HOW TO –

B. Mock DataService maken (zonder database)

(Voorbeeldapplicatie van bieren, Brouwers en soorten bieren)

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

1. **Open Bier.cs onder de folder Models en Wijzig/ voeg de volgende code toe. We gebruiken hier OnPropertyChanged van de basis klasse ObservableObject omdat we een 2-way binding willen met de BierenDetailsView (UI).**

```
public class Bier : ObservableObject
{

    private int _bierNr;
    private string _naam;
    private double? _alcohol;
    private Brouwer _brouwer;
    private BierSoort _bierSoort;
    public int BierNr { get { return _bierNr; } set { OnPropertyChanged(ref _bierNr, value); } }
    public string Naam { get { return _naam; } set { OnPropertyChanged(ref _naam, value); } }
    public double? Alcohol { get { return _alcohol; } set { OnPropertyChanged(ref _alcohol, value); } }
    public Brouwer Brouwer
    {
        get { return _brouwer; }
        set
        {
            OnPropertyChanged(ref _brouwer, value);
        }
    }
    public BierSoort BierSoort
    {
        get { return _bierSoort; }
        set
        {
            OnPropertyChanged(ref _bierSoort, value);
        }
    }
}
```

2. **Open Brouwer.cs onder de folder Models en voeg de volgende code toe :**

```
public class Brouwer : ObservableObject
{

    #region fields
    private int _brouwerNr;
    private string _brNaam;
    private string _straat;
    private short? _postcode;
    private string _gemeente;
    private double? _omzet;
    #endregion
    #region properties
    public int BrouwerNr { get { return _brouwerNr; } set { OnPropertyChanged(ref _brouwerNr, value); }
}
    public string BrNaam { get { return _brNaam; } set { OnPropertyChanged(ref _brNaam, value); } }
    public string Straat { get { return _straat; } set { OnPropertyChanged(ref _straat, value); } }
    public short? PostCode { get { return _postcode; } set { OnPropertyChanged(ref _postcode, value); }
}
    public string Gemeente { get { return _gemeente; } set { OnPropertyChanged(ref _gemeente, value); }
}
    public double? Omzet { get { return _omzet; } set { OnPropertyChanged(ref _omzet, value); } }

    #endregion


}
```

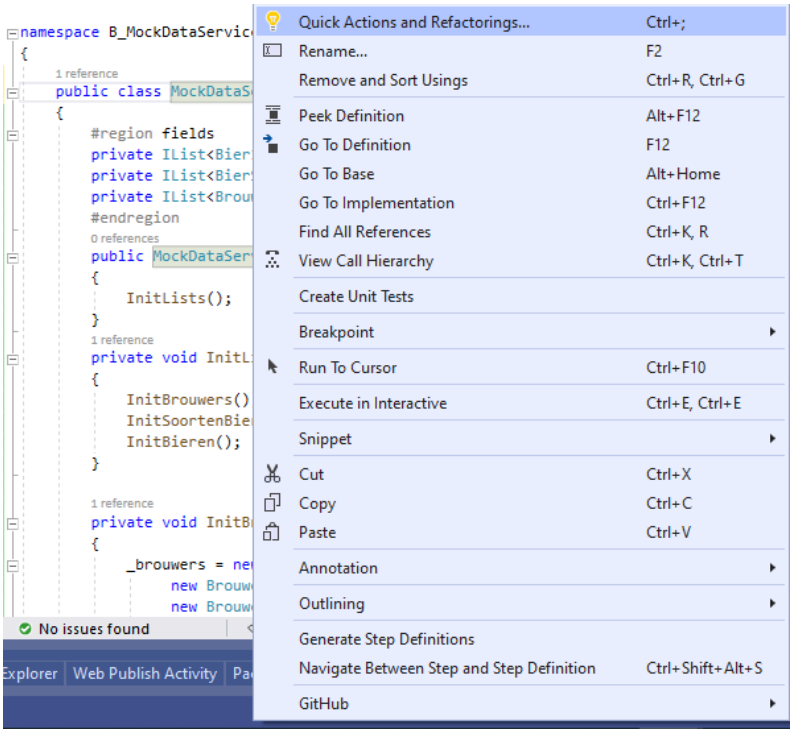
3. **Open SoortBier.cs onder de folder Models en voeg de volgende code toe :**

```
public class BierSoort : ObservableObject
{

    private int _soortNr;
    private string _soortNaam;

    public int SoortNr { get { return _soortNr; } set { OnPropertyChanged(ref _soortNr, value); } }
    public string SoortNaam { get { return _soortNaam; } set { OnPropertyChanged(ref _soortNaam, value); } }
}
```

