



# Bessere ViewModels mit dem MVVM Community Toolkit

Olaf Lischke

# Speaker: Olaf Lischke



macht .NET, seit es .NET Framework gibt



versucht, Projekte und Seminare zu kombinieren



singt Tenor in Chören und Musikprojekten



zockte schon auf dem ZX 81, heute ausschließlich auf PC



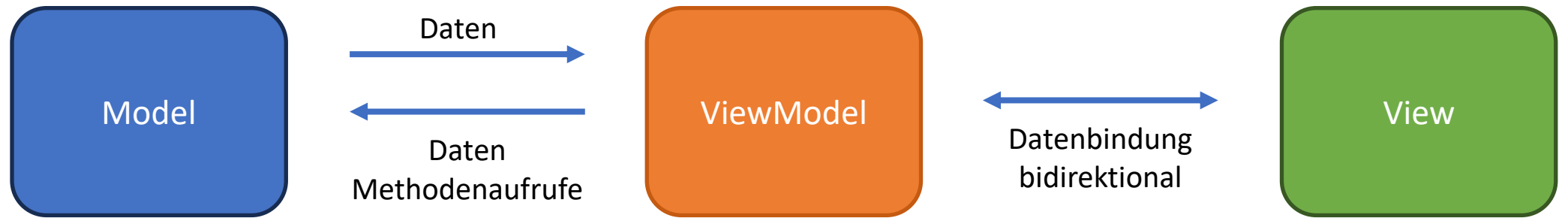
fotografiert, seit er eine Kamera halten kann



fliegt, wenn Wetter und Zeit es zulassen (TMG/SEP)



# MVVM-Pattern: Architektur

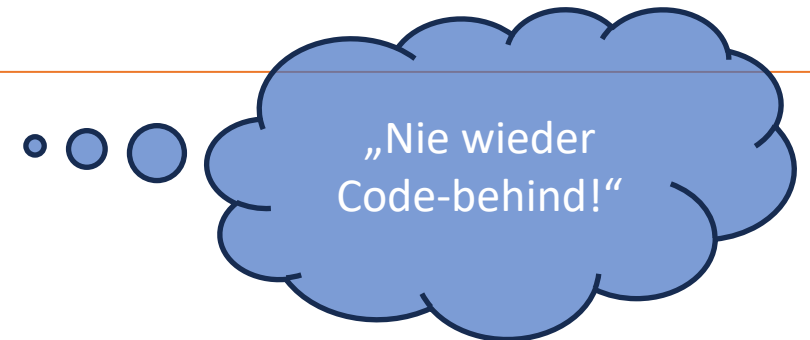


Domain Model  
Logik-Schicht  
Datenzugriffsschicht

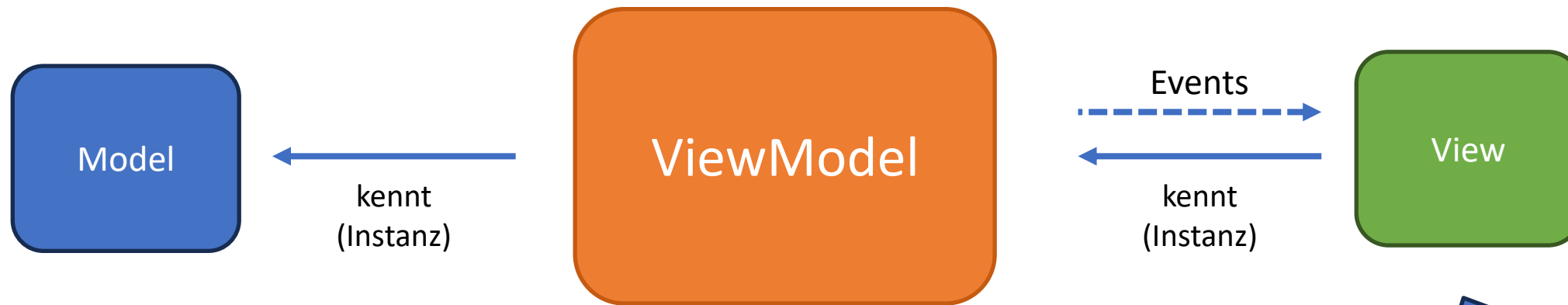
Datenaufbereitung  
Commanding

Benutzeroberfläche

- Ziele:**
- Trennung von Darstellung und Logik
  - Wartbarkeit des Codes
  - Testbarkeit des Codes



# MVVM-Pattern: Entkoppelung



`INotifyPropertyChanged`  
`PropertyChanged`-Event

→ Fullqualified Properties!

