Hans Verbeeck
is werkzaam bij Microsoft

De revival van Windows-toepassingen

Bouwen met de krachtige Windows Forms package uit het .NET Framework

Als ontwikkelaar wil je je intellectueel ei leggen waarmee je tegemoet komt aan de eisen van je werkgever of klant. Als het even kan, dan wil je Windows-applicaties bouwen met 'coole' technologie en 'coole' tools. Het goede nieuws is dat dit inderdaad mogelijk is, met Microsoft .NET.

Microsoft .NET is er nu al een tijdje en er is al een Service Pack 1 voor het .NET Framework beschikbaar. In dit artikel vind je de belangrijkste redenen voor Visual Basic-ontwikkelaars om over te stappen op .NET. Wij beginnen echter met het bouwen van een Windows-applicatie. Doordat in de beginfase van Microsoft .NET zo sterk de nadruk werd gelegd op XML Web Services en ASP.NET, leek het alsof er eigenlijks niets nieuws inzat voor Windows development. Niets is minder waar!

In dit artikel bouwen we een Windows-applicatie en tonen we:

- hoe je menu's aanmaakt;
- dat je geen resize code meer hoeft te schrijven;
- hoe je een icoon voor je applicatie in de system tray plaatst met een context menu;
- hoe je items toevoegt aan de eventlog.

Ervaren Visual Basic-programmeurs zeggen misschien 'big deal, dat kan ik nu ook al' en dat klopt. Je kan de meeste dingen al met VB6 maar vaak moest je daarvoor zeer ingewikkelde code schrijven, bijvoorbeeld Windows API calls maken. In Visual Studio is het vaak eenvoudiger door enkele drag & drop-operaties en het schrijven van een eenvoudige lijncode. Let's begin…

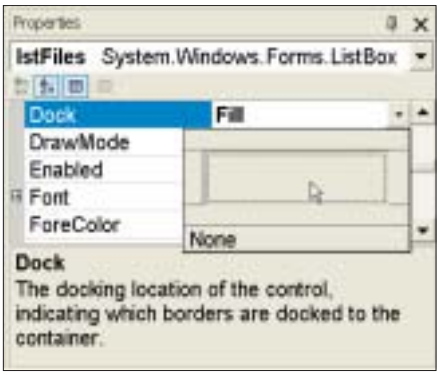
Afbeelding 1. De Windows Form

Een Windows form zonder resize code

Wanneer je een Windows-applicatie bouwt, open je Visual Studio.NET, je selecteert New Project en dan Windows Application. Dit creëert dan een nieuw project met daarin een default form 'Form1' genaamd. Wanneer je deze naam wijzigt, moet je ook het Startup Object van het project veranderen. Het voorbeeld dat we hier uitwerken is een applicatie die in de gaten houdt wanneer er in een bepaalde folder nieuwe bestanden worden geplaatst (zie afbeelding 1). Als je wilt dat de listbox control de hele Form inneemt, ongeacht hoe de gebruiker de Form 'resized', dan kan je hiervoor de Dock property gebruiken zoals in afbeelding 2. Een andere zeer aardige property die je kan instellen is de Anchor property. Hiermee kan je de afstanden tussen een control en de rand van zijn container (bijvoorbeeld een Form of een panel control) vastleggen. Deze twee properties zorgen ervoor dat je praktisch geen resize code meer hoeft te schrijven.

10 redenen om over te stappen naar VB .NET

1. VB .NET is volledig objectgeoriënteerd. VB .NET ondersteunt multi-threading en structured exception handling.
2. Visual Studio .NET is nu een complete ontwikkelingsomgeving met onder andere de Server Explorer, Task Window en een visuele designer voor componenten.
3. Met Visual Studio.NET kun je sneller Windows-toepassingen maken.
4. De productiviteit neemt tevens toe door de drag & drop-functionaliteit voor componenten.
5. ASP.NET is een flinke stap voorwaarts. ASP.NET brengt het event-driven programmeermodel van Visual Basic 6 naar web development.
6. Je kan XML Web Services aanbieden en consumeren zonder met de complexiteit van XML geconfronteerd te worden.
7. Wil je ook applicaties voor Pocket PC's bouwen, dan kan je nu hetzelfde programmeermodel gebruiken dan voor Windows- of webapplicaties.
8. ADO.NET is een data access-technologie, afgestemd op gedistribueerde toepassingen en volledig op XML gebaseerd.
9. De .NET-componenten die je bouwt hoeven niet meer geregistreerd te worden in de registry waardoor deployment sterk wordt vereenvoudigd en de 'DLL hell' achter onze rug ligt.
10. Er is een brede community die je in je programmeeractiviteiten bijstaat en ondersteunt.



Afbeelding 2. De Dock-eigenschap

Indien je de Form in de rechter-onder hoek wil plaatsen (ongeacht de resolutie) van het scherm, dan kan je dit bereiken met de eenvoudige lijncode in afbeelding 3.

