

```
textBoxOrt.Margin = new Thickness(1, 1, 0, 0);
```

tabindex="120"

```
public MainWindow()
{
    InitializeComponent();
    // Steuerelement anlegen und minimale Eigenschaften definieren
    TextBox eingabeName = new TextBox();
    eingabeName.Name = "EingabeName";
    eingabeName.Text = "Textbox aus Code behind";
    eingabeName.VerticalAlignment = VerticalAlignment.Top;
    eingabeName.HorizontalAlignment = HorizontalAlignment.Left;
    eingabeName.Margin = new Thickness(10, 40, 0, 0);
    eingabeName.Height = 23;
    eingabeName.Width = 150;
    // Handler registrieren
    eingabeName.LostFocus += new RoutedEventHandler(this.MacheAktion);
    // Anzeige in Fensterhierarchie definieren
    topGrid.Children.Add(eingabeName);
}
```

```
<Button x:Name="Speichern" Content="Speichern" IsDefault="True"/>
Esc auf einen Knopf umleiten
<Button x:Name="Schliessen" Content="Schliessen" IsCancel="True"/>
Weitere, beliebige Tastenabkürzungen (Shortcuts) werden später behandelt.
```

```
private void zeigeKunden_Click(object sender, RoutedEventArgs e)
{
    if (aktuellesModul != Module.Kunden)
    {
        aktuellesModul = Module.Kunden;
        standardKnopfe();
        zeigeKunden.Background = Brushes.White;

        Kunden kunden = new Kunden();
        kunden.HorizontalAlignment = HorizontalAlignment.Left;
        kunden.VerticalAlignment = VerticalAlignment.Top;
        inhalt.Content = null;
        inhalt.Content = kunden;
    }
}
```

```
foreach (DB.Kunde kunde in APP.Kunde.Lesen_Name(name))
{
    Console.WriteLine("KundeID: " + kunde.KundeID + .....
}
```

Sample Patterns	
Pattern	Will Match
<code>[A-Za-z0-9-]+</code>	Letters, numbers and hyphens
<code>(d{1,2})\d(1,2)\d(4)</code>	Date (e.g. 21/3/2006)
<code>((^s)+)?=\.(jpg gif png))\.\2</code>	Jpg, gif or png image
<code>^(1-9)(1){4}^(1-4)(1){0-9}{1}{1}{1}^50\$</code>	Any number from 1 to 50 inclusive
<code>(#?([A-Fa-f0-9]{3}){3}([A-Fa-f0-9]{3}){3})?</code>	Valid hexadecimal colour code
<code>((?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,15})</code>	String with at least one upper case letter, one lower case letter, and one digit (useful for passwords).
<code>(w+@[a-zA-Z_]+?\.[a-zA-Z]{2,6})</code>	Email addresses
<code>(\</?[^\>]+\>)</code>	HTML Tags
Note: These patterns are intended for reference purposes and have not been extensively tested. Please use with caution and test thoroughly before use.	

Anchors		Character Classes	
<code>^</code>	Start of string	<code>\c</code>	Control character
<code>\A</code>	Start of string	<code>\s</code>	White space
<code>\$</code>	End of string	<code>\S</code>	Not white space
<code>\Z</code>	End of string	<code>\d</code>	Digit
<code>\b</code>	Word boundary	<code>\D</code>	Not digit
<code>\B</code>	Not word boundary	<code>\w</code>	Word
<code>\<</code>	Start of word	<code>\W</code>	Not word
<code>\></code>	End of word	<code>\x</code>	Hexadecimal digit
		<code>\O</code>	Octal digit

```
Grid
<Grid>
    <Grid.RowDefinitions>
        <RowDefinitions Height="30"/>
        <RowDefinitions Height="30"/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="100"/>
        <ColumnDefinition Width="100"/>
    </Grid.ColumnDefinitions>
    <Button Grid.Column="0" Grid.Row="0" Content="Button1"/>
    <Button Grid.Column="1" Grid.Row="0" Content="Button2"/>
    <Button Grid.Column="0" Grid.Row="1" Content="Button3"/>
    <Button Grid.Column="1" Grid.Row="1" Content="Button4"/>
</Grid>
```

```
Grid UserControl change
Kunden kunden = new Kunden();
GRIDNAME.Children.Clear();
GRIDNAME.Children.Add(kunden);
```

```
StackPanel
<StackPanel>
    <Button Content="Button1"/>
    <Button Content="Button2"/>
</StackPanel>
```

```
Click=""
TextChanged=""
GotKeyboardFocus
GotMouseCapture
```

```
Canvas
<Canvas>
    <Button Canvas.Left="10" Canvas.Top="10" Content="Button1" />
    <Button Canvas.Left="20" Canvas.Top="20" Content="Button2" />
</Canvas>
```