`ICommand`-Implementierung

→ `RelayCommand`-Klasse!

Wenn das ViewModel  
keine Views kennt, wer  
öffnet dann wie welche  
Views?



# MVVM Community Toolkit

- Attribute  
→ SourceGenerator erzeugt Standardcode
- Observables  
→ liefern die PropertyChanged-Implementierung
- Commanding  
→ vereinfachen die Implementierung von Commands
- Messenger  
→ Kommunikation der Komponenten
- Dependency Injection  
→ Integration aller Komponenten an einer Stelle

nuget:  
CommunityToolkit.Mvvm



# Attribute

## Property bisher

```
private Artist? _selectedArtist;

public Artist? SelectedArtist
{
    get { return _selectedArtist; }
    set
    {
        if (value != _selectedArtist)
        {
            OnArtistChanging();
            OnPropertyChanged();
            _selectedArtist = value;
            OnPropertyChanged();
            OnArtistChanged();
            OnCanExecuteChanged(EditArtistCommand);
            OnCanExecuteChanged(RemoveArtistCommand);
        }
    }
}
```

## MVVM Community Toolkit

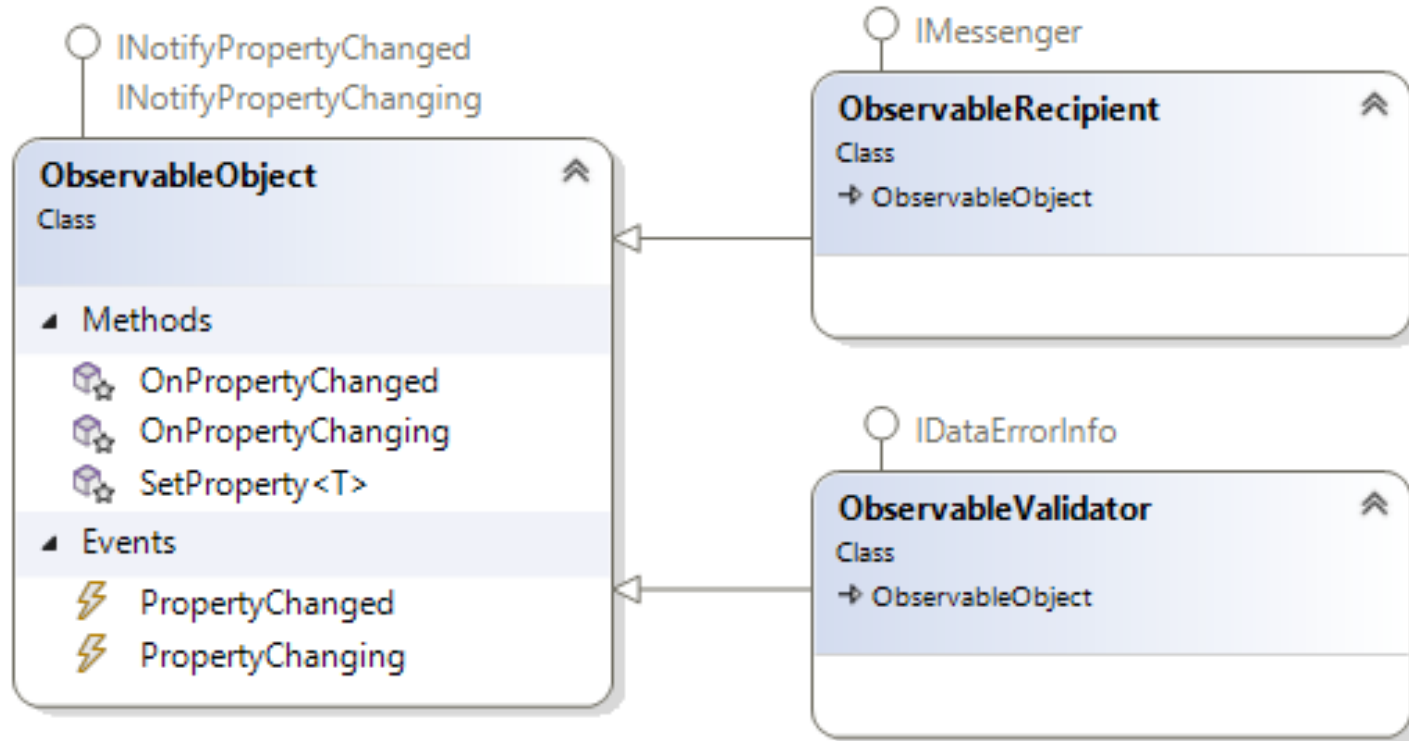
```
[ObservableProperty]
[NotifyCanExecuteChangedFor(nameof(EditArtistCommand))]
[NotifyCanExecuteChangedFor(nameof(RemoveArtistCommand))]
private Artist? selectedArtist;
```

...den Rest macht der  
Source Generator!

