Window Presentation Foundation

- Dipl.-Inf., Dipl.-Ing. (FH) Michael Wilhelm
- Hochschule Harz
- FB Automatisierung und Informatik
- mwilhelm@hs-harz.de
- http://www.miwilhelm.de
- Raum 2.202
- Tel. 03943 / 659 338

·Inhalt

- Einführung WPF
- Layouts
- C# Sprache
- **Dialog-Elemente, Menüs**
- 2D- / 3D-Grafik, Audio, Video, Animation
- Routet Events, Dependency Properties, Command
- Textdarstellung (Flow-FixedDocuments)
- Datenbanken
- Navigation / Browser
- Eigene Komponenten

MessageBox

- Show(String)
- Show(IWin32Window, Text, Caption, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions, String, HelpNavigator, Object)
- Show(Text, Caption)
- Show(IWin32Window, Text)
- Show(Text, Caption, MessageBoxButtons)
- Show(IWin32Window, Text, Caption)
- Show(Text, Caption, MessageBoxButtons, MessageBoxIcon)
- Show(IWin32Window, Text, Caption, MessageBoxButtons)
- Show(Text, Caption, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton)

MessageBox

- Show(IWin32Window, Text, Caption, MessageBoxButtons, MessageBoxIcon)
- Show(IWin32Window, Text, Caption, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions)
- Show(Text, Caption, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions, Text, Caption)
- Show(Text, Caption, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions, String, HelpNavigator)

MessageBox

- Show(IWin32Window, Text, Caption, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions, String)
- Show(Text, Caption, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions, String, HelpNavigator, Object)
- Show(IWin32Window, Text, Caption, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions, Text, Caption)
- Show(IWin32Window, Text, Caption, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions, String, HelpNavigator)

Konstanten der MessageBox

MessageBoxButtons.YesNoCancel
MessageBoxButtons.YesNo
MessageBoxButtons.RetryCancel
MessageBoxButtons.OKCancel
MessageBoxButtons.AbortRetryIgnore
MessageBoxButtons.OK

DialogResult.Yes
DialogResult.OK
DialogResult.None
DialogResult.No
DialogResult.Ignore
DialogResult.Cancel
DialogResult.Abort

MessageBoxIcon.Warning
MessageBoxIcon.Stop
MessageBoxIcon.Question
MessageBoxIcon.None
MessageBoxIcon.Information
MessageBoxIcon.Hand
MessageBoxIcon.Exclamation
MessageBoxIcon.Error
MessageBoxIcon.Asterisk

- Show(Text, Caption, MessageBoxButtons, MessageBoxIcon)

Konstanten der MessageBox

MessageBoxIcon.Warning
MessageBoxIcon.Stop
MessageBoxIcon.Question
MessageBoxIcon.None
MessageBoxIcon.Information
MessageBoxIcon.Hand
MessageBoxIcon.Exclamation
MessageBoxIcon.Error
MessageBoxIcon.Asterisk

Eigenschaften der GUI-Elemente: Frame

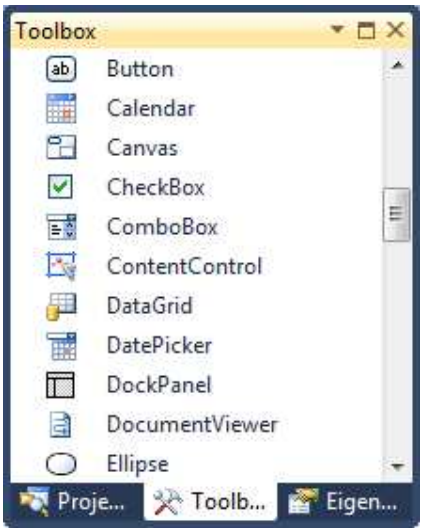
- Name
- AutoScroll False/True
- AutoSize False/True
- AutoSizeMode: GrowOnly, GrowAndShrink
- BackColor
- BackGroundImage
- CancelButton
- MaximizeBox
- MinimizeBox
- Cursor
- DoubleBuffered
- Enabled
- Font
- ForeColor
- FormBorderStyle:
  None
  FixedSingle,
  Fixed3D,
  FixedDialog,
  Sizable,
  FixedToolWindow,
  SizableToolWindow

Eigenschaften der GUI-Elemente: Frame

- Icon
- KeyPreview
- Location
- Locked
- MainMenuStrip
- Opacity
- ShowIcon
- Size
- StartPosition:
  Manual,
  CenterScreen,
  WindowsDefaultLocation,
  WindowsDefaultBounds,
  CenterParent
- Tag
- Text: (Caption)
- TopMost False/True
- WindowState:
  Normal,
  Minimized,
  Maximized

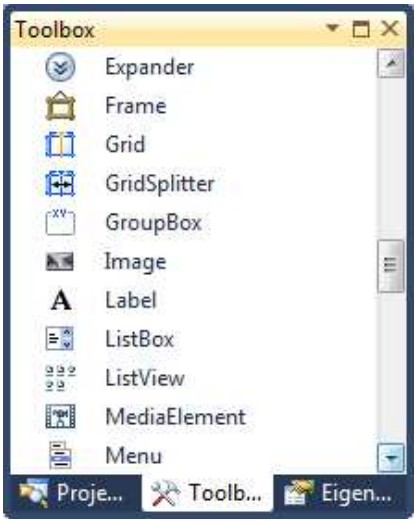
GUI-Elemente

- Button /ToggleButton
  RepeatButton
- Calendar
- Canvas
- CheckBox
- ComboBox
- ContentControl
- DataGrid
- DatePicker
- DockPanel
- DocumentViewer
- Ellipse



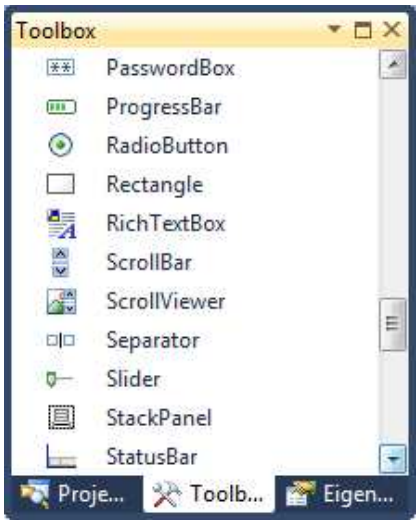
GUI-Elemente

- Expander
- Frame
- Grid
- GridSplitter
- GroupBox
- Image
- Label
- ListBox
- ListView
- MediaElement
- Menu



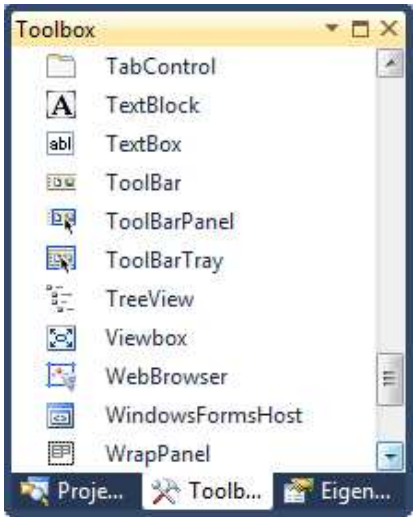
GUI-Elemente

- PasswordBox
- ProgressBar
- RadioButton
- Rectangle
- RichTextBox
- ScrollBar
- ScrollViewer
- Separator (Menüs)
- Slider
- StackPanel
- StatusBar



GUI-Elemente

- TabControl
- TextBlock
- TextBox
- ToolBar
- ToolBarPanel
- TreeView
- Viewbox
- WebBrowser
- WindowsFormsHost
- WrapPanel



Eigenschaften Button:

- Name
- BackGround
- Cursor
- Content statt Text
- FontSize etc.
- IsCancel
- IsDefault
- IsEnabled
- ClickMode:
  - Release
  - Press
  - Hover
- Min- MaxHeight
- Min- MaxWidth
- Height / Width
- Margin
- Padding
- Opacity
- HorizontalAligment
  - Left / Right
  - Center / Stretch
- Tag
- ToolTip
- Visibility
- Z-Index

