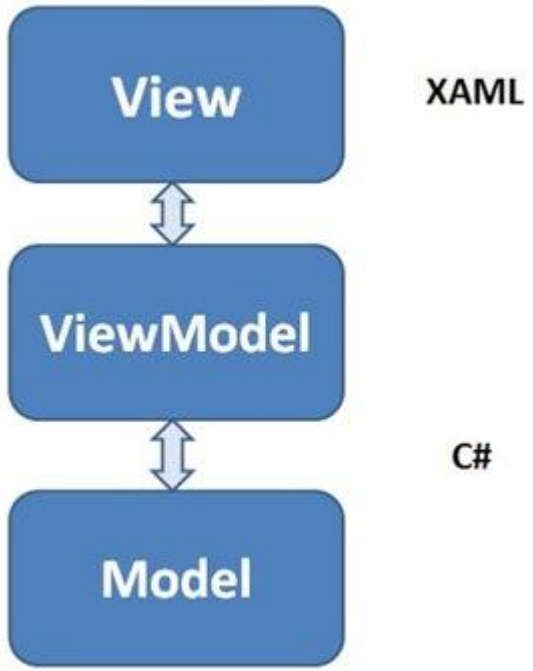
MVVM

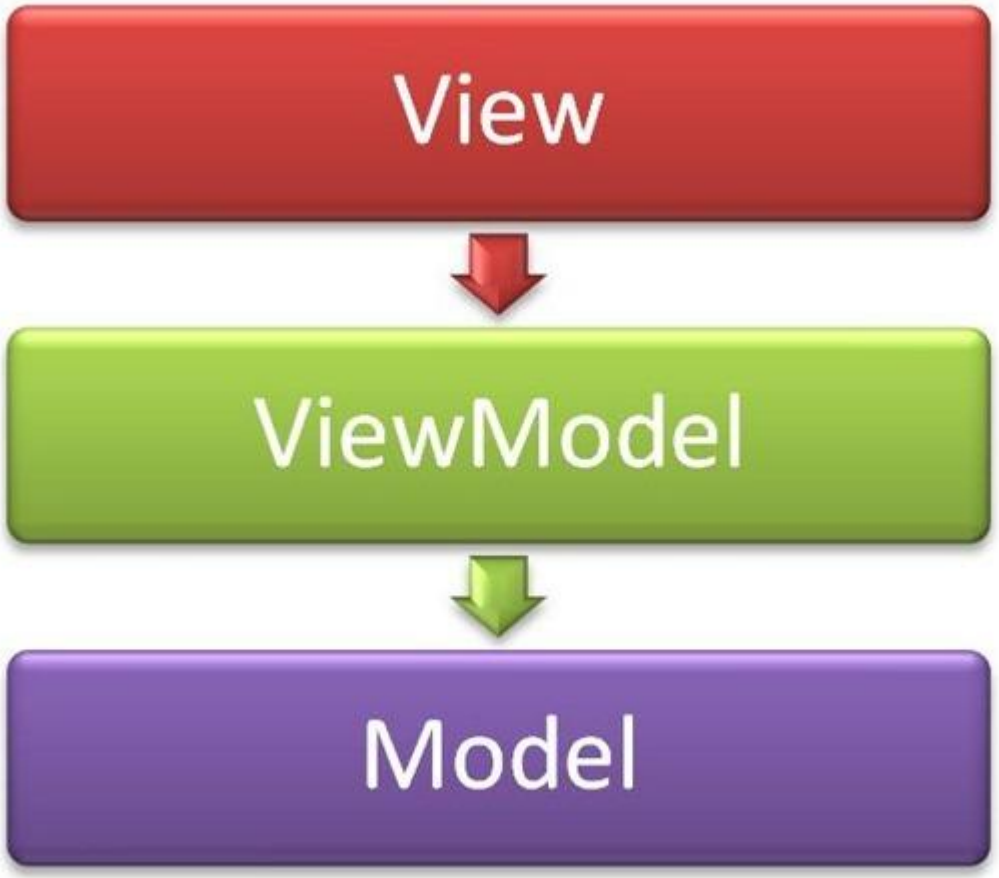
beyond the basics

DI Thomas Mutzl
Software Architect

MVVM – simple explained

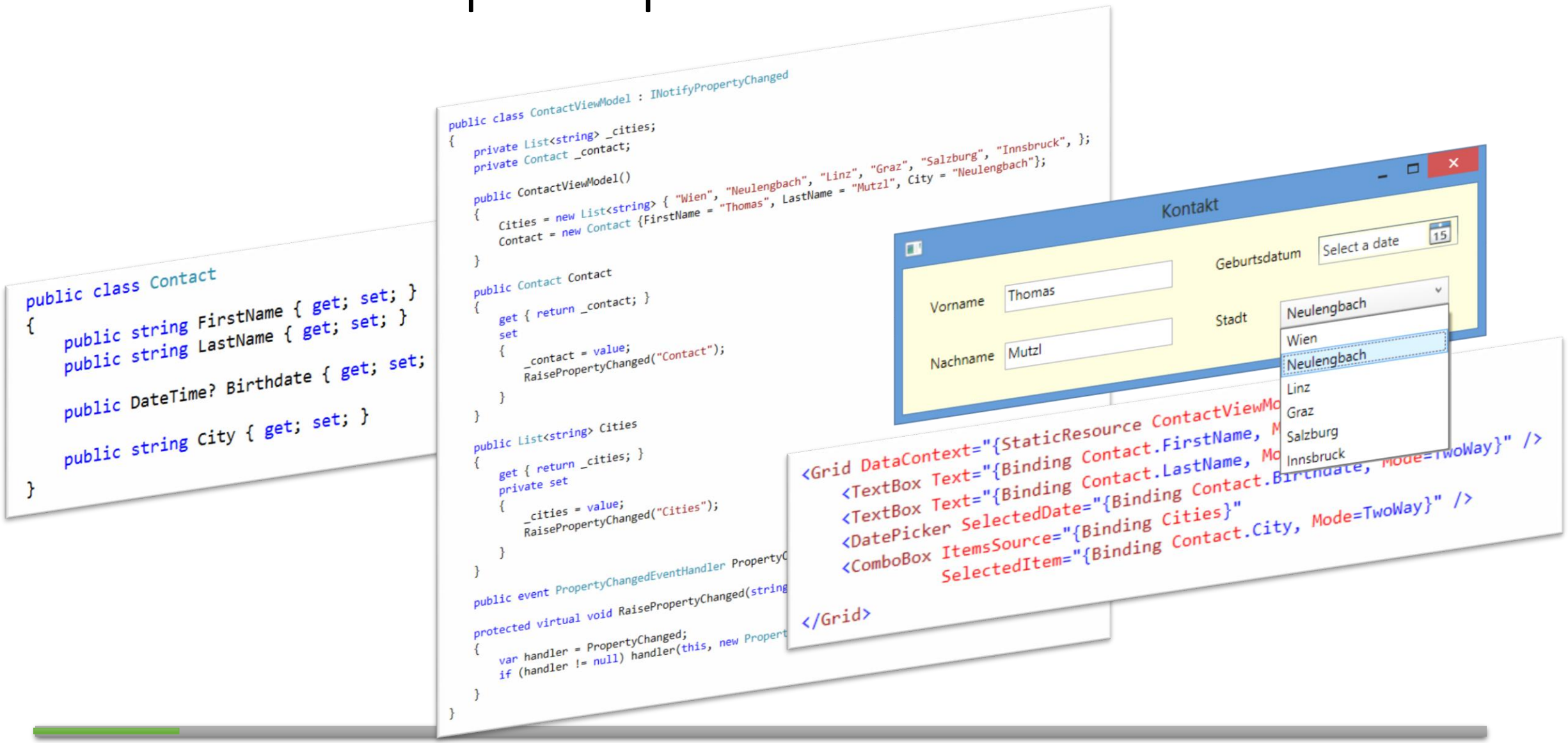


MVVM – simple explained



MVVM – simple explained

MVVM – simple explained



MVVM – basics

- Model – View – ViewModel
- Implementierung von INPC (INotifyPropertyChanged)
- Frameworks
  - MVVM Light
  - Caliburn Micro
  - Prism
  - …

Properties mit NotifyPropertyChangeInvocator

- Code Snippets
- CallerMemberNameAttribute
- ReSharper
- Aspekt Orientierte Programmierung

