# (Data)Binding oefeningen

Contents

[(Data)Binding oefeningen 1](#_Toc432000386)

[Aanpassen van de StudentModulePuntApp 2](#_Toc432000387)

[Student visualisatie in de ComboBox 2](#_Toc432000388)

[StudentModulePunt visualisatie in de ListBox 3](#_Toc432000389)

[StudentModulePunt visualisatie: een StyleSelector voor de Grid 4](#_Toc432000390)

[Module statistiek tonen 4](#_Toc432000391)

[StudentModulePunt app: stand van zaken 5](#_Toc432000392)

[Aanpassen van de ParkingApp 5](#_Toc432000393)

[Parking app: wegwerken van de usercontrols 5](#_Toc432000394)

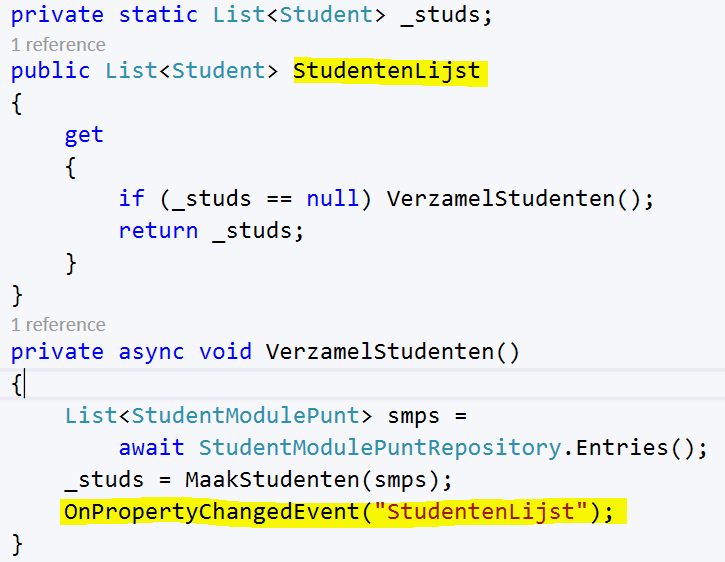
[Parking app: gebruik van MapControl class 6](#_Toc432000395)

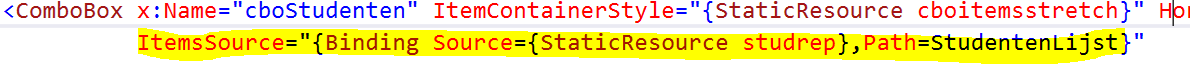
## Aanpassen van de StudentModulePuntApp

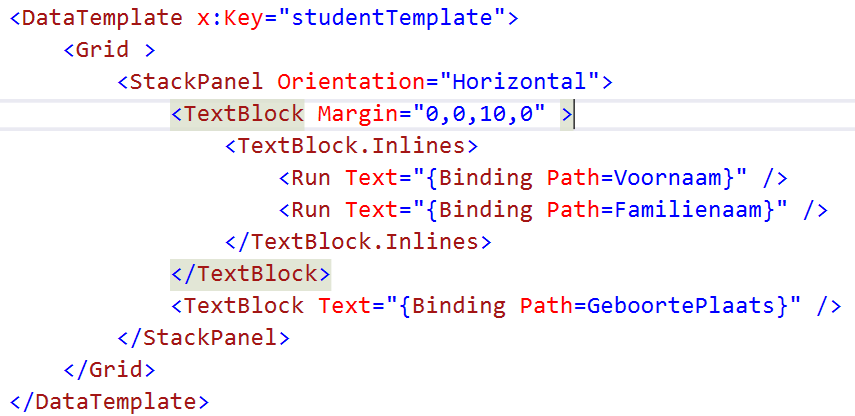
Het is de bedoeling om alle usercontrols te vervangen door standaard controls en datatemplates (met binding). Volg hiervoor het onderstaande stappenplan:

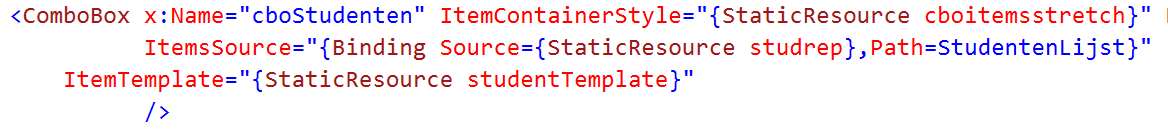
* Vergeet niet de webservice app te starten
* Maak een folder DataTemplates, met daarin een ResourceDictionary DataTemplates en voeg deze toe aan de merged dictionary in app.xaml. Zoals de naam reeds aangeeft zullen we in deze file de DataTemplate resources definiëren

### Student visualisatie in de ComboBox

* Maak een resource StudentenRepository met als key **studrep**
* Implementeer de INotifyPropertyChanged interface op de klasse StudentenRepository. Indien de PowerPoint en het Word document onvoldoende duidelijk zijn verwijs ik graag naar [INotifyPropertyChanged interface](https://msdn.microsoft.com/en-us/library/system.componentmodel.inotifypropertychanged(v=vs.110).aspx)
* Voorzie op de klasse StudentenRepository een ReadOnly (alleen een Get) property StudentenLijst die een List<Student> objecten terug geeft
  + Het is een property, er zijn dus geen argumenten
  + We zullen de lijst, eens ze berekend is bijhouden in \_studs, zodat we ze onmiddellijk beschikbaar hebben indien ze nog eens worden opgevraagd
  + Indien \_studs nog niet werd bepaald zullen we ze verzamelen via een Async method. Omdat een property niet async gedefinieerd mag worden kunnen we deze call niet awaiten ☹.
    - De verzamelmethode moet asynchroon omdat ze de Entries methode van de StudentModulePunt repository aanroept, en deze is async gedefinieerd (omdat ze een webservice raadpleegt)
  + Het laatste commando van de verzamel methode gebruikt de INotifyPropertyChanged interface om eventuele bindings te verwittigen dat de StudentenLijst property werd gewijzigd. De UI zal de lijst opnieuw opvragen om deze te tonen.
    - Zonder de INotifyPropertyChanged interface zou de Binding de oorspronkelijke, lege lijst, blijven tonen
    - Bemerk dat we als propertynaam **StudentenLijst** meegeven. Dit is niet de naam van de uitvoerende routine: de routine VerzamelStudenten is trouwens geen property, en dus helemaal niet de property waarvan we de Binding willen verwittigen
* Op de studenten combobox stellen we de ItemsSource in op de **StudentenLijst** property van de eerder aangemaakte static resource **studrep:**



* Indien u nu de toepassing zou uitvoeren wordt in de combobox voor elke student de ToString waarde getoond.
* Definieer een DataTemplate (x:Key=”StudentTemplate”) in de DataTemplates.xaml file waarvan de structuur identiek is aan deze van uw usercontrol ctlStudent
  + Hint: copieer de ganse Grid- tag
  + Verwijder alle x:Name definities (die waren nodig voor de C# code)
  + Definieer de nodige bindings om de inhoud van een Student object naar de juiste UI properties te copieren
  + Stel deze static resource in als ItemTemplate voor de ComboBox:



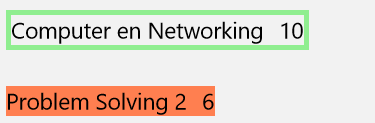
* Indien u nu de toepassing opstart zal voor elke student de DataTemplate getoond worden
  + Op dit punt mag u de usercontrol ctlStudent verwijderen

### StudentModulePunt visualisatie in de ListBox

Indien u in de combobox een item kiest is het selected item een Student object. Om de student modulepunten van dit Student- object te visualiseren gaan we als volgt te werk:

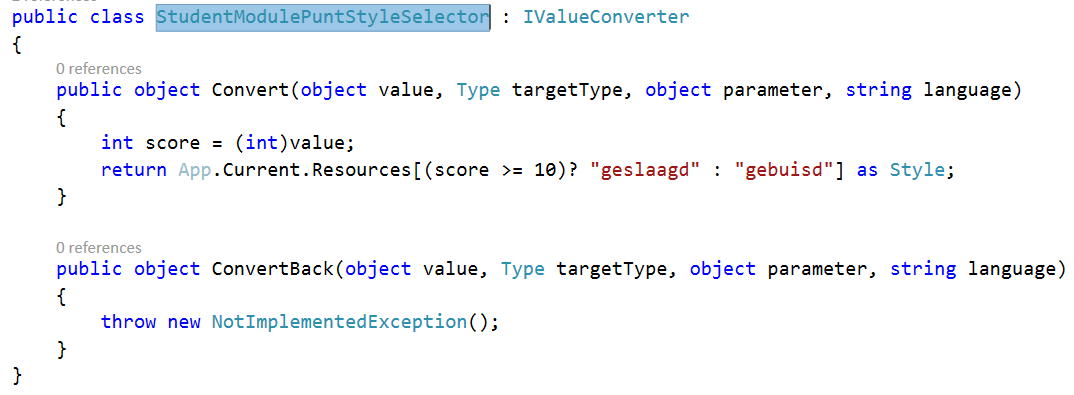
* Verwijder de usercontrol die alle ctlStudentModulePunt uit de MainPage.xaml en vervang hem door een ListBox
  + Op dit punt mag u die usercontrol klasse verwijderen
* In de vorige versie van de app was u reeds in staat om voor een gekozen student zijn StudentModulePunt lijst te verzamelen. Gebruik deze code om, bij het kiezen van een student in de combobox (het SelectionChanged event weetjewel?) de ItemsSource van de zonet toegevoegde ListBox in te stellen op de StudentModulePunten van de gekozen student
  + Op dit moment zal per studentmodulepunt de ToString waarde getoond worden
* Voeg in de datatemplates file een datatemplate toe voor de StudentModulePunt gegevens. Copieer ook hier de Grid van de overeenkomstige usercontrol (geen screenshot, want gelijkaardig aan de StudentTemplate)
  + Verwijder de Name attributes
  + Voeg Bindings toe zodat de juiste informatie uit een StudentModulePunt datacontext getoond wordt
  + Verwijder de usercontrol. Eventuele styles die nog aanwezig zijn worden bij de application Styles geplaatst
* Stel de zopas aangemaakt template in als ItemTemplate van de ListBox
* Indien u nu de toepassing start zal u na het kiezen van een student zijn StudentModulePunt informatie te zien krijgen, zoals gedefinieerd in de DataTemplate

### StudentModulePunt visualisatie: een StyleSelector voor de Grid

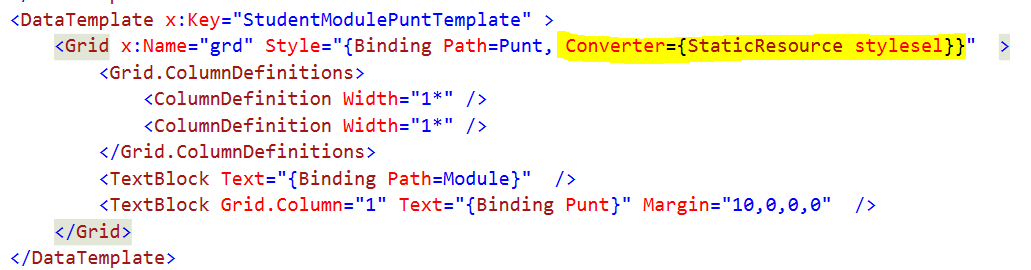


De grid van de StudentModulePunt template kan gestyled worden met een **geslaagd** of een **gebuisd** style. We veronderstellen dat deze styles reeds werden gedefinieerd en in de application resources beschikbaar zijn gesteld. Om deze styles nu in de DataTemplate te selecteren gaan we als volgt te werk:

* Ontwikkel de klasse StudentModulePuntStyleSelector in de Helpers folder. Deze implementeert de IValueConverter interface die een punt zal omvorm tot een style:



* Definieer in de DataTemplates file een StaticResource voor deze klasse met als Key **stylesel**
* Pas de Grid in de template aan en bind de Style property aan de Punt property van de DataContext (het StudentModulePunt). Stel de converter in op de zopas gemaakte static resource om het punt om te vormen tot een Style:



### Module statistiek tonen

* Indien in de ListBox een element wordt gekozen (SelectionChanged) moet de detail van de gekozen module getoond worden. Dit is gelijkaardig aan de oplossing van vorige week.