Eigenschaften RepeatButton:

- **Interval**
- Name
- BackGround
- Cursor
- Content statt Text
- FontSize etc.
- IsCancel
- IsDefault
- IsEnabled
- ClickMode:
  - Release
  - Press
  - Hover
- Min- MaxHeight
- Min- MaxWidth
- Height / Width
- Margin
- Padding
- Opacity
- HorizontalAligment
  - Left / Right
  - Center / Stretch
- Tag
- ToopTip
- Visibility
- Z-Index

Eigenschaften ToggleButton: Manuelle Verarbeitung

- **IsThreeState: 2 oder 3 Zustände**
- **IsChecked**
- Name
- BackGround
- Cursor
- Content statt Text
- FontSize etc.
- IsCancel
- IsDefault
- IsEnabled
- ClickMode:
  - Release
  - Press
  - Hover
- Min- MaxHeight
- Min- MaxWidth
- Height / Width
- Margin
- Padding
- Opacity
- HorizontalAligment
  - Left / Right
  - Center / Stretch
- Tag
- ToopTip
- Visibility
- Z-Index

Eigenschaften Label:

- Name
- BackGround
- Cursor
- Content statt Text
- FontSize etc.
- HorizontalAligment
  - Left / Right
  - Center / Stretch
- VerticalAligment
  - Top / Bottom
  - Center / Stretch
- Min- MaxHeight
- Min- MaxWidth
- Height / Width
- Margin
- Padding
- Opacity
- Tag
- ToopTip
- Visibility
- Z-Index

Eigenschaften TextBox:

- Name
- AcceptsReturn
- AcceptsTab
- FlowDirection
- BackGround
- Cursor
- Text statt Content
- IsReadOnly
- MaxLines
- FontSize etc.
- HorizontalAligment
- VerticalAligment
- TextDecoration
- TextAligment
- TextWrapping
- UndoLimit
- Min- MaxHeight
- Min- MaxWidth
- Height / Width
- Margin
- Padding
- Opacity
- Tag

Eigenschaften TextBox:

TextBox mit mehreren Zeilen (Multiline):

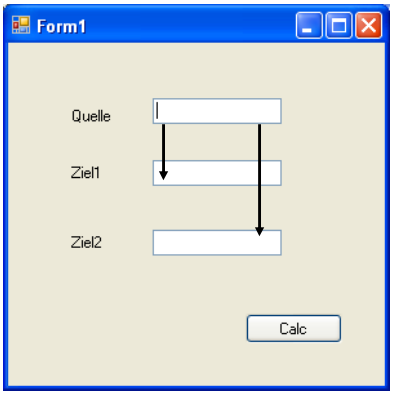
- tInputElement.TextWrapping = TextWrapping.Wrap;
- tInputElement.VerticalScrollBarVisibility = ScrollBarVisibility.Visible;
- tInputElement.AcceptsReturn = true;

- tInputElement.TextWrapping = TextWrapping.NoWrap;
- tInputElement.HorizontalScrollBarVisibility = ScrollBarVisibility.Visible;
- tInputElement.VerticalScrollBarVisibility = ScrollBarVisibility.Visible;
- tInputElement.AcceptsReturn = true;

Eigenschaften TextBlock:

- Name
- TextTrimming
- Inlines
  - Bold
  - Italic
  - Run
  - Underline
- FlowDirection
- BackGround
- Cursor
- Text statt Content
- FontSize etc.
- HorizontalAligment
- VerticalAligment
- TextDecoration
- TextAligment
- TextWrapping
- Min- MaxHeight
- Min- MaxWidth
- Height / Width
- Margin
- Padding
- Opacity
- Tag

Beispiel ueb13: Strings

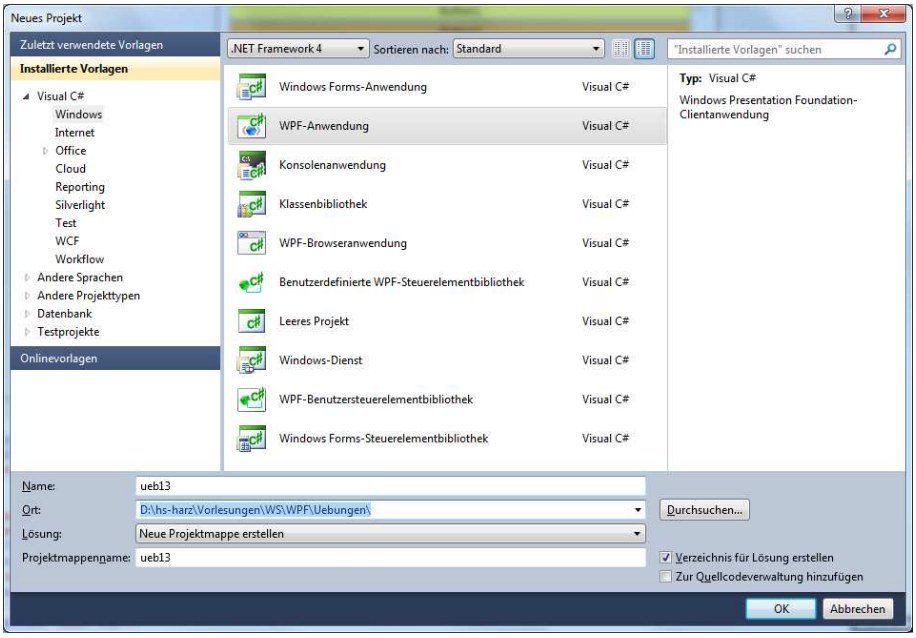


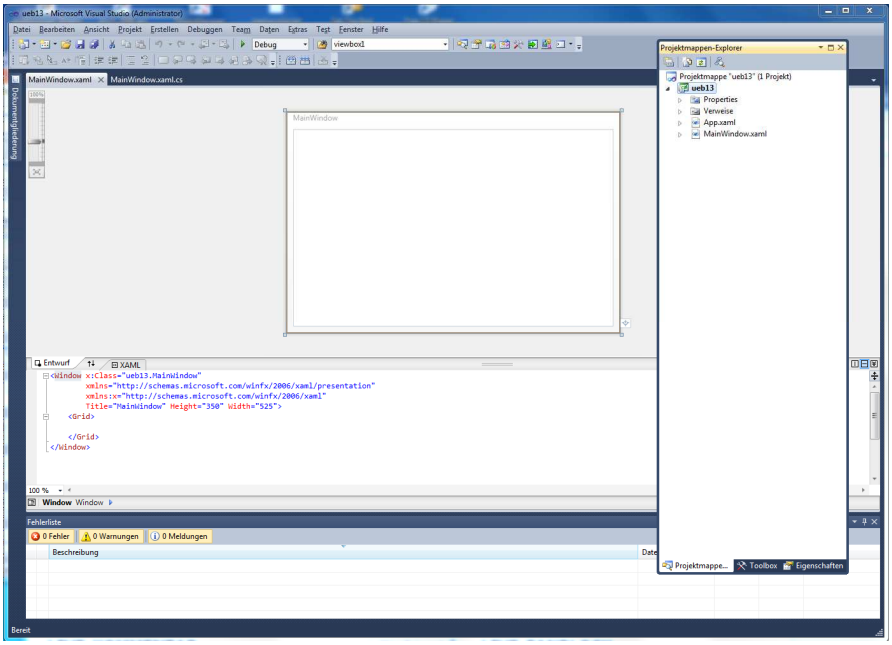
Namen:

- lbQuelle
- tQuelle
- lbZiel1
- tZiel1
- lbZiel1
- tZiel2
- bnCalc

Aufgaben

- Transportieren eines String per Schalter
- Transportieren eines String bei Änderung

