André Obelink is technical manager bij AcouSoft

Nieuwe taalelementen van Visual Basic 2005

KLEINE STAPPEN GROTE STAPPEN

Visual Basic heeft de afgelopen jaren een ware evolutie doorgemaakt. De grootste stap was de overstap van Visual Basic 6 naar Visual Basic .NET. De eerste .NET-versie, Visual Basic .NET 2002, werd al snel opgevolgd door Visual Basic .NET 2003. Tussen deze twee versie waren de verschillen marginaal. Visual Basic 2005 biedt daarentegen wel weer een groot aantal veranderingen. Deze uitbreidingen en verbeteringen leiden ertoe dat je nog sneller en beter kunt programmeren. In dit artikel zet ik een aantal van de nieuwe taalelementen voor je op een rij.

Eén van de handigste en meest tijdbesparende uitbreidingen van Microsoft Visual Basic 2005 is ongetwijfeld de namespace *My*. Kort samengevat bevat deze namespace een collectie van klassen, waarvan de meeste niets meer zijn dan een vereenvoudigde schil om bestaande klassen uit het hele .NET Framework 2.0. De code die nodig is om bepaalde zaken voor elkaar te krijgen, wordt hierdoor aanzienlijk gereduceerd. Hierbij kun je bijvoorbeeld denken aan het benaderen van bestanden, informatie over je applicatie of het benaderen van het register van Windows. In aanvulling op al die classwrappers beschikt de namespace My ook over enkele dynamische objecten en eigenschappen, die worden gemaakt wanneer je bepaalde klassen of bestanden aan je project toevoegt. Elk formulier is bijvoorbeeld in je project te benaderen als een afzonderlijke eigenschap op het object *My.Forms*. Interessanter is het wanneer deze techniek wordt toegepast op de configuratie-instellingen of resources in je project. Deze zijn met behulp van Intellisense gemakkelijk te gebruiken. Je kunt in je programma een instelling opnemen binnen de tab *Settings*. Bijvoorbeeld de naam van een bepaalde server:

Deze instelling of setting bevindt zich in het bestand *applicatienaam.exe.config*. Het is vervolgens in je code erg eenvoudig uit te lezen en, afhankelijk van de scope, ook eenvoudig in te stellen:

```
lblServerNaam.Text = My.Settings.NaamServer
```

In nieuwsgroepen en fora wordt helaas nog wel eens denigrerend over de namespace *My* gesproken. Dit is onterecht. De enige opmerking die je zou kunnen maken, is het feit dat jij je als ontwikkelaar niet moet blindstaren op de functionaliteit in deze namespace. Wanneer een bepaalde mogelijkheid zich niet in *My* bevindt, wil dat niet zeggen dat het helemaal niet mogelijk zou zijn. Vaak kun je het dan wel bereiken met de oorspronkelijke en ingewikkelder klassen, waarvoor *My* een vereenvoudigde laag is. Zolang je dit niet uit het oog verliest, moet je gewoon gebruikmaken van deze mogelijkheden.

Afbeelding 1. Het toevoegen van configuratiedata is erg eenvoudig.

De namespace *My* bestaat uit de volgende objecten:

- Application
- Computer
- User
- Resources
- Settings
- Forms
- Webservices

Zoals reeds gemeld kent *My* een aantal dynamische typen. Deze typen of objecten zijn: *Resources, Settings, Forms* en *Webservices*. *My* kent ook drie statische objecten. In *My.Application* vind je informatie over de applicatie zelf, zoals versie, commandline-argumenten of werkdirectory. Je kunt het bijvoorbeeld ook gebruiken om vast te stellen of de applicatie via ClickOnce verspreid is; zie codevoorbeeld 1.
My.Computer geeft toegang tot de eigenschappen van de computer, de hardware en eventueel aangesloten randapparatuur. My.Computer kent properties zoals Mouse, Audio, Network, Registry en Clipboard; zie codevoorbeeld 2.

```
Dim strBedrijfsnaam As String
Dim blnClickOnce As Boolean

With My.Application
   strBedrijfsnaam =.Info.CompanyName
   blnClickOnce = .IsNetworkDeployed
End With
```

Codevoorbeeld 1.

```
Dim intVrijGeheugen As Integer
Dim strComputerNaam As String

With My.Computer
   .Clipboard.SetText("www.VBcentral.nl")

   intVrijGeheugen = .Info.AvailablePhysicalMemory
   strComputerNaam = .Name
End With
```

Codevoorbeeld 2.

```
Dim blnIsBeheerder As Boolean
Dim strGebruikersNaam As String

With My.User
   blnIsBeheerder = .IsInRole("Administrators")
   strGebruikersNaam = .Name
End With
```
Codevoorbeeld 3.

