Taakgroep: Data

Context

Bij het implementeren van verschillende data bronnen moest je uiteindelijk 1 data bron tonen op je console. Het werd al snel duidelijk dat de console niet echt geschikt is om snel even wat aan te passen en uit te breiden. Met WPF heb je deze limitaties niet. Je kan makkelijk nieuwe componenten op je scherm slepen. Mocht je scherm geen ruimte meer hebben dan maak je gewoon een nieuw scherm aan. Uiteraard is het vooraf nadenken over de schermontwerpen wel belangrijk.

In deze laatste level, waar je nog taken krijgt, ga je aan de slag om verschillende data bronnen te koppelen aan verschillende componenten. Hierbij maak je gebruik van `Data Context` en `Data Binding`, meer hierover in de volgende taken.

Taak: More Windows

Tijdens de brainstormsessie heb je besloten om de WPF applicatie in te delen in 3 schermen. Het hoofdscherm waarin je de race visueel weergeeft. Een 2e scherm waarin je statistieken van de huidige race toont. En een 3e scherm waarin je statistieken over de deelnemers en competitie toont. Deze schermen moeten te openen zijn via een menu.

Aanpak

- In de solution explorer maak in je WPF project nog 2 extra schermen aan. Dat doe je in het context menu `Add -> Windows (WPF)`.

  Wanneer je de 2 schermen, met passende namen, hebt aangemaakt heb je 2 extra XAML bestanden. Deze hebben beide ook een cs bestand, net zoals bij de MainWindow.

  Gebruikers van je applicatie moeten uiteraard zelf de controle hebben welk scherm ze graag willen openen. Bijna alle applicaties hebben hiervoor een menu bovenaan staan. Een menu toevoegen aan je scherm doe je in 2 stappen.

- Voeg aan je MainWindow het component `DockPanel`. Let wel dat je dit component toevoegt aan je Grid component. Gebruik de XAML editor om te kijken of het component goed geplaatst is.

  Met het `DockPanel` component geef je het `Menu` component de mogelijkheid om een vaste plek te krijgen op je scherm.

- Voeg aan het `DockPanel` component het `Menu` component.
- Controleer de XAML of de opbouw correct is. `DockPanel` is een kind van `Grid` en `Menu` is weer een kind van `DockPanel`. Corrigeer dit wanneer dit niet het geval is.

- Geef het `Menu` component, via het properties scherm, de volgende properties. `Dock` = `Top`, `Height` = `20` en `VerticalAlignment` = `Top`. Deze properties vind je onder het kopje `Layout`.

  Als het goed is heb je nu een grijze balk bovenaan je scherm staan. Dit is de basis van het menu. De volgende stap is om items toe te voegen aan het menu.

- In het properties scherm van het `Menu` open het kopje `Common` en klik op de de knop bij `Items`.

  Er opent zich nu een nieuw venster. Links een lijst, die nu nog leeg is, met alle items in je menu.

- Selecteer in de dropdown `MenuItem` en druk op de `Add` knop. In het popup venster selecteer `MenuItem` onder `System.Windows.Controls`. En druk vervolgens op `Ok`.

  Je ziet in het linker kolom dat een `MenuItem` is toegevoegd. Wanneer je de `MenuItem` selecteert zie je in de rechterkolom de properties die horen bij die `MenuItem`.

- Geef de `MenuItem` bij de property `Header` een passende naam bijvoorbeeld "Menu".

  De properties van het `MenuItem` bevat ook weer de property `Items`. Hiermee kan je het geselecteerde `MenuItem` sub items geven.

- Geef de aangemaakte `MenuItem` 3 extra items. 2 voor het openen van de aangemaakte schermen. 1 om de applicatie helemaal af te sluiten. Geef deze menuitems passende teksten zodat het duidelijk is waarvoor ze dienen.

Ondersteunende informatie

Taak: UI Events

Bij de vorige taak heb je de extra schermen en een menu aangemaakt. In deze taak ga je verdere invulling geven aan het menu waarbij de extra schermen geopend kunnen worden.

Aanpak

Bij grotere WPF applicaties wordt meestal gebruik gemaakt van `Commands`. Bij kleine applicaties, zoals de race simulator, is het prima om te werken met events.

Elk component heeft naast alle properties ook veel events. De lijst aan events kan je vinden door, recht bovenin het properties scherm, op het bliksemicoontje te klikken. Je kan weer terug naar de properties door op het steeksleutelicoontje te klikken.