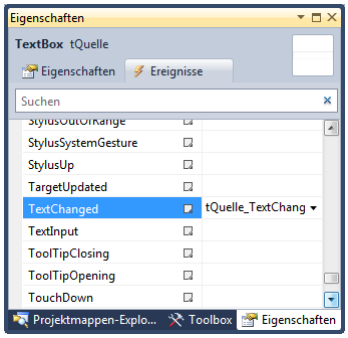




1. Beispiel

- Einbau der GUI-Elemente
  - Label lbQuelle
  - TextBox tQuelle
  - Label lbZiel1
  - TextBox tZiel1
  - Label lbZiel2
  - TextBox tZiel2
  - Schalter „Calc“ bnCalc
- Doppelklick auf den Schalter
- Eintragen des Quellcodes
- Einbauen der Event-Methode
- Eintragen des Quellcodes

tZiel1.Text = tQuelle.Text;

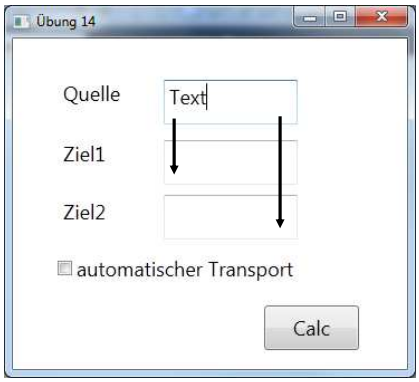


tZiel2.Text = tQuelle.Text;

Eigenschaften CheckBox:

- Name
- IsChecked
- IsEnabled
- IsThreeState
- BackGround
- Cursor
- Content statt Text
- FontSize etc.
- ClickMode:
  - Release
  - Press
  - Hover
- Min- MaxHeight
- Min- MaxWidth
- Height / Width
- Margin
- Padding
- Opacity
- HorizontalAligment
  - Left / Right
  - Center / Stretch
- Tag
- ToopTip
- Visibility
- Z-Index

Beispiel ueb14: Strings mit CheckBox



Namen:

- lbQuelle
- tQuelle
- lbZiel1
- tZiel1
- lbZiel1
- tZiel2
- bnCalc
- chkMove2Ziel2

Aufgaben

- Transportieren eines String per Schalter
- Transportieren eines String bei Änderung nur bei angeklickter CheckBox

Beispiel ueb14: Strings mit CheckBox

```
private void bnCalc_Click(object sender, RoutedEventArgs e) {
        tZiel1.Text = tQuelle.Text;
}

private void tQuelle_TextChanged(object sender,
                                 TextChangedEventArgs e)  {
        // ThreeState   true, false, null
        if (chkMove2Ziel2.IsChecked==true)
        {
           tZiel2.Text = tQuelle.Text;
        }
}
```


Eigenschaften GroupBox: Speichert Elemente mit Rahmen

- Name
- BackGround
- Cursor
- Header statt Content statt Text
- HeaderStringFormat
- FontSize etc.
- HorizontalAligment
  - Left / Right
  - Center / Stretch
- VerticalAligment
  - Top / Bottom
  - Center / Stretch
- Min- MaxHeight
- Min- MaxWidth
- Height / Width
- Margin
- Padding
- Opacity
- Tag
- ToopTip
- Visibility
- Z-Index

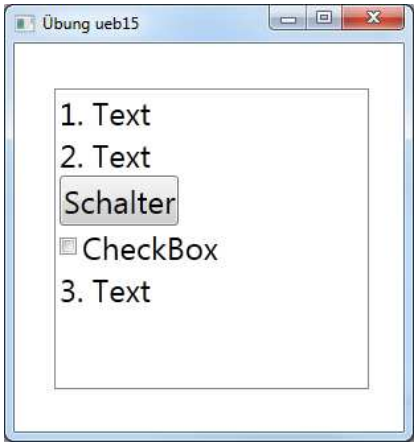
Eigenschaften RadioButton:

- Name
- BackGround
- Cursor
- Content statt Text
- IsChecked
- IsThreeState
- IsEnabled
- FontSize etc.
- HorizontalAligment
  - Left / Right / Center / Stretch
- VerticalAligment
  - Top / Bottom / Center / Stretch
- Min- MaxHeight
- Min- MaxWidth
- Height / Width
- Margin
- Padding
- Opacity
- Tag
- ToopTip
- Visibility
- Z-Index
- ClickMode:
  - Release
  - Press
  - Hover

Eigenschaften ListBox: Speichert beliebige Elemente

- Name
- AlternationCount
- BackGround
- Cursor
- Content statt Text
- GroupStyle
- FontSize etc.
- SelectionIndex
- SelectionValue
- SelectionValuePath
- SelectionMode
- Items // speichert die Elemente
- HorizontalAligment
  - Left / Right / Center / Stretch
- VerticalAligment
  - Top / Bottom / Center / Stretch
- Min- MaxHeight
- Min- MaxWidth
- Height / Width
- Margin / Padding
- Opacity
- Tag
- ToopTip
- Visibility
- Z-Index

Beispiel ListBox: ueb15




```
<ListBox Name="listBox1" >
    <ListBoxItem>1. Text</ListBoxItem>
</ListBox>
```


Beispiel ListBox: ueb15

```
private void setGUI()    {
        listBox1.Items.Clear();
        listBox1.Items.Add("1. Text");        listBox1.Items.Add("2. Text");
        Button bn = new Button();
        bn.Content = "Schalter";
        bn.Click += button1_Click;
        listBox1.Items.Add(bn);
        CheckBox chk = new CheckBox();
        chk.Content = "CheckBox";
        listBox1.Items.Add(chk);
        listBox1.Items.Add("3. Text");
    }
    private void button1_Click(object sender, RoutedEventArgs e)    {
        MessageBox.Show("Hier in OnClick");
    }
```

Eigenschaften ComboBox:

- Name
- AlternationCount (odd/even-Farbe)
- BackGround
- Cursor
- Content statt Text
- GroupStyle
- **IsEditable**
- SelectionIndex
- SelectionValue
- SelectionValuePath
- Items // speichert die Elemente

- HorizontalAligment
  - Left / Right / Center / Stretch
- VerticalAligment
  - Top / Bottom / Center / Stretch
- Min- MaxHeight
- Min- MaxWidth
- Height / Width
- Margin / Padding
- FontSize etc.
- Opacity / Tag
- ToopTip
- Visibility / Z-Index

Kein Sort / DropDownStyle / MaxDropDown

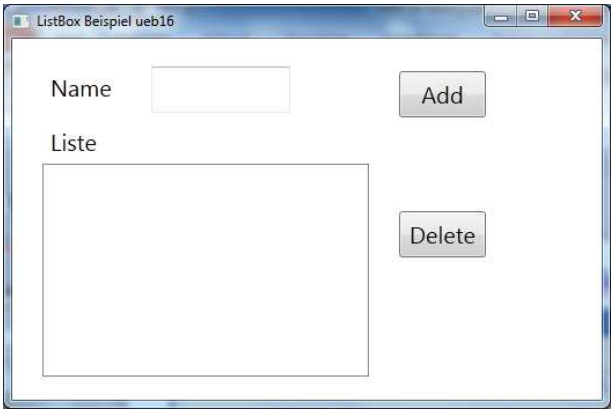
Eigenschaften ComboBox:

Kein direktes Sortieren

```
using System.ComponentModel;  // Sortieren

comboBox1.Items.Clear();
comboBox1.Items.Add("xx1. Text");
comboBox1.Items.Add("2. Text");
comboBox1.Items.Add("1. Text");
comboBox1.Items.SortDescriptions.Add(
        new SortDescription("Content", ListSortDirection.Descending));
);
```

"Content" = propertyName = Der Name der Eigenschaft (Attribut),
nach dem sortiert werden soll (Klassen)

Beispiel ueb16 mit einer ListBox



- tName
- bnAdd
- bnDelete
- lstItems (Liste)