Fody

- NuGet Package PropertyChanged.Fody
- https://github.com/Fody/PropertyChanged
- Injecting code into IL

```
1 reference
public string FirstName { get; set; }
1 reference
public string LastName { get; set; }

0 references
public string FullName
{
    get
    {
        return string.Format("{0}, {1}", LastName, FirstName);
    }
}
```

```
public string FirstName
{
  [CompilerGenerated]
  get
  {
    return this.<FirstName>k__BackingField;
  }
  [CompilerGenerated]
  set
  {
    if (!string.Equals(this.<FirstName>k__BackingField, value, StringComparison.Ordinal))
    {
      this.<FirstName>k__BackingField = value;
      this.RaisePropertyChanged("FullName");
      this.RaisePropertyChanged("FirstName");
    }
  }
}
```

ViewModelLocator

- Klasse mit Properties für jedes ViewModel
- Ermöglicht Verwendung von Dependency Injection

```
1 reference
public class ViewModelLocator
{
    0 references
    public ViewModelLocator()
    {
        ServiceLocator.SetLocatorProvider(() => SimpleIoc.Default);

        SimpleIoc.Default.Register<MainViewModel>();
    }
    0 references
    public MainViewModel MainViewModel { get {return ServiceLocator.Current.GetInstance<MainViewModel>();} }

}
```

- Statische Ressource

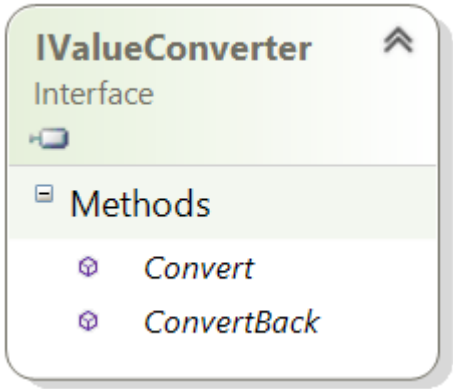
```
<viewModels:ViewModelLocator x:Key="Locator" />
```

- Deklaratives Mapping von View zu ViewModel

```
DataContext="{Binding MainViewModel, Source={StaticResource Locator}}"
```

ValueConverter

- Konvertiert von einem Datentyp in einen beliebigen anderen Datentyp (und zurück)
- Converter implementiert IValueConverter
- Ermöglicht Trennung von UI-Aspekten (z.B. Visibility, Color) von Business Logik



Observable Collection

- Hinzufügen oder Löschen von Elementen einer Liste

- Statt `List<T>`
  Verwendung von `ObservableCollection<T>`

- Implementiert `INotifyCollectionChanged`

Commands

- ButtonBase nicht nur Click Event sondern auch Command Property

```
namespace System.Windows.Controls.Primitives
{
    ...public abstract class ButtonBase : ContentControl
    {
        ...public ICommand Command { get; set; }

        ...public event RoutedEventHandler Click;
    }
}
```

```
<Button Click="ButtonBase_OnClick"
        Command="{Binding DemoCommand}"/>
```

- ICommand
- Keine Implementierung in .NET BCL
- RelayCommand, DelegateCommand, …

Command ohne Button

- Codebehind

```
((MainViewModel)DataContext).DemoCommand.Execute(null);
```

- EventToCommandBehavior

DependentRelayCommand

- Erweiterung zum `RelayCommand`
- Automatisches `RaiseCanExecuteChanged()`
  wenn sich Properties ändern,
  welche Einfluss auf den Return-Value von `CanExecute()` haben

```
AddColorCommand
= new DependentRelayCommand(AddColor, CanAddColor, this, "NewColor");
```

```
SaveCommand
= new DependentRelayCommand(Save, CanSave, this, () => FirstName, () => LastName);
```

- Als NuGet Package verfügbar.

MVVM Light DependentRelayCommand (PCL)
Extends the RelayCommand of the MVVM Light Toolkit. Fires the CanExecuteChanged event for the command automatically if a PropertyChanged event of a dependent pro...


www.mutzl.com


thomas@mutzl.com


/thomas.mutzl


@mutzl

Danke!

DI Thomas Mutzl
Software Architect