

Hand-out

Programmeren

Periode 2

02. Controls

1. Inleiding

In de vorige les hebben we het gehad over formulieren in het algemeen. In deze hand-out gaan we wat dieper in op controls. Controls zijn de onderdelen die je formulier opmaken, met controls kun je het formulier bedienen. Controls kun je op meerdere manieren opmaken en instellen, in deze hand-out gaan we een aantal van die controls onder de loep nemen en toepassen.

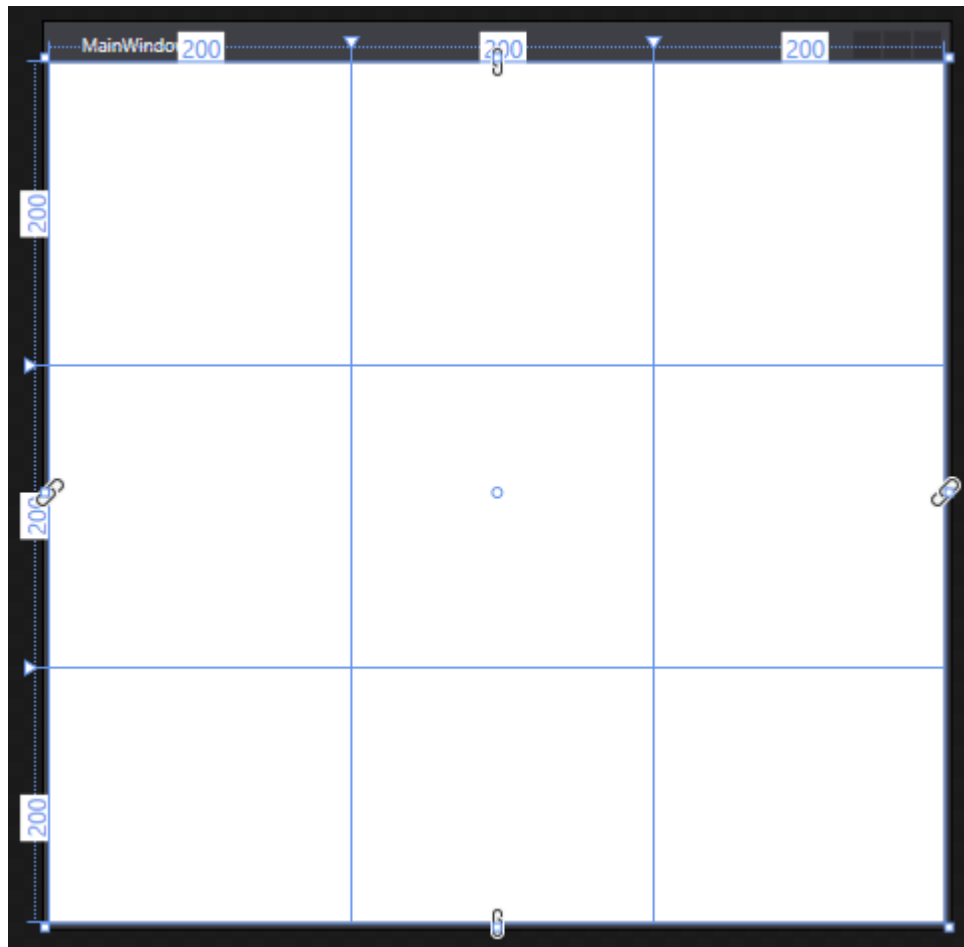
2. Grid

Het Grid control is een van de controls die standaard inbegrepen is wanneer je een nieuw WPF project begint. Je XAML bestand heeft een Grid tag maar daar staan verder nog geen eigenschappen in gedefinieerd.

Een grid is een object dat je kunt indelen in kolommen en rijen, vervolgens kun je andere controls opmaken binnen een specifiek kolom of rij. Om dit te doen maken we gebruik van Column en Row definitions, deze maken we los van het Grid object op als volgt :

```
1  <Window x:Class="P2_02.Controls.MainWindow"
2      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
5      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
6      xmlns:local="clr-namespace:P2_02.Controls"
7      mc:Ignorable="d"
8      Title="MainWindow" Height="600" Width="600">
9      <Grid>
10         <Grid.ColumnDefinitions>
11             <ColumnDefinition Width="200"/>
12             <ColumnDefinition Width="200"/>
13             <ColumnDefinition Width="200"/>
14         </Grid.ColumnDefinitions>
15         <Grid.RowDefinitions>
16             <RowDefinition Height="200"/>
17             <RowDefinition Height="200"/>
18             <RowDefinition Height="200"/>
19         </Grid.RowDefinitions>
20     </Grid>
21 </Window>
22
```

