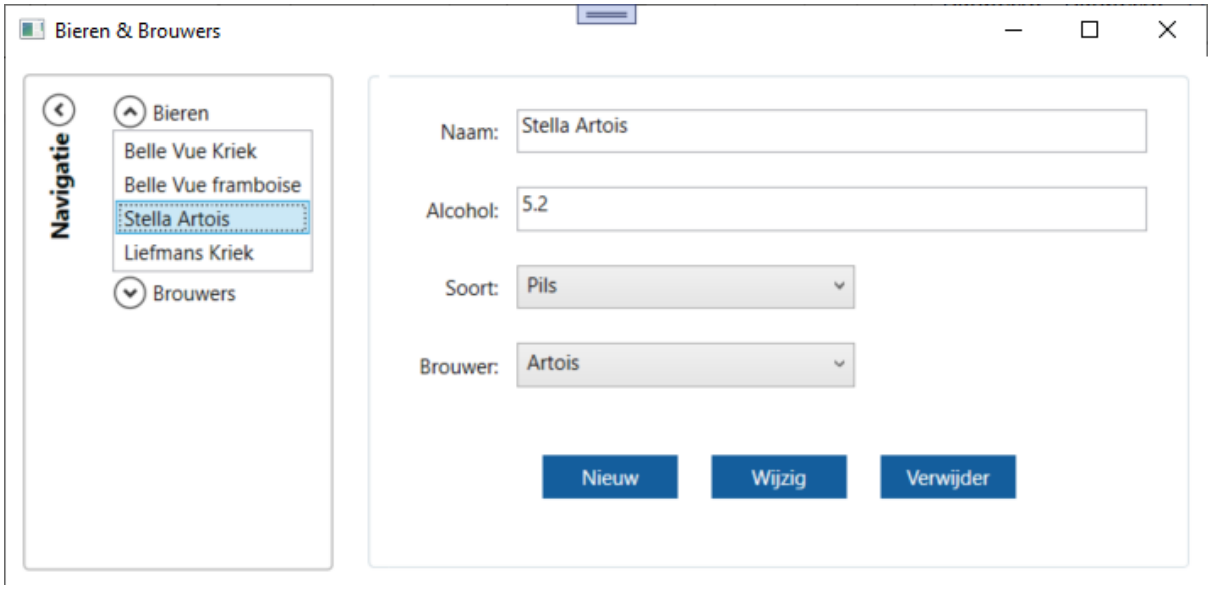
HOW TO –
D. WPF MVVM Command Binding

(Voorbeeldapplicatie van bieren, Brouwers en soorten bieren)

https://github.com/CSharpSyntraWest/HOWTO-WPF-MVVM

1. **In dit hoofdstuk plaatsen we Command bindings op de Buttons Nieuw, Wijzig en Verwijder in de BierDetailsView en voegen de noodzakelijke code toe aan de BierenViewModel klasse en de MockDataService om een Bier toe te voegen, te wijzigen en te verwijderen uit de lijst van bieren.**



