|  |
| --- |
|  |
| Inhoudsopgave |
|  |
|  |

**GROEP 2 LAYOUT**

May 29, 2017

Opgesteld door: Janneke van Veltom

Inhoudsopgave

Bestelapplicatie

Inhoud

[Introductie 2](#_Toc483835080)

[Lijst lay-out 3](#_Toc483835081)

[Scroll view 4](#_Toc483835082)

[Grid lay-out 5](#_Toc483835083)

[Doorsturen naar een nieuw form 6](#_Toc483835084)

[Bronnen 7](#_Toc483835085)

# Introductie

De bestelapplicatie dient mogelijkheden te hebben om uit een reeks van producten te kiezen en die te bestellen. Tijdens deze workshop komt het onderdeel lay-outs in Xamarin naar voren. Wij gaan u vertellen over het gebruik van lijst lay-out, het gebruik van scroll view, het gebruik van een grid lay-out en het doorsturen naar nieuwe form. In deze workshop worden enkele voorbeelden gegeven over het gebruik van deze onderdelen in Xamarin en welke voorbeelden u beter niet kunt gebruiken.

# Lijst lay-out

ListView is een belangrijk UI component van Android applicaties, die overal gebruikt wordt voor 'short lists' vanuit de menu opties tot aan 'long lists' van bijvoorbeeld contacten of internetbrowser favorieten.

ListView biedt een simpele manier om een 'scroll list' van rijen te presenteren die geformateerd met een ingebouwde stijl kunnen worden.

Ook heeft ListView een Adapter nodig om data in te kunnen laden die in rijen worden weergeven.

## Items toevoegen aan lay-out

De makkelijkste manier om items toe te voegen, doe je doormiddel van een array gevuld met strings te vullen.

**Hier een voorbeeld:**

var listView = new ListView();

listView.ItemsSource = new string[]{

"mono",

"monodroid",

"monotouch",

"monorail",

"monodevelop",

"monotone",

"monopoly",

"monomodal",

"mononucleosis"

};

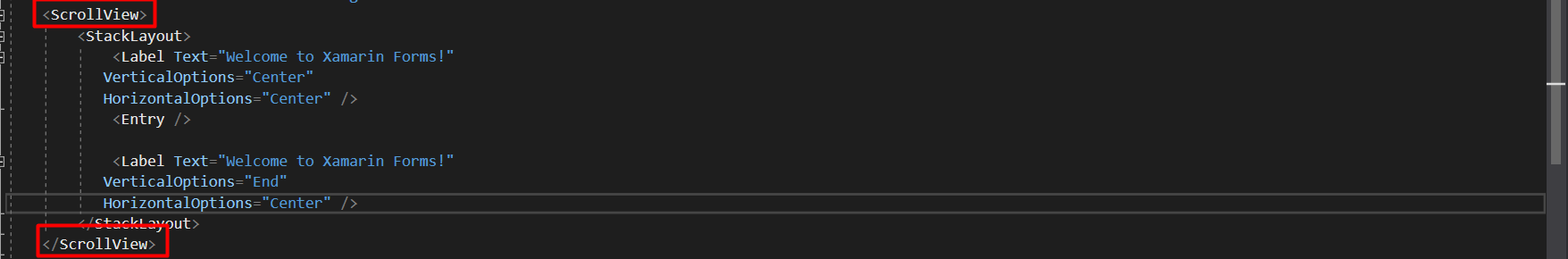
Hier wordt een nieuw ListView aangemaakt, waarbij de ItemsSource bestaat uit een string.

De waarden worden hierbij ingevuld.

# Scroll view

De ScrollView is een geweldige layout-optie één die je bijna op elke pagina wil gebruiken. Het zorgt ervoor dat in principe je pagina zo lang mogelijk kan maken.

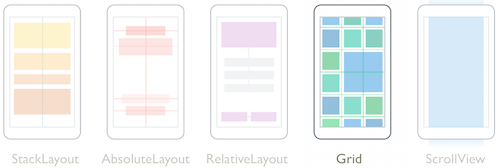
Om een ScrollView te implementeren heb je maar 2 simpele regels code nodig!