In dit voorbeeld zie je hoe we kolommen en rijen kunnen definiëren binnen een XAML document en hoe we deze een breedte en hoogte kunnen geven. Dit doen we door gebruik te maken van definitions, in dit geval ColumnDefinitions en RowDefinitions. Het resultaat is als volgt :

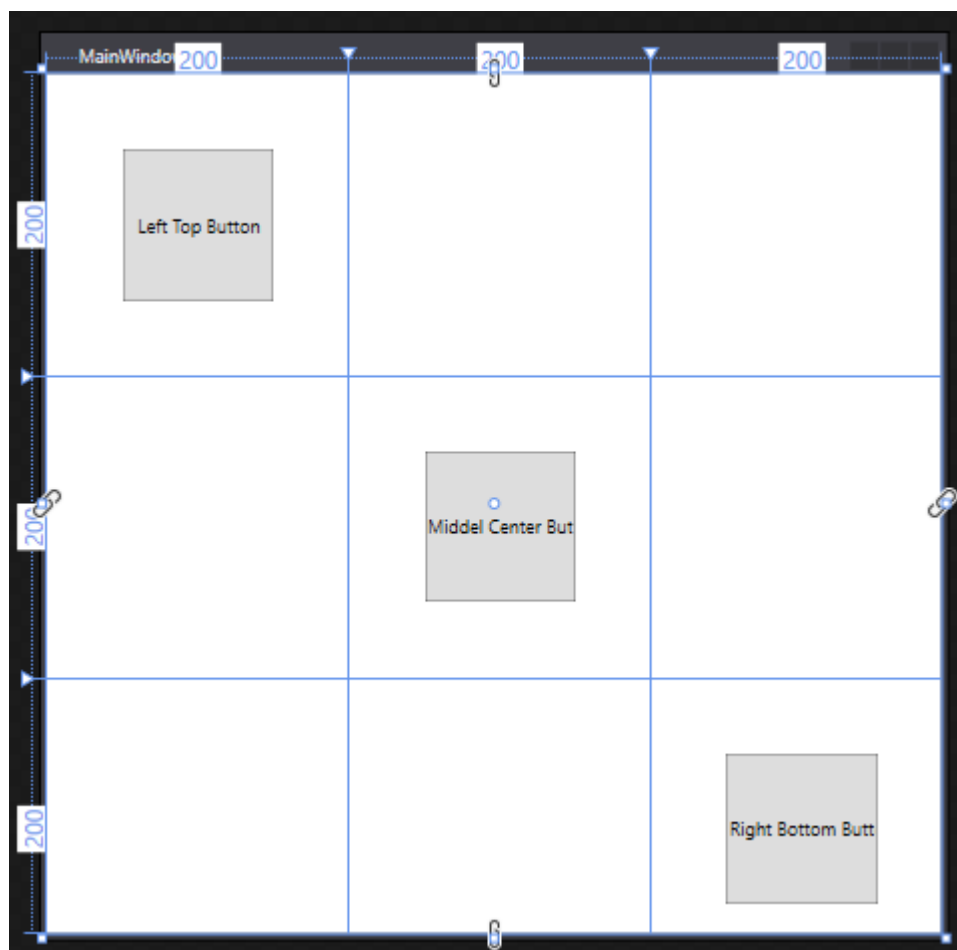


Nu je de definities hebt gemaakt kun je andere controls aan je pagina toevoegen, en deze op een positie binnen het Grid zetten. Dit doe je door gebruik te maken van de Grid.Column en Grid.Row eigenschappen, deze zijn op ieder ander control aanwezig. Grid.Row="1" wil dus zeggen dat je de control op rij 1 gaat plaatsen. Kolommen en rijen beginnen te tellen bij 0. 0,0 is dus links bovenin en 2,2 is dus rechts onder.

