Wouter Devinck

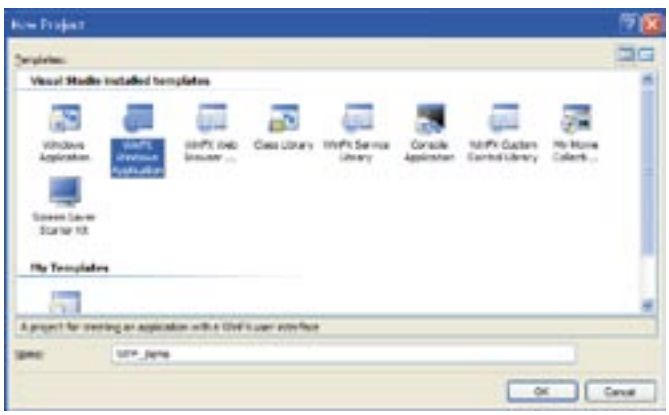
WPF in de praktijk

BETERE EN MOOIERE INTERFACES MET WINDOWS PRESENTATION FOUNDATION

Samen met Windows Vista staat de derde versie van het .NET Framework (WinFX) voor de deur. Deze uitbreiding op .NET 2.0 zal enkele nieuwe technologieën bevatten. Eén van die nieuwe technologieën is WPF (Windows Presentation Foundation, codenaam Avalon). In dit artikel ga ik ervan uit dat je al een zekere kennis hebt van wat WPF en XAML zijn. In dit artikel maak ik gebruik van een voorbeeld om je warm te maken voor deze nieuwe technologie.

Om te beginnen starten we een nieuwe 'WinFX Windows Application' in Visual Studio 2005, hiervoor moet Cider geïnstalleerd zijn; zie afbeelding 1. Alles wat hier staat, is gebaseerd op CTP-versies van de software. Deze software kan dus nog veel wijzigingen doormaken voor het pakket in de winkelrekken ligt. Vergeet niet het project een duidelijke naam te geven. Ik raad je aan dit project ook meteen op te slaan, omdat de CTP-producten af en toe vastlopen. Als je net als ik Visual Basic Express of C# Express gebruikt, zul je al snel merken dat de designer, een onderdeel van Cider dat sterk doet denken aan de Windows Forms Designer, ontbreekt. Of deze designer in de definitieve versie van Cider wel zal werken in de Express-edities van Visual Studio zal de toekomst moeten uitwijzen.
In dit voorbeeld gebruiken we de Microsoft Expression Interactive Designer niet alleen voor de opmaak en grafische effecten, maar ook voor de schikking van de controls. In grote projecten is het echter de bedoeling om ontwikkelaars en designers samen aan één project te laten werken. Ontwikkelaars kunnen dan Visual Studio 2005 gebruiken voor de algemene lay-out van het formulier en het programmeerwerk, terwijl de designer hetzelfde project opent in een programma van de Expression-familie.

Als je gebruikmaakt van Visual Basic Express zul je merken dat de Solution Explorer slechts twee bestanden bevat: App.xaml en Window1.xaml. In C# Express zie je diezelfde twee bestanden, maar met een plusje ervoor. Als je op dit plusje klikt, verschijnt het zogenaamde codebehind-bestand van het window (Window1.xaml.vb) of van de applicatie (App.xaml.vb). In dit voorbeeld gaan we verder werken met Visual Basic Express. Om de plustekens



Afbeelding 1: Het aanmaken van een WinFX-project App.xaml

zichtbaar te maken moet je op het knopje 'Show All Files' klikken, bovenaan in de Solution Explorer. Om te tonen hoe je in XAML het object kiest dat als eerste start, gaan we een nieuw window toevoegen. Dit doe je op de gebruikelijke manier om een formulier toe te voegen, kies echter wel voor 'WinFX Window' in plaats van 'Windows Form'. We noemen dit window 'winCalc.xaml'. Na het toevoegen van dit formulier kunnen we Window1.xaml verwijderen. Om er voor te zorgen dat het nieuwe formulier ook als eerste start, moet een wijziging in App.xaml gemaakt worden; zie codevoorbeeld 1. In dit bestand kun je ook resources plaatsen die beschikbaar moeten zijn voor de hele applicatie.

Microsoft Expression Interactive Designer

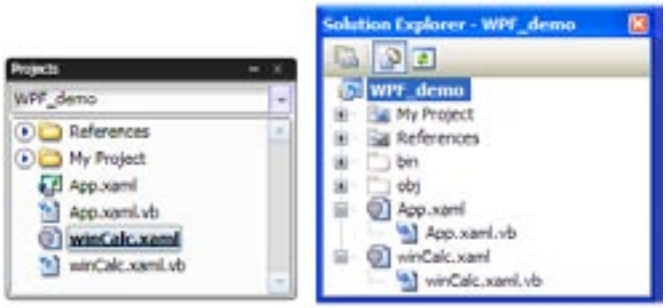
Zoals gezegd gaan we het formulier volledig ontwerpen in Microsoft Expression Interactive Designer. Na het openen van het project in dit gloednieuwe programma (wel eerst opslaan met Visual Studio) zien we in het *Projects*-venster exact dezelfde bestanden verschijnen als in de Solution Explorer (zie afbeelding 2). Een interessant gegeven is dat de interface van Interactive Designer ontworpen is met Interactive Designer zelf. Een mooi voorbeeld hiervan is de 'Workspace zoom' rechts bovenaan in het programma. Met dit schuifbalkje kun je in en uitzoomen op de interface. Het zoomen zonder kwaliteitsverlies is te danken aan vector-afbeeldingen en het feit dat de controls zich ten opzichte van elkaar schikken. Dit is te danken aan enkele krachtige controls zoals *grids* en *WrapPanels* die beschikbaar zijn in WPF.