Simple! Het resultaat zorgt ervoor dat je naar onder en boven kan swipen!

# Grid lay-out

## Introductie

Grid lay-out geeft iemand de mogelijkheid om de pagina weer te geven met rijen en kolommen. Rijen en kolommen kunnen zelf worden ingedeeld met hun eigen vaste grote. De grid lay-out zou niet de verwarrend moeten zijn, gezien de traditionele tabellen. Verder is het geen concept van rijen, kolommen of cel formateren. Anders dan HTML tabellen, is Grid uitsluitend bedoeld om inhoud uit te leggen.

Grid kan worden gebruikt voor het weergeven van je pagina in een raster. Dit kan handig zijn in de volgende omstandigheden:

* Ordenen van knoppen.
* Organiseren van knoppen / keuzes in een raster, zoals bijvoorbeeld de iOS- of Android startschermen.
* Arrangeren van de weergaven zodat alles op gelijke grootte is, zoals bijvoorbeeld in sommige werkbalken.

In tegenstelling tot traditionele tabellen, maakt het bij Grid niet uit hoeveel en in welke grote de rijen en kolommen inhouden. In plaats daarvan heeft Grid RowDefinitions en ColumnDefinitions collecties. Deze bevatten definities van hoeveel rijen en kolommen worden uitgelegd. Uitzicht wordt toegevoegd aan Grid met opgegeven rij- en kolomindices, die bepalen welke rij en kolom een ​​weergave moet worden geplaatst.

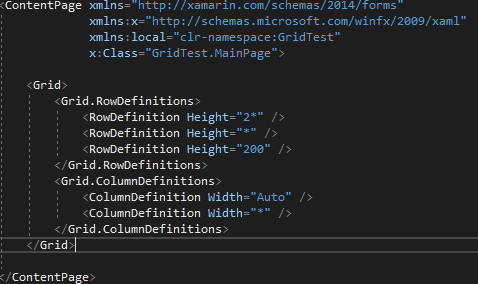
Alle code bij deze workshop zal in XAML zijn.

## Rijen en kolommen toevoegen

Rij- en kolominformatie wordt opgeslagen in de eigenschappen van RowDefinitions & ColumnDefinitions van Grid, die respectievelijk zijn van respectievelijk verzamelingen van RowDefinition en ColumnDefinition-objecten. RowDefinition heeft een enkele eigenschap, hoogte en ColumnDefinition heeft een enkele eigenschap, breedte. De opties voor hoogte en breedte zijn als volgt:

* Auto  
  Maakt automatisch de maten om de rij of kolom aan te passen. Word gegeven als GridUnitType.Auto in C# of als Auto in XAML.
* Proportional  
  Maakt de maten van de rijen en kolommen als een deel van de resterende ruimte. Gegeven als een waarde en GridUnitType.Star in C# en als #\* in XAmL, waarbij # de gewenste warde is. Een rij of kolom worden opgegeven met \*, en zal er voor zorgen dat de beschikbare ruimte wordt gevuld.
* Absolute  
  Maakt de maten van de rijen en kolommen met een specifieke hoogte en breedte waardes. Word gegeven als GridUnitType.Absolute in C# en als # in XAML, waarbij # uw gewenste waarde is.

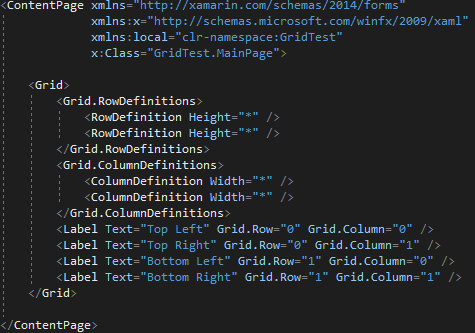
Ga ervan uit dat een app drie rijen en twee kolommen nodig heeft. De onderste rij moet precies 200px lang zijn en de bovenste rij moet tweemaal zo lang zijn als de middelste rij. De linker kolom moet breed genoeg zijn om de inhoud te passen en de rechter kolom moet de resterende ruimte invullen.



