = (User)LayoutRoot.Resources["user"];

13 }

14

15 private void OnClickMultiplication(object sender, RoutedEventArgs e)

16 {

17 tBlock.Text = " ... arbeite ...";

18 string s = tbox.Text;

19 int i;

20 int.TryParse(s, out i);

21 MatheServiceReference.MatheServiceClient client

22 = new MatheServiceClient();

23 client.MalZweiCompleted +=

24 new EventHandler<MalZweiCompletedEventArgs>(client\_MalZweiCompleted);

25 client.MalZweiAsync(i);

26 }

27

28 void client\_MalZweiCompleted(object sender,

29 MalZweiCompletedEventArgs e)

30 {

31 if (e.Error != null)

32 {

33 tBlock.Text = e.Error.Message + " "

34 + e.Error.InnerException + " "

35 + e.Error.GetType(); ;

36 }

37 else

38 {

39 tBlock.Text = e.Result;

40 }

41 }

42

43 private void OnShowLoginDialog(object sender, RoutedEventArgs e)

44 {

45 LoginDialog dialog = new LoginDialog();

46 dialog.Closed += new EventHandler(OnLoginDialogClosed);

47 dialog.Show();

48 }

49

50 void OnLoginDialogClosed(object sender, EventArgs e)

51 {

52 LoginDialog dialog = (LoginDialog)sender;

53 bool? result = dialog.DialogResult;

54

55 if (result.HasValue && result.Value)

56 {

57 User.Username = dialog.Username;

58 }

59 else

60 {

61 User.Username = null;

62 }

63 }

64 }

## Listing 14.20 XAML-Code des LoginDialogs

<controls:ChildWindow x:Class="SecureServices.LoginDialog"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:controls="clr-namespace:System.Windows.Controls;assembly=System.Windows.Controls"

Width="400" Height="300"

Title="LoginChildWindow">

<Grid x:Name="LayoutRoot" Margin="2">

<Grid.RowDefinitions>

<RowDefinition />

<RowDefinition Height="Auto" />

</Grid.RowDefinitions>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition Height="Auto" />

<RowDefinition Height="Auto" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="3\*" x:Name="theCol"/>

</Grid.ColumnDefinitions>

<TextBlock Grid.Column="0" Grid.Row="0" Text="User" />

<TextBlock Grid.Column="0" Grid.Row="1" Text="Passwort" />

<TextBox Grid.Column="1" Grid.Row="0"

Width="{Binding ElementName=theCol, Path=ActualWidth}"

Text="{Binding Path=Username, Mode=TwoWay, NotifyOnValidationError=True,

ValidatesOnExceptions=True}" />

<PasswordBox Grid.Column="1" Grid.Row="1"

Width="{Binding ElementName=theCol, Path=ActualWidth}"

Password="{Binding Path=Passwort, Mode=TwoWay, NotifyOnValidationError=True,

ValidatesOnExceptions=True}" />

<TextBlock Grid.Row="2" Grid.ColumnSpan="2" TextWrapping="Wrap"

x:Name="textBlockError" />

</Grid>

<Button x:Name="CancelButton" Content="Cancel"

Click="CancelButton\_Click" Width="75" Height="23"

HorizontalAlignment="Right" Margin="0,12,0,0" Grid.Row="1" />

<Button x:Name="OKButton" Content="LogIn"

Click="LogInButton\_Click" Width="75" Height="23"

HorizontalAlignment="Right" Margin="0,12,79,0" Grid.Row="1" />

</Grid>

</controls:ChildWindow>

## Listing 14.21 *SLUser*, von *INotifyPropertyChanged* abgeleitet

public class SLUser : INotifyPropertyChanged

{

private string username;

public string Username

{

get { return username; }

set { username = value;

OnPropertyChanged("Username");

}

}

private string passwort;

public string Passwort

{

get { return passwort; }

set { passwort = value;

OnPropertyChanged("Passwort");

}

}

public event PropertyChangedEventHandler PropertyChanged;

void OnPropertyChanged(string property)

{

if (PropertyChanged != null)

this.PropertyChanged(this,

new PropertyChangedEventArgs(property));

}

}

## Listing 14.22 Codebehind-Datei des Anmeldesteuerelements

1 public partial class LoginDialog : ChildWindow

2 {

3 public bool IsLoggedIn { get; set; }

4 public string Username { get; set; }

5 SLUser user = new SLUser() { Username = "derUser", Passwort = "" };

6

7 public LoginDialog()

8 {

9 InitializeComponent();

10 LayoutRoot.DataContext = user;

11 }

12

13 private void LogInButton\_Click(object sender, RoutedEventArgs e)

14 {

15 AuthenticationServiceClient authClient

16 = new AuthenticationServiceClient();

17 authClient.LoginCompleted

18 += new OnLoginCompleted;

19 authClient.LoginAsync(user.Username, user.Passwort, "", true);

20 }

21

22 void OnLoginCompleted(object sender, LoginCompletedEventArgs e)

23 {

24 if (e.Error == null && e.Result)

25 {

26 IsLoggedIn = true;

27 Username = this.user.Username;

28 this.DialogResult = true;

29 }

30 else

31 {

32 Run run = new Run();

33 run.Text = "Login nicht erfolgreich";

34 textBlockError.Text = "";

35 textBlockError.Inlines.Add(run);

36 textBlockError.Inlines.Add(new LineBreak());

37 if (e.Error != null && e.Error.Message != null)

38 {

39 textBlockError.Inlines.Add(e.Error.Message);

40 if (e.Error.InnerException != null)

41 {

42 textBlockError.Inlines.Add(new LineBreak());

43 Run r = new Run();

44 r.Text = e.Error.InnerException.Message;

45 textBlockError.Inlines.Add(r);

46 }

47 }

48 }

49 }

50

51 private void CancelButton\_Click(object sender, RoutedEventArgs e)

52 {

53 IsLoggedIn = false;

54 Username = null;

55 this.DialogResult = false;

56 }

57 }