Quantifiers	
<code>*</code>	0 or more
<code>+</code>	1 or more
<code>?</code>	0 or 1
<code>{3}</code>	Exactly 3
<code>{3,}</code>	3 or more
<code>{3,5}</code>	3, 4 or 5

```
DockPanel
<DockPanel>
    <Button DockPanel.Dock="Left" Content="Links"/>
    <Button DockPanel.Dock="Right" Content="Rechts"/>
    <Button DockPanel.Dock="Top" Content="Oben"/>
    <Button DockPanel.Dock="Bottom" Content="Unten"/>
    <Button Content="Mitte"/>
</DockPanel>
```

```
WrapPanel
<WrapPanel>
    <Button Content="Button 1"/>
    <Button Content="Button 2"/>
    <Button Content="Button 3"/>
</WrapPanel>
```

```
ScrollView UserControl change
Kunden kunden = new Kunden();
kunden.HorizontalAlignment = HorizontalAlignment.Left;
kunden.VerticalAlignment = VerticalAlignment.Top;
inhalt.Content = null;
inhalt.Content = kunden;
```

```
using System.Text.RegularExpressions;
```

```
Dynamisch je nach grösse
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto"/>
        <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <Button Grid.Column="0" Content="Grösse"/>
    <Button Grid.Column="1" Content="frei"/>
</Grid>
```

```
Dynamisch mit Stern
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="100"/>
        <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <Button Grid.Column="0" Content="Links fix"/>
    <Button Grid.Column="1" Content="Rechts frei"/>
</Grid>
```

```
ViewBox
<Grid>
<Viewbox Stretch="MOEGlichkeit">
    <Button Content="t">
</Viewbox>
</Grid>
MOEGlichkeit: Fill, Uniform, UniformToFill
```

```

public partial class UC_Einzelansicht_Kunde : UserControl
{
    #region "Klassenvariablen"
    11 Verweise
    enum Zustand
    {
        Leer,
        Anzeige,
        Neu,
        Veraendert,
        Ungespeichert
    }
    17 Verweise
    enum Uebergang{
        Abbrechen,
        Suchen,
        Erstellen,
        Loeschen,
        Eingabe,
        Speichern
    }
    private Boolean eingabeIgnorieren;
    private Zustand aktuellerZustand;
    17 Verweise

```

```

private Zustand AktuellerZustand{
    get { return aktuellerZustand; }
    // Anrede nie/ausschalten je nach Zustand
    set{
        if (value == Zustand.Leer){
            DeaktiviereSteuerelemente();
            suchen.IsEnabled = true;
            suchen.IsEnabled = true;
            neu.IsEnabled = true;
            aktuellerZustand = value;
            // GUI
            LeereFelder();
            EinzelansichtGrid.IsEnabled = false;
        }
        else if (value == Zustand.Anzeige){
            DeaktiviereSteuerelemente();
            abbrechen.IsEnabled = true;
            neu.IsEnabled = true;
            loeschen.IsEnabled = true;
            aktuellerZustand = value;
            // Felder abfüllen
            eingabeIgnorieren = true;
            LeereFelder();
            EinzelansichtGrid.IsEnabled = true;
            if (kunde.Anrede.Equals("Herr")){
                anredeHerr.IsChecked = true;
            }
            else if (kunde.Anrede.Equals("Frau")){
                anredeFrau.IsChecked = true;
            }
            else if (kunde.Anrede.Equals("Familie")){
                anredeFamilie.IsChecked = true;
            }
            tbVorname.Text = kunde.Vorname;
            tbName.Text = kunde.Name;
            tbNameZusatz.Text = kunde.NameZusatz;
            tbPLZ.Text = kunde.PLZ.ToString();
            tbOrt.Text = kunde.Ort;
            tbTel.Text = kunde.Telefon;
            tbMobile.Text = kunde.Mobile;
            tbEmail.Text = kunde.Email;
            tbWeb.Text = kunde.Web;
            eingabeIgnorieren = false;
        }
        else if (value == Zustand.Veraendert){
            DeaktiviereSteuerelemente();
            abbrechen.IsEnabled = true;
            speichern.IsEnabled = true;
            aktuellerZustand = value;
        }
        else if (value == Zustand.Neu){
            DeaktiviereSteuerelemente();
            EinzelansichtGrid.IsEnabled = true;
            abbrechen.IsEnabled = true;
            aktuellerZustand = value;
        }
        else if (value == Zustand.Ungespeichert){
            etc.
        }
    }
}
}
//ende DB.Kunde.kunde;
}
endregion