- bnAdd_Click

Check Eintrag gültig

- bnDelete_Click

Check Eintrag selektiert

- Trim
- Liste.Items.Add(sStr);


```
private void BnAdd_Click(object sender, EventArgs e)
{
        string sStr = tName.Text.Trim();
        if (! sStr.Equals("") )
        {
           tName.Text = "";
           lstItems.Items.Add(sStr);
        }
        else
        {
           MessageBox.Show("Bitte einen Eintrag in das Editorfeld");
           tName.Focus();
        }

      }
```

```
private void BnDelete_Click(object sender, EventArgs e)    {
    int i = lstItems.SelectedIndex;
    string sStr = i.ToString();
    MessageBox.Show(sStr);
    if (i >= 0)   {
      MessageBoxResult retCode = MessageBox.Show("Wollen
        Sie wirklich den "+(i+1).ToString()+". Eintrag löschen?",
            "Remove Item", MessageBoxButton.YesNo,
        MessageBoxButton.YesNo,MessageBoxImage.Question);
      if (retCode == MessageBoxResult.Yes) {
          MessageBox.Show("Löschen");
          lstItems.Items.RemoveAt(i);
      } // if
    } // if
} // BnDelete_Click
```


Eigenschaften ProgressBar:

- Name
- Orientation
  - Horizontal
  - Vertical
- SmallChange als Double
- Minimum
- Maximun
- Value
- **Background**
- **Foreground**
- **Balken hat Farbverlauf**
  Cursor
- HorizontalAligment
  - Left / Right / Center / Stretch
- VerticalAligment
  - Top / Bottom / Center / Stretch
- Min- MaxHeight
- Min- MaxWidth
- Height / Width
- Margin / Padding
- FontSize etc.
- Opacity
- Tag / ToopTip
- Visibility / Z-Index

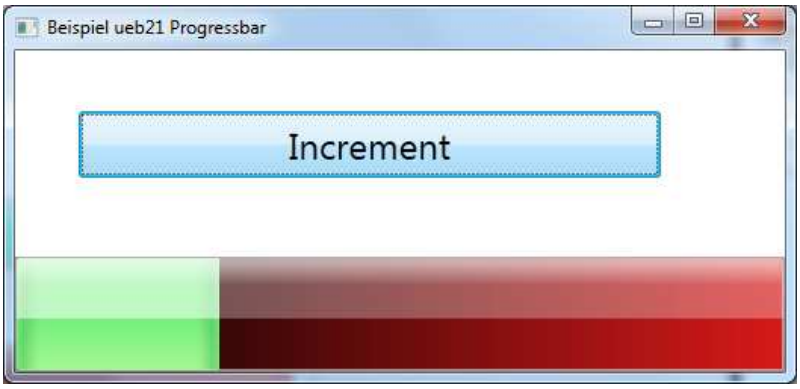
Beispiel ProgressBar: ueb21

```
<RepeatButton Interval="50"  Content="Increment" Height="44"
  Name="button1" Margin="42,22,82,33" Click="button1_Click"
        Background="Aquamarine" />

<ProgressBar Grid.Column="0" Grid.Row="1" Name="progressBar1"
        Foreground="#FF28E249">
     <ProgressBar.Background>
          <LinearGradientBrush EndPoint="1,0.5" StartPoint="0,0.5">
             <GradientStop Color="Black" Offset="0" />
             <GradientStop Color="#FFD81919" Offset="1" />
          </LinearGradientBrush>
      </ProgressBar.Background>
</ProgressBar>
```


Beispiel ProgressBar: ueb21

```
private void setGUI()        {
        progressBar1.Minimum = 0;
        progressBar1.Maximum = 200;
        progressBar1.Value = 0;
  }

private void button1_Click(object sender, RoutedEventArgs e)    {
        progressBar1.Value++;
}
```

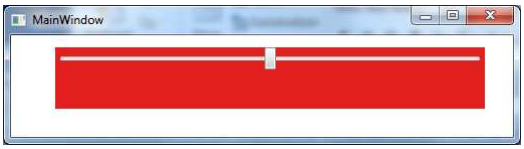
Beispiel ProgressBar: ueb21



Eigenschaften Slider:

- Name
- Orientation
  - Horizontal
  - Vertical
- SmallChange als Double
- Minimum
- Maximun
- Value
- Background
- Foreground
- Cursor
- HorizontalAligment
  - Left / Right / Center / Stretch
- VerticalAligment
  - Top / Bottom / Center / Stretch
- Min- MaxHeight
- Min- MaxWidth
- Height / Width
- Margin / Padding
- FontSize etc.
- Opacity
- Tag / ToopTip
- Visibility / Z-Index

Beispiel Slider:

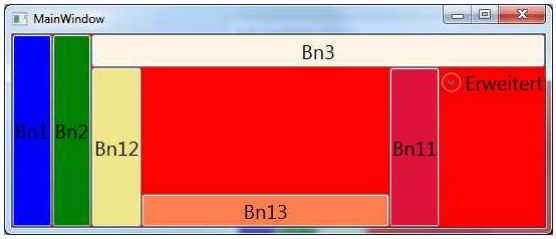


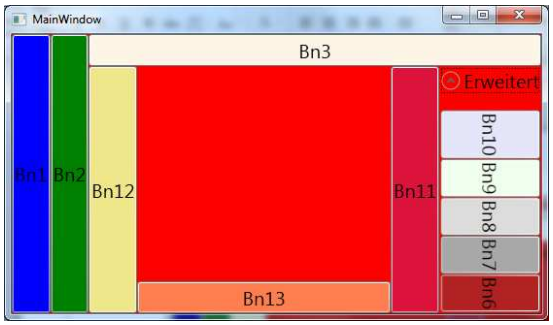

```
<Slider Height="61" Name="slider1" Width="428"
        Background="#FFE22020" Foreground="#FF44AA44"
        Minimum="0,, Maximum="100" Value="50" />

private void slider1_ValueChanged(object sender,
        RoutedPropertyChangedEventArgs<double> e)    {
        MessageBox.Show( slider1.Value.ToString() );
}
```


Eigenschaften Expander: ueb17

- Name
- IsExpanded
- ExpandDirection
  - Down, Schalter oben
  - Up, Schalter unten
  - Left, Schalter rechts
  - Right, Schalter links
- Header statt Content
- HorizontalAligment
  - Left / Right / Center / Stretch
- VerticalAligment
  - Top / Bottom / Center / Stretch
- Min- MaxHeight
- Min- MaxWidth
- Height / Width
- Margin / Padding
- FontSize etc.
- Opacity
- Tag
- ToopTip
- Visibility
- Z-Index

Beispiel Expander: ueb17





Beispiel Expander: ueb17

```
<DockPanel Background="Red"  LastChildFill="False" >
  <Button DockPanel.Dock="Left" Background="Blue"  Content="Bn1"  Name="bn1"  />
  <Button DockPanel.Dock="Left" Background="Green" Content="Bn2"  Name="bn2"   />
  <Button DockPanel.Dock="Top" Background="OldLace" Content="Bn3"  Name="bn3"   />
  <Expander DockPanel.Dock="Right" Header="Erweitert" ExpandDirection="Down">
     <DockPanel Background="Red"  LastChildFill="False" >
       <Button DockPanel.Dock="Right" Bg="Firebrick" Content="Bn6"  Name="bn6"   />
                    …
       <Button DockPanel.Dock="Right" Bg="Lavender" Content="Bn10"  Name="bn10"   />
       <DockPanel.LayoutTransform>
           <RotateTransform Angle="90" />
        </DockPanel.LayoutTransform>
     </DockPanel>
  </Expander>
   <Button DockPanel.Dock="Right" Bg="Crimson" Content="Bn11"  Name="bn11"   />
   <Button DockPanel.Dock="Left" Bg="Khaki" Content="Bn12"  Name="bn12"   />
    <Button DockPanel.Dock="Bottom" Bg="Coral" Content="Bn13"  Name="bn13"   />
</DockPanel>
```

Eigenschaften SplitContainer (WinForms):

