Bart Janson is werkzaam als Professional Software Engineer bij Qurius ETX te Rijswijk

Visual Studio 2005 IDE Extensibility

OP WEG NAAR EEN GEÏNTEGREERDE ONTWIKKELOMGEVING

Als ontwikkelaar besteed je veel tijd in de Visual Studio.NET Integrated Development Environment (IDE), waarbij deze programmeeromgeving het ontwikkelwerk ondersteunt en taken vereenvoudigt. Toch loop je vaak tegen het probleem aan dat de IDE net niet helemaal aansluit bij jouw wensen of dat er veel repeterende handelingen moeten worden verricht. Om dit te verhelpen stelt Microsoft in Visual Studio.NET een aantal uitbreidingsmogelijkheden beschikbaar. In dit artikel bespreek ik welke mogelijkheden dit zijn, wat je er nu eigenlijk mee kunt doen en hoe dit in Visual Studio 2005 verder is verbeterd.

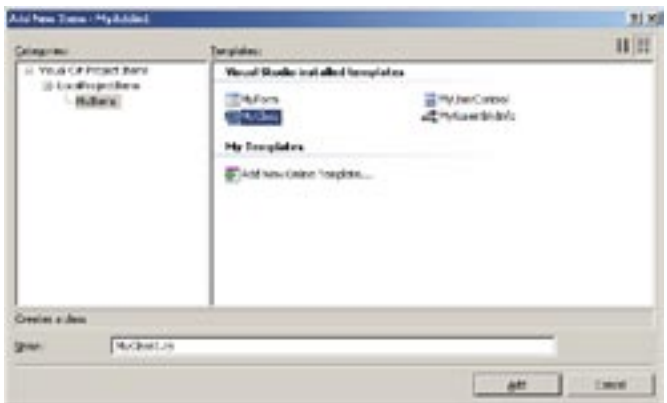
De Visual Studio.NET 2003 IDE is inmiddels uitgegroeid tot een vertrouwde, volwassen omgeving, die de ontwikkelaar hulpmiddelen biedt om zijn werk zo efficiënt mogelijk uit te voeren. Visual Studio 2005 heeft weer nieuwe features toegevoegd; lees het artikel van Robert van der Kleij, "Productiviteitsverbeteringen in nieuwe Visual Studio" in het .Net Magazine #6. Eén van de minder bekende uitbreidingen is de mogelijkheid om eigen templates aan VS2003 toe te voegen, waardoor sneller ontwikkeld kan worden. Daarnaast zijn er vaak handelingen uit het ontwikkelproces die niet aansluiten bij de IDE. Denk bijvoorbeeld aan het uitvoeren van een automatische test, bugtracking of FxCop. Hierbij wordt tooling ingezet of iets zelf ontwikkeld dat buiten Visual Studio wordt opgestart. Deze applicaties zou je echter het liefst binnen de IDE houden, en wel om de volgende redenen:

- alles wat met de solution of het project te maken heeft zou op een centrale locatie opgeslagen moeten zijn, zodat voordelen van efficiëntie, toegankelijkheid en overzicht maximaal benut kunnen worden
- de functionaliteit in de externe applicatie vertoont vaak veel overeenkomsten met de IDE. Denk hierbij aan project/itemhiërarchie, zoeken en SourceSafe-integratie.

Om deze redenen is het zinvol om Visual Studio als raamwerk te gebruiken bij het inzetten van tooling, zoals ook onderdelen van BizTalk Server 2004 zich diep in de IDE nestelen. Met de uitbreidingsmogelijkheden, die bijna allemaal standaard tot onze beschikking staan, kunnen we de ontwikkelomgeving meer naar onze hand zetten. En dat werkt voor ons, ontwikkelaars, sneller en zeker prettiger. Laten we daarom de verschillende uitbreidingsmogelijkheden van Visual Studio .NET 2003 eens onder de loep nemen en kijken wat Visual Studio 2005 hieraan heeft toegevoegd.

Project-templates en projectitem-templates

Net als in bijvoorbeeld Word, is een projectitem-template een raamwerk dat wordt gebruikt om een nieuw document aan te maken. Kies in project 'Add New Item' en er verschijnt een dialoog met de beschikbare projectitems. Deze getoonde items zijn de aanwezige 'projectitem-templates'. De inhoud van deze items kan eenvoudig worden aangepast. Ook het zelf toevoegen van nieuwe templates is niet ingewikkeld. Nuttige toevoegingen in een eigen class-template kunnen bijvoorbeeld SourceSafe-headers, regions en enkele vaak gebruikte namespaces zijn. Afbeelding 1 toont de dialoog van Visual Studio 2005 met een eigen map en vier eigen projectitem-templates.



Afbeelding 1. De dialoog van Visual Studio 2005 met een eigen map en vier eigen projectitem-templates

Afbeelding 2. De 'External Tools'-dialoog

```
Sub CollapseAllProjects()
  Try
    Try
      ' levert exception op als geen open vensters
      DTE.ExecuteCommand("Window.CloseAllDocuments")
    Catch ex As System.Exception ' do nothing
    End Try

    Dim solutionName As String = _
      DTE.Solution.Properties.Item("Name").Value
    Dim projects As Projects = DTE.Solution.Projects
    Dim solutionExplorer As Window = _
      DTE.Windows.Item(Constants.vsWindowKindSolutionExplorer)

    ' collapse all projects in the solution
    For Each project As Project In projects
      Try ' nodig, omdat .Name soms niet beschikbaar is
        Dim itemName As String = _
          solutionName + "\" + project.Name
        Dim item = solutionExplorer.Object.GetItem(itemName)
        item.UIHierarchyItems.Expanded = False
      Catch ex As System.Exception ' do nothing
      End Try
    Next

   Catch ex As System.Exception
    ' do nothing
  End Try
End Sub
```

Codevoorbeeld 1.

Naast projectitem-templates biedt versie 2005 ook de mogelijkheid om zelf project-templates toe te voegen. Wanneer je 'Add New Project' kiest, kun je zo een complete projectstructuur met verschillende items in één keer opzetten. VS2003 biedt geen uitleg in de documentatie over hoe je de projectitem-templates aan kunt maken; hiervoor verwijs ik dan ook naar http://www.code-magazine.com/article.aspx?quickid=0411091. Visual Studio 2005 bevat daarentegen wel een duidelijke uitleg, zowel over de project-templates als de projectitem-templates.

Projectitem-templates zijn handig bij aanvang van een nieuw project. De hoofdontwikkelaar richt namelijk een solution-structuur in voor het project en er worden de benodigde namespaces en assembly-referenties bepaald. Als deze ontwikkelaar vooraf tijd besteedt aan het opstellen van enkele projectitem-templates en deze vervolgens beschikbaar stelt aan de rest van het team kan er veel tijd worden bespaard. De teamleden kunnen de templates via xcopy of een zip-bestand met elkaar uitwisselen. Een nettere oplossing wordt ons in versie 2005 geboden, waarbij het pad naar de templates configureerbaar is. Dit pad kan dan naar een netwerklocatie in de eigen organisatie verwijzen. Deze instelling is te vinden bij de opties voor 'Solutions and Projects' in het Tools->Options-menu. De in afbeelding 1 getoonde 'Add New Online Template'-optie (nog niet werkend in bèta 1 van Visual Studio 2005) zal in het uiteindelijke product bedoeld zijn voor het ophalen van templates van een zelf te configureren URL. Op deze manier zijn de organisatie-templates ook vanaf de eigen werkplek te benaderen