2. **Open BierDetailsView.xaml en voeg de volgende bindings toe op de Command property van de 3 buttons onderaan:**

```
<UserControl x:Class="D_BindingCommandsWPFMVVM.Views.BierDetailsView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:D_BindingCommandsWPFMVVM.Views"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800">
    <StackPanel Margin="10">
            <Grid Margin="10">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="Auto" />
                    <ColumnDefinition Width="6" />
                    <ColumnDefinition Width="*" />
                </Grid.ColumnDefinitions>

                <Grid.RowDefinitions>
                    <RowDefinition Height="Auto" />
                    <RowDefinition Height="20" />
                    <RowDefinition Height="Auto" />
                    <RowDefinition Height="20" />
                    <RowDefinition Height="Auto" />
                    <RowDefinition Height="20" />
                    <RowDefinition Height="Auto" />
```

```
                    <RowDefinition Height="20" />
                    <RowDefinition Height="Auto" />
                </Grid.RowDefinitions>
                <Label  Grid.Row="0" Content="Naam:"
                       HorizontalAlignment="Right" />
                <TextBox Text="{Binding SelectedBier.Naam, UpdateSourceTrigger=PropertyChanged}"
Grid.Column="2" />
                <Label Content="Alcohol:"
                       Grid.Row="2"
                       HorizontalAlignment="Right" />
                <TextBox Text="{Binding SelectedBier.Alcohol,
UpdateSourceTrigger=PropertyChanged}" Grid.Row="2"
                         Grid.Column="2" />
                <Label Content="Soort:" Grid.Row="4"
                       HorizontalAlignment="Right" />
                <ComboBox  Grid.Row="4" Grid.Column="2" Width="200" HorizontalAlignment="Left"
ItemsSource="{Binding BierSoorten}" SelectedItem="{Binding
SelectedBier.BierSoort,UpdateSourceTrigger=PropertyChanged}" DisplayMemberPath="SoortNaam"/>
                <Label Content="Brouwer:" Grid.Row="6"
                       HorizontalAlignment="Right" />
                <ComboBox  Grid.Row="6" Grid.Column="2" Width="200" HorizontalAlignment="Left"
ItemsSource="{Binding Brouwers}" SelectedItem="{Binding
SelectedBier.Brouwer,UpdateSourceTrigger=PropertyChanged}" DisplayMemberPath="BrNaam"/>

                <StackPanel Grid.Row="8" Grid.Column="0" Grid.ColumnSpan="3"
HorizontalAlignment="Center" Orientation="Horizontal" Margin="10">
                    <Button Style="{StaticResource DetailsFormButton}"
                                Command="{Binding AddBierCommand}" Margin="10"
                        Grid.Row="8"
                        Content="Nieuw" />

                    <Button Style="{StaticResource DetailsFormButton}"
                                Command="{Binding UpdateBierCommand}" Margin="10"
                        Grid.Row="8"
                        Content="Wijzig" />
                    <Button Style="{StaticResource DetailsFormButton}"
                                Command="{Binding DeleteBierCommand}" Margin="10"
                        Grid.Row="8"
                        Content="Verwijder" />
                </StackPanel>

            </Grid>
    </StackPanel>
</UserControl>
```

3. De **AddBierCommand**, **UpdateBierCommand** en **DeleteBierCommand** moeten nu nog als public properties worden toegevoegd aan de BierenViewModel klasse. Deze zijn van het type ICommand, een ingebouwde Interface die specifiek bij o.a. MVVM wordt gebruikt om Button Commando's te binden aan code in de ViewModel die een bepaalde actie uitvoert (bv een Bier-element toevoegen, aanpassen of verwijderen).

4. Open **BierenViewModel**.cs en voeg deze public properties toe. De set van deze properties zal enkel worden gedaan door de constructor van de klasse en zijn daarom private :

```
public ICommand AddBierCommand { get; private set; }
public ICommand UpdateBierCommand { get; private set; }
public ICommand DeleteBierCommand { get; private set; }
```