Als je het bestand winCalc.xaml opent, zie je dat er al een grid op het formulier staat. Een grid is een krachtige control, omdat hij in rijen en kolomen kan worden verdeeld die dynamisch meebewegen als je het formulier vergroot of verkleint. In dit voorbeeld gaan we daar geen gebruik van maken, omdat we het formulier een vaste grootte gaan geven. Om aan te geven dat het window een vaste grootte heeft, moet de eigenschap *ResizeMode*

```
<Application x:Class="App"
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   StartupUri="winCalc.xaml">
   <Application.Resources>

   </Application.Resources>
</Application>
```

Codevoorbeeld 1. App.xaml



Afbeelding 2: Het Projects-venster en de Solution Explorer

Afbeelding 3: Een eerste lay-out

op *NoResize* of *CanMinimize* gezet worden. De eigenschappen van een object kun je, net zoals in de Windows Forms Designer van Visual Studio, aanpassen in het *Properties*-venster. Om een object eenvoudig te selecteren, kun je gebruikmaken van het linkerdeel van het *Timeline*-venster, daar zie je namelijk een boomstructuur van alle objecten.

Een eerste lay-out

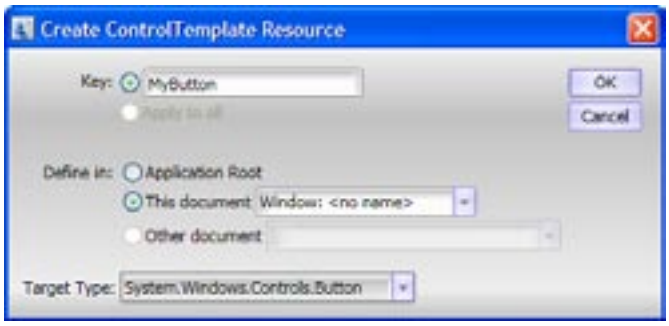
Het is hoog tijd dat we beginnen met het schikken van het formulier. In Interactive Designer staan de controls niet in een toolbox, maar in het *Library*-venster. Op afbeelding 3 zie je hoe de schikking van ons formulier in elkaar zit; bovenaan een tekstvak en een knopje en eronder een *WrapPanel* waarop later de knoppen van de rekenmachine zullen komen. Een WrapPanel is een soort panel dat zijn geneste controls automatisch schikt. Voor het positioneren van de controls is het *Layout*-venster erg handig. In dit venster kun je de afstand van de control tot aan de randen van het grid bepalen (let op: dit is enkel geldig voor de randen waaraan het control vastgemaakt is met een anker). In tabel 1 zie je de eigenschappen van de basiselementen van ons voorbeeld.

Templates en styles

In WPF kun je op twee verschillende manieren controls opmaken, met styles en met templates. In een style kun je voor een reeks controls zaken zoals kleurenschema's, lijndiktes en kleurverlopen vastleggen. Templates gaan echter nog een stapje verder, een template kun je naar wens vormen, effecten en animaties toevoegen, laten reageren op muisacties, enzovoort. Om een template voor de buttons te maken, tekenen we eerst een button van 50 bij 50 op het WrapPanel, let hierbij wel op dat het WrapPanel geselecteerd

Object	Eigenschap	Waarde
TextBox Ankers: Links, boven en rechts	Height	25px
	Top & Left	5px
	Right	60px
Button Ankers: Boven en rechts	Height	25px
	Width	50px
	Top & Right	5px
WrapPanel Ankers: Links, boven, rechts en onder	Top	32,5px
	Left	2,5px
	Bottom & Right	0px

Tabel 1. De eigenschappen van enkele objecten in ons voorbeeld



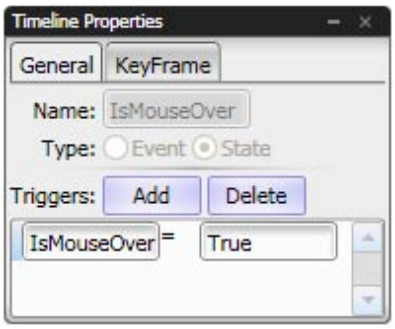
Afbeelding 4: Een template aanmaken

moet zijn. Een lege template maken we aan door rechts te klikken op de button en vervolgens in het submenu 'Edit Template' op 'Create Empty Template' te klikken. In het venster dat verschijnt (afbeelding 4), kun je een naam aan de template geven en kiezen waar de XAML-code moet komen. Aangezien we de template enkel binnen winCalc nodig hebben, kiezen we voor 'This Document'. Nadat we op OK geklikt hebben, zien we een lege button, let ook op de boomstructuur in het Timeline-venster.

