André Obelink

Wat is nieuw in Visual Basic 14?

De release van Visual Studio 2015 staat om de hoek. Naast significante uitbreidingen in de IDE, ASP.NET en cloudtoepassingen, is het .NET Compiler Platform misschien wel het meest noemenswaardig.

Onder de codenaam 'Roslyn' heeft Microsoft de afgelopen jaren hard gewerkt om de compilers van C# en Visual Basic .NET volledig te herschrijven. Beide (inmiddels) open-source compilers zijn geschreven in taal zelf. Dus een Visual Basic .NET applicatie wordt gecompileerd door een compiler die ook in Visual Basic .NET is geschreven. Een betere manier van 'dog fooding' kun je je haast niet voorstellen. Maar naast al dit geweld, hebben gelukkig ook de talen C# en Visual Basic .NET de nodige aandacht gekregen. In dit artikel zet ik eerst de vernieuwingen in de taal Visual Basic voor jou op een rijtje. Daarna laat ik je kort kennismaken met enkele vernieuwingen in de ontwikkelomgeving. Een saillant detail is overigens dat de versie van Visual Basic opgehoogd is van versie 12 naar 14. Men slaat dus versie 13 over. Volgens ingewijden is de belangrijkste reden om aan te sluiten bij de interne versie van Visual Studio.

Microsoft heeft zich in deze release in het bijzonder ingezet op een tweetal aandachtsgebieden:

1. Bestaande codepatronen met eenvoudiger code mogelijk te maken;
2. Oplossen van een aantal irritante issues die eigenlijk al in eerdere versies van Visual Basic .NET aangepakt hadden moeten worden.

De ?. operator

De ?. operator, door sommigen ook wel de Elvis operator genoemd, maakt het mogelijk om veel eenvoudiger te controleren of een variabele Nothing is, alvorens daar iets mee te gaan doen. Als ik naar mijn eigen code kijk, staat het vol met dit soort testen. In onderstaande code kun je zien hoe je je eigen code kunt vereenvoudigen met behulp van de ?. operator.

```
Dim gebruikersGroep As New GebruikersGroep With {.Naam =
"SDN"}

' Visual Basic 12
If gebruikersGroep IsNot Nothing AndAlso gebruikersGroep.Naam
= "VBcentral" Then
  Console.WriteLine("Visual Basic rocks!")
End If

' Visual Basic 14
If gebruikersGroep?.Naam = "VBcentral" Then
  Console.WriteLine("Visual Basic rocks!")
End If
```

Je kun de ?. operator ook gebruiken bij aanroepen naar een methode op een object:

```
Dim gebruikersGroepen As New List(Of GebruikersGroep)
Dim dotNed = gebruikersGroepen?.SingleOrDefault(Function(x)
x.Naam = "dotNed")
```

Daarnaast kun je deze operator ook gebruiken in een statement die genest is:

```
Dim sdn As New GebruikersGroep
If sdn?.Magazines?.Count > 125 Then
  Console.WriteLine("Dat is een mooi aantal!")
End If
```

Het is interessant te weten wat de .Count property in bovenstaande retourneert wanneer één van de twee parent-objecten daadwerkelijk Nothing is. Je krijgt dan een Nullable-type van de variabele die je uitvraagt terug. In dit geval dus een Nullable(Of Integer). Je code kan er dan zo uit zien.

```
Dim aantalMagazines = sdn?.Magazines?.Count
If aantalMagazines.HasValue Then
  '....
End If
```

String Interpolation

Met de nieuwe String Interpolation functionaliteit kun je eenvoudiger Strings opbouwen en voorzien van variabelen. Het is eigenlijk een kortere variant van de welbekende String.Format() functie. Door de tekenreeks vooraf te laten gaan van een $-teken, kan men de variabelen direct injecteren. Met name in scenario's waarin veel variabelen geïnjecteerd moeten worden, leidt dit tot veel leesbaarder code. Oude placeholders zoals {0} of {1} zeggen op zich natuurlijk niet veel.

```
' Visual Basic 12
Dim gebruikersGroep = "WAZUG NL
Dim result = String.Format("{0} is de Windows Azure User
Group", gebruikersGroep)

' Visual Basic 14
Dim gebruikersGroep = "DIWUG"
Dim result = $"{gebruikersGroep} is de Dutch Information
Worker User Group"
```