+ weitere Attribute:  
[RelayCommand]  
[NotifyPropertyChanged]



# Observables



```
private List<Presentation.Genre> _genres;  
  
public List<Presentation.Genre> Genres  
{  
    get => _genres;  
    set => SetProperty(ref _genres, value);  
}
```

Nie wieder  
`INotifyPropertyChanged`  
implementieren!



# Commands

## ICommand bisher

- RelayCommand.cs aus anderem Projekt kopieren
- Namespace anpassen
  - Tippfehler im Namespace korrigieren ;-)
- Commands als Properties im ViewModel anlegen

## MVVM Community Toolkit

- RelayCommand bereits implementiert.

Deklaration:

```
public ICommand AddArtistCommand { get; set; }
```

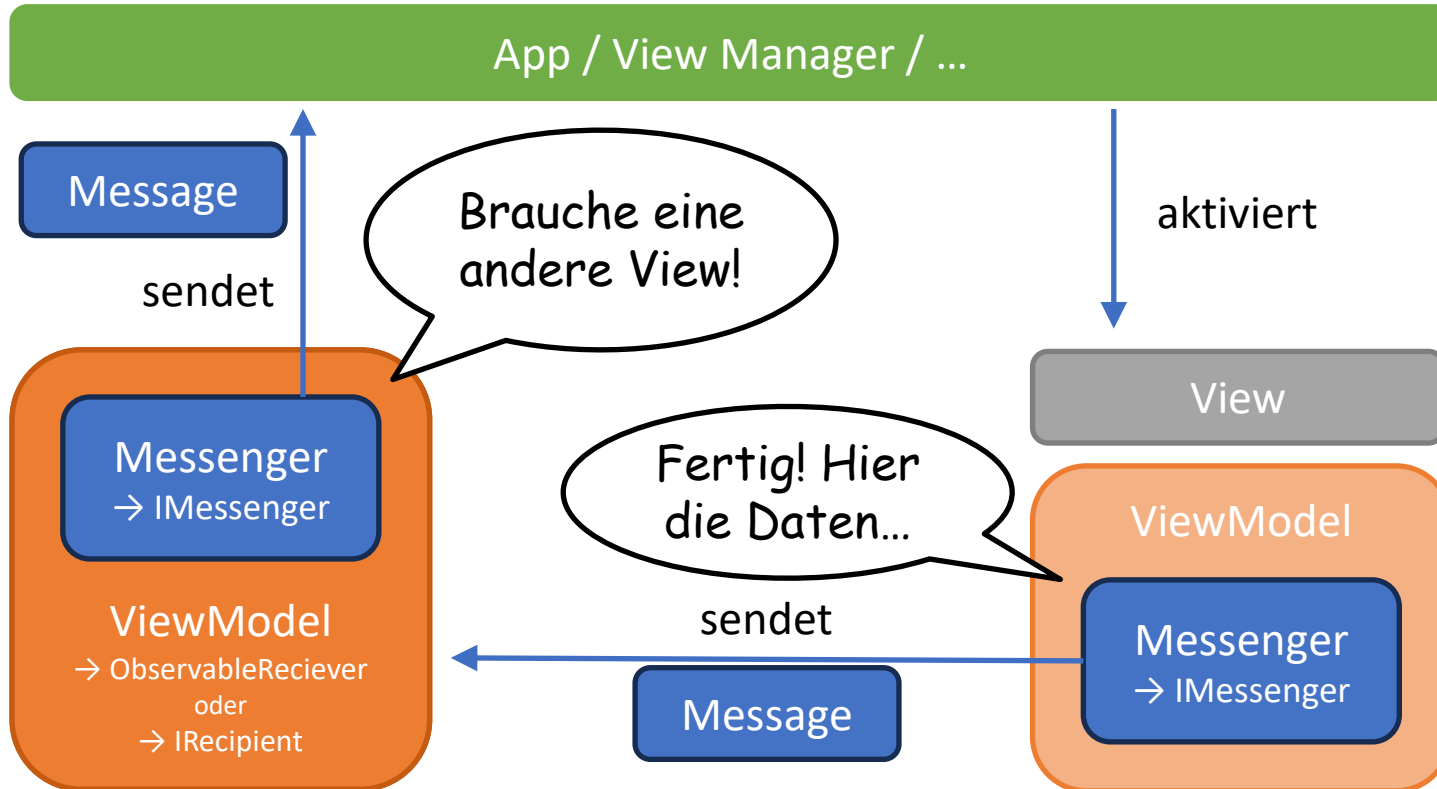
Instanziierung:

```
AddArtistCommand = new RelayCommand(AddArtist);
```





