# MVVM oefeningen

Contents

[MVVM oefeningen 1](#_Toc432582688)

[LandInfo MVVM 1](#_Toc432582689)

[MasterDetailVM 1](#_Toc432582690)

[MasterDetailDemo.xaml (de View) 2](#_Toc432582691)

[LandInfoFilteringVM 2](#_Toc432582692)

[FilteringDemo.xaml (de View) 2](#_Toc432582693)

[StudentModulePunt MVVM 2](#_Toc432582694)

[Aangepaste webservice 2](#_Toc432582695)

[StudentModulePuntVM 2](#_Toc432582696)

[MainPage.xaml (de View) 3](#_Toc432582697)

[MainPage.xaml.cs (de codebehind file) 3](#_Toc432582698)

[Round Up 3](#_Toc432582699)

[Parking MVVM 4](#_Toc432582700)

Opmerking: bij deze opgaves wordt verondersteld dat u het theorie document van deze les ook doorneemt.

## LandInfo MVVM

U krijgt een startproject waarin de visualisatie van een Master/Detail en Filtering pagina zijn opgenomen. Ook de repository is reeds in het project aanwezig. De start ViewModel klasses zijn ook aanwezig en reeds voorzien van een correcte INotifyPropertyChanged implementatie.

### MasterDetailVM

Op basis van de UI rechts bepaalt u welke controls in de UI informatie uit het ViewModel zullen ophalen

* Welke UI properties worden gebind?
  + Welke worden TwoWay gebind?
* Op basis van deze informatie voorzien we properties voor de ViewModel klasse MasterDetailVM
  + Welke properties (naam en return type)
  + Indien ze TwoWay gebind zullen worden zullen we gebruik maken van een eigen backing field, zodat we in de setter eigen logica kunnen coderen
  + Door welke andere ViewModel properties wordt een property beïnvloedt? Dit geeft aan welke setters welke OnPropertyChanged oproepen nodig zullen hebben (eventueel meer dan 1!)
* Programmeer de MasterDetailVM klasse op basis van voorgaande informatie
  + Vergeet niet waar nodig de Bindings te verwittigen!
* Voorzie (in App.xaml) een static resource voor deze ViewModel klasse

### MasterDetailDemo.xaml (de View)

* Stel de DataContext van de Page in op de StaticResource van zijn ViewModel
* Bind de control properties aan de overeenkomstige ViewModel properties
  + Gebruik TwoWay binding waar nodig!

### LandInfoFilteringVM

Op een gelijkaardige manier zullen we het Filtering model opbouwen:

* Welke UI properties worden gebind?
  + Welke worden TwoWay gebind?
* Op basis van deze informatie voorzien we properties voor de ViewModel klasse FilteringVM
  + Welke properties (naam en return type)
  + Indien ze TwoWay gebind zullen worden zullen we gebruik maken van een eigen backing field, zodat we in de setter eigen logica kunnen coderen
  + Door welke andere ViewModel properties wordt een property beïnvloedt? Dit geeft aan welke setters welke OnPropertyChanged oproepen nodig zullen hebben (eventueel meer dan 1!)
* Programmeer de FilteringVM klasse op basis van voorgaande informatie
  + Vergeet niet waar nodig de Bindings te verwittigen!
* Voorzie (in App.xaml) een static resource voor deze ViewModel klasse

### FilteringDemo.xaml (de View)

* Stel de DataContext van de Page in op de StaticResource van zijn ViewModel
* Bind de control properties aan de overeenkomstige ViewModel properties
  + Gebruik TwoWay binding waar nodig!