- Das Splitten zweier Elemente wird nun über das Grid-Element und einem GridSplitter realisiert


```
<Grid Name="splitContainer1" SizeChanged="Grid_SizeChanged" >
    <Grid.RowDefinitions>
      <RowDefinition Height="*" />
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
      <ColumnDefinition Width="300"/>
      <ColumnDefinition Width="*"/>
    </Grid.ColumnDefinitions>
    <ListBox Grid.Column="0" Grid.Row="0" Name="list1"
        HorizontalAlignment="Stretch" VerticalAlignment="Stretch"
        Background="#FFeeee00" />
        <TextBox Grid.Column="1" Grid.Row="0"  TextWrapping="Wrap"
        VerticalScrollBarVisibility="Visible"  AcceptsReturn="True"
        HorizontalAlignment="Stretch"  Name="textBox2"
        VerticalAlignment="Stretch" Background="#FFccee00"  />
    <GridSplitter Name="gridSplitter1" VerticalAlignment="Top" Width="5"
                  Background="Red" Height="311" />
  </Grid>
```

Beispiel SplitContainer durch einen Grid

```
private void setGUI()    {
        list1.Items.Clear();
        for (int i = 1; i < 100; i++) list1.Items.Add(i.ToString());
}

private void Grid_SizeChanged(object sender, SizeChangedEventArgs e)  {
        RowDefinitionCollection liste = splitContainer1.RowDefinitions;
        if (liste.Count > 0)
        {
           RowDefinition row = liste[0];
           if (!double.IsNaN(row.ActualHeight))
           {
              gridSplitter1.Height = row.ActualHeight;
           }
        }
}
```


OpenFileDialog: using Microsoft.Win32;

```
OpenFileDialog openFileDialog1 = new OpenFileDialog();
openFileDialog1.Filter =
  "txt files (*.txt)|*.txt|pdf files (*.pdf)|*.pdf|All files (*.*)|*.*";
openFileDialog1.FilterIndex = 2;// zählt von 1 !
openFileDialog1.DefaultExt = ".txt";
openFileDialog1.InitialDirectory="c:\\daten\\0";
openFileDialog1.Multiselect=false;  // true
openFileDialog1.RestoreDirectory = true;
if ( openFileDialog1.ShowDialog() = = true ) {
   // Verarbeitung
}
```

OpenFileDialog: using Microsoft.Win32;

```
if ( openFileDialog1.ShowDialog() = = true ){
  string sFile=openFileDialog1.FileName.ToString();
}

if ( openFileDialog1.ShowDialog() = = true ){
      string[] sFiles = openFileDialog1.FileNames;
      foreach (string sFile in sFiles)
      {
          MessageBox.Show(sFile, Application.ProductName);
       }
}
```


SaveFileDialog: using Microsoft.Win32;

```
 SaveFileDialog saveFileDialog1 = new SaveFileDialog();
 saveFileDialog1.Filter =
   "txt files (*.txt)|*.txt|pdf files (*.pdf)|*.pdf|All files (*.*)|*.*";
saveFileDialog1.FilterIndex = 1;// zählt von 1 !
saveFileDialog1.DefaultExt = ".txt";
saveFileDialog1.InitialDirectory = "c:\\daten";
saveFileDialog1.RestoreDirectory = true;
if (saveFileDialog1.ShowDialog() == DialogResult.OK)
{
    string sFile = saveFileDialog1.FileName.ToString();
    MessageBox.Show(sFile, Application.ProductName);
}
```

Eigenschaften Menüs:

- Menüs sind normale Elemente
- Beinhaltet Objekte der Klasse MenuItem
- FontSize wird nicht vom Parent übernommen
- IsCheckable, IsChecked
- IsEnabled
- Untermenüs speichert man in Items
- Über Command greift man auf vorgefertigte Aktion zu
  - Command="ApplicationCommands.Copy"
  - Command="ApplicationCommands.Paste"
- CommandTarget: welches Elemente bekommt „Paste“
  - CommandTarget="{Binding ElementName=Editor2}"
- <Separator/>
- Fontsize etc.

Beispiel Menü: ueb18

```
<Menu DockPanel.Dock="Top"  FontSize="20">
  <MenuItem Header="_Datei" Name="mainFile" >
    <MenuItem Header="_Öffnen" Name="mnOpen" InputGestureText="Strg+O"/>
    <MenuItem Header="_Speichern" Name="mnSave"
                  Command="ApplicationCommands.SaveAs"/>
   <Separator/>
         <MenuItem Header="_Schließen" Name="mnClose"
                          Command="ApplicationCommands.Close"/>
  </MenuItem>
  <MenuItem Header="_Bearbeiten" Name="mainEdit">
       <MenuItem Header="_Kopieren" Name="mnCopy"
                          Command="ApplicationCommands.Copy"/>
      <MenuItem Header="_Einfügen" Name="mnPaste"
                          Command="ApplicationCommands.Paste"/>
   </MenuItem>
</Menu>
```

Eigenschaften ICommand:

- Mehrere Aktionen für
  - Menü „OpenFile“
  - Schalter „OpenFile“
  - Kurztaste „Strg+O“
- Vorgefertigte Command
  - Paste, Copy, Close etc.
- Eigene Command
  - using System.Windows.Input;
  - class OpenFile:Icommand
  - {

    Methoden: CanExecute, Execute, CanExecuteChanged
  - }

Beispiel ueb18: Klasse OpenFileCommand

```
using System.Windows;
using System.Windows.Input;
using Microsoft.Win32;

public class OpenFileCommand : ICommand   {
    public  bool CanExecute(object parameter)    {
      return true;
    }

    public  void Execute(object parameter)    {
      OpenFileDialog openFileDialog1 = new OpenFileDialog();
    }

    public  event EventHandler CanExecuteChanged;
  }
```

Beispiel ueb18: Klasse AppCommands

```
using System.Windows.Input;

// Singleton Pattern

public static class AppCommands
{
    private static ICommand openfileCommand = new OpenFileCommand();

    public static ICommand Openfile
    {
        get { return openfileCommand; }
    }

}
```


Beispiel ueb18: Klasse MainWindow.xaml

```
<Window x:Class="ueb18.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="clr-namespace:ueb18"
    Title="Beispiel Menu ueb18" Height="350" Width="525" FontSize="20">

  <DockPanel>
    <Menu DockPanel.Dock="Top"  FontSize="20">
      <MenuItem Header="_Datei" Name="mainFile" >
        <MenuItem Header="_Öffnen"  Name="mnOpen"
                Command="{x:Static local:AppCommands.Openfile}"/>
      </MenuItem>
    </Menu>

</DockPanel>
</Window>
```

Beispiel ueb19: Klasse OpenFileCommand

```
using System.Windows;
using System.Windows.Input;
using Microsoft.Win32;

public class OpenFileCommand : ICommand   {
      private static OpenFileCommand  instance  =  new  OpenFileCommand();
      public  static  ICommand   Instance   {
              get {  return instance;  }
       }
      public  bool CanExecute(object parameter)    {
         return true;
      }
      public  void Execute(object parameter)    {
         OpenFileDialog openFileDialog1 = new OpenFileDialog();
      }
      public  event EventHandler CanExecuteChanged;
   }
```


Beispiel ueb19: Klasse MainWindow.xaml

```
<Window x:Class="ueb19.MainWindow"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:local="clr-namespace:ueb19"
      Title="Beispiel Menu ueb19" Height="350" Width="525" FontSize="20">

   <DockPanel>
      <Menu DockPanel.Dock="Top"  FontSize="20">
         <MenuItem Header="_Datei" Name="mainFile" >
            <MenuItem Header="_Öffnen"  Name="mnOpen"
                     Command="{x:Static local:OpenFileCommands.Openfile}"/>
         </MenuItem>
      </Menu>
     <Button Content="Open" Command="{x:Static local:OpenFileCommands.Openfile}"/>
</DockPanel>
</Window>
```

Beispiel ueb20: Klasse MyCommands

```
using System.Windows.Input;
 public static class MyCommands
   {
      private static RoutedUICommand openfile;
      private static RoutedUICommand export2file;

      static MyCommands()     {
         openfile = new RoutedUICommand("Datei öfnen", "OpenFile", typeof(MyCommands));
         openfile.InputGestures.Add(
                        new KeyGesture(Key.O, ModifierKeys.Alt | ModifierKeys.Control));
         export2file = new RoutedUICommand("Export nach HTML", "Export2File",
                        typeof(MyCommands));
         export2file.InputGestures.Add(new KeyGesture(Key.E, ModifierKeys.Alt |
                        ModifierKeys.Control));
         // openfile.InputGestures.Add(new MouseGesture(MouseAction.LeftDoubleClick,
            ModifierKeys.Alt | ModifierKeys.Control));
      }
```


