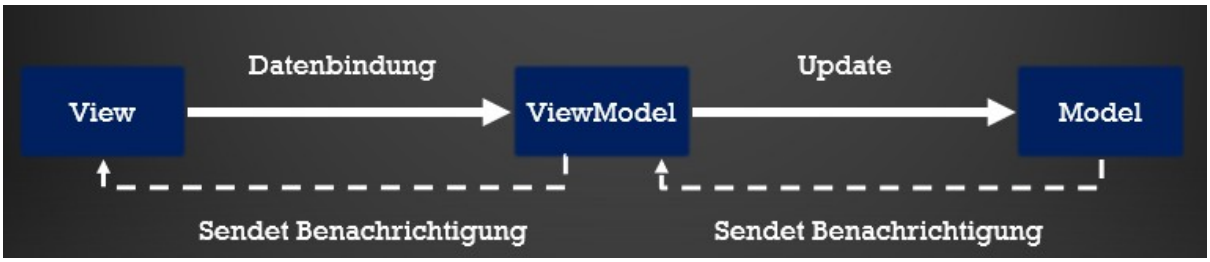
erstellt von:
Zaft, Grant, Wollny

Handout zu Datenbindung und MVVM

Was ist MVVM?

MVVM steht für Model-View-ViewModel und bezeichnet ein Architekturmuster für interaktive Systeme. Hierbei werden Programmlogik und das Benutzerinterface voneinander getrennt.



Das **Model** beschäftigt sich mit der Programmlogik, darunter fallen die Kommunikation mit Datenbanken oder APIs. Es kommuniziert nur mit dem ViewModel und hat keine Verbindung zur eigentlichen Oberfläche (View)

Die **View** ist alleinig für die Benutzeroberfläche zuständig und besitzt deshalb keine Programmlogik. Es reagiert auf Benutzeraktionen z. B. Eingabe von Text in einer TextBox, und gibt diese Daten an das ViewModel weiter. Es hat keinerlei Anbindung zum Model.

Das Herzstück stellt das **ViewModel** da. Das ViewModel stellt das Bindestück zwischen Model und View dar. Die UI-Relevante Logik und Verarbeitung von Benutzereingaben sind Aufgaben des ViewModel. Dieses leitet die Daten an das Model weiter, welches diese verarbeitet. Bei dieser Vorgehensweise können Funktionen des ViewModels getestet werden, ohne eine Abhängigkeit an die UI.

View und Viewmodel verbinden

Um View mit dem ViewModel zu verbinden, muss der DataContext auf eine Instanz des ViewModels gesetzt werden

```
2 references
public partial class MainWindow : Window
{
    0 references
    public MainWindow()
    {
        InitializeComponent();
        DataContext = new MainWindowViewModel();
    }
}
```

Datenbindung

• Dient zur Synchronisation von Daten zwischen View und ViewModel
• Änderungen im Code werden direkt in der UI sichtbar

```
0 references
public class ViewModel
{
    0 references
    public int Telefonnummer { get; set; }
}
```

Code im ViewModel

```
<TextBox Text="{Binding Telefonnummer, Mode=TwoWay}" />
```

Code im View

Datenbindung bei Listbox

```
1 reference
public class ViewModel
{
    1 reference
    ObservableCollection<string> Namen { get; set; }
    0 references
    public ViewModel()
    {
        Namen = new ObservableCollection<string>();
    }
}
```

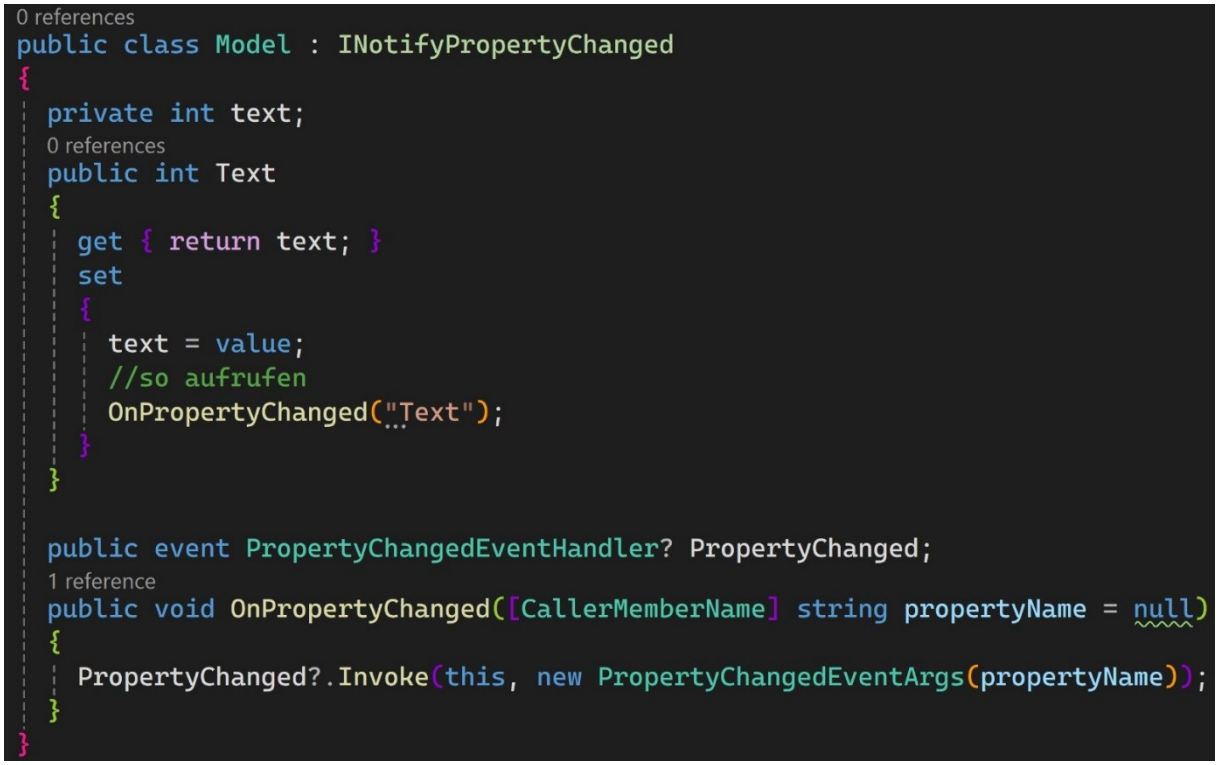
```
<ListBox ItemsSource="{Binding Namen}"/>
```

Eigenschaften von Datenbindung

Um direkt Änderungen an UI-Elementen weiterzugeben, wird die Eigenschaft „UpdateSourceTrigger“ auf „PropertyChanged“ gesetzt

```
<TextBox Text="{Binding Nachricht, UpdateSourceTrigger=PropertyChanged}" />
```

Dieses Interface dient zur Aktualisierung der UI bei Änderungen, die direkt an den Bindings geschehen.



```
0 references
public class Model : INotifyPropertyChanged
{
    private int text;
    0 references
    public int Text
    {
        get { return text; }
        set
        {
            text = value;
            //so aufrufen
            OnPropertyChanged("Text");
        }
    }

    public event PropertyChangedEventHandler? PropertyChanged;
    1 reference
    public void OnPropertyChanged([CallerMemberName] string propertyName = null)
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
    }
}
```