Folders monitoren

Bestanden monitoren is een van die dingen die de rijkdom van het .NET Framework goed aantonen. In een Visual Basic 6-applicatie moet je een aantal Windows API's aanroepen of een control kopen die API's voor je inpakte. Nu zit deze functionaliteit gewoonweg in het .NET Framework, om precies te zijn in de System.IO namespace. Helemaal bovenaan in de module van de form voegen we Imports System.IO toe. Hierdoor kunnen we de classes in deze namespace gebruiken zonder telkens de hele namespace te moeten typen. In afbeelding 4 staat de voledige code hiervoor.
We creëren een nieuw FileSystemWatcher object en geven aan welke folder we willen monitoren. Dan specificeren we dat we geïnformeerd willen worden wanneer er een bestandsnaam verandert. Vervolgens geven we aan dat we geïnteresseerd zijn in alle bestanden met een extensie. De magie zit hem in het AddHandler keyword waarmee we een eventhandler verbinden aan onze watcher. In deze eventhandler (OnChanged) voegen we de FullPath toe aan de ListBox control. Tot slot zorgen we ervoor dat de watcher effectief begint te monitoren. Heel recht toe recht aan. Vind je deze code nog teveel, dan kan je de FileSystemWatcher control gebruiken die je vindt in de Toolbox onder de Components tab. Je vindt daar tevens de Eventlog control. Je sleept deze control op de form (frmVBDino) en je ziet dat Visual Studio automatisch onder je form een paneeltje toevoegt waarop alle controls komen die de eindgebruikers van de applicatie ook niet zullen zien (afbeelding 5).
Via de Property Browser stel je de Log property van dit object in op 'Application' - zodat de gegevens toegevoegd worden aan de 'Application' log - en de Source property stel je in op bijvoorbeeld de naam van de applicatie. Voor de MachineName property specifieer je '.' om aan te geven dat de applicatie entries in de locale eventlog moet toevoegen. In je code kan je dit EventLog1 object gaan aanspreken. De lijn in afbeelding 6 kan je aan de Form_Load eventhandler toevoegen om de start van de applicatie te loggen.

Menu's

De manier waarop je menu's toevoegt is ook veel intuïtiever geworden. Je sleept een MainMenu control op je Form en zoals je ziet in afbeelding 5, kun je dan gewoon typen wat je wil tonen in de menu's. In het properties-venster kan je dan aan de MenuItems een naam geven.

Het NotifyIcon en het Contextmenu

Je hebt vandaag de dag heel wat applicaties die, wanneer je ze opstart, een icoontje toevoegen naast je klok in je taakbalk. Dit wordt de System Tray genoemd. Meestal kun je op dat icoontje rechtsklikken om de applicatie te sluiten of zichtbaar te maken. Een populair voorbeeld hiervan is de Messenger. Dit is iets waarmee ik in VB6 serieus wat tijd mee verloren had en ik was dan ook zeer aangenaam verrast toen ik zag

```
Me.SetDesktopBounds(SystemInformation.WorkingArea.Right - Me.Width,
SystemInformation.WorkingArea.Bottom - Me.Height, Me.Width,
Me.Height)
```

Afbeelding 3.

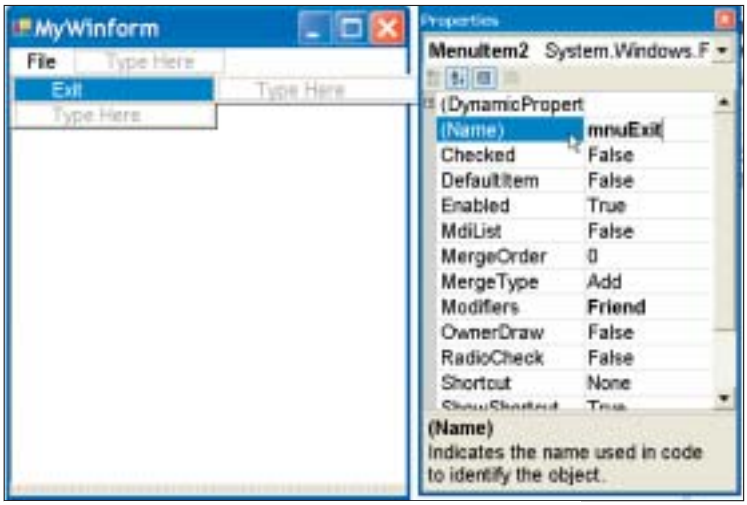
```
Private Sub frmMain_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Dim watcher As New FileSystemWatcher()
    watcher.Path = "c:\Temp\"
    watcher.NotifyFilter = (NotifyFilters.FileName)

    watcher.Filter = "*.*"

    AddHandler watcher.Created, AddressOf OnFileCreated

    watcher.EnableRaisingEvents = True
End Sub

Public Sub OnFileCreated(ByVal source As Object, ByVal e As FileSys-
temEventArgs)
    lstFiles.Items.Add(e.FullPath)
End Sub
```

Afbeelding 4.