## StudentModulePunt MVVM

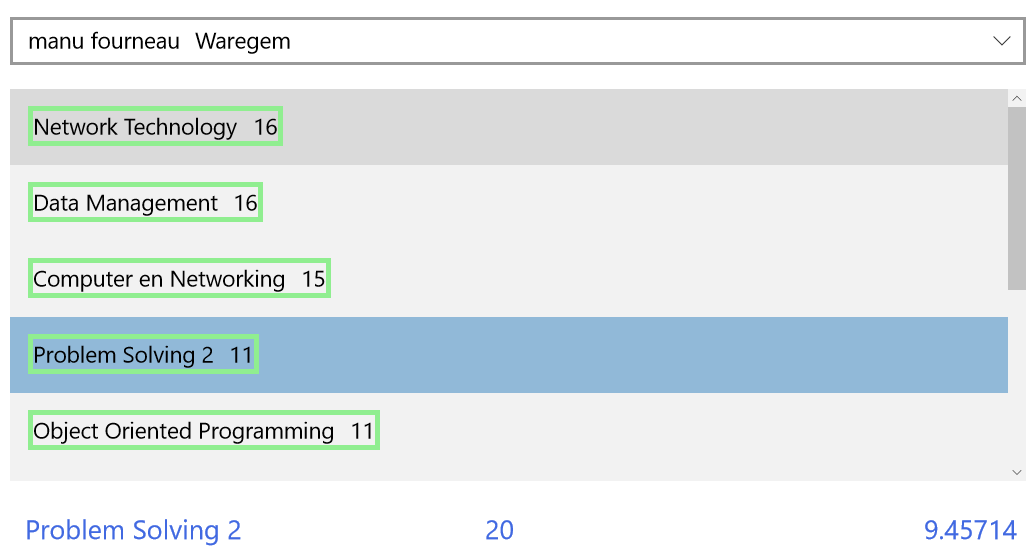
### Aangepaste webservice

De webservice werd uitgebreid met een aantal extra mogelijkheden (en bug fixes). De aangepaste webservice en bijhorende repository klasses zijn op leho te vinden. Deze nieuwe web-API stemt meer overeen met wat u in realiteit mag verwachten.

### StudentModulePuntVM

Maak een folder ViewModel in de solution en maak hierin een ViewModel klasse voor uw Page. In de veronderstelling dat u MainPage gebruikt heten we de ViewModel klasse MainPage VM.

* Voorzie bij de static resources (App.xaml) een static resource voor deze ViewModel klasse
  + Indien er nog static resources voor repositories gedefinieerd zijn mag u deze verwijderen
* Voorzie onmiddellijk de INotifyPropertyChanged code (dezelfde als in de vorige labos) en plaats deze in een region die u dan voor de rest van de opgave kan dichtklappen;
* Op basis van de te tonen UI bepaal je de nodige properties van de ViewModelKlasse:



* Welke controls worden gebruikt
  + welke properties worden gebind?
  + Welke properties worden TwoWay gebind?
* Op basis van de control properties die worden gebind kunnen we de properties van het ViewModel bepalen
  + Geef ze een naam
  + Welk return type hebben deze properties?
  + Door welke andere ViewModel properties worden ze beïnvloedt?
    - Bv: de getoonde module lijst wordt beïnvloedt door de geselecteerde student
* Programmeer de ViewModel klasse op basis van de voorgaande informatie
  + Vergeet niet op de juiste moment momenten de Bindings te verwittigen (door gebruik te maken van de INotifyPropertyChanged interface!)

### MainPage.xaml (de View)

* Pas de XAML code van de page aan:
  + Gebruik de static resource van het ViewModel als DataContext van de MainPage
  + Bind de controls aan de properties van het ViewModel, TwoWay waar nodig
  + Vergeet de SelectedItem properties niet!

### MainPage.xaml.cs (de codebehind file)

* Verwijder de onnodige C# code in de codebehind file van MainPage (enkel de ctor met de InitializeComponent oproep blijft nodig)

### Round Up

Indien u er in slaagt om de StudentModulePunt applicatie zonder extra code behind werkend te krijgen hebt u het MVVM pattern voldoende toegepast!

## Parking MVVM

Zoals u al verwacht: naar analogie van de vorige oefeningen:

* Op basis van de vorige oplossing (de UI) ontwikkelt u een ViewModel
* Voorzie een static resource voor dit model
* Bind de XAML- elementen aan de overeenkomstige ViewModel elementen
* Verwijder de C# code die u niet langer nodig hebt

Veel Succes!