Het object *My* geeft, zoals de naam al doet vermoeden, toegang tot eigenschappen van de huidige gebruiker. Daarnaast kun je door het implementeren van de interfaces *IPrincipal* en *Iidentity* ook je eigen authenticatiemethoden maken. Codevoorbeeld 3 toont twee voorbeelden van het gebruik van *My.User*:

Ik raad je aan deze namespace aan een nader onderzoek te onderwerpen. Je zult snel ontdekken dat het een groot aantal handige mogelijkheden bevat, die je direct in je applicaties kunt toepassen.
Nieuwe datatypen
Met ondermeer de introductie van de zogenaamde 'unsigned' datatypen UShort, UInteger en ULong voldoet Visual Basic 2005 nu ook volledig aan het Common Type System (CTS). De datatypen, die geen negatieve waarden kunnen bevatten, kunnen hierdoor wel een groter getal bevatten.

```
Dim shtGetal As UShort
' Bereik: 0 to 65535
shtGetal = 65535
```

Operator overloading

De mogelijkheid operators te overloaden, geeft je de kans het gedrag van intrinsieke operatoren zoals +, -, =, <, > en <> op je *eigen objecten* te beïnvloeden. Het concept van het toepassen van bijvoorbeeld de operator + op twee objecten is natuurlijk niet nieuw; zie codevoorbeeld 4.

In bovenstaand voorbeeld worden twee strings samengevoegd. Dit is ook een logische actie wanneer we + toepassen op strings. Ook het toepassen van de operator > geeft een logisch resultaat. Namelijk een vergelijking op alfabetische volgorde. Maar wat moet de compiler doen wanneer we de operator > toepassen op twee Leerling-objecten?

```
If objLeerlingJan > objLeerlingPiet Then
   ...
End If
```

Zegt de vergelijking wat over de lengte van de leerlingen? Of over de leeftijd? Of misschien over het gemiddelde rapportcijfer? Jij, als ontwikkelaar, kunt aangeven op welke wijze er vergeleken moet worden. Bekijk de definitie van de klasse *Leerling* in codevoorbeeld 5 maar eens.

Je ziet dat operator is toegevoegd. Deze moet gemarkeerd zijn als *Shared*. Ook moet ten minste een van de argumenten van het desbetreffende type zijn; in ons geval dus *Leerling*. Vervolgens kun je elke vergelijking maken, zolang deze maar een True of een False retourneert. Overigens werkt deze code alleen maar wanneer ook de tegenovergestelde operator, dus <, is geïmplementeerd.
Een speciaal type operator is de *Ctype*-operator. *CType* wordt gebruikt om een object van het ene datatype naar een ander data-

```
Dim strVoornaam As String = "André"
Dim strAchteraam As String = "Obelink"
Dim strVolledigeNaam As String = ""

strVolledigeNaam = strVoornaam + " " + strAchteraam
```
Codevoorbeeld 4.

```
Private mintLeeftijd As Integer
   Private mintLengte As Integer
   ...

   Public Property Lengte() As Integer
      Get
         Return mintLengte
      End Get
      Set(ByVal Value As Integer)
         mintLengte = Value
      End Set
   End Property

   Public Property Leeftijd() As Integer
      Get
         Return mintLeeftijd
      End Get
      Set(ByVal Value As Integer)
         mintLeeftijd = Value
      End Set
   End Property
   ...

   Public Shared Operator >(ByVal objLinks As Leerling, _
              ByVal objRechts As Leerling) As Boolean

      ' Code toevoegen die test of een van de objecten
      ' Nothing is een juiste waarde retourneert

      If objLinks.Leeftijd > objRechts.Leeftijd Then
         Return True
      Else
         Return False
      End If

   End Operator
   ...
End Class
```
Codevoorbeeld 5.

type te converteren. Je hebt zelf volledig invloed op hoe jouw objecten naar de diverse datatypen worden geconverteerd. Enkele voorbeelden zijn te zien in codevoorbeeld 6.