Afbeelding 5. Controls die de eindgebruikers niet zullen zien

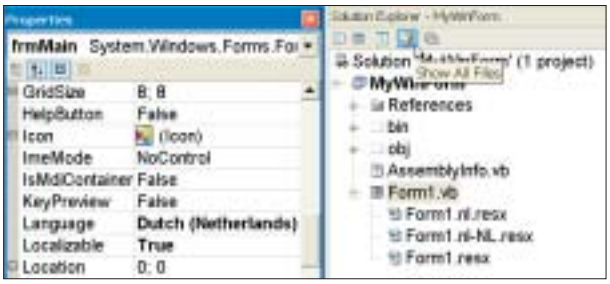
```
'wanneer de applicatie wordt gestart, gaan we dat loggen in de eventlog
EventLog1.WriteEntry("Application Start",EventLogEntryType.Information)
```

Afbeelding 6.



Afbeelding 7. Menu's toevoegen gebeurt heel intuïtief

Afbeelding 8. De Nederlandse vlag als NotifyIcon met ContextMenu



Afbeelding 10. Internationalisatie

dat de toepassing van system tray in VS.NET zonder enige moeite mogelijk is. Je sleept uit je Toolbox – bij de Windows Forms controls – een NotifyIcon op je form die tevens bij je EventLog1 object geplaatst wordt (zie afbeelding 5). Je duidt dit NotifyIcon1 object aan en stelt de Icon property in. Hier geef je aan welk icoontje je in de system tray wil tonen. Als je een popup-menu wil tonen wanneer de gebruiker rechtsklikt op dat icoontje, dan moet je eerst een ContextMenu control aan je form toevoegen. Je sleept die control op je form. Naast je NotifyIcon1 control komt nu een ContextMenu1 control. Als je deze aanduidt, verschijnt dit menu op je form waar eerst het andere menu stond. Dit menu wordt nooit op die plaats getoond aan de gebruiker. Hij staat daar alleen om de menu-items toe te voegen en namen te geven. Ik geef toe, dat is een beetje raar.

In afbeelding 8 zie je rechts onderaan dat er een Restore en Exit MenuItem is toegevoegd aan het ContextMenu. Wanneer je dubbelklikt op het 'Restore' MenuItem kom je in de eventhandler voor het click event daarvan. 'Me.WindowState = FormWindowState.Maximized' zorgt ervoor dat je applicatie terugkomt op normale grootte. Dit kan zeer nuttig zijn wanneer je de ShowInTaskBar property van de form op False hebt ingesteld. In deze applicatie hebben we een Exit MenuItem in de MainMenu control én in de ContextMenu control. De code in afbeelding 9 toont hoe je één eventhandler voor events van verschillende controls kan hebben.

Na het Handles keyword kun je opsommen welke events er door deze procedure worden afgehandeld.

Internationalisatie

Ook de manier waarop je je Windows-applicatie meertalig kan maken, toont de grote kracht van dit Windows Forms-model. De eerste stap is de Localizable property van de Form op True instellen. Dan stel je de Language property van de Form in op de taal die je wilt aanbieden, bijvoorbeeld Dutch (Netherlands). Visual Studio.NET maakt op dat moment resource files aan die vertalingen van bepaalde strings, of de andere resources zoals images, gaan bevatten. Je kan deze files zien door in de Solution Explorer de Show All Files knop in te drukken (zie afbeelding 10). Nu de Language property op de gewenste taal is ingesteld, kun je de Text property van de menu's gaan wijzigen of zelfs images gaan wijzigen. De originele waardes blijven bewaard in het originele resources-bestand en de nieuwe zullen in het resources bestand voor het Nederlands komen. Wanneer je deze bestanden met de rest van de applicatie 'deployed', dan zal de runtime de resources laden afhankelijk van de Culture van de thread waarop de applicatie draait. Het standaardgedrag is dat de taal uit de regional settings wordt getoond of de Default-taal indien er geen passend resource-bestand kan gevonden worden.

.NET Connected Windows-applicaties

De Windows Forms package is een krachtig onderdeel van het .NET Framework. Meer Windows API's zijn netjes verpakt in .NET Framework classes en Windows Forms kunnen nu als client gebruikt worden voor webapplicaties via XML Web Services. Verder zijn eindelijk de deployment-problemen verdwenen die we kennen van VB6/COM-toepassingen. Windows-applicaties die gebouwd zijn op het .NET Framework, noemen we daarom graag .NET Connected Windows-applicaties. Ben je klaar voor deze revival van Windows-toepassingen?

```
Private Sub Exit(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles mnuExit.Click, ctxmnuExit.Click
        EventLog1.WriteEntry("Application Ended")
        Application.Exit()
End Sub
```

Afbeelding 9.

Nuttige internetadressen

- http://www.windowsforms.com/ (web site van het Windows Forms product team)
- http://www.syncfusion.com/faq/winforms/ (frequently asked questions about Windows Forms)
- http://msdn.microsoft.com/vbasic/downloads/samples/visualbasic.asp (101 code samples, 24 about Windows Forms)