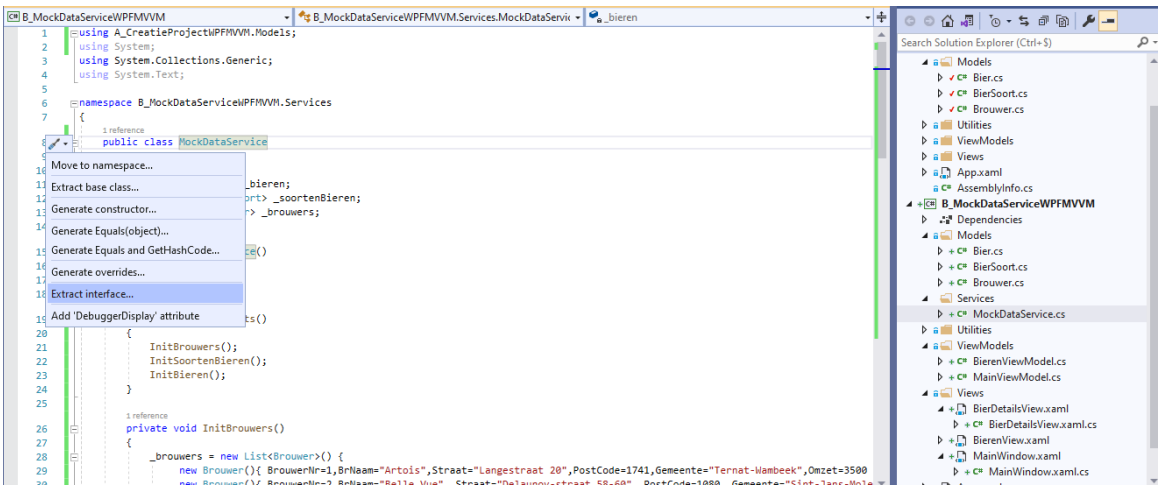
4. Maak een nieuwe folder **Services** aan. Maak hieronder een **MockDataService** Klasse. Deze zal gebruikt worden om een database source te imiteren. Later kunnen we eventueel een echte database service aanmaken voor de BierenDb sql server database.
5. In de MockDataService klasse houden **we lijsten bij van brouwers, bieren en biersoorten**. Deze initialiseren we met een aantal elementen in de constructor.
6. Voeg eveneens code toe aan de MockDataService klasse om de lijst van alle Brouwers, Bieren en biersoorten terug te geven :

```
public class MockDataService : IDataService
{
    #region fields
    private IList<Bier> _bieren;
    private IList<BierSoort> _soortenBieren;
    private IList<Brouwer> _brouwers;
    #endregion
    public MockDataService()
    {
        InitLists();
    }
    private void InitLists()
    {
        InitBrouwers();
        InitSoortenBieren();
        InitBieren();
    }

    private void InitBrouwers()
    {
        _brouwers = new List<Brouwer>() {
             new Brouwer(){ BrouwerNr=1,BrNaam="Artois",Straat="Langestraat 20",PostCode=1741,Gemeente="Ternat-Wambeek",Omzet=3500 },
             new Brouwer(){ BrouwerNr=2,BrNaam="Belle Vue", Straat="Delaunoy-straat 58-60", PostCode=1080 ,Gemeente="Sint-Jans-Molenbeek",Omzet= 300000.00},
             new Brouwer(){ BrouwerNr=3,BrNaam="Liefmans",Straat="Aalststraat 200",PostCode=9700,Gemeente="Oudenaarde",Omzet=10000 }
        };
    }

    private void InitSoortenBieren()
    {
        _soortenBieren = new List<BierSoort>() {
            new BierSoort(){ SoortNr=1, SoortNaam="Lambik"},
            new BierSoort(){ SoortNr=2, SoortNaam="Pils"},
            new BierSoort(){ SoortNr=3, SoortNaam = "Geuze"}
        };
    }
    private void InitBieren()
    {
        _bieren = new List<Bier>() {
            new Bier(){ BierNr=1,Naam="Belle Vue Kriek", Alcohol=5.2, BierSoort = _soortenBieren[2],Brouwer=_brouwers[1]},
            new Bier(){ BierNr=2,Naam="Belle Vue framboise", Alcohol=5.2, BierSoort = _soortenBieren[2],Brouwer=_brouwers[1]},
            new Bier(){ BierNr=3,Naam="Stella Artois", Alcohol=5.2, BierSoort = _soortenBieren[1],Brouwer=_brouwers[0]},
            new Bier(){ BierNr=3,Naam="Liefmans Kriek", Alcohol=6.5, BierSoort = _soortenBieren[0],Brouwer=_brouwers[2]}
        };

    }
    public IList<Bier> GeefAlleBieren()
    {
        return _bieren;
    }
    public IList<Brouwer> GeefAlleBrouwers()
    {
        return _brouwers;
    }
    public IList<BierSoort> GeefAlleBierSoorten()
    {
        return _soortenBieren;
    }
```

