

Praktische Informatik

Vorlesung 05

Einfache Steuerelemente

Zuletzt haben wir gelernt...

- Was die Windows Presentation Foundation genau ist.
- Wie man mit Hilfe von XAML Benutzeroberflächen definiert.
- Was der sog. Code Behind ist.
- Was Routed Events sind.
- Was Dependency Properties sind.

Inhalt heute

- Steuerelemente in der WPF
- Generelle Eigenschaften der Klasse Control
- Content Controls
- Label, Button, RadioButton und Checkbox
- TextBox
- Slider

Steuerelemente

- Um eine Benutzeroberfläche zu erstellen, werden **Steuerelemente (engl. Controls)** benötigt.
 - Das WPF-Framework verfügt über eine reichhaltige Auswahl an vordefinierten Steuerelementen.
 - Button, TextBox, Menü, Toolbar, ...
- Es existieren entsprechende Klassen, aus denen Objekte erzeugt werden.
 - Das kann in XAML oder im Code Behind geschehen.
- Die Objekte stellen die eigentlichen Steuerelemente in einer Oberfläche dar.
 - Das Aussehen der Steuerelemente wird über Eigenschaften (oft Dependency Properties) angepasst.

FrameworkElement und Control

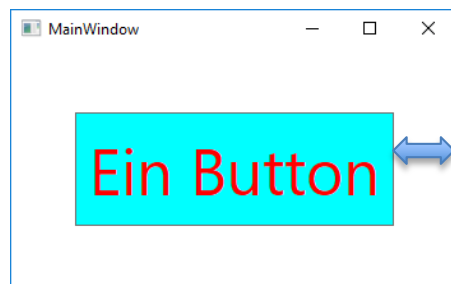
- Viele (nicht alle) Klassen für Steuerelemente erben von der Basisklasse `Control`, die wiederum von `FrameworkElement` ableitet.
 - `Window`, `Button`, ...
- Entsprechend besitzen alle Kindklassen die Eigenschaften und Ereignisse dieser Basisklassen.
 - Siehe auch in der [online Dokumentation](#).

Eigenschaft	Bedeutung
Background	Definiert den Hintergrund in Form eines Objekts vom Typ <code>Brush</code> .
Foreground	Definiert die Vordergrundfarbe.
FontSize	Die Schriftgröße.
Padding	Legt den inneren freien Rand in Form eines Objekts vom Typ <code>Thickness</code> fest.
Margin	Legt den äußeren Rand in Form eines Objekts vom Typ <code>Thickness</code> fest.
Width, Height	Breite und Höhe in einer geräteunabhängigen Einheit von 1/96 Zoll pro Einheit.

Beispiel für Eigenschaften

- Ein Button (Klasse Button) in XAML erstellen:

```
<Button Padding="10" Margin="50" FontSize="50" Background="Aqua"
Foreground="Red">Ein Button</Button>
```



Dies ist der äußere
Rand, der sog.
Margin.

- Der gleiche Button, aber im Code Behind erstellt:

```
Button b = new Button();
b.Content = "Ein Button";
b.Margin = new Thickness(50);
b.Padding = new Thickness(10);
b.Foreground = Brushes.Red;
b.Background = Brushes.Aqua;
b.FontSize =
panel.Children.Add(b);
```

Ereignisse

- Die Basisklassen `Control` und `FrameworkElement` stellen auch bereits eine Menge von Ereignissen bereit.
 - Über entsprechende Ereignis-Handler im Code Behind kann dann auf das Eintreten des Ereignisses reagiert werden.
- Die folgenden Ereignisse sind z.B. bereits definiert:

Ereignis	Bedeutung
GotFocus	Tritt ein, wenn das Steuerelement den Eingabefocus erhält.
Loaded	Tritt ein, wenn das Steuerelement geladen und gelayoutet wurde.
MouseDown	Tritt ein, wenn auf dem Steuerelement eine Maustaste gedrückt wurde.
MouseEnter	Tritt ein, wenn der Mauszeiger auf das Steuerelement verschoben wurde.
MouseLeave	Tritt ein, wenn der Mauszeiger vom Steuerelement herunter bewegt wurde.
KeyDown	Tritt ein, wenn eine Taste der Tastatur betätigt wurde, während das Steuerelement den Eingabefocus besitzt.