Controls zet je op een specifieke plek in het grid als volgt :

```
1 <Window x:Class="P2_02.Controls.MainWindow"
2   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4   xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
5   xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
6   xmlns:local="clr-namespace:P2_02.Controls"
7   mc:Ignorable="d"
8   Title="MainWindow" Height="600" Width="600">
9
10  <Grid>
11    <Grid.ColumnDefinitions>
12      <ColumnDefinition Width="200"/>
13      <ColumnDefinition Width="200"/>
14      <ColumnDefinition Width="200"/>
15    </Grid.ColumnDefinitions>
16    <Grid.RowDefinitions>
17      <RowDefinition Height="200"/>
18      <RowDefinition Height="200"/>
19      <RowDefinition Height="200"/>
20    </Grid.RowDefinitions>
21    <Button Name="btnLeftTop" Margin="50" Content="Left Top Button" Grid.Column="0" Grid.Row="0"/>
22    <Button Name="btnMiddleCenter" Margin="50" Content="Middel Center Button" Grid.Column="1" Grid.Row="1"/>
23    <Button Name="btnRightBottom" Margin="50" Content="Right Bottom Button" Grid.Column="2" Grid.Row="2"/>
24  </Grid>
25 </Window>
26
```

Het resultaat ziet er zo uit :

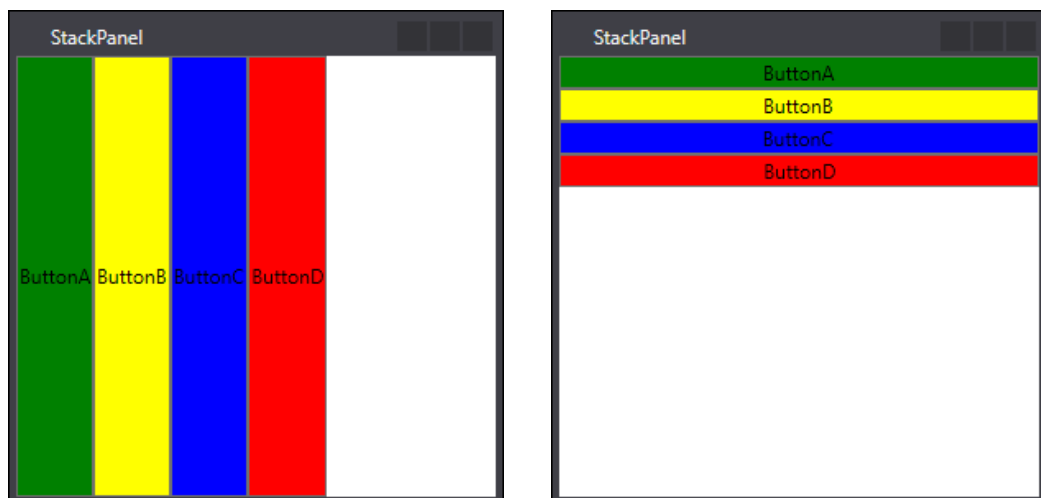


3. Dock, Wrap en StackPanel

Soms wil je controls op een bepaalde manier opgemaakt hebben, maar is een Grid net iets teveel van het goede. Dan is een StackPanel perfect. De StackPanel is een paneel waar je andere controls in kunt plaatsen en er vervolgens voor kunt kiezen om deze horizontaal of verticaal op te maken via de Orientation eigenschap :

```
1 <Window x:Class="P2_02.Controls.StackPanel"
2     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
5     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
6     xmlns:local="clr-namespace:P2_02.Controls"
7     mc:Ignorable="d"
8     Title="StackPanel" Height="300" Width="300">
9     <StackPanel Orientation="Horizontal">
10         <Button Name="btnButtonA" Content="ButtonA" Background="Green"></Button>
11         <Button Name="btnButtonB" Content="ButtonB" Background="Yellow"></Button>
12         <Button Name="btnButtonC" Content="ButtonC" Background="Blue"></Button>
13         <Button Name="btnButtonD" Content="ButtonD" Background="Red"></Button>
14     </StackPanel>
15 </Window>
16
```

Zetten we nu de Orientation van Horizontal naar Vertical hebben we het volgende verschil :



Een StackPanel wordt dus vaak gebruikt om controls netjes onder of langs elkaar te plaatsen, denk hierbij vooral aan radio buttons en checkboxes.

De DockPanel en WrapPanel zijn nagenoeg hetzelfde als de StackPanel, alleen kun je bij een DockPanel een control aan een van de kanten docken en een WrapPanel wrappt bijvoorbeeld tekst om een image heen.