# Messaging



- CommunityToolkit.Mvvm.Messaging
  - IMessenger
  - IMessengerExtensions
  - IRecipient<in TMessage>
  - MessageHandler<in TRecipient, in TMessage>
  - StrongReferenceMessenger
  - WeakReferenceMessenger

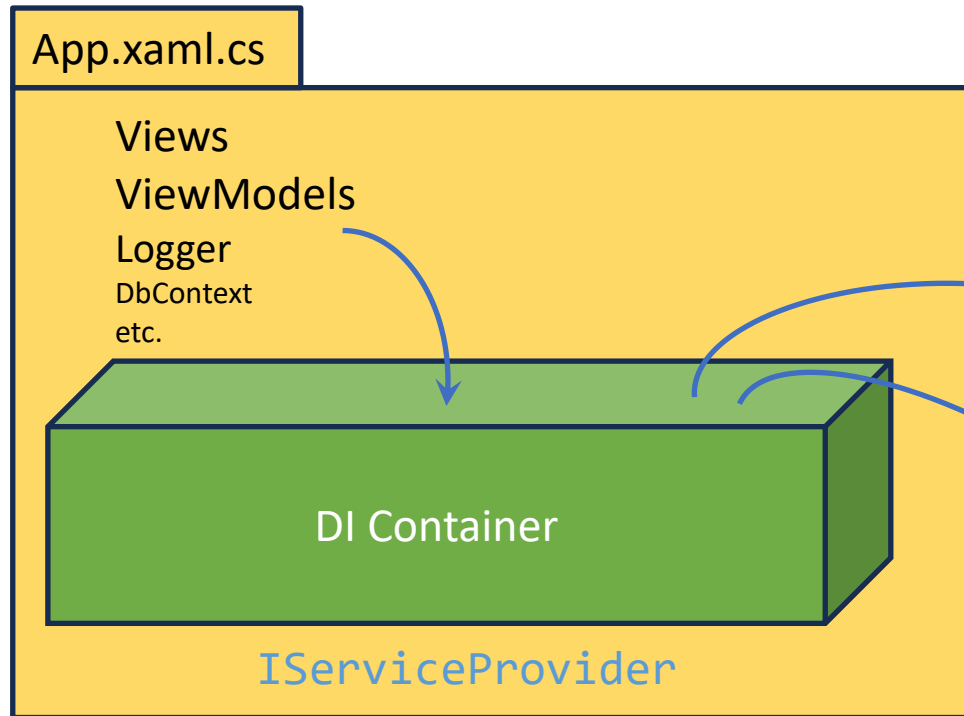
**Message** → eine gewöhnliche Klasse

```
public class ShowArtistEditDialogMessage
{
    public ShowArtistEditDialogMessage(Artist artist)
    {
        this.Artist = artist;
    }
    public Artist Artist { get; set; }
}
```



# Dependency Injection

**nuget:**  
`Microsoft.Extensions.DependencyInjection`



- instanziiert Komponenten auf Anfrage
- weiß, wie diese zu instanziiieren sind
- kennt und kontrolliert die Lebensdauer

## Constructor Injection

```
public MainWindowViewModel(IMessenger messenger, ILogger<Main
```

**Debug-Tipp:**  
**Interface!**

Instanzen automatisch  
aus DI Container

## GetService-Methode

```
viewModel = (ACViewModel)App.Current  
                .Services  
                .GetService(typeof(ACViewModel));
```



# Fazit

- „Alles da, alles drin“:
  - Source Generator erzeugt Boilerplate Code
  - Alle Elemente für MVVM bereits an Bord (RelayCommand, Observables, Messaging)
- Erleichtert die vollständige(!) Umsetzung des MVVM Patterns
- Für jede XAML-basierte UI-Technologie (WPF, WinUI, MAUI)



Echte  
Arbeitserleichterung



# Vielen Dank!



## Kontaktinfos:

[olaf.lischke@lischke-edv.de](mailto:olaf.lischke@lischke-edv.de)

0172 / 937 24 94



`olaflischke.vcf`

## Slides und Code Sample:

<https://github.com/olaflischke/do-dotnet-mvvm>