Beispiel ueb20: Klasse MyCommands

```
using System.Windows.Input;
 public static class MyCommands
   {
      private static RoutedUICommand openfile;
      private static RoutedUICommand export2file;

      public static RoutedUICommand Openfile
      {
         get { return openfile; }
      }

      public static RoutedUICommand Export2file
      {
         get { return export2file; }
      }

   }
```

Beispiel ueb20: Klasse MainWindow.xaml

```
<Window x:Class="ueb20.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="clr-namespace:ueb20"
    Title="Beispiel Menu ueb19" Height="350" Width="525" FontSize="20">
  <DockPanel>
    <Menu DockPanel.Dock="Top"  FontSize="20">
      <MenuItem Header="_Datei" Name="mainFile" >
        <MenuItem Header="_Speichern" Name="mnSave"
                Command="{x:Static local:MyCommands.Openfile}"/>
         <MenuItem Header="_Export" Name="mnExport"
                Command="{x:Static local:MyCommands.Export2file}"/>
      </MenuItem>
    </Menu>
</DockPanel>
</Window>
```


Beispiel ueb20: Klasse MainWindow.xaml.cs

```
using Microsoft.Win32;   // OpenDialog

 private void Window_Loaded(object sender, RoutedEventArgs e)     {
    CommandBindings.Add( new CommandBinding(
            ApplicationCommands.Open, OpenExecuted));
    CommandBindings.Add( new CommandBinding(
          MyCommands.Export2file, Export2FileExecuted, Export2FileCanExecuted));
  }

  private void OpenExecuted(object sender, ExecutedRoutedEventArgs e)     {
  }
  private void Export2FileCanExecuted(object sender, CanExecuteRoutedEventArgs e) {
        e.CanExecute = true;              //  this.sFilename!= null;
  }
   private void Export2FileExecuted(object sender, ExecutedRoutedEventArgs e)   {
   }
```

Eigenschaften StatusBar:

- Name
- Elemente heißen StatusBarItem
- Separator mit Width

- AlternationCount
- BackGround
- Cursor

- HorizontalAligment
  - Left / Right / Center / Stretch
- VerticalAligment
  - Top / Bottom / Center / Stretch
- Min- MaxHeight
- Min- MaxWidth
- Height / Width
- Margin / Padding
- FontSize etc.
- Opacity / Tag
- ToopTip
- Visibility / Z-Index

Beispiel StatusBar:

```
<StatusBar  Name="statusBar1"  >
        <StatusBarItem>"3333"</StatusBarItem>
        <Separator Width="40,,  Background="Red" />
        <StatusBarItem >"3333"</StatusBarItem>
</StatusBar>
```

Eigenschaften ToolBar und ToolBarTray

- Name
- AlternationCount
- BackGround
- Cursor
- Beinhaltet bel. Elemente
  - Variable Items
- Orientation (readOnly)
  - Horizontal
  - Vertical
- ToolBarTray.IsLocked
- ToolBar OverflowMode
  - ToolBar.OverflowMode="Always"
  - ToolBar.OverflowMode="AsNeeded"
  - ToolBar.OverflowMode="None"
- HorizontalAligment
  - Left / Right / Center / Stretch
- VerticalAligment
  - Top / Bottom / Center / Stretch
- Min- MaxHeight
- Min- MaxWidth
- Height / Width
- Margin / Padding
- FontSize etc.
- Opacity / Tag
- ToopTip
- Visibility / Z-Index

Beispiel ToolBar und ToolBarTray: ueb22

Beispiel ToolBar und ToolBarTray: ueb22

```
<Grid>
    <Grid.RowDefinitions>
      <RowDefinition Height="Auto" />
      <RowDefinition Height="Auto" />
      <RowDefinition Height="1*" />
      <RowDefinition Height="Auto" />
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
      <ColumnDefinition Width="*"/>
    </Grid.ColumnDefinitions>

        …

  </Grid>
```


Beispiel ToolBar und ToolBarTray: ueb22

```
<Menu Grid.Column="0" Grid.Row="0" FontSize="18" >
    <MenuItem Header="_Datei" Name="mainFile" >
      <MenuItem Header="_Öffnen" Name="mnOpen"  />
      <MenuItem Header="_Speichern" Name="mnSave" />
      <MenuItem Header="_Export" Name="mnExport" />
      <Separator/>
      <MenuItem Header="_Schließen" Name="mnClose" />
    </MenuItem>

    <MenuItem Header="_Bearbeiten" Name="mainEdit">
       <MenuItem Header="_Rückgängig" Name="mnUndo" />
        <Separator/>
        <MenuItem Header="_Kopieren" Name="mnCopy" />
    </MenuItem>
</Menu>
```

Beispiel ToolBar und ToolBarTray: ueb22

```
<ToolBarTray Grid.Column="0" Grid.Row="1">
        <ToolBar Band="1">
          <Button Name="BnOpen">Öffnen</Button>
          <Button Name="BnNew">Neu</Button>
          <Button Name="BnSave">Speichern</Button>
        </ToolBar>
        <ToolBar Band="2">
          <Label>Zoom</Label>
          <ComboBox>
            <ComboBoxItem>400%</ComboBoxItem>
            <ComboBoxItem>300%</ComboBoxItem>
           <ComboBoxItem>10%</ComboBoxItem>
          </ComboBox>
        </ToolBar>
</ToolBarTray>
```


Beispiel ToolBar und ToolBarTray: ueb22

```
<TextBox Grid.Column="0" Grid.Row="2"  Name="editor"
          TextWrapping="Wrap"  VerticalScrollBarVisibility="Visible"
          AcceptsReturn="True" />

<StatusBar  Grid.Column="0" Grid.Row="3" Name="statusBar1"  FontSize="14" >
        <StatusBarItem>x: 33</StatusBarItem>
        <Separator Width="2"   />
        <StatusBarItem >y: 123</StatusBarItem>
        <Separator Width="2"  Background="Black" />
        <StatusBarItem>Maßstab:</StatusBarItem>
        <StatusBarItem >
          <TextBox Width="100">1:10000</TextBox>
        </StatusBarItem>
  </StatusBar>
          ...
   </Grid>
```

Eigenschaften ListView, SubKlasse von ListBox

- Name
- AlternationCount
- BackGround
- Cursor
- Content statt Text
- GroupStyle
- FontSize etc.
- SelectionIndex
- SelectionValue
- SelectionValuePath
- SelectionMode: Extended
- Items // speichert die Elemente
- **View, bestimmt Aussehen**
- HorizontalAligment
  - Left / Right / Center / Stretch
- VerticalAligment
  - Top / Bottom / Center / Stretch
- Min- MaxHeight
- Min- MaxWidth
- Height / Width
- Margin / Padding / Opacity
- Tag / ToopTip
- Visibility / Z-Index
- Arbeitet mit
  - GridView
  - GridViewColumn

Beispiel ListView ueb25

```
<ListView Grid.Column="0" Grid.Row="0"   Name="listView1"
          GridViewColumnHeader.Click="ColumnHeader_Click" >
  <ListView.View>
    <GridView>
      <GridView.Columns>
        <GridViewColumn Width="60"  Header="FirstName"
                        DisplayMemberBinding="{Binding FirstName}" />
        <GridViewColumn Width="70"  Header="LastName"
                        DisplayMemberBinding="{Binding LastName}" />
        <GridViewColumn Width="135" Header="City"
                        DisplayMemberBinding="{Binding City}" />
      </GridView.Columns>
    </GridView>
  </ListView.View>
</ListView>
```