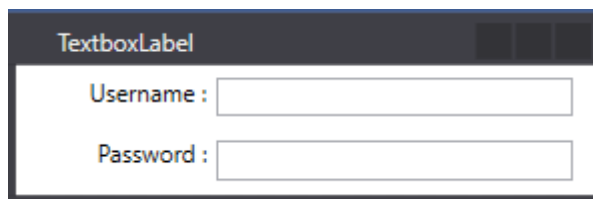
4. Textbox en Label

Textboxes en Labels zijn controls die we gebruiken om tekst in weer te geven. Het verschil tussen een Textbox en een Label is dat we een Textbox gebruiken op het moment dat we tekst in willen voeren of aan willen passen. Een label daar in tegen is puur om tekst weer te geven.

In een formulier zie je vaak dat een label en een tekstvlak bij elkaar horen, het label wordt gebruikt om aan te geven wat voor soort informatie er in de textbox moet komen staan.

Hier een voorbeeld van een login scherm :

```
<Grid>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="100"/>
    <ColumnDefinition Width="auto"/>
  </Grid.ColumnDefinitions>
  <Grid.RowDefinitions>
    <RowDefinition Height="30"/>
    <RowDefinition Height="30"/>
  </Grid.RowDefinitions>
  <Label Name="lblUsername" Content="Username :" Grid.Column="0" Grid.Row="0" HorizontalAlignment="Right"></Label>
  <TextBox Name="tbxUsername" Height="20" Margin="0,2,10,0" Width="180" HorizontalAlignment="Left" Grid.Column="1" Grid.Row="0"/>
  <Label Name="lblPassword" Content="Password :" Grid.Column="0" Grid.Row="1" HorizontalAlignment="Right"></Label>
  <PasswordBox Name="pwbPassword" Height="20" Margin="0,2,10,0" Width="180" HorizontalAlignment="Left" Grid.Column="1" Grid.Row="1"/>
</Grid>
```



Zoals je ziet staan de controls binnen een grid object en op de juiste positie, dit zorgt voor de juiste alignment en een veel mooiere opmaak. Dit is weer beter verkoopbaar aan de klant. De tekst is misschien een beetje klein in het voorbeeld maar de uitwerking is beschikbaar op Github.

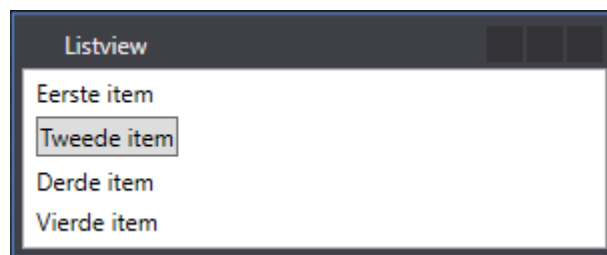
5. ListView

Een ListView is een wat complexere control, dit omdat het items bevat die de list opmaken. Dit noemen we ListItems. Elk ListView control kan worden voorzien van een of meerdere ListViewItems. Een beetje zoals een array.

Om een ListViewItem toe te voegen aan een ListView hebben we meerdere opties. Of we kunnen dit handmatig doen door ze er statisch in te laden, of we kunnen dit programmatisch doen. Als we dit met de hand willen doen kunnen we in de properties window bij de optie Items het knopje er langs aanklikken en krijgen we een nieuw scherm, hier kunnen we items toevoegen. Dit kan je ook doen in de XAML door ListViewItem tags op te maken. En als je dit programmatisch wilt doen gebruik je de Items.Add(); methode van het ListView object.

Het resultaat van een ListView met items ziet er als volgt uit :

```
<ListView Name="lvwMyListView">
  <ListViewItem Content="Eerste item"></ListViewItem>
  <ListViewItem>
    <Button Content="Tweede item"></Button>
  </ListViewItem>
  <ListViewItem Content="Derde item"></ListViewItem>
  <ListViewItem Content="Vierde item"></ListViewItem>
</ListView>
```



Hopelijk valt het je op, want in het tweede listitem hebben we een button geplaatst. Het is dus mogelijk om andere controls binnen een listitem te plaatsen.