Multi-line String Literals

Het opbouwen van een string over meerdere regels is eigenlijk altijd al echt vervelend werk. Het aan elkaar plakken van regels met een vbCrLf of Environment.NewLine() geeft lelijke code, terwijl het naar nu blijkt een stuk fraaier kan. Ik moet wel eerlijk zeggen dat in de nieuwe variant de tekst niet zo mooi uitgelijnd wordt, omdat ik het prettiger zou vinden dat in code het woord 'Wilt' onder het woord 'De' geplaatst zou worden. Dat zou extra spaties opleveren. Als regelscheidingsteken(s) worden de karakters gebruikt die standaard ook in je codebestand gebruikt wordt, in dit geval dus *vbCrLf* (Zie Afbeelding 1).

```
' Visual Basic 12
Dim vraag = "De wijzigingen zijn nog niet opgeslagen." &
            Environment.NewLine() & Environment.NewLine() &
            "Wilt u dat nu doen?"

' Visual Basic 14
Dim vraag = "De wijzigingen zijn nog niet opgeslagen.

Wilt u dat nu doen?"
```

Fig.1: Multi-line Strings

Hiermee samenhangend kan men nu commentaarregels ook over meerdere regels tekst verspreiden. Dit kan met name handig zijn bij het opbouwen van ingewikkelde LINQ statements. Dit eenvoudige voorbeeld illustreert wel de goed de mogelijkheden.

```
Dim vbCentral = From g In gebruikersGroepen ' De 'From'
                Where g.Naam = "VBCentral"  ' De Where-clause
                Select g                    ' In VB is deze
regel niet nodig
```

Read-Only Auto Properties

Een functionaliteit waar ik zelf al een tijdje op zit te wachten is de mogelijkheid om naast de gewone Auto Properties, ook *Read-Only Auto Properties* te kunnen gebruiken. Ze werken zoals je dat zou verwachten. Je kunt ze van waarden voorzien in de declaratie zelf of eventueel in de constructor van de klasse waarin ze gedefinieerd zijn.

```
Public Class Magazine

  Public ReadOnly Property Nummer As Integer = 125
  Public ReadOnly Property Jaar As Integer

  Public Sub New()
    Jaar = 2015
  End Sub

End Class
```

De NameOf operator

De NameOf operator maakt het eenvoudiger te verwijzen naar de naam van een variabele in je code. Deze operator wordt niet geëvalueerd tijdens het uitvoeren, maar het is een compile-time constante. De NameOf operator helpt je voorkomen dat code, die normaal gesproken tussen dubbele quotes staan, synchroon blijven lopen met je andere code. Deze functionaliteit is vooral handig in classes waarbij je bijvoorbeeld de INotifyPropertyChanged interface implementeert. Bij het hernoemen van de eigenschap Naam, zal de constructor van PropertyChangedEventArgs altijd goed zijn. Een ander voorbeeld waarbij deze nieuwe feature goed van pas kan komen is bij het opgooien van een InvalidArgumentException() waarbij je naam van het ongeldige argument wilt meegeven.

```
Private _naam As String
Property Naam As String
  Get
    Return _naam
  End Get
  Set
    _naam = Value
    ' Visual Basic 12
    RaiseEvent PropertyChanged(Me, New PropertyChangedEven-
tArgs("Naam"))
    ' Visual Basic 14
    RaiseEvent PropertyChanged(Me, New PropertyChangedEven-
tArgs(NameOf(Naam)))
  End Set
End Property
```

Wat kun je nog meer verwachten in Visual Studio 2015?

Zoals je hebt kunnen lezen is de taal Visual Basic .NET wederom uitgebreid met een aantal zeer handige nieuwe mogelijkheden. Ook al zijn de wijzigingen misschien niet zo schokkend als in voorgaande versies, waarin bijvoorbeeld Async and Await werden aangekondigd, ze zijn toch meer dan de moeite waard. Door uitbreidingen binnen de IDE van Visual Studio 2015 zal de ervaring van de gemiddelde Visual Basic developer echter nog meer op vooruit gaan.

Een van de belangrijkste redenen waarom ik zo verknocht ben aan de third-party tool Resharper zijn de uitgebreide mogelijkheden van ingebouwde refactorings. Visual Basic 2015 zal standaard al veel van deze refactorings aan boord hebben. Dit helpt je oner meer om beter leesbare en onderhoudbare code te schrijven. Een cadeautje dat we krijgen door de release van het .NET Compiler Platform (Roslyn), waardoor de code editor veel beter weet wat een regel code precies doet en hoe dat eventueel vereenvoudigd kan worden. Een andere nieuwe feature die uit dezelfde koker komen zijn de zogenaamde Analyzers.

Analyzers geven jou – of anderen – de mogelijkheid om light-bulbs met code acties in de kantlijn te plaatsen of code met een squiggle te onderstrepen. Hiermee kun je codestandaarden of best-practices binnen je team afdwingen. Denk hierbij bijvoorbeeld aan het opgooien van echte Warnings of Errors wanneer een collega een bibliotheek van jou verkeerd toepast. Je kunt je collega's dus aanwijzingen geven, net zoals je deze nu krijgt wanneer je een variabele wel hebt gedeclareerd, maar nergens hebt toegepast. Er krachtig!