Beispiel für Ereignisse

- Ein Label in XAML:

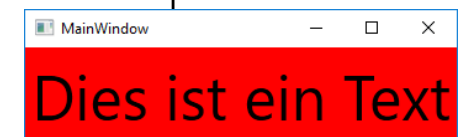
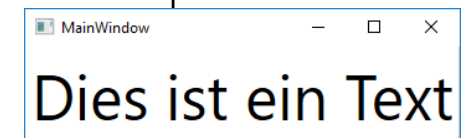
```
<Label FontSize="50" MouseEnter="Label_MouseEnter" MouseLeave="Label_MouseLeave">Dies ist ein Text</Label>
```

- Der zugehörige Code Behind:

```
public partial class MainWindow : Window
{
    public MainWindow()
    {
        InitializeComponent();
    }

    private void Label_MouseEnter(object sender, MouseEventArgs e)
    {
        var label = (Label)sender;
        label.Background = Brushes.Red;
    }

    private void Label_MouseLeave(object sender, MouseEventArgs e)
    {
        var label = (Label)sender;
        label.Background = Brushes.White;
    }
}
```



Zugriff auf Elemente

- Man kann im Code Behind auf alle Steuerelemente des zugehörigen XAML zugreifen.
 - Man kann dann zur Laufzeit deren Eigenschaften ändern.
 - Die Hintergrundfarbe oder der Inhalt einer Textbox verändern oder auslesen.
- Dafür muss das Element in XAML aber benannt werden.
 - Die Eigenschaft „x:Name“ muss gesetzt werden.
 - Im Code Behind ist das Element unter diesem Namen verfügbar.

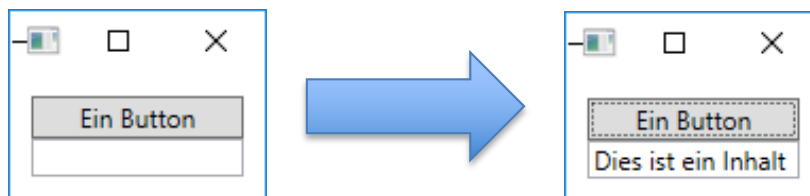
Beispiel

- Ein Button und eine Textbox im XAML:

```
<Button Click="Button_Click">Ein Button</Button>
<TextBox x:Name="textbox"></TextBox>
```

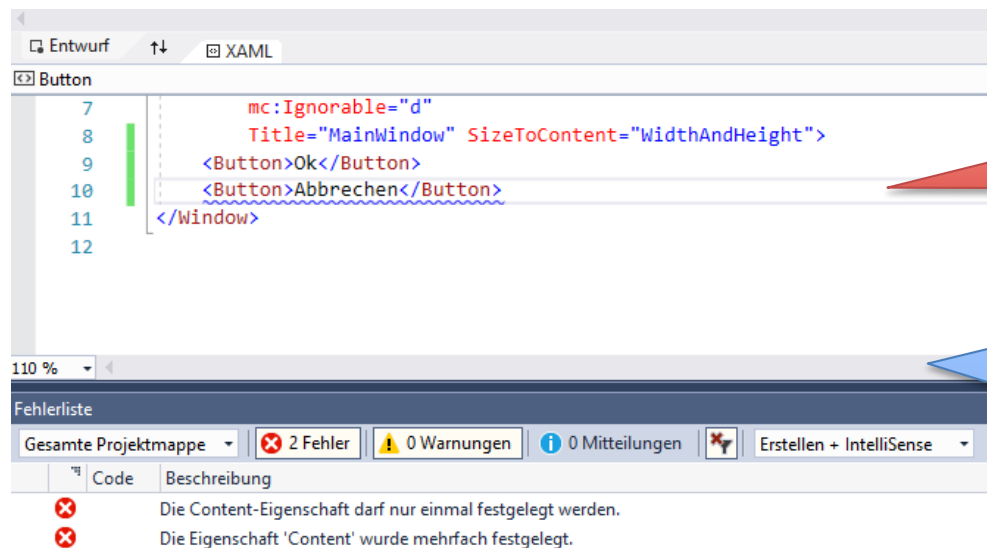
- Verändern des Inhalts der Textbox durch Click des Buttons:

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    textbox.Text = "Dies ist ein Inhalt";
}
```



Mehrere Elemente in der Oberfläche

- Im letzten Beispiel hatten wir schon zwei Steuerelemente in unserer Oberfläche.
 - Einen Button und eine TextBox.
- Wenn wir mehrere Elemente in den Content-Bereich unseres Window hinzufügen wollen, bekommen wir allerdings eine Fehlermeldung.



Achtung: Ein Window kann nur ein Kind-Element aufnehmen!

Für mehrere Elemente benötigen wir zwingend sog. Layoutcontainer. Diese werden wir in der nächsten Vorlesung kennen lernen!

Content Controls

- Der Button aus dem vorherigen Beispiel ist ein sog. ContentControl.
 - ContentControls besitzen die Eigenschaft „Content“.
- Dieser Eigenschaft lässt sich ein beliebiges Objekt zuweisen.
 - z.B. Text, als Bezeichnung des Buttons.
 - Im Gegensatz dazu existieren die sog. ItemsControls, die viele Objekte verwalten können (Siehe Vorlesung 8).
- Man kann dem Content aber auch andere Steuerelemente zuweisen.
 - Diese Elemente werden dann auf dem Button dargestellt.
 - Dies wird auch als **flexibles Inhaltsmodell** der WPF bezeichnet.

Beispiel

- Der Eigenschaft „Content“ eines ContentControls lässt sich auf zwei Arten etwas zuweisen:

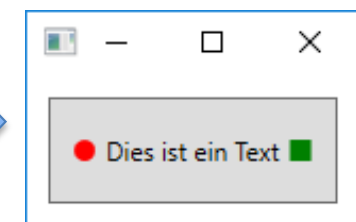
```
<Button Content="Dies ist ein Button" />
```

- Oder so:

```
<Button>Dies ist ein Button</Button>
```

- Der Content eines ContentControls kann auch etwas ganz anderes sein, als nur einfacher Text:

```
<Button Margin="10">
  <StackPanel Orientation="Horizontal" Margin="10">
    <Ellipse Width="10" Height="10" Fill="Red" />
    <TextBlock Padding="5">Dies ist ein Text</TextBlock>
    <Rectangle Width="10" Height="10" Fill="Green" />
  </StackPanel>
</Button>
```

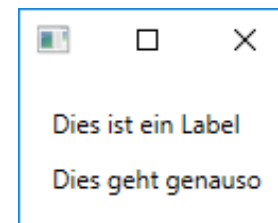


Label

- Ein Label ist ein sehr einfaches ContentControl.
 - Es wird üblicherweise dazu genutzt, um Textboxen mit einer Beschriftung zu kennzeichnen.
- Wie bei ContentControls üblich wird der Bezeichner über das „Content“-Property zugewiesen.
 - Man kann über das Target-Property auch angeben, wofür das Label die Beschriftung darstellt.
- Beispiele:

```
<Label>Dies ist ein Text</Label>
```

```
<Label Content="Dies geht genauso" />
```



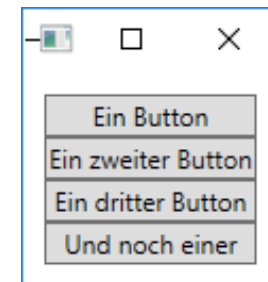
Button

- Die Klasse Button ist ebenfalls ein ContentControl.
 - Sie kann in XAML oder Code Behind benutzt werden, um einen Druckknopf zu realisieren.
- Die Klasse definiert nur wenige besondere Eigenschaften:
 - IsDefault: Legt fest, ob der Button betätigt wird, wenn die Enter-Taste gedrückt wird.
 - IsCancel: Legt fest, ob der Button betätigt wird, wenn die ESC-Taste gedrückt wird.
- Das wichtigste Ereignis ist sicher „Click“.
 - Dieses wird ausgelöst, wenn der Nutzer auf den Button klickt, oder bei gleichzeitigem Focus die Enter/Leertaste-Taste gedrückt wird.

Beispiel Button

- Einige Buttons definieren und das Click-Ereignis mit einem Handler verbinden:

```
<Button Click="Button_Click">Ein Button</Button>
<Button Click="Button_Click">Ein zweiter Button</Button>
<Button Click="Button_Click">Ein dritter Button</Button>
<Button Click="Button_Click">Und noch einer</Button>
```



- Der Click-Handler im Code Behind:

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    var button = (Button)sender;
    MessageBox.Show(button.Content.ToString());
}
```