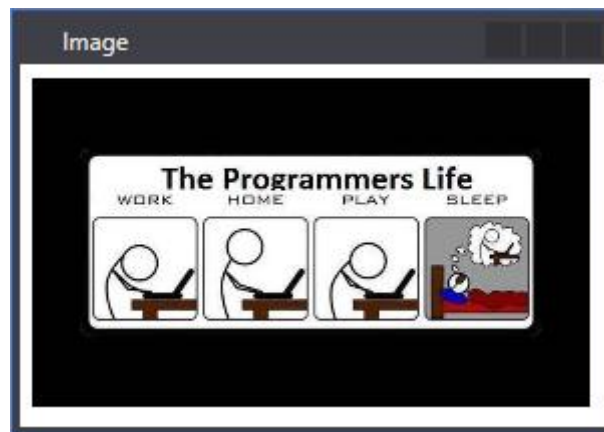
6. Buttons

Buttons of knoppen zijn misschien wel de makkelijkste controls van allemaal en hebben daarom ook niet zoveel uitleg nodig.

7. Images

Images zijn eigenlijk een soort van placeholder voor plaatjes en afbeeldingen. Als je een Image control op je scherm sleept zal je ook zien dat er nog geen afbeelding wordt weergegeven. Je kunt ook een Image control op het scherm plaatsen door een plaatje van je harde schijf rechtsreeks op je formulier te slepen, je zal dan merken dat het plaatje een Image control is geworden.

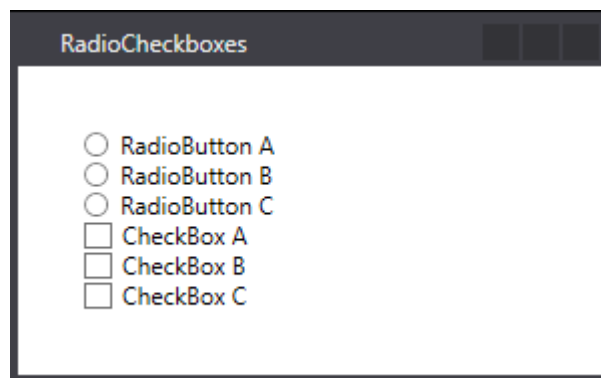
```
<Image Margin="6,7,8,10" Source="joke--comic_sleep-cycle.jpg" Stretch="Fill"/>
```



8. Checkboxes & Radio buttons

Checkboxes en Radiobuttons vallen eigenlijk in de zelfde categorie, omdat ze beide vaak in meervoud worden gebruikt. Het enige verschil tussen checkboxes en radiobuttons is de mogelijkheid om meerdere checkboxes te selecteren en maar een radiobutton. Radiobuttons worden altijd opgemaakt in een groep zodat we weten welke radiobuttons bij elkaar horen.

```
<StackPanel Orientation="Vertical">
  <RadioButton Content="RadioButton A" Name="rbnRadioA" GroupName="radioButtonGroup" ></RadioButton>
  <RadioButton Content="RadioButton B" Name="rbnRadioB" GroupName="radioButtonGroup" ></RadioButton>
  <RadioButton Content="RadioButton C" Name="rbnRadioC" GroupName="radioButtonGroup" ></RadioButton>
  <CheckBox Content="CheckBox A" Name="cbxCheckboxA"></CheckBox>
  <CheckBox Content="CheckBox B" Name="cbxCheckboxB"></CheckBox>
  <CheckBox Content="CheckBox C" Name="cbxCheckboxC"></CheckBox>
</StackPanel>
```



9. Demo

In dit stuk gaan we klassikaal een demo maken. Het is de bedoeling dat je mee kijkt en zo stapsgewijs houvast krijgt op het onderwerp, maak daar waar nodig is aantekeningen in je schrift of in je kladblok. Als je klaar bent ga je zelf aan de slag dus let goed op!

10. Opdracht

In deze opdracht ga je aan de slag met het maken van het eerste deel van Boter Kaas en Eieren. De bedoeling is dat je een Grid gebruikt om het speelveld in op te maken. Dit doe je door gebruik te maken van grid en ColumnDefinitions.

Vervolgens kun je controls gebruiken op de plaats van de cirkels en kruisjes, bijvoorbeeld een Image Control maar dit kan ook een Button of een Label zijn.

Het is op dit moment nog niet de bedoeling dat je het spel helemaal werkend krijgt maar wel het speelveld opmaakt en er voor zorgt dat wanneer er ergens geklikt wordt er een kruisje op het scherm komt te staan. De volgende les gaan we verder aan deze opdracht en maken we de logica achter het spel op.