- Open voor de menuitem, om de applicatie af te sluiten, de lijst met events.
- Vul bij het event `Click` de waarde "MenuItem_Exit_Click" en druk dan op enter.

Nadat je op enter hebt gedrukt opent automatisch `MainWindow.xaml.cs` en wordt er een nieuwe methode aangemaakt. De naam van de methode komt overeen met de waarde dat je bij `Click` hebt ingevuld. Dit is de eventhandler geworden voor de `Click` van de menuitem. Wanneer de gebruiker klikt op dat menuitem wordt deze methode dan uitgevoerd.

- Geef de andere 2 menuitems ook een eigen eventhandler. Zorg voor een duidelijke naam.

Nu je voor de events verschillende eventhandlers hebt aangemaakt kan je de gevraagde functionaliteit programmeren. De eerste functionaliteit is de WPF applicatie af te sluiten wanneer de gebruiker klikt op de exit menuitem.

Zoals al eerder vertelt is WPF multithreaded daardoor volstaat het sluiten van het scherm niet. Dus de code `this.Close();` zal wel het scherm sluiten maar niet de applicatie. Om de applicatie zelf te sluiten moet je `Application.Current.Shutdown();` uitvoeren.

- Implementeer de eventhandler `MenuItem_Exit_Click` zodanig dat de applicatie gesloten wordt.

Wanneer je nu de applicatie start kan je deze weer sluiten via het menu.

De volgende stap is om de 2 andere schermen te openen wanneer de gebruiker op het menuitem klikt. Het openen van een scherm is eenvoudig te doen door een instantie van de bijbehorende klasse te maken. De naam van de klasse is de vinden door de bijbehorende, van de XAML, het cs bestand te openen. Om controle te blijven houden over de extra schermen is het verstandig om de klassen als private variabele aan te maken in de `MainWindow.xaml.cs`.

- Voor elk extra scherm maak in de klasse `MainWindow` een private variabele.

Nu moet je alleen nog de eventhandlers van de andere menuitems implementeren.

- Initialiseer de private variabelen in de eventhandlers.

  Als je alles goed gedaan hebt zal het juiste scherm openen wanneer je in het menu op het menuitem klikt.

Ondersteunende informatie

Wil je inlezen in `Commands` lees dan deze informatie.

Taak: Data Context

WPF is door Microsoft data driven gemaakt. Dit is terug te zien bij de properties van de componenten. Je kan met de hand een waarde invullen maar je kunt de property ook koppelen aan een datapunt. Voordat je aan de slag kan met `Data Binding` zal je eerst een `Data Context` moeten aanmaken en koppelen. In deze taak ga je een klasse schrijven die je vervolgens gaat gebruiken als `Data Context`.

Aanpak

Het project, waar je de klasse gaat aanmaken voor de `Data Context`, mag je zelf beredeneren en bepalen.

- Maak in de, door jouw gekozen project, een normale klasse aan. Deze klasse ga je straks gebruiken als `Data Context` voor het hoofdscherm `MainWindow`

  Je zult ondertussen wel doorhebben dat WPF bijna volledige event driven is. Dat is bij `Data Context` en `Data Binding` niet anders. WPF zal pas een wijziging aan de data pas doorvoeren op het scherm wanneer de er een event afgetrapt is. Binnen C# is er een interface die door WPF gebruikt wordt voor dit doel. Door deze interface te implementeren zorg je ervoor dat WPF straks de data op het scherm aanpast wanneer de data verandert.

- Implementeen in de aangemaakte klasse de interface `INotifyPropertyChanged`.

  Nu moet je alleen nog een manier vinden om het nieuwe event af te vuren wanneer iets aan de data is verandert. Gelukkig heb jij al een event `DriversChanged` die regelmatig afgevuurd wordt. Deze kan je daarom prima gebruiken.

- Maak een eventhandler voor het event `DriversChanged`. Gebruik de constructor om de eventhandler te abonneren op het event.

  De laatste stap is om het `PropertyChanged` event af te vuren. Deze gebruikt `PropertyChangedEventArgs` als event arguments voor het event. De constructor van de `PropertyChangedEventArgs` vraagt de naam van de property dat is gewijzigd. Voor nu is het prima om een lege string mee te geven, hiermee geef je eigenlijk aan dat alle properties aangepast zijn.

- Vuur, vanuit de aangemaakt eventhandler, het `PropertyChanged` event af. Geef een lege string mee bij de constructor van de `PropertyChangedEventArgs`.