```

```

public partial class TextBoxMitValidierung : UserControl{
    private String regex;
    private SolidColorBrush okColor = new SolidColorBrush(Colors.LightGreen);
    private SolidColorBrush nokColor = new SolidColorBrush(Colors.Red);
    1 Verweise
    public TextBoxMitValidierung(String tooltipText, String regex)
    {
        InitializeComponent();
        this.Input.ToolTip = tooltipText;
        this.regex = regex;
        validieren();
    }
    3 Verweise
    public void Input_GotKeyboardFocus(object sender, KeyboardFocusChangedEventArgs e)
    {
        markiereAlles();
    }
    1 Verweise
    public void Input_GotMouseCapture(object sender, MouseEventArgs e)
    {
        markiereAlles();
    }
    1 Verweise
    public void Input_TextChanged(object sender, TextChangedEventArgs e)
    {
        validieren();
    }
    2 Verweise
    private void markiereAlles() {
        input.SelectionStart = 0;
        input.SelectionLength = input.Text.Length;
    }
    2 Verweise
    private void validieren(){
        if (Regex.IsMatch(input.Text, regex)){
            input.Background = okColor;
            input.BorderBrush = okColor;
        } else {
            input.Background = nokColor;
            input.BorderBrush = nokColor;
        }
    }
}

```

```

<TextBox x:Name="input" Height="20" Width="190" Margin="1,1,0,0"
HorizontalAlignment="Left" VerticalAlignment="Top" GotKeyboardFocus="input_GotKeyboardFocus"
GotMouseCapture="input_GotMouseCapture" TextChanged="input_TextChanged"></TextBox>
Grid>

```

```

1-Verweise
private void setupForm() {
    TextBoxMitValidierung textBoxOrt = new TextBoxMitValidierung("Bitte Ortschaft eingeben", @"^[A-Z]{1}[a-z]+$");
    textBoxOrt.Name = "tbOrt";
    //textBoxOrt.GotKeyboardFocus += textBoxOrt.Input_GotKeyboardFocus;
    //textBoxOrt.TabIndex =
    Grid.SetRow(textBoxOrt, 2);
    Grid.SetColumn(textBoxOrt, 1);
    hotelGrid.Children.Add(textBoxOrt);
    TextBoxMitValidierung textBoxAnzahlZimmer = new TextBoxMitValidierung("Bitte Anzahl Zimmer eingeben", @"^\d{1,5}$");
    textBoxAnzahlZimmer.Name = "tbAnzahlZimmer";
    Grid.SetRow(textBoxAnzahlZimmer, 6);
    Grid.SetColumn(textBoxAnzahlZimmer, 1);
    hotelGrid.Children.Add(textBoxAnzahlZimmer);
    //TextBoxMitValidierung textBoxEmail = new TextBoxMitValidierung("Bitte Emailadresse eingeben",@"")
}

```

```

public UC_Einzelansicht_Kunde(){
    InitializeComponent();
    AktuellerZustand = Zustand.Leer;
}
10 Verweise
private void macheUebergang(Uebergang uebergang){
    if (uebergang == Uebergang.Abbrechen){
        if (aktuellerZustand == Zustand.Veraendert || aktuellerZustand == Zustand.Ungespeichert){
            if (MessageBox.Show("Änderungen gehen verloren, wirklich abbrechen?", "Achtung!", MessageBoxButton.YesNo, MessageBoxImage.Warning) ==
                MessageBoxButton.No){
                return;
            }
            else{
                Abbrechen();
            }
        }
        else if (uebergang == Uebergang.Erstellen){
            Erstellen();
        }
        else if (uebergang == Uebergang.Loeshen){
            if (MessageBox.Show("Wirklich löschen?", "Achtung", MessageBoxButton.YesNo, MessageBoxButtonImage.Warning,
                MessageBoxButton.No, MessageBoxButton.No) ==
                MessageBoxButton.No){
                return;
            }
            else if (uebergang == Uebergang.Suchen){
                if (suchenNr.Text.Length > 0){
                    AblaufSuchen();
                }
                else {
                    MessageBox.Show("Bitte Kundennummer eingeben", "Fehler", MessageBoxButton.OK, MessageBoxButtonImage.Exclamation);
                }
            }
            else if (uebergang == Uebergang.Eingabe){
                AblaufEingabe();
            }
            else if (uebergang == Uebergang.Speichern){
                AblaufSpeichern();
            }
        }
    }
}