## Weergaven plaatsen in Grid

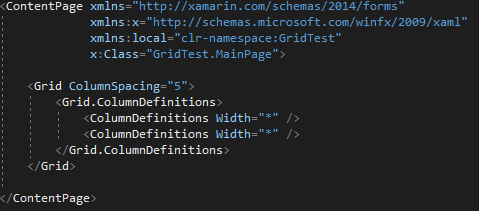
Om weergaven in een raster te plaatsen, moet u deze als kinderen toevoegen aan het raster en vervolgens opgeven welke rij en kolom ze behoren.

In XAML, gebruik Grid.Row en Grid.Column op elke afzonderlijke weergave om plaatsing te specificeren. Merk op dat Grid.Row en Grid.Column locatie specificeren op basis van de nul gebaseerde lijsten van rijen en kolommen. Dit betekent dat in een 4x4-rooster de linker bovenste cel (0,0) is en de rechter onderkant (3,3) is.



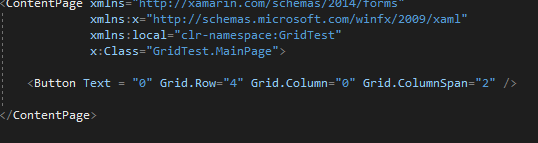
## Ruimte

Grid heeft eigenschappen om de afstand tussen rijen en kolommen te regelen. De volgende eigenschappen zijn beschikbaar om het raster aan te passen: ColumnSpacing - de hoeveelheid ruimte tussen kolommen. RowSpacing - de hoeveelheid ruimte tussen rijen. De volgende XAML specificeert een raster met twee kolommen, één rij en 5 px tussenruimte tussen kolommen.



## Gebruik van spans

Vaak bij het werken met een raster is er een element dat meer dan één rij of kolom moet bezetten. In XAML kan een button als volgt worden toegevoegd:



# Doorsturen form

`App.Navigation.PushAsync(new MyCustomContentPage());`

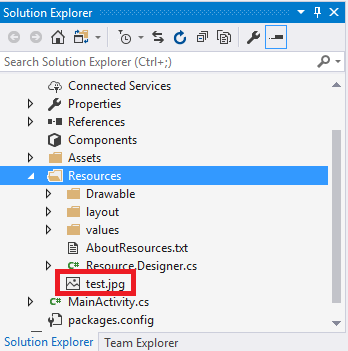
Met dit stukje code kan je een naar een nieuwe page door in de pushasync een nieuwe class aan te maken met de page die je wilt openen. Bijvoorbeeld

als je een class Secondpage hebt kun je App.Navigation.PushAsync(new Secondpage()); gebruiken. De code maakt dan een nieuwe page aan en switched hiernaartoe. Ook kan je new page veranderen in een al bestaande page.

# Toevoegen van afbeelding aan project

Om een afbeelding toe te voegen aan je project zijn er verschillende manieren beschikbaar.

* Voor iOS plaats je je gewenste afbeelding in de “Resources” map die je kunt vinden in de Solution Explorer.
* Voor Android plaats je de afbeelding in de “Resources/Drawable” map.
* Voor Windows Phone plaats je de afbeelding in de root van je project.
* Voor Windows/UWP plaats je ook de afbeelding in de root van het project.



Om de afbeelding in te laden gebruik je de volgende code:

var testImage = new Image { Aspect = Aspect.AspectFit };

testImage.Source = ImageSource.FromFile("test.jpg");

Om verschillende afbeeldingen voor de verschillende operating systems kun je de volgende code gebruiken:

testImage.Source = Device.OnPlatform(

iOS: ImageSource.FromFile("Images/testt.jpg"),

Android: ImageSource.FromFile("test.jpg"),

WinPhone: ImageSource.FromFile("Images/test.png"));

Hierbij word steeds dezelfde afbeelding ingeladen op de verschillende systemen.

# Bronnen

<https://developer.xamarin.com/guides/xamarin-forms/user-interface/layouts/grid/>

https://developer.xamarin.com/guides/xamarin-forms/user-interface/images/