De operator IsNot

De oplettende lezer zal het niet ontgaan zijn dat er in codevoorbeeld 6 gebruik is gemaakt van nog een nieuwe operator: de *IsNot*-operator. In vorige versies van Visual Basic moest je, om te bepalen of een object *Nothing* was, de *Not*-operator toepassen op het *Is*-keyword. Je kent vast en zeker wel de volgende type instructie:

```
Public Shared Widening Operator CType(ByVal objL As Leerling) As String
   Return objL.Voornaam & " " & objL.Achternaam
End Operator

Public Shared Widening Operator CType(ByVal objL As Leerling) As Date
   If objL IsNot Nothing Then
      Return objL.GeboorteDatum
   End If
End Operator

Public Shared Widening Operator CType(ByVal objL As Leerling) As Integer
   If objL IsNot Nothing Then
      Return objL.Leeftijd
   End If
End Operator
```
Codevoorbeeld 6.

```
If Not objLeerling Is Nothing Then
```

Zowel technisch als logisch gezien, valt er niets op dit statement aan te merken. De vraagstelling en vergelijking kloppen precies, maar het kan erg lastig zijn om tijdens het lezen van de code, de logica te snappen. Het volgende statement leest naar mijn mening veel prettiger:

```
If objLeerling IsNot Nothing Then
```

Het keyword Global

Het definiëren van je eigen namespaces is niet nieuw in Visual Basic 2005. Dit concept bestaat al sinds de eerste versie van het .NET-platform. Visual Basic 2005 kent wel een nieuw keyword met betrekking tot namespaces, namelijk *Global*. Misschien ben jezelf al wel eens tegen het probleem aangelopen dat je in je eigen namespaces gebruik wilt maken van namen die ook binnen de rootnamespace bestaan. Een voorbeeld zou kunnen zijn: *Bedrijfsnaam.System*. Het is in deze klasse *System* niet mogelijk een variabele te declareren van het type *System.Integer*. *Integer* maakt immers geen onderdeel uit van die klasse waarin het zich bevindt. Dit probleem kun je oplossen door de variabele vooraf te laten gaan door het keyword Global.

```
Dim intGetal As Global.System.Integer
```

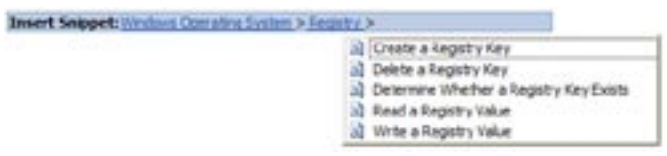
Het keyword Using

De Garbage Collector neemt je bij het opruimen en vrijgeven van geheugen veel werk uit handen. Deze 'vuilniswagen' komt eens in de zoveel tijd voorbij en ruimt dan alle objecten op die niet meer gebruikt worden en opgeruimd mogen worden. Een mooi principe, maar je hebt eigenlijk geen invloed op het moment wanneer dat gebeurt. De GC roept de zogenaamde *Finalizer* aan, maar of en zo ja wanneer het geheugen en eventuele handles worden vrijgegeven, is onbekend. In veel gevallen is dit geen enkel probleem, maar met name voor objecten die bijvoorbeeld een bestand open houden, kan het lastig zijn. Het is deze gevallen gebruikelijk dat het desbetreffende object de interface *IDisposable* implementeert, zodat je door het aanroepen van de methode *Dispose*, direct het object kunt opruimen. Het nieuwe keyword *Using* (en *End Using*) maakt gebruik van de besproken interface; zie codevoorbeeld 7.

Het keyword Continue

Een veel voorkomende programmeeropdracht is het vroegtijdig verlaten van een bepaalde lusstructuur zoals een *For… Next* of *Do…. Loop*. In vorige versies van Visual Basic was er geen eenvoudige manier om naar de volgende teller van een loop te springen.

Afbeelding 2. Gebruik van het keyword Global.



Afbeelding 3. Code snippets toevoegen

```
Using objSr As New StreamReader("c:\MijnApplicatie.log")

   Dim strRegel As String = ""

   Do While Not strRegel Is Nothing
      strRegel = objSr.ReadLine
      Console.WriteLine(strRegel)
   Loop

   objSr.Close()

End Using
```

Codevoorbeeld 7.

Gelukkig is daar nu, met de komst van het keyword *Continue*, verandering in gekomen. Hiermee kun je direct naar de volgende lusdoorgang gaan; zie codevoorbeeld 8.

Hoewel je in bovenstaand codevoorbeeld 8 de instructie om het Getal te tonen ook binnen de *If… Else* zou kunnen zetten, geeft het wel aan dat deze code niet wordt uitgevoerd bij alle veelvouden van vijf. Je kunt deze instructie ook gebruiken in geneste lusstructuren.