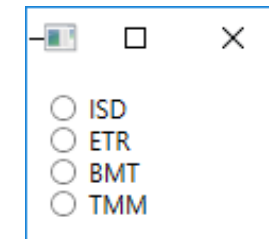

Optionsfelder

- **Optionsfelder (eng. Radiobuttons)** dienen dazu, eine einzelne Option aus wenigen Möglichkeiten auszuwählen.
 - Eine einzelne Option kann entweder ausgewählt (IsChecked==true) oder nicht ausgewählt sein (IsChecked==false).
 - Wird eine einzelne Option ausgewählt, so kann keine andere der Möglichkeiten ausgewählt sein.
 - z.B. das Geschlecht einer Person.
- In der WPF wird ein einzelnes Optionsfeld durch die Klasse `RadioButton` realisiert.
 - Ein `RadioButton` ist ebenfalls ein `ContentControl`.
- Die `Radiobuttons` gehören zu einer Gruppe, wenn sie zu einem gemeinsamen Panel gehören.
 - Innerhalb einer Gruppe kann nur ein `Radiobutton` selektiert sein.
 - Panels werden wir in der nächsten Vorlesung behandeln.

Beispiel Radiobutton

- Einige Radiobuttons in XAML:

```
<RadioButton Checked="RadioButton_Checked">ISD</RadioButton>
<RadioButton Checked="RadioButton_Checked">ETR</RadioButton>
<RadioButton Checked="RadioButton_Checked">BMT</RadioButton>
<RadioButton Checked="RadioButton_Checked">TMM</RadioButton>
```



- Im Code Behind kann im Event Handler dann auf eine Auswahl reagiert werden.

```
private void RadioButton_Checked(object sender, RoutedEventArgs e)
{
    var radiobutton = (RadioButton)sender;
    MessageBox.Show("Es wurde '" + radiobutton.Content + "' ausgewählt!");
}
```

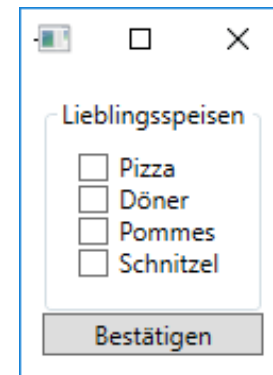
Kontrollkästchen

- **Kontrollkästchen** (engl. Checkboxes) ermöglichen die Abfrage von mehreren gleichberechtigten Optionen vom Benutzer eines Dialogs.
 - Eine einzelne Option kann entweder ausgewählt (`true`) oder nicht ausgewählt (`false`) worden sein.
 - Das Auswählen einer einzelnen Option hat keine Auswirkungen auf die anderen Optionen.
 - z.B. Angabe von Lieblingsfilmen, Aktivieren von Zugriffsrechten bei einem Benutzer, ...
- In der WPF wird ein einzelnen Kontrollkästchen durch die Klasse `CheckBox` realisiert.
 - Eine `CheckBox` ist ebenfalls ein `ContentControl`.

Beispiel Kontrollkästchen

- Es werden ein paar Checkboxes in einer GroupBox mit Überschrift angeordnet.
 - Beim Click auf den Button soll die Auswahl angezeigt werden.
 - Das Panel mit den Checkboxes wird benannt, damit man auf die Elemente im Code Behind zugreifen kann.

```
<StackPanel Margin="10">
    <GroupBox Header="Lieblingsspeisen" Padding="10">
        <StackPanel x:Name="checkboxes">
            <CheckBox x:Name="pizza">Pizza</CheckBox>
            <CheckBox x:Name="doener">Döner</CheckBox>
            <CheckBox x:Name="pommes">Pommes</CheckBox>
            <CheckBox x:Name="schnitzel">Schnitzel</CheckBox>
        </StackPanel>
    </GroupBox>
    <Button Click="Button_Click">Bestätigen</Button>
</StackPanel>
```



Beispiel Kontrollkästchen

- Die Collection „Children“ enthält alle Elemente in dem Panel.
 - Entsprechend kann man über diese iterieren und so die selektierten Elemente herausfinden:

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    var options = new List<string>();
    foreach (var child in checkboxes.Children)
    {
        if (child is CheckBox)
        {
            var checkbox = (CheckBox)child;
            if (checkbox.IsChecked.HasValue && checkbox.IsChecked.Value)
                options.Add(checkbox.Content.ToString());
        }
    }

    var options_as_string = String.Join(", ", options.ToArray());
    MessageBox.Show(options_as_string);
}
```

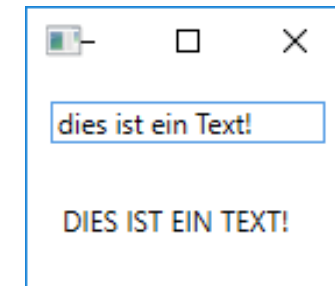
TextBox

- Die Klasse TextBox repräsentiert ein Eingabefeld für Text.
 - Die Klasse ist kein ContentControl, sondern erbt direkt von der Basisklasse Control.
 - Entsprechend verfügt die TextBox über alle Eigenschaft der Klasse Control.
- Die TextBox verfügt über einige eigene Eigenschaften und Ereignisse:

Art	Name	Beschreibung
Eigenschaft	Text	Der Inhalt der TextBox als string.
Ereignis	SelectionChanged	Wird geworfen, wenn der Nutzer einen Text markiert.
Ereignis	TextChanged	Wird geworfen, wenn der Inhalt der TextBox geändert wird.

Beispiel TextBox

- Der Inhalt einer TextBox soll in einem Label in Großbuchstaben angezeigt werden.



- XAML:

```
<TextBox Margin="10" x:Name="text" TextChanged="TextBox_TextChanged"></TextBox>
<Label Margin="10" x:Name="label"></Label>
```

- Code Behind:

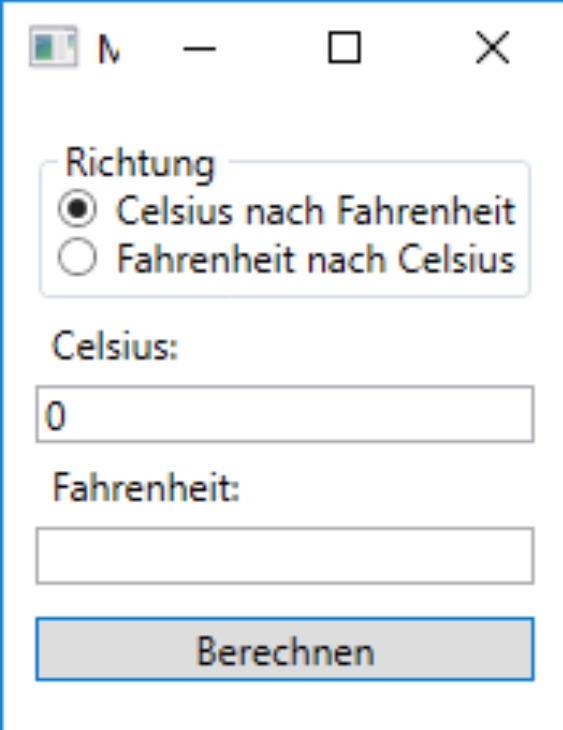
```
private void TextBox_TextChanged(object sender, TextChangedEventArgs e)
{
    label.Content = text.Text.ToUpper();
}
```

Erstellen einer GUI

- Mit Hilfe der nun gesehen Controls lassen sich schon recht gut kleine Anwendungen erstellen.
 - Die Controls können gut miteinander kombiniert werden.
 - Es entsteht eine grafische Benutzeroberfläche (GUI).
- Mit Hilfe von TextBoxen, Radio- und Checkboxes können z.B. Daten vom Benutzer eingelesen werden.
 - Durch Buttons kann man dann Berechnungen anstoßen.
- Wir wollen an einem echten Beispiel sehen, wie man diese Fähigkeiten nutzen kann.

Aufgabe

- Erstellen Sie ein Programm mit GUI, welches in der Lage ist, Temperaturen zwischen Celsius und Fahrenheit umzurechnen.
- Die Richtung der Umrechnung (C->F oder F->C) soll über RadioButtons eingestellt werden können.
- Die Temperaturen sollen in TextBoxen ein- bzw. ausgegeben werden.



The screenshot shows a Windows-style application window with a title bar containing a maximize button, a minus button, and a close button. The window contains the following elements:

- A group box labeled "Richtung" containing two radio buttons:
 - ☒ Celsius nach Fahrenheit
 - ☐ Fahrenheit nach Celsius
- A label "Celsius:" followed by a text box containing the value "0".
- A label "Fahrenheit:" followed by an empty text box.
- A button labeled "Berechnen" at the bottom.

XAML des Temperaturumrechners