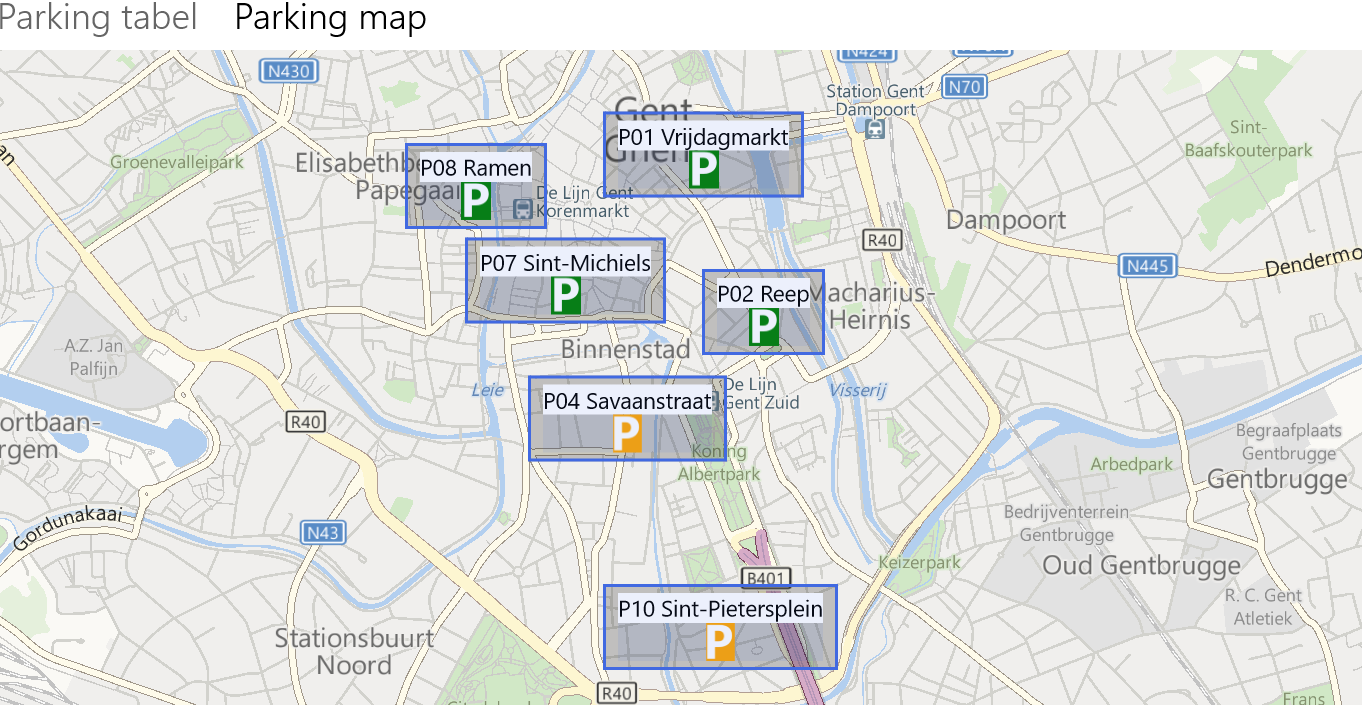
### StudentModulePunt app: stand van zaken

* Alle usercontrols zijn verwijderd
* Selecties in een ItemsControl (de ComboBox en de ListBox) worden via een EventHandler opgevangen om de nodige logica uit te voeren. Volgende week, bij het uitwerken van een geschikt ModelView zal dit niet langer nodig zijn.