```

```

#region "Schnittstelle zu Applikationsschicht"
2 Verweise
private void Abbrechen(){// DB
    kunde = null; // GUI
    AktuellerZustand = Zustand.Leer;
}
1-Verweise
private void Erstellen(){// DB
    kunde = new DB.Kunde(); // GUI
    LeereFelder();
    AktuellerZustand = Zustand.Neu;
}
1-Verweise
private void AblaufLoeshen(){// DB
    APP.Kunde.Loeshen(kunde); // GUI
    AktuellerZustand = Zustand.Leer;
}
1-Verweise
private void AblaufSuchen() { // DB
    kunde = APP.Kunde.Lesen_KundeID(Convert.ToInt64(suchenNr.Text)); // TODO: wenn nichts gefunden
    if (kunde == null) {
        MessageBox.Show("Kein Kunde gefunden", "Info", MessageBoxButton.OK, MessageBoxButtonImage.Information,
            MessageBoxButton.No, MessageBoxButton.No);
        AktuellerZustand = Zustand.Leer;
    }
    else {
        // GUI
        AktuellerZustand = Zustand.Anzeige;
    }
}
1-Verweise
private void AblaufSpeichern(){ // Auslesen
    if (anredeHerr.IsChecked == true) {
        kunde.Anrede = "Herr";
    }
    else if (anredeFrau.IsChecked == true) {
        kunde.Anrede = "Frau";
    }
    else if (anredeFamilie.IsChecked == true) {
        kunde.Anrede = "Familie";
    }
    kunde.Vorname = tbVorname.Text;
    kunde.Name = tbName.Text;
    kunde.NameZusatz = tbNameZusatz.Text;
    try {
        kunde.PLZ = Convert.ToInt64(tbPLZ.Text);
    }
    catch (Exception ex) {
        kunde.PLZ = 0;
    }
    kunde.Ort = tbOrt.Text;
    kunde.Telefon = tbTel.Text;
    kunde.Mobile = tbMobile.Text;
    kunde.Email = tbEmail.Text;
    kunde.Web = tbWeb.Text; // DB
    if (aktuellerZustand == Zustand.Ungespeichert) {
        // Neu hinzufügen
        APP.Kunde.Erstellen(kunde);
    }
    else if (aktuellerZustand == Zustand.Veraendert) {
        // Änderungen speichern
        APP.Kunde.Aktualisieren(kunde);
    }
    // GUI
    AktuellerZustand = Zustand.Anzeige;
}
1-Verweise
private void AblaufEingabe(){ // DB (leer) // GUI
    if (AktuellerZustand == Zustand.Anzeige || AktuellerZustand == Zustand.Veraendert) {
        AktuellerZustand = Zustand.Veraendert;
    }
    else if (AktuellerZustand == Zustand.Neu) {
        AktuellerZustand = Zustand.Ungespeichert;
    }
}

```

```

#region "GUI Ereignisse"
1-Verweise
private void suchen_Click(object sender, RoutedEventArgs e){
    macheUebergang(Uebergang.Suchen);
}
1-Verweise
private void suchenNr_KeyDown(object sender, KeyEventArgs e){
    if (e.Key == Key.Return) {
        macheUebergang(Uebergang.Suchen);
    }
}
1-Verweise
private void speichern_Click(object sender, RoutedEventArgs e){
    macheUebergang(Uebergang.Speichern);
}
10 Verweise
private void eingabe(object sender, TextChangedEventArgs e) {
    if (eingabeIgnorieren) {
        macheUebergang(Uebergang.Eingabe);
    }
}
1-Verweise
private void anredeFrau_Checked(object sender, RoutedEventArgs e) {
    if (eingabeIgnorieren) {
        macheUebergang(Uebergang.Eingabe);
    }
}
1-Verweise
private void anredeHerr_Checked(object sender, RoutedEventArgs e){
    if (eingabeIgnorieren) {
        macheUebergang(Uebergang.Eingabe);
    }
}
1-Verweise
private void anredeFamilie_Checked(object sender, RoutedEventArgs e) {
    if (eingabeIgnorieren) {
        macheUebergang(Uebergang.Eingabe);
    }
}
1-Verweise
private void neu_Click(object sender, RoutedEventArgs e) {
    macheUebergang(Uebergang.Erstellen);
}
1-Verweise
private void loeshen_Click(object sender, RoutedEventArgs e) {
    macheUebergang(Uebergang.Loeshen);
}
1-Verweise
private void abbrechen_Click(object sender, RoutedEventArgs e) {
    macheUebergang(Uebergang.Abbrechen);
}
}
endregion

```