```
<StackPanel Margin="10">
  <GroupBox Header="Richtung">
    <StackPanel>
      <RadioButton x:Name="cToF" IsChecked="True">Celsius zu Fahrenheit</RadioButton>
      <RadioButton x:Name="fToC">Fahrenheit zu Celsius</RadioButton>
    </StackPanel>
  </GroupBox>
  <Label Target="c">Celsius:</Label>
  <TextBox x:Name="c">0</TextBox>
  <Label Target="f">Fahrenheit:</Label>
  <TextBox x:Name="f"></TextBox>
  <Button Margin="0 10 0 0" Click="Button_Click" IsDefault="True">Berechnen</Button>
</StackPanel>
```

Code Behind

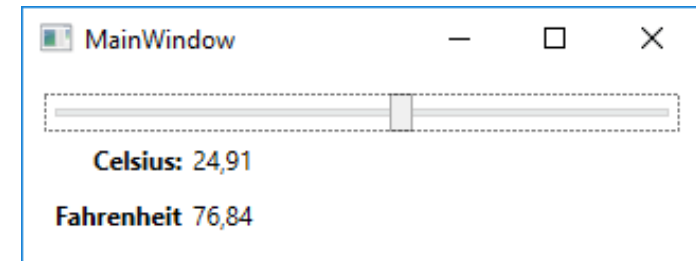
```
private void Button_Click(object sender, RoutedEventArgs e)
{
    if (cToF.IsChecked.HasValue && cToF.IsChecked.Value)
    {
        double c_value = Convert.ToDouble(c.Text);
        double f_value = c_value * 1.8 + 32;
        f.Text = f_value.ToString();
    }
    else
    {
        double f_value = Convert.ToDouble(f.Text);
        double c_value = (f_value - 32) * 0.5555;
        c.Text = c_value.ToString();
    }
}
```

Slider

- Die Klasse **Slider** stellt einen sog. **Schiebebalken** dar.
 - Dadurch kann man aus einem vorher eingestellten Wertebereich einen einzelnen Wert sehr einfach auswählen.
- Der Slider gehört zu den sog. **RangeControls**.
 - Neben dem Slider gehören auch die ProgressBar (Fortschrittsbalken) und die ScrollBar (Scrollbalken) zu diesen Controls.
- Für die Nutzung des Sliders sind zunächst die folgenden Eigenschaften wichtig:
 - Value: Der aktuell eingestellte Wert.
 - Minimum: Der kleinste einstellbare Wert.
 - Maximum: Der größte einstellbare Wert.
- Das wichtigste Ereignis ist sicher ValueChanged.
 - Dieses wird genau dann geworfen, wenn der eingestellte Wert verändert wurde.

Aufgabe

- Erstellen Sie ein neues Programm, welches einen Slider benutzt, um eine Temperatur in Grad Celsius einzustellen.
- In zwei Labels soll jeweils die Temperatur in Celsius und in Fahrenheit angezeigt werden.



XAML + Code Behind

Welches Alternative zur
Temperaturumrechnung finden
Sie besser?

```
<Slider
  x:Name="slider"
  Minimum="-200"
  Maximum="200"
  Value="0"
  valueChanged="c_ValueChanged" />

<TextBlock VerticalAlignment="Center" x:Name="cText" Text="0,00" />
<TextBlock VerticalAlignment="Center" x:Name="fText" Text="32,00"/>
```

```
private void c_ValueChanged(object sender, RoutedEventArgs<double> e)
{
    cText.Text = slider.Value.ToString("N2");
    fText.Text = (slider.Value * 1.8 + 32).ToString("N2");
}
```

Weitere Controls

- Die WPF stellt noch viele weitere Controls bereit, die man üblicherweise in Benutzeroberflächen benötigt.
 - Nicht alle können wir uns hier in der Vorlesung ansehen.
- Einige weitere einfache Controls sind z.B.:
 - DatePicker → Ein Control, um einzelne Datumswerte einzulesen.
 - Calendar → Ebenfalls für Datumswerte, kann aber noch mehr.
 - PasswordBox → Eine TextBox für Passwörter.
- Bitte experimentieren Sie einmal selbst damit!

Wir haben heute gelernt...

- Was Steuerelemente sind.
- Welche Basisklassen die WPF für Steuerelemente anbietet und welche Eigenschaften sich daraus für die meisten Controls ergeben.
- Was ContentControls sind.
- Der Umgang mit den ContentControls Label, Button, RadioButton und CheckBox.
- Wie man Text mit Hilfe des Controls TextBox einliest.
- Der Umgang mit dem Slider.