  Nu je een klasse hebt aangemaakt die je gaat gebruiken als `Data Context` is de volgende stap om deze klasse te koppelen aan een WPF component. De `Data Context` die je zet wordt gebruikt door het component en alle kinderen van dat component. Vooral bij schermen met veel informatie loont het om eerst goed na te denken waar je welke `Data Context` plaatst. De `Data Context` kan je plaatsen via het properties scherm.

- Open het properties scherm van de `MainWindow` en selecteer in de XAML het `Window` element.
- Onder het kopje `Common` klik op de knop `New` bij de property `DataContext`.

  Na het klikken op het knopje `New` opent zich een nieuw venster met allerlij verschillende namespaces.

- Zoek de namespace waarin jij de klasse hebt aangemaakt. Wanneer je namespace en klasse niet in de lijst kan vinden dan moet je eerst je solution opnieuw builden.
- Selecteer de naam van de klasse en druk op de `Ok` knop.

  Je hebt nu voor alle componenten in de `MainWindow` een `Data Context` aangemaakt. Wanneer je nu de properties opent van een component zal je bij de `Data Context` property een groen vierkantje zien staan. Dit betekent dat het component de waarde, voor die property, overneemt van het bovenliggende component.

Ondersteunende informatie

Informatie over `Data Context` is hier te vinden.

Beschrijving van de `INotifyPropertyChanged` is hier te raadplegen.

Taak: Data Binding

Nu je een data context hebt wil je hier uiteraard gebruik van gaan maken. In deze taak ga je het hoofdscherm verder uitbreiden zodat ook de naam van de huidige track getoond wordt.

Aanpak

De properties bij een component zijn onderwater hetzelfde als normale properties in je code. Wil je dus vanuit je code data gebruiken bij een WPF component moet je de data context klasse een extra property geven. Het type van de property moet wel aansluiten met het type property van het WPF component.

- Pas het WPF scherm `MainWindow` aan en plaats een extra component van het type `Label`. Dit component gaat straks de naam van de huidige track tonen.

  De tekst van een label kan je aanpassen bij de property `Content`. De type van het `Content` property van een `Label` component is een string.

- Geef de data context klasse een extra property, van het type string, voor de naam van de huidige track. Gebruik een `Lambda Expression` om de naam van de huidige track op te halen.

  Nu de data context klasse een property heeft kan je deze gaan gebruiken.

- Open de properties van het `Label` component en zorg dat de property `Content` in beeld is.
- Klik nu op het vierkantje dat bij de property `Content` hoort.
- Kies uit het contextmenu `Create Data Binding`

  Er opent nu een extra venster. In het witte vlak onder `Path` staat al de naam van de data context klasse met daaronder alle properties. Als dit niet het geval is rebuild eerst je solution en herhaal de stappen.

- Selecteer de aangemaakte property en klik op `Ok`.

  Je hebt nu de property `Content` van het `Label` component gekoppeld aan de property in de data context klasse. Door de koppelingen met de events verandert de waarde van deze property mee met de waarde van de property van de data context klasse. Data Bindings kan je herkennen door de gele vierkantjes bij de properties van de componenten.

Ondersteunende informatie

Uitleg over hoe `Lambda Expressions` gebruikt kunnen worden kan je hier vinden.

Meer informatie over `Data Binding` is hier te lezen.

Taak: Showing information

Je hebt alle stappen een keer doorlopen om data uit je applicatie te tonen op het scherm. In deze taak ga je de 2 extra schermen voorzien van verschillende gegevens die je in het vorige level hebt aangemaakt.

Aanpak

Deze taak mag je zelf helemaal vorm geven en invullen. Je mag alle opgedane kennis, inzicht en creativiteit gebruiken om de volgende stap uit te voeren. Gebruik de data, die je hebt aangemaakt in de vorige level, om te tonen. Er wordt verwacht om zoveel mogelijk `LINQ` expressies te gebruiken waar mogelijk.

- Geef het scherm met informatie over de competitie tenminste 2 componenten en bind de data met de data uit je applicatie. Daarbij moet tenminste 1 component een lijst zijn.
- Geef het scherm met informatie over de huidige race tenminste 3 componenten en bind de data met de data uit je applicatie. Daarbij moet tenminste 2 componenten een lijst zijn.

Vergeet bij het afronden van deze taak niet om de unit tests uit te breiden.

Ondersteunende informatie

Uitleg hoe `LINQ` werkt en gebruikt kan worden is hier te vinden