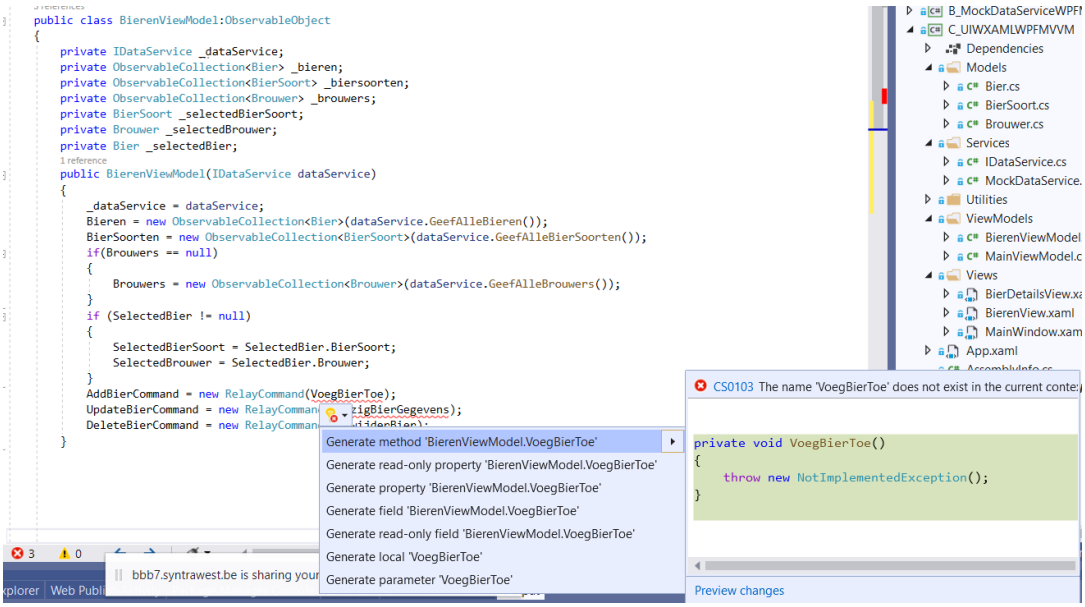
5. Nu moeten we deze Properties in de constructor van BierenViewModel initialiseren. We koppelen deze aan een methode die zal worden uitgevoerd wanneer er op de button wordt geklikt. We maken gebruik van een Utility-klassen **RelayCommand<**T> en **RelayCommand** (soms wordt deze klasse ook BaseCommand (zie vroegere oefeningen over MVVM) genoemd). We plaatsen deze klasse onder de folder **Utilities**:

```
public class RelayCommand<T> : ICommand
{
    private readonly Action<T> _execute = null;
    private readonly Func<T, bool> _canExecute = null;

    public RelayCommand(Action<T> execute, Func<T, bool> canExecute = null)
    {
        _execute = execute ?? throw new ArgumentNullException(nameof(execute));
        _canExecute = canExecute ?? (_ => true);
    }

    public event EventHandler CanExecuteChanged
    {
        add => CommandManager.RequerySuggested += value;
        remove => CommandManager.RequerySuggested -= value;
    }

    public bool CanExecute(object parameter) => _canExecute((T)parameter);

    public void Execute(object parameter) => _execute((T)parameter);
}

public class RelayCommand : RelayCommand<object>
{
    public RelayCommand(Action execute)
        : base(_ => execute()) { }

    public RelayCommand(Action execute, Func<bool> canExecute)
        : base(_ => execute(), _ => canExecute()) { }
}
```

6. **Open BierenViewModel.cs en voeg aan de constructor de initialisatie toe van de `AddBierCommand`, `UpdateBierCommand` en `DeleteBierCommand` public ICommand properties. We gebruiken hiervoor RelayCommand utility class (zie code hierboven):**

```
public BierenViewModel(IDataService dataService)
{
    _dataService = dataService;
    Bieren = new ObservableCollection<Bier>(dataService.GeefAlleBieren());
    BierSoorten = new ObservableCollection<BierSoort>(dataService.GeefAlleBierSoorten());
    if(Brouwers == null)
    {
        Brouwers = new ObservableCollection<Brouwer>(dataService.GeefAlleBrouwers());
    }
    if (SelectedBier != null)
    {
        SelectedBierSoort = SelectedBier.BierSoort;
        SelectedBrouwer = SelectedBier.Brouwer;
    }
    AddBierCommand = new RelayCommand(VoegBierToe);
    UpdateBierCommand = new RelayCommand(WijzigBierGegevens);
    DeleteBierCommand = new RelayCommand(VerwijderBier);
}
```