Microsoft heeft ook Edit-and-Continue verbeterd. In situaties waarbij men asynchrone code gebruikt of iterators toepaste, werkte Edit-and-Continue niet altijd even goed. Dit zou nu verbeterd moeten zijn. Daarnaast zou men nu ook betere foutmeldingen in de Error-list getoond moeten krijgen. Los van deze betere omschrijvingen, is het aantal niet meer beperkt tot het (vreemde?) aantal van 101. In Visual Studio 2015 worden nu alle fouten getoond. Iets wat C# ontwikkelaars al veel langer kennen, maar wat voor Visual Basic ontwikkelaars nieuw is, is dat de node References in je Solution Window ook zichtbaar is zonder dat gelijk alle bestanden getoond moeten worden middels de button 'Show All Files'.

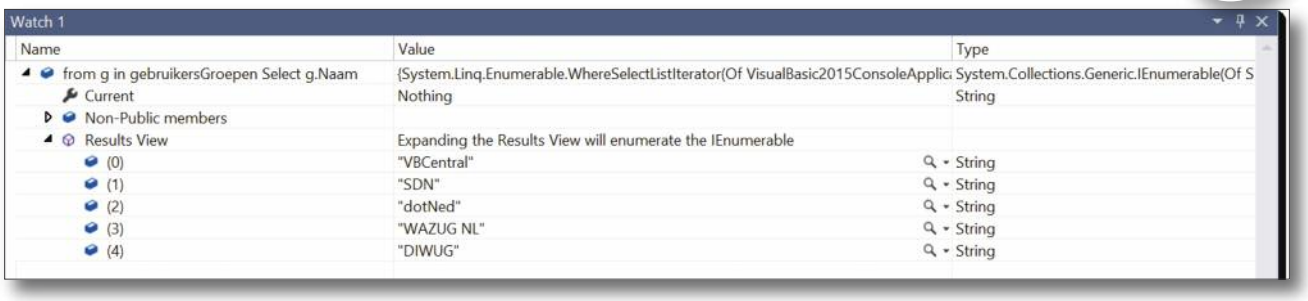

Fig.2: Toepassen van LINQ in de Watch Window

Een kleine, maar zeer handige uitbreiding, waarvan je je wel afvraagt waarom we hierop zolang hebben moeten wachten.
Ik heb het zelf nog niet echt getest, maar het compileren van projec ten is in Visual Studio 2015 ongeveer 50% sneller geworden dan in Visual Studio 2013. Dit zal je misschien verbazen als je weet dat de voorgaande versie van de Visual Basic compiler geschreven is in C++ en de nieuwe in Visual Basic .NET zelf. Het toepassen van betere algoritmes, slimmere datastructuren en gebruik van Async and Await hebben tot dit mooie resultaat geleid.
Waar ik zelf het meeste naar uitkijk is de mogelijkheid om lambdas en LINQ expressies uit te kunnen voeren in de Watch - of Immediate Window. Dit kan echt handig zijn, met name wanneer je waarden uit een grote collectie met een diepere structuur wilt bekijken. Onderstaand voorbeeld is natuurlijk wat eenvoudiger, maar geeft zeker de kracht en mogelijkheden hiervan aan.

Het statement 'from g in gebruikersGroepen Select g.Naam' is direct ingevoerd en geëvalueerd. Voorwaarde is natuurlijk wel dat je ergens in je code een breakpoint hebt staan en dat er de variabele gebruikersGroepen binnen scope is.

Conclusie
Het blijft me telkens wel weer verbazen dat een nieuwe versie van een tool zoals Visual Studio me elke keer weer weet te verrassen. Natuurlijk verandert het applicatielandschap erg snel, waardoor ondersteuning voor dat soort type applicaties voor de nodige nieuwe mogelijkheden zorgt. Maar ook je dagelijkse werkzaamheden in codebases waar je misschien al wel jaren aan werkt, worden prettiger met deze nieuwe versie. Hoewel er nog geen harde releasedate is afgegeven, verwacht men een release in de zomer van 2015. Van mij mag deze versie echter morgen al wel gereleased worden! •

André Obelink

André Obelink is eigenaar van MarYor | software & consultancy. Een bedrijf dat software maakt voor uiteenlopende bedrijven en ontwikkelteams steunt om betere software te bouwen. André is zelf een substantieel deel van zijn tijd werkzaam als consultant, trainer en softwarearchitect in voornamelijk de financiële sector. Hij heeft meerdere boeken geschreven over Visual Basic .NET en is coauteur van het Visual C# kookboek. Sinds 2006 heeft hij voor zijn bijdrage aan de (inter)nationale .NET community, elk jaar een Microsoft MVP award mogen ontvangen.