Beispiel ListView ueb25

```
class MyListItem  {
    private string sFirstName = "";
    private string sLastName = "";
    private string sCity = "";

    public MyListItem(string FirstName,
      string LastName, string City)   {
      this.sFirstName = FirstName;
      this.sLastName = LastName;
      this.sCity = City;
    }
        public string FirstName
        {
          get { return sFirstName; }
          set { sFirstName = value; }
        }
```

```
public string FirstName
{
  get { return sFirstName; }
  set { sFirstName = value; }
}

public string LastName
{
  get { return sLastName; }
  set { sLastName = value; }
}
public string City
{
  get { return sCity; }
  set { sCity = value; }
}
```


Beispiel ListView ueb25

```
private void setGUI()
    {
      listView1.Items.Add(new MyListItem("Ralf", "Meier", "WR"));
      listView1.Items.Add(new MyListItem("Frank", "Schulze", "MD"));
    }

private void ColumnHeader_Click(object sender, RoutedEventArgs e)  {
    GridViewColumnHeader h = e.OriginalSource as GridViewColumnHeader;
    ListView lv = (ListView) sender;
      // sortBy ist entweder FirstName, LastName oder City
     string sortBy = (h.Column.DisplayMemberBinding as Binding).Path.Path;

      lv.Items.SortDescriptions.Clear();
      SortDescription sd = new SortDescription(sortBy, ListSortDirection.Ascending);
      lv.Items.SortDescriptions.Add(sd);
      lv.Items.Refresh();
}
```

Beispiel ListView ueb26: Alles per c#

```
private void setGUI()      {
        GridView gridView = new GridView();
        GridViewColumn col;
        col = new GridViewColumn();
        col.Header="FirstName";
        col.DisplayMemberBinding = new Binding("FirstName");
        col.Width = 60;
        gridView.Columns.Add(col);

          …

        listView1.View = gridView;

        listView1.Items.Add(new MyListItem("Ralf", "Meier", "WR"));
        listView1.Items.Add(new MyListItem("Frank", "Schulze", "MD"));
    }
```


Beispiel ListView ueb27:

- ListView mit CheckBox

Beispiel ListView ueb28:

- ListView mit Bilder

Beispiel ListView ueb29:

- ListView mit Editoren

Eigenschaften TabControl:

- Name
- AlternationCount
- BackGround
- Cursor
- Items // speichert die Elemente
  - TabItem
    - Speichert die Elemente
    - Darf aber nur **ein** root haben
    - Header (Text)
- SelectionItem
- SelectionIndex
- SelectionValue
- HorizontalAligment
  - Left / Right / Center / Stretch
- VerticalAligment
  - Top / Bottom / Center / Stretch
- Min- MaxHeight
- Min- MaxWidth
- Height / Width
- Margin / Padding
- FontSize etc.
- Opacity / Tag
- ToopTip
- Visibility / Z-Index

Eigenschaften TabControl:

- Einfügen eines TabControls
- Einfügen des ersten TabItem
  - Beschriftung, Header, setzen
  - Layout definieren
  - Weitere Elemente einfügen
- Einfügen des zweiten TabItem
  - Beschriftung, Header, setzen
  - Layout definieren
  - Weitere Elemente einfügen

Beispiel TabControl: ueb30

Beispiel TabControl: ueb30

Beispiel TabControl: ueb30

Beispiel TabControl: ueb31: Manuelles Einfügen

```
private void BnInsert_Click(object sender, RoutedEventArgs e)    {
        TextBox editor = new TextBox();
        editor.Name = "textBox" + (tabControl1.Items.Count + 1).ToString();
        editor.TextWrapping = TextWrapping.Wrap;  // WrapWithOverflow  NoWrap
        editor.VerticalScrollBarVisibility = ScrollBarVisibility.Auto;
        editor.AcceptsReturn = true;
        editor.HorizontalAlignment = HorizontalAlignment.Stretch;
        editor.VerticalAlignment = VerticalAlignment.Stretch;
        editor.Background = Brushes.Bisque;

        TabItem reg = new TabItem();
        reg.Header = (tabControl1.Items.Count + 1).ToString() + ". Register";
        reg.Content = editor;

        tabControl1.Items.Add(reg);
        tabControl1.SelectedIndex = tabControl1.Items.Count - 1;
}
```

Tree / Baum

- **Eigenschaften**
  - Darstellung hierarchischer Elemente
  - Unterscheidung Verzweigung / Blätter
  - Symbole (Geschlossen / aufgeklappt)
  - Häufig in Verbindung mit einer ListView / Explorer-Fenster
  - Erweiterte Eigenschaften in Knoten (aufklappbar, Node-Id))

- **Beispiele:**
  - Dateisystem (Explorer)
  - Darstellen der Objektstruktur einer Datei
  - Darstellen aller Teiler einer Zahl

Eigenschaften TreeView:

- Name
- AlternationCount
- BackGround
- Cursor
- Items // speichert die Elemente
  - **TreeViewItem**
  - Header="root" (Caption)
  - IsExpanded="True„
  - IsSelected
  - MultiElemente !
  - **z. B. RadioButton**
- SelectionIndex
- SelectionValue
- SelectionValuePath
- HorizontalAligment
  - Left / Right / Center / Stretch
- VerticalAligment
  - Top / Bottom / Center / Stretch
- Min- MaxHeight
- Min- MaxWidth
- Height / Width
- Margin / Padding
- FontSize etc.
- Opacity / Tag
- ToopTip
- Visibility / Z-Index

Beispiel TreeView: ueb32/33

- WrapPanel: ueb32
- DockPanel: ueb33

Beispiel TreeView: ueb34

Beispiel TreeView: ueb34

```
private void BnInsert_Click(object sender, RoutedEventArgs e)    {
        TreeViewItem root, item1, item2, item3;
        tree1.Items.Clear();
        root = new TreeViewItem();
        root.Header = "root";
        root.Foreground = Brushes.Chocolate;
        root.IsExpanded = true;
        tree1.Items.Add(root);

        item1 = new TreeViewItem();
        item1.Header = "1. Knoten";
        item1.Foreground = Brushes.Chocolate;
        root.Items.Add(item1);
}
```



```
<TreeView Grid.Column="0" Grid.Row="0"  Name="tabControl1" >
    <TreeViewItem Header="root" IsExpanded="True" Foreground="#FF0000FF">
      <TreeViewItem Header="1. Knoten" IsExpanded="True">aaaaa</TreeViewItem>
      <TreeViewItem Header="2. Knoten">
        <GroupBox>
          <WrapPanel>
            <RadioButton Margin="10">2a. Knoten</RadioButton>
            <RadioButton Margin="10">2b. Knoten</RadioButton>
            <RadioButton Margin="10">2c. Knoten</RadioButton>
          </WrapPanel>
        </GroupBox>
      </TreeViewItem>
      <TreeViewItem Header="3. Knoten" Tag="abcs" >
        <TreeViewItem Header="4. Knoten">
          <CheckBox>Hier eine CheckBox</CheckBox>
        </TreeViewItem>
        <TreeViewItem Header="5. Knoten">5. Knoten</TreeViewItem>
      </TreeViewItem>
      <TreeViewItem Header="6. Knoten"></TreeViewItem>
    </TreeViewItem>
```

Tree: Beispiel

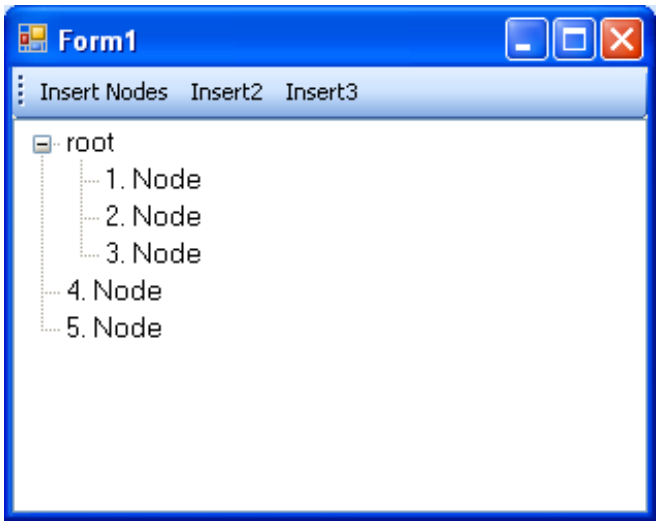
Tree: 1. Beispiel

```
treeView1.Nodes.Clear();

treeView1.Nodes.Add(new TreeNode("1. Node"));
treeView1.Nodes.Add(new TreeNode("1. Node"));
treeView1.Nodes.Add(new TreeNode("2. Node"));
treeView1.Nodes.Add(new TreeNode("3. Node"));
```