7. **De constructor van RelayCommand verwacht een methode als parameter. De methoden VoegBierToe(), WijzigBierGegevens() en VerwijderBier() moeten nu nog aangemaakt worden in de BierenViewModel klasse. Rechtsklik met de muis op VoegBierToe en selecteer "Generate method 'BierenViewModel.VoegBierToe'" uit het context menu :**



8. **De volgende code wordt nu gegenereerd :**

```
private void VoegBierToe()
{
    throw new NotImplementedException();
}
```

9. **Vervang de lijn met NotImplementedException() met de volgende code die een nieuw Bierobject aanmaakt en deze toevoegt via de dataservice. De dataservice methode VoegBierToe geeft de nieuwe lijst van bieren terug met een nieuw bier. We zetten property Bieren op de gewijzigde lijst van bieren. Verder zetten we het SelectedBier property op de laatst toegevoegde bier in de lijst.**

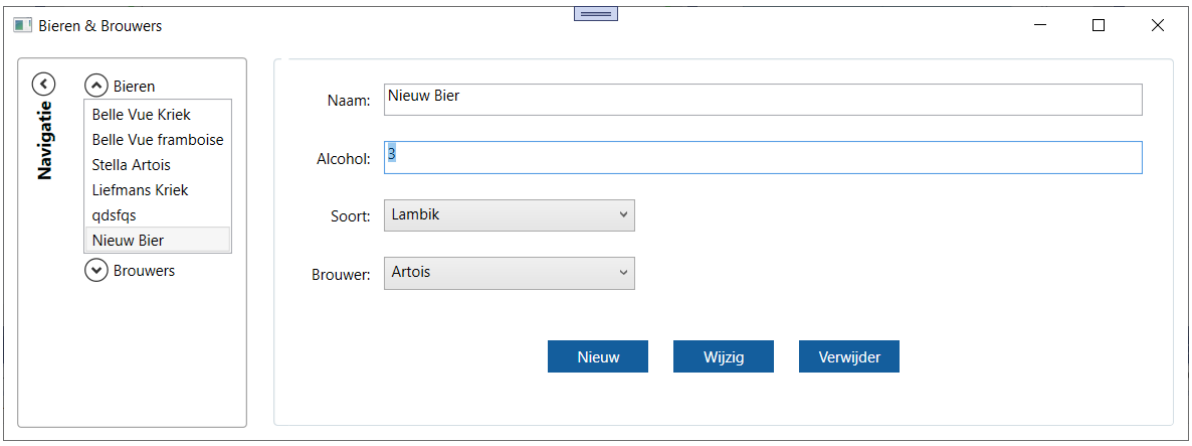
```
private void VoegBierToe()
{
    Bier bier = new Bier() { Naam = "Nieuw Bier", BierSoort = BierSoorten[0], Brouwer = Brouwers[0] };
    Bieren = new ObservableCollection<Bier>(_dataService.VoegBierToe(bier));
    SelectedBier = Bieren[Bieren.Count - 1];
}
```

10. **Voeg op dezelfde manier de methoden WijzigBierGegevens() en VerwijderBier() toe :**

```
private void WijzigBierGegevens()
{
    _dataService.WijzigBier(SelectedBier);
}

private void VerwijderBier()
{
    Bieren = new ObservableCollection<Bier>(_dataService.VerwijderBier(SelectedBier));
    if (_bieren.Count > 0) SelectedBier = _bieren[0];
}
```

De verwijderBier methode past eveneens de Bieren property aan en de SelectedBier wordt standaard op het eerste bier in de lijst gezet (indien er nog bieren in de lijst zijn)

11. **Run de app en test de buttons Nieuw, Wijzig en Verwijder van de BierDetailsView : Voorbeeld wanneer op de button Nieuw wordt geklikt, wordt er een bier toegevoegd aan de lijst met naam «Nieuw Bier » :**



12. **Test eveneens de buttons Wijzig en Verwijder uit:**