WPF

**MessageBox**

MessageBox.Show(„Content“, „Title“, MessageBoxImage.\_, MessageBoxButton.\_);

**New Window**

WPF Fenster einfügen

NewWindow n = new NewWindow();

n.Show();

**Controls**

1. **Label**

<Label Content=“Text“/>

1. **TextBlock**

<TextBlock TextAlignment=“Left/Right/Center“ TextTrimming=“a“ TextWrapping=“Wrap“/>

1. **TextBox**

<TextBox Text=“Text“ TextWrapping=“Wrap“ AcceptsRetrun=“True/False“ />

1. **Button**

<Button Content=“Text“ Command=“{Binding Name}“/>

1. **RadioButton**

<RadioButton Content=“Text“ GroupName=“Name“ IsChecked=“true/false“/>

1. **ComboBox**

<ComboBox ItemSource=“{Binding Name}“ SelectedItem=“{Binding Name}“ DisplayMemberPath=““/>

1. **Slider**

<Slider Maximum=“2“ Minimum=“0“ Value=“0.5“ TickFrequency=“Int wo stehen soll“/>

1. **Progressbar**

<ProgressBar Maximum=“10“ Minimum=“0“ Value=“0.5“/>

1. **Datagrid**

<DataGrid ItemSource=“{Binding Name}“ SelectedItem=“{Binding Name}“/>

1. **ListBox**

<ListBox ItemSource=“{Binding Name}“ SelectedItem=“{Binding Name}“/>

1. **ListView**

<ListView ItemSource=“{Binding Name}“ SelectedItem=“{Binding Name}“/>

1. **CheckBox**

<CheckBox Content=“Name“ IsChecked=“True/False“/>

**Panels**

1. **Canvas**

<Canvas>

<Button Canvas.\_=“2“/> Top, Left, Right, Bottom

</Canvas>

1. **StackPanel**

<StackPanel Orientation=“Horizontal/Vertical“/>

1. **WrapPanel**

<WrapPanel Orientation=“Horizontal/Vertical“/>

1. **DockPanel**

<DockPanel LastChildFill=“True/False“>

<Button DockPanel.Dock=“Top/Bottom/Left/Right“/>

</DockPanel>

1. **Grid**

<Grid>

<Grid.ColoumnDefinition>

<ColoumnDefinition Width=“1\*“/>

</Grid.ColoumnDefinition>

<Grid.RowDefinition>

<RowDefinition Width=“2\*“/>

</Grid.RowDefinition>

</Grid>

1. **TabControl**

<TabControl>

<TabItem Header=“Text“>

<TabItem.DataContext>

<local: VM/>

</TabItem.DataContext>

<TabItem.Resources>

<local: VM x.key =“Name“/>

</TabItem.Resources>

**Data Binding**

1. **MVC vs. MVVM**

|  |  |  |
| --- | --- | --- |
|  | Windows Form | WPF |
| View | Klasse Partial + 2 cs | Xaml + cs Datei |
| Zwischenschicht | Logiklasse  Methoden parametrisiert, bzw. Rückgabewert in View | Properties für Databinding & CommandBinding im xaml verwendet |
| Modelschicht | Modelklasse von Entity Framework | |
| Persistente Daten | Csv, txt, mdf, … beliebige Datenbank | |

1. **INotifyPropertyChanged**

abstract class Name : INotifyPropertyChanged{

public event PropertyChangedEventHandler parameter;

void OnPropertyChanged([CallerMemberName] string propertyName = null){

if(paramter != null)

paramter(this, new PropertyChangedEventArgs(propertyName);

}

}

1. **ObservableCollection**

private ObservableCollection<Klasse> name;

public ObservableCollection<Klasse> Name {

get { return name;}

set { name = value;}

}

1. **Binding Modes**
2. **UpdateSourceTrigger**

**Command Binding**

1. **ICommand + Ausimplementieren einfügen**

interface ICommand

{

event EventHandler CanExecuteChanged;

bool CanExecute(object parameter);

void Execute(object parameter);

}

class Name : ICommand{

public event EventHandler CanExecuteChanged;

public bool CanExecute(object parameter){

(return (vm.Name != null))

(File.Exists();)

}

public void Execute(object parameter){

(vm.List.Add(item);

(Vm m = new Vm()

{ Name = vm.Name;}

Vm.toFile(m)

m.Name = String.Empty;)

}

1. **Binding Properties**

ItemSource à Liste zum Content generieren

SelectedItem à bind zu instanze vom Objekt

**Converter**

1. **IValueConverter**

class Name : IValueConverter{

public object Convert( object value, Type tt, object parameter, CulturInfo culture){

return ((ENum)value)==((ENum)parameter); à ENum to Bool

return (int)value; à ENum to Int

}

public object ConvertBack (object vlaue, Type tt, object parameter, CulturInfo culture){

return ((bool)value) ? parameter : Binding.DoNothing; à ENum to Bool

return (ENum).System.Convert.Int32(value); à ENum to Int

}

1. **Converter als Resource**

Converter = {StaticResource Name}

ConverterParameter={x:Static local:ENum.Wert}