Tree: 1. Beispiel

```
TreeNode node;
TreeNode root;
treeView1.Nodes.Clear();
root = new TreeNode("root");
treeView1.Nodes.Add(root);
node=new TreeNode("1. Node");
  root.Nodes.Add(node);
node=new TreeNode("2. Node");
  root.Nodes.Add(node);
node=new TreeNode("3. Node");
  root.Nodes.Add(node);
treeView1.Nodes.Add(new TreeNode("4. Node"));
treeView1.Nodes.Add(new TreeNode("5. Node"));
```

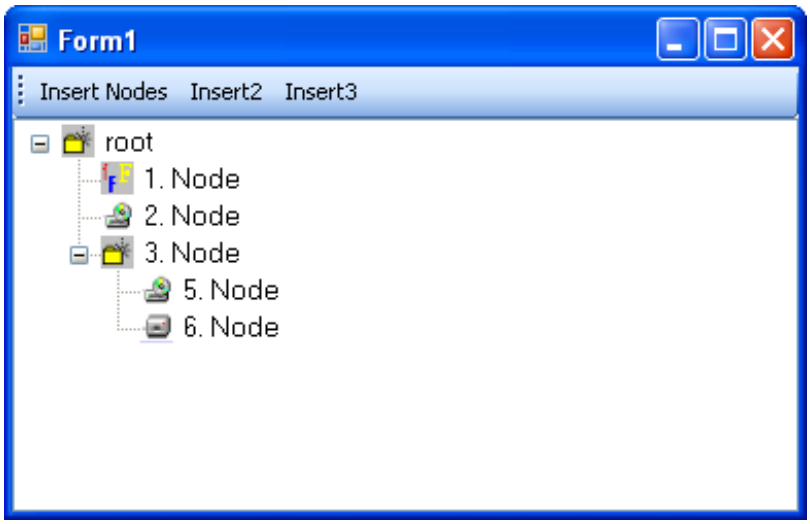

Tree: Beispiel



Tree: 3. Beispiel

```
TreeNode node, node2;
TreeNode root;
treeView1.BeginUpdate(); // Unterdrücken des Neuzeichnens
treeView1.ImageList = imageList1;          // an ImageList binden
treeView1.Nodes.Clear();
root = new TreeNode("root",1,2);       // Symbole (markiert und nicht markiert)
  treeView1.Nodes.Add(root);
node = new TreeNode("1. Node",2,3);
root.Nodes.Add(node);
  node = new TreeNode("2. Node",3,4);
root.Nodes.Add(node);
  node2 = new TreeNode("3. Node",1,3);
treeView1.EndUpdate();  // Neu Zeichnen
root.ExpandAll();
```


Tree: 3. Beispiel



Klasse als "Pointer" an jedem TreeNode

```
class CStudent
  {
    public string name;
    public int matrnr;

    public CStudent(string name, int matrnr)
    {
      this.name = name;
      this.matrnr = matrnr;
    }
  }
```



```
private void BnInsert_Click(object sender, EventArgs e)
{
        TreeNode node, node1, node2, node3;
        TreeNode root;
        treeView1.BeginUpdate(); // Unterdrücken des Neuzeichnens
        treeView1.ImageList = imageList1;          // an ImageList binden
        treeView1.Nodes.Clear();
        root = new TreeNode("root", 1, 2);  // Symbole geschlossen offen
        treeView1.Nodes.Add(root);

        node = addNode(root, "1. Node", 12345,1,2);  // Erzeugt Instanz CStudent
        node1 = addNode(root, "2. Node", 13345,2,3);
        node2 = addNode(root, "3. Node", 13345, 1, 4);
        node = addNode(node2, "4. Node", 43345, 3, 4);
        node3 = addNode(node2, "5. Node", 23345, 6, 1);
        treeView1.EndUpdate();  // Neu Zeichnen
        root.ExpandAll();
}
```

Klasse als "Pointer" an jedem TreeNode

```
private TreeNode addNode(TreeNode parent,
        string name, int matrnr, int sym1, int sym2)
{
        CStudent std;
        TreeNode node;
        std = new CStudent(name, matrnr);
        node = new TreeNode(name, sym1, sym2);
        node.Tag = std;
        parent.Nodes.Add(node);
        return node;
}
```

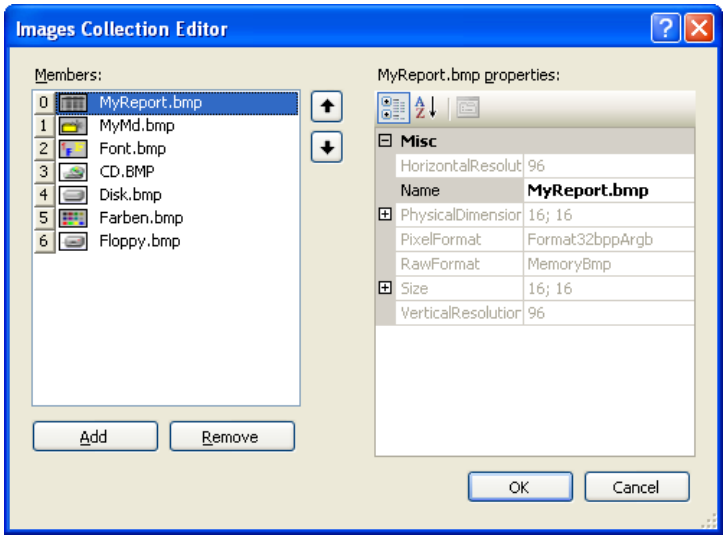


```
private void treeView1_AfterSelect(object sender, TreeViewEventArgs e)  {
    TreeNode node = treeView1.SelectedNode;
    if (node != null)
    {
      CStudent std;
      string sStr1, sStr2, sStr3;
      std = (CStudent) node.Tag;
      sStr1 = "Pfad  : " + node.FullPath;
      if (std != null)  // wenn root angeklickt !
      {
        sStr2 = "Name  : " + std.name;
        sStr3 = "Matrnr: " + std.matrnr.ToString();
        MessageBox.Show(sStr1 + "\r\n" + sStr2 + "\r\n" + sStr3, "Tree Click");
      }
      else
      {
        MessageBox.Show(sStr1, "Tree Click (root click)");
      }
    }
}
```

Tree: Aufgabe

- Erstellen eines neuen Projektes
- Einfügen eines Trees
- Doppelklick, form_load erstellen
  - treeView1.Dock = DockStyle.Fill;
- Imagelist einfügen
- Anklicken
- Property-Fenster
- Eintrag "Collection"
- Einfügen der Symbole (Zip-Datei, siehe Homepage)
- Toolstrip einfügen
- Schalter einfügen
- Doppelklick: **Bestimme alle Teiler einer Zahl (1 bis 50)**

Tree: Aufgabe



Tree: Aufgabe

```
private TreeNode addNode(TreeNode parent,
        string name, int sym1, int sym2)
{
    TreeNode node = new TreeNode(name, sym1, sym2);
    parent.Nodes.Add(node);
    return node;
}

private void insertTeiler(TreeNode parent, int nr)      {
        int i, j;
        TreeNode node1, node2;
        …
}
```


Tree: Aufgabe

```
private void BnInsert_Click(object sender, EventArgs e)
{
        TreeNode root;
        int i,j;
        treeView1.BeginUpdate(); // Unterdrücken des Neuzeichnens
        treeView1.ImageList = imageList1;        // an ImageList binden
        treeView1.Nodes.Clear();
        root = new TreeNode("root", 1, 2);  // Symbole geschlossen offen
        treeView1.Nodes.Add(root);
          // Aktion
        treeView1.EndUpdate();  // Neu Zeichnen
        root.ExpandAll();
    }
```