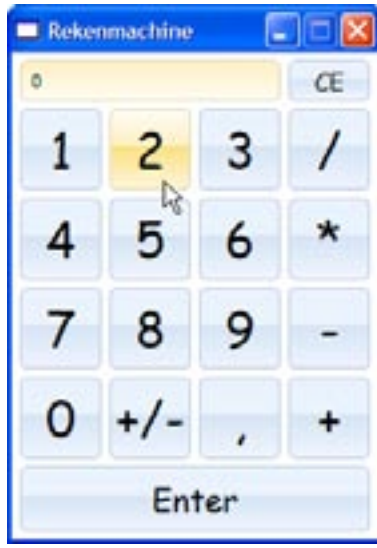
Nu moeten we een grid aanmaken die de volledige button vult, dubbelklik hiervoor op grid in het *Library*-venster. Op deze grid tekenen we een rechthoek uit het Tools-venster. Vanaf hier kun je de button naar wens opmaken door er vormen op te plaatsen en kleuren te wijzigen in het Appearance-venster, lijndiktes te wijzigen in het *Stroke*-venster en eigenschappen te wijzigen in het Properties-venster. Ook hier is het *Layout*-venster erg handig. Ik raad aan de nodige ankers te plaatsen, zodat de template ook gebruikt kan worden voor kleinere en grotere knoppen. Tot slot moet je niet vergeten een *ContentPresenter* op de button te plaatsen, dit object is noodzakelijk om de inhoud van de button (tekst of een afbeelding) weer te geven. Als je op F5 drukt (net zoals in Visual Studio) zie je de button in actie. Nu ja, in actie is veel gezegd, want door een template te maken, zijn de effecten bij muisacties ook weg. Om de button op de muis te laten reageren, kun je zogenaamde states instellen. Een state maak je aan met het knopje 'Create New State' in het *Timeline*-venster. Verder moet je in het *Timeline Properties*-venster (afbeelding 5) een trigger toevoegen. Voorbeelden van triggers zijn 'IsMouseOver' en 'IsPressed'. Je kunt ook verschillende triggers aan een state toevoegen. Om aan te geven hoe een template er in een bepaalde state uit moet zien, klik je de state gewoon aan in het *Timeline*-venster en maak je enkele wijzigingen in de opmaak van de objecten.

Verdere afwerking

Nu we een knopje hebben dat er uit ziet zoals we het altijd al wilden, kunnen we het gewoon kopiëren en de WrapPanel zorgt voor de schikking. Vergeet niet de marges van de buttons in te stellen, want deze worden niet mee gekopieerd; als je de waardes van tabel 1 hebt gebruikt, ziet een marge 2,5px er goed uit. We kunnen ook het kleine knopje naast het tekstvak aan de template koppelen door het te selecteren, er rechts op te klikken en voor 'Apply Resource' te kiezen in het submenu 'Edit Template'. Ook



Afbeelding 5: Een trigger toevoegen aan een state



Afbeelding 6: Het afgewerkte programma

voor een tekstvakje kun je eigen templates maken op een soortgelijke manier als voor knopjes. Dit zal in de toekomst ook mogelijk zijn voor zowat alle controls. Nu kan het soms nog foutmeldingen opleveren in Interactive Designer. Als alle knopjes er op staan, kunnen we in Visual Studio de nodige code toevoegen. De voorbeeldcode bij dit artikel is een werkende rekenmachine (zie afbeelding 6), maar het belangrijkste aan de voorbeeldcode is uiteraard dat de interface in XAML geschreven is. De voorbeeldcode kun je downloaden van www.microsoft.nl/netmagazine15.

Tot besluit

WPF is een krachtig middel om nog betere en mooiere interfaces voor onze programma's te maken. Het schrijven van een interface in XAML-code is best te doen, maar Microsoft zorgt ervoor dat een hele suite nieuwe tools klaar zal staan om de ontwikkelaar het leven weer wat makkelijker te maken.
Alle code in dit artikel is geschreven voor en met de June 2006 CTP (Community Technology Preview) van de volgende sofware:

- Microsoft .NET Framework 3.0 (beter bekend als WinFX)
- Microsoft Expression Interactive Designer
- Microsoft Visual Studio Code Name 'Orcas' - Development Tools for .NET Framework 3.0 (bevat Cider).

Wouter Devinck is een 16-jarige student die graag op de hoogte blijft van de nieuwste technologieën en een zwak heeft voor het maken van oogverblindende interfaces. Met vragen en opmerkingen kun je terecht op wouter.devinck@gmail.com of zie www.wouterdevinck.net.

Referenties

http://www.microsoft.com/products/expression
http://www.microsoft.com/downloads
http://www.microsoft.com/products/expression/en/demos.mspx
http://msdn.microsoft.com/msdntv/episode.aspx?xml=episodes/en/20051020CiderMB/manifest.xml