## Aanpassen van de ParkingApp

Het aanpassen van deze app heeft twee doelstellingen:

* Het wegwerken van de usercontrols door de introductie van DataTemplates (met style selecties);
* Het gebruik maken van de Bing- maps om de parking situatie te visualiseren op een kaart:

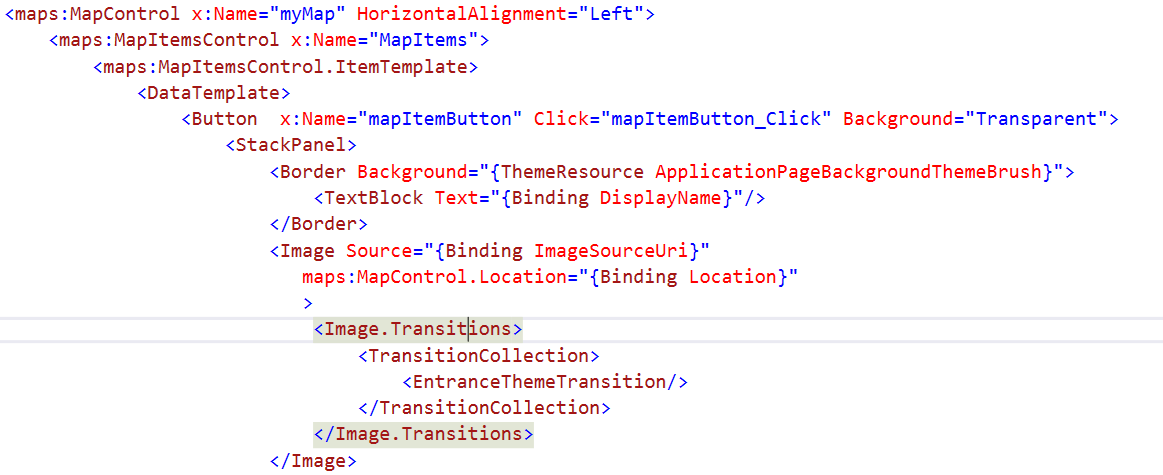
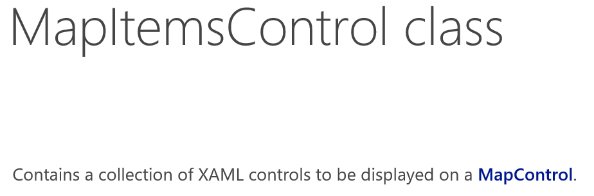


### Parking app: wegwerken van de usercontrols

* Maak een DataTemplates folder met hierin een DataTemplates.xaml ResourceDictionary
* Definieer in deze ResourceDictionary een DataTemplate om Parking model objecten te visualiseren
  + Copieer de ganse grid van de usercontrol in een DataTemplate tag
  + Verhuis eventuele styles naar de application resources
  + Verwijder de usercontrols in de toepassing
* Definieer een StaticResource voor de ParkingRepository klasse
* Bind de ItemsSource van de GridView aan de Parkings property van de ParkingRepository static resource
  + Omdat deze bij het opstarten nog null is zal de GridView geen elementen tonen
  + Implementeer INotifyPropertyChanged op de repository klasse
  + De async methode die de parkings verzamelt zal op het einde van zijn uitvoering de Binding verwittigen dat de Parkings- property een nieuwe waarde kreeg (dit is volledig gelijkaardig aan de StudentenLijst invulling van de vorige oefening).
* Omdat de ParkingStatus niet bij elke Parking aanwezig is heb ikzelf een ParkingStatusExtra property gedefinieerd. Deze geeft de ParkingStatus terug indien deze werd ingevuld, anders wordt een default parkingstatus terug gegeven. Mijn Bindings die de parkingstatus nodig hebben gebruiken dan ook de extra versie ☺
* Definieer een ParkingStyleSelector die, op basis van een ParkingCapacityLevel de juiste style selecteer uit de application resources
* Maak van deze klasse een static resource en gebruik deze om de Style property van de Grid in de Parking DataTemplate te binden aan de ParkingCapacityLevel van de DataContext. Ook dit is volledig gelijkaardig aan de StudentModulePunt oplossing

### Parking app: gebruik van [MapControl class](https://msdn.microsoft.com/en-us/library/windows/apps/xaml/windows.ui.xaml.controls.maps.mapcontrol.aspx)

We zullen dezelfde informatie visualiseren op een kaart. Om beide oplossingen op dezelfde page te kunnen plaatsen gaan we als volgt te werk:

* Voeg een Pages folder toe aan de project structuur
* Voeg aan de Pages- folder een Page toe met als naam ParkingGridView. Verplaats alle XAML code en alle .cs code van de MainPage (waar uw huidige oplossing gecodeerd werd) naar deze nieuwe pagina
* In de nu lege (op een Grid na) MainPage.xaml file plaatsen we een Pivot- control.
* In een eerste PivotItem (Header Parking tabel) plaatsen we een instantie van de zopas gemaakte Page class
  + Indien u nu de app start zou alles normaal moeten werken, op eventueel wat opmaak na (die u zelf kan corrigeren)
* Voeg aan de Pages folder de Page ParkingKaart toe. Voorzie in de Pivot control van de MainPage een PivotItem en plaats hierin een ParkingKaart Page object
* In de [Windows Universal Samples](https://github.com/Microsoft/Windows-universal-samples) vindt u een MapControl sample die (scenario 3) toelicht hoe points of interest aan een kaart worden toegevoegd. Op basis van dit voorbeeld zullen we onze eigen parkings aan de kaart toevoegen
  + De MapControl is in de toolbox terug te vinden onder ‘All XAML Controls’. Zet er één op de kaart page
  + Vervang de MapControl in uw toepassing door de MapControl tag (met zijn inhoud) uit het sample:
    - De maps namespace start hier met een kleine letter m
    - U herkent een DataTemplate zoals we die zelf reeds maakten
  + Zoek de breedtegraad en de lengtegraad van Gent en gebruik dit om de kaart op Gent te centreren. Stel ook een accepteerbaar zoomlevel in voor uw toepassing
    - Hint: in scenario 3 van het sample, kijk de code van de MyPage\_Loaded na
* In het sample gebruikt men een klasse PointOfInterestsManager, wij zullen omwille van consistente naamgeving **PointOfInterestRepository** gebruiken
* Bemerk in de addXamlChildrenButton\_Click routine dat het toevoegen van de PointOfInterest elementen gebeurt door de MapItems.ItemsSource in te stellen. De documentatie van [MapItemsControl class](https://msdn.microsoft.com/en-us/library/windows/apps/xaml/windows.ui.xaml.controls.maps.mapitemscontrol.aspx) leert ons:
* Om het onszelf gemakkelijk te maken, en niet uit te hoeven pluizen welke properties ingesteld moeten worden om goed op de MapControl getoond te kunnen worden copiëren we gewoon de PointOfInterest klasse van het sample.
* De PointOfInterestRepository klasse zal op basis van de Parkings van de ParkingRepository een lijst van PointOfInterest objecten samenstellen
  + Indien u die lijst kan aanmaken met behulp van LINQ bent u goed bezig
  + Ook nu zal de repository klasse nood hebben aan een INotifyPropertyChanged uitwerking om de kaart te verwittigen wanneer alle Points of interest verzameld zijn.
* Indien u het sample meer in detail napluist merkt u dat de merkers op de kaart uit de assets folder worden gekozen (op basis van de ImageSourceUri).
* Maak 3 png files met een juiste kleur P en pas de ImageSourceUri van elke parking aan zodat deze de juiste image gebruikt

Opmerkingen: indien u offline kaarten hebt gedownload worden deze door de mapcontrol gebruikt. Dit is een Windows 10 feature, die dus ook op de windows mobile versie beschikbaar is (niet op de emulator ☺).