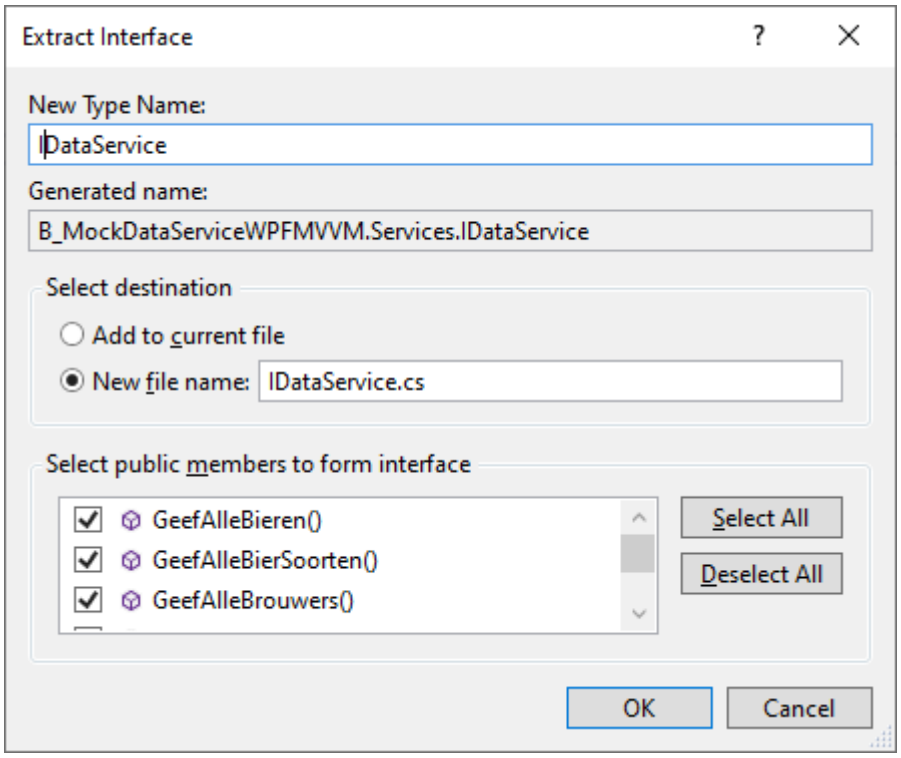
7. Maak een interface die door de MockDataService klasse wordt geïmplementeerd. Door het gebruik van een **interface kunnen we later gemakkelijker overschakelen naar een andere DataService, bv json dataservice of Sql data service,…**
   Extract een interface van de klasse MockDataService: Rechtsklik op de naam van de klasse**, en kies « Quick Actions and Refactorings.. » uit het Context menu**



8. Kies vervolgens in het context menu "**Extract interface**":



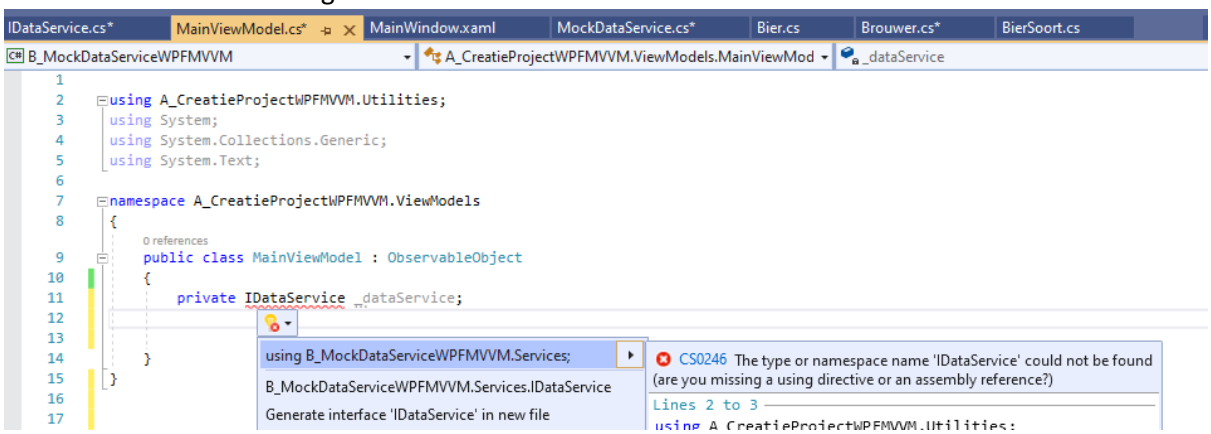
9. Je krijgt een pop-up window, geef als naam voor de interface **IDataService** en laat de destination staan op New file name. Klik op de OK-Button.