Generics

Generics vormen een extensie op het .NET Framework 2.0 en zijn dus pas beschikbaar vanaf Visual Basic 2005. Met Generics kun je klassen en methoden zo definiëren dat het gegevenstype van tevoren niet altijd bekend hoeft te zijn. Het .NET Framework bevat een groot aantal Generics. Generics is het hulpmiddel om je code 'strongly typed' te maken. Stel dat je een collectie van Leerlingobjecten wilt maken; een collectie waar je enkel en alleen objecten van het type Leerling kunt toevoegen. In vorige versies van Visual Basic moest je heel veel code schrijven om dit mogelijk te maken. Nu kan het met slechts één regel code.

```
Dim LeerlingenList As New System.
Collections.Generic.List(Of Leerling)
```

Je kunt ook zelf generieke methoden maken. Bekijk maar eens de onderstaande 'typesafe'- implementatie van de aloude VB6-functie *IIf()*. Zie codevoorbeeld 9 voor het gebruik van IIF().

Wanneer je de volgende code gebruikt om de functie aan te roepen én je hebt Option Strict aan staan, dan zal het bovenste statement niet gecompileerd kunnen worden. Omdat strResult1 gedeclareerd is als string, accepteert het niet de waarde 0. Echt heel cool!

```
Dim strResult1 As String = IIf((4 > 5), "Juist", 0)
Dim strResult2 As String = IIf((4 > 5), "Juist", "Onjuist")
```

Buiten de klassen in *System.Collections.Generic* is het maken van generieke code best lastig in de vingers te krijgen. Mijn advies is toch hier zeker aandacht aan te besteden, je code wordt niet alleen veel krachtiger, maar ook minder foutgevoelig.

```
For intTeller As Integer = 0 To 50

   If intTeller Mod 5 = 0 Then
      Console.WriteLine("Waarde: " & intTeller)
   Else
      Continue For
   End If

   Console.WriteLine("Getal: " & (intTeller / 5).ToString)

Next
```

Codevoorbeeld 8.

```
Public Function IIf(Of T)(ByVal Expressie As Boolean, _
            ByVal WaardeTrue As T, _
            ByVal WaardeFalse As T) As T
   If Expressie Then
      Return WaardeTrue
   Else
      Return WaardeFalse
   End If

End Function
```

Codevoorbeeld 9.

Code snippets

Geen nieuw taalelement, maar wel erg handig zijn code snippets. Visual Basic 2005 wordt geleverd met zo'n 500 code snippets. Code snippets zijn kleine stukjes code die je eenvoudig aan je code kunt toevoegen. Deze code snippets zijn gecategoriseerd per taak en zijn in de codewindow eenvoudig op te roepen onder je rechtermuisknop of door de sneltoets '? + Tab'. Je kunt trouwens zelf ook je eigen code snippets toevoegen met behulp van de Snippets Editor, een tooltje dat je gratis kunt downloaden. Code snippets zijn niet alleen erg praktisch in gebruik, maar je leert ook hoe een ander een bepaalde taak zou oplossen.

Wat niet aan bod is gekomen

Niet alleen de taal Visual Basic 2005 is sterk uitgebreid, maar met name het aantal klassen in het .NET Framework 2.0. Hierdoor is het eigenlijk onmogelijk alle vernieuwingen in dit artikel de revue te laten passeren. Enkele van deze onderwerpen die interessant zijn om nog eens goed te bestuderen zijn: *Code Access Security* (CAS), het uitrollen van je software met behulp van ClickOnce en het toevoegen van XML-commentaren. Ook op het gebied van het bouwen van webapplicaties valt er veel nieuws te beleven. Concepten als Membership, Profiles en Roles worden nu native ondersteund. Andere uitbreidingen zijn bijvoorbeeld de mogelijkheden van Master Pages en Themes. Last but not least…. Edit & Continue is gelukkig ook weer terug.
Buiten het feit dat de IDE van Visual Studio 2005 sterk verbeterd is, de debug-mogelijkheden sterk uitgebreid zijn, zijn alleen de nieuwe taalelementen en -constructen al een reden om met deze versie van Visual Basic aan de slag te gaan.

André Obelink (MCSD) is een Microsoft MVP voor Visual Basic. In het dagelijks leven is hij technical manager bij AcouSoft, een bedrijf dat audiologische software ontwikkelt. In zijn vrije tijd runt hij samen met Willem van den Broek de populaire portal VBcentral.nl. Hij schrijft al jaren voor zowel nationale als internationale tijdschriften en is auteur van het boek: Visual Basic 2005 – De Basis (Pearson Education, ISBN: 9043012890)

Referenties

http://www.VBcentral.nl
http://msdn2.microsoft.com/en-us/vbasic/aa463382.aspx
Visual Basic http://msdn2.microsoft.com/en-us/vbasic/default.aspx
Visual Basic video's: http://msdn2.microsoft.com/en-us/vbasic/ms789071.aspx
Visual Basic blogs: http://msdn2.microsoft.com/en-us/vbasic/ms789067.aspx