# Adaptive UI oefeningen

Contents

[Adaptive UI oefeningen 1](#_Toc436236710)

[Oefening 1 2](#_Toc436236711)

## Oefening 1

Op de windows 10 developers day werd een volledig uitgewerkte oefening omtrent adaptive UI voorgesteld. Deze opgave is voorzien van een volledig uitgewerkt stappenplan, alle bronmateriaal en een model oplossing, ideaal dus om in deze les te recupereren. U vindt deze oefening op leho onder de naam **02. Adaptive UI**.

## Oefening 2

### Inleiding

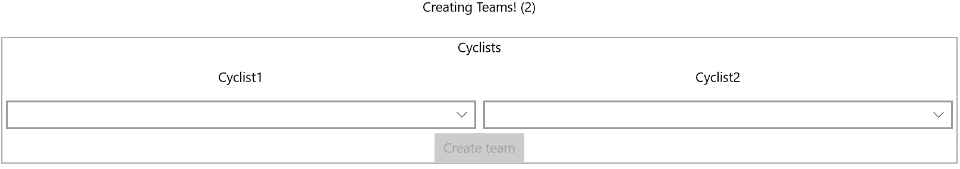
U vindt op leho een niet adaptive solution die wat zesdaagse logica implementeert:

* Er is een LandingPage (de Home page) die een image toont;
* Er is een create team page waarmee renners tot een team gekoppeld kunnen worden. De logica in de app laat enkel toe twee renners met dezelfde sponsor in één team te plaatsen;
* Een pagina die dernys punten toekent: er wordt een team geselecteerd en op het +- teken geklikt. De logica in de app laat enkel toe om per dernys geen twee keer punten aan hetzelfde team toe te kennen. Per dernys worden 5, 4, 3,2 of 1 punt toegekend, in dalende volgorde. Nadat alle punten zijn toegekend keren we terug naar de home page

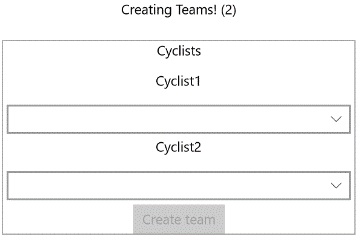
Bemerk dat de navigatie in deze app gebeurt mbv van een commandbar. Deze is voorzien in de MainPage. De DataContext van de Mainpage is gebind aan het application viewmodel zodat de navigatie commando’s ook daar gezocht moeten worden (dat wil zeggen: het is de MainPage die de navigatie verzorgt, niet elk van de verschillende pagina’s (wat ook een valabele optie was geweest)).

### Aanpassing 1: create team: SelectTeamMembers DataTemplate

We wensen voor smalle UI’s de standaard visualisatie aan te passen:

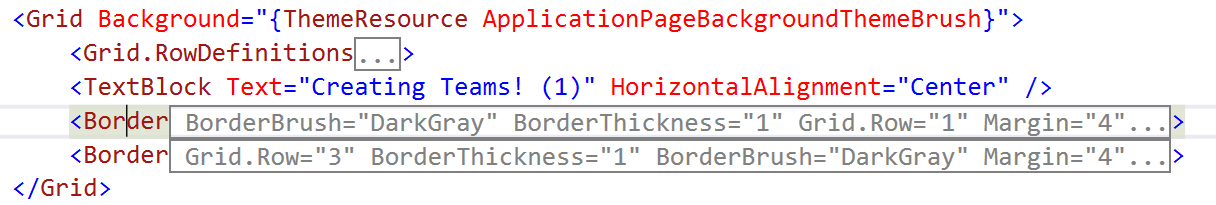


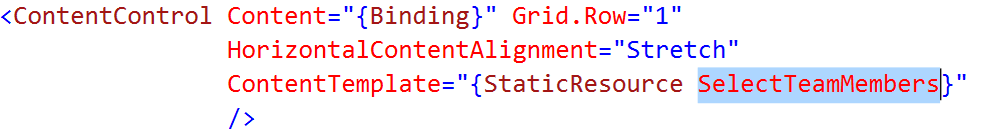
Bovenstaande UI gebruiken we bij brede schermen, onderstaande UI bij smalle schermen.



Wanneer we de CreateTeamPage er op nakijken blijkt de volledige inhoud van de tweed rij (Grid.Row 1) een samenhangend geheel te vormen: we selecteren twee renners en maken er een team van. Om hier met een datatemplate te werken gaan we als volgt te werk:

* Klap het border toe zodat we het gemakkelijk kunnen knippen



* Knip de ganse Border tag (Row 1) en plak deze in een DataTemplate in de DataTemplates.xaml file. Geef deze DataTemplate de naam SelectTeamMembers.
* Plaats een ContentControl waar de geknipte Border tag stond (Grid.Row = 1).
  + Bind de Content aan het ViewModel
  + stel de ContentTemplate in op de zopas aangemaakte SelectTeamMembers DataTemplate (intellisense krijg je enkel na een succesvolle compilatie)
  + zet de HorizontalContentAlignement op Stretch

Tot nu toe is er visueel geen verschil: de xaml- code is verhuisd naar de DataTemplates.xaml file. We zullen op basis van VisualStates de twee UI’s aansturen.

* Bemerk dat de smalle UI twee extra rijen bevat. Wijzig de SelectTeamMembers template en voorzie de twee extra rijen (kijk na op welke rij posities deze rijen moeten geplaatst worden. Geef ze ook alletwee een x:Name zodat ze straks in de State setters benoemd kunnen worden (State setters werken op basis van x:Name!).
* Definieer Visual States voor een minimum breedte van 0 en 1024 (vergeet de UserControl tag niet, zie het theorie document).
  + Definieer de statetriggers
  + Definieer de setters. Ik verwacht dat de smalle state volgende zaken instelt:
    - De twee nieuwe rijen krijgen een gepaste hoogte (kijk die na voor de overeenkomstige rijen die reeds voorkomen) en de controls voor de tweede wielrenner worden verplaatst (instellen van de Grid.Row en Grid.Column properties)
    - Vergeet niet de breedte van de tweede kolom op 0 te plaatsen (u moet die dan wel eerst een naam geven).
  + Omdat de default XAML waardes deze voor de brede situatie zijn moeten er in de brede state geen setters gedefinieerd worden
* Bemerk dat de CyclistDataTemplate te breed is voor de smalste situatie. Definieer een CyclistDataTemplateVerySmall waarin enkel de voornaam en familienaam van de renner wordt getoond, en gebruik deze in de smalle state van de SelectTeamMembers template:

 werkt, evenals

 U moet deze wel voor beide comboboxen instellen (en vergeet niet de eerste combobox een naam te geven!

Indien u fouten maakt in (minstens) 1 setter van een state wordt geen enkele setter uitgevoerd. Foutboodschappen bij het falen heb ik nog niet kunnen terugvinden. Very annoying!!

### Aanpassing 2: create team: TeamDataTemplateAdaptive DataTemplate

Voor de getoonde TeamData moeten we een gelijkaardige oefening maken. Ze wordt wel iets complexer (er zijn meer verschillende situaties):



De breedste situatie toont alle gegevens van de twee renners naast elkaar, met de sponsor erboven. De dubbele sponsor informatie is weinig zinnig maar dient hier enkel ter illustratie van ‘veel data elementen’. De kolommen van de wielrenner informatie hebben vastgecodeerde breedtes.



Indien er per renner onvoldoende plaats is wordt enkel de familienaam en de voornaam getoond (elk krijgt 50% van de beschikbare plaats).



Wanneer de beschikbare breedte nog smaller wordt worden de twee renners van het team onder elkaar geplaatst (zoals ook bij de teamselectie gebeurde)

Indien we nadenken over bovenstaande visualisatie kunnen we dit op verschillende manieren oplossen. In het onderstaande stappenplan zullen we een adaptive datatemplate maken voor het team, die gebruik maakt van een adaptive datatemplate voor de renners:

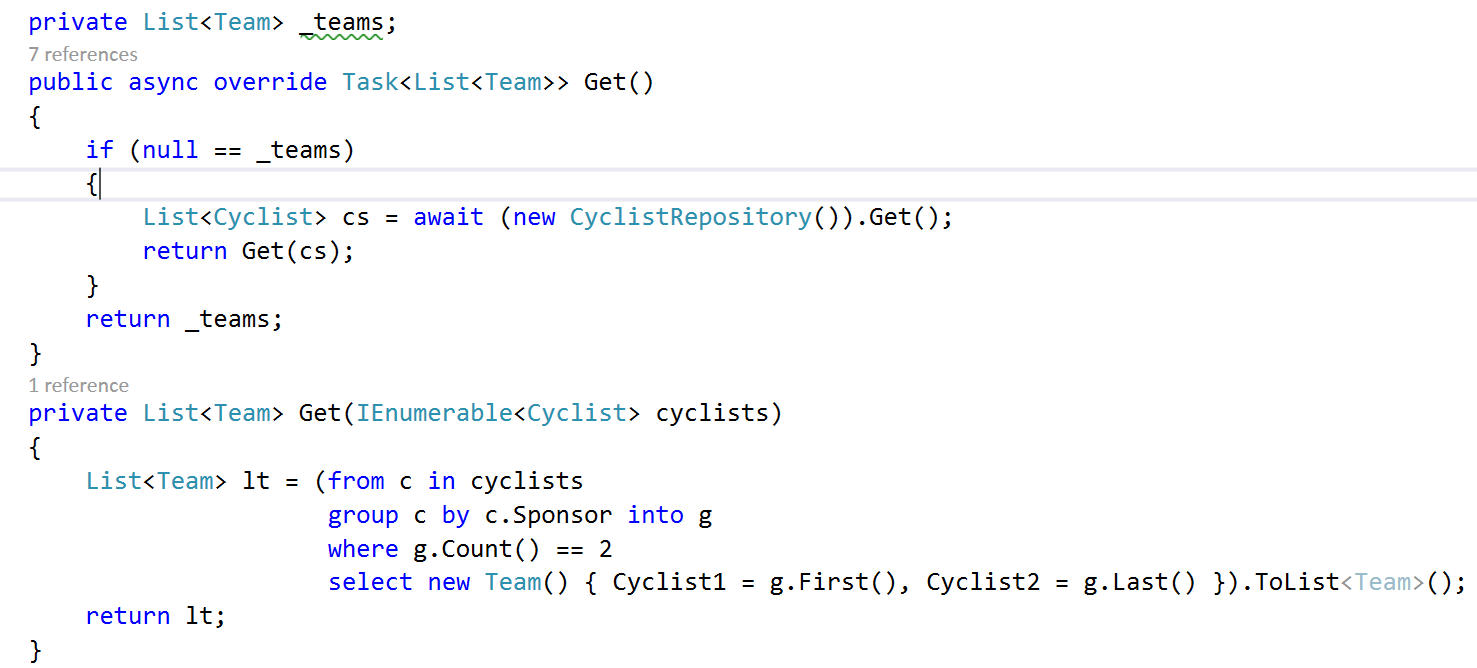
* Copieer de CyclistDataTemplate en heet deze copy CyclistDataTemplateAdaptive. Voorzie hierin twee states: minimum width 0 en minimum width 1024.
  + In de smalle state zullen we de land- en sponsor kolom een breedte 0 geven (ze worden onzichtbaar) en de familienaam en voornaam kolom krijgen elk de helft van de beschikbare breedte (1\*).
  + De brede state is leeg en gebruikt de default instellingen.
* Copieer de TeamDataTemplate naar een nieuwe datatemplate met naam TeamDataTemplateAdaptive.
  + Voorzie een state voor minimum 0 pixels en een (lege) state voor 548 pixels;
  + In de smalle state verplaatsen we de tweede renner opnieuw onder de eerste renner, dit verloopt volledig gelijkaardig aan het vorige stuk van de oefening.
  + Vanaf 548 pixels (de state met de lege setters) wordt teruggevallen op de default XAML code waardoor de twee renners terug naast elkaar geplaatst worden.
  + Stel de ContentTemplate in de default code in op de zopas gedefinieerde CyclistDataTemplateAdaptive template.

Doordat beide templates op een verschillende breede properties wijzigen krijgen we een getrapte aanpassing.

## Oefening 3

Zoals in de theorie cursus werd geïllustreerd is het behoorlijk eenvoudig om zelf een StateTrigger te coderen. Programmeer een trigger om na te gaan of er internet toegang is en demonstreer zijn werking (bijvoorbeeld door een save knop te disablen wanneer het netwerk niet beschikbaar is).

## Oefening 4

Indien u het samenstellen van de teams automatisch wenst te laten gebeuren kan u een nieuwe TeamRepository klasse ontwikkelen (u start met een copy van de huidige TeamRepository klasse) waarin u volgende code voorziet:

* We vragen alle renners op;
* We groeperen deze op sponsor  
  en stellen Teams samen met de  
  eerste en laatste renner van   
  elk groepje. Indien er in elk  
  groepje exact 2 renners zijn   
  (dit is het geval met de demo renners) dan krijgen we een correcte team indeling!
* Het is natuurlijk ook mogelijk om zelf wat deelproblemen te schrijven die op basis van de sponsor de renners in teams verdeeld. Indien u bovenstaande query zelf vindt (zelfs met wat Google werk) mag u tevreden zijn over uw LINQ kennis.

## Oefening 5

Voor het dernys venster wensen we ook een adaptive UI die het team horizontaal visualiseert indien voldoende ruimte, en de team members onder elkaar plaatst indien we een smalle display hebben.