Nu is er een nieuw bestand gegenereerd IDataService.cs met de volgende Interface code:

```
public interface IDataService
{
    IList<Bier> GeefAlleBieren();
    IList<BierSoort> GeefAlleBierSoorten();
    IList<Brouwer> GeefAlleBrouwers();
}
```

10. Open **MainViewModel**.cs onder de folder ViewModels. We gaan hier de dataservice initialiseren die we in onze applicatie (en in alle viewmodels) gaan gebruiken. Merk op dat we hier een private field van het type IDataService declareren, omdat we later een andere dataservice willen kunnen gebruiken dan de MockDataService.



11. Rechtsklik op IDataService en kies voor het toevoegen van de using directive (eerste in het context menu). Zo wordt de juiste namespace van de IDataService bovenaan toegevoegd

12. We initialiseren de private field **_dataService** in de constructor van de klasse met een nieuw object van **MockDataService**:

```
using B_MockDataServiceWPFMVVM.Utilities;
using B_MockDataServiceWPFMVVM.Services;
using System;
using System.Collections.Generic;
using System.Text;

namespace A_CreatieProjectWPFMVVM.ViewModels
{
    public class MainViewModel : ObservableObject
    {
        private IDataService _dataService;

        public MainViewModel()
        {
            _dataService = new MockDataService();

        }

    }
```

13. OPen **BierenView.xaml** onder de folder views Hierin tonen we de lijst van biernamen in een **Listbox (Verwijder het TextBlock)**

```
<UserControl x:Class="B_MockDataServiceWPFMVVM.Views.BierenView"
             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:B_MockDataServiceWPFMVVM.Views"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800">

    <Grid>
        <ListBox ItemsSource="{Binding Bieren}" SelectedItem="{Binding SelectedBier}" DisplayMemberPath="Naam"/>
    </Grid>
</UserControl>
```

14. Open **BierenViewModel**.cs onder de folder ViewModels en zet **ObservableObject** als basisklasse.
    Voeg een constructor toe die een IDataService als parameter neemt die een private field van hetzelfde type initialiseert. Roep de dataservice aan om alle bieren terug te geven en zet deze als ObservableCollection<Bier> in Bieren (public property met bijhorende private field)
    Voeg nog een public Property SelectedBier om de SelectedItem van de listbox hierop te kunnen binden:

```
    public class BierenViewModel:ObservableObject
    {
        private IDataService _dataService;
        private ObservableCollection<Bier> _bieren;
        private Bier _selectedBier;
        public BierenViewModel(IDataService dataService)
        {
            _dataService = dataService;
            Bieren = new ObservableCollection<Bier>(dataService.GeefAlleBieren());
        }
```

```
        public ObservableCollection<Bier> Bieren {
            get { return _bieren; }
            set { OnPropertyChanged(ref _bieren, value); }
        }

        public Bier SelectedBier
        {
            get { return _selectedBier; }
            set { OnPropertyChanged(ref _selectedBier, value); }
        }
    }
        }
```

15. Open MainWindow.xaml en zet de DataContext van Grid `DataContext="{StaticResource MainViewModel}"` en van BierView `DataContext="{Binding BierenVM}"`

```
<Window x:Class="B_MockDataServiceWPFMVVM.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:B_MockDataServiceWPFMVVM" xmlns:viewmodels="clr-
namespace:B_MockDataServiceWPFMVVM.ViewModels" xmlns:views="clr-namespace:B_MockDataServiceWPFMVVM.Views"
        mc:Ignorable="d"
        Title="MainWindow" MinHeight="350" MinWidth="525">
    <Window.Resources>
        <viewmodels:MainViewModel x:Key="MainViewModel"/>
    </Window.Resources>
    <Grid  DataContext="{StaticResource MainViewModel}">
        <views:BierenView DataContext="{Binding BierenVM}"/>
    </Grid>
         </Window>
```

16. Open **MainViewModel**.cs en voeg een constructor toe die een BierenViewModel aanmaakt. Voor ook een public property `BierenVM` toe, die wordt gebonden op de BierenView. Deze is van het type **BierenViewModel en deze wordt geïnitialiseerd in de constructor van MainViewModel.**

```
    public class MainViewModel : ObservableObject
    {
        private IDataService _dataService;
        private BierenViewModel _bierenVM;
        public MainViewModel()
        {
            _dataService = new MockDataService();
            BierenVM = new BierenViewModel(_dataService);

        }
        public BierenViewModel BierenVM
        {
            get { return _bierenVM; }
            set { OnPropertyChanged(ref _bierenVM, value); }
        }
    }
```

Run Nu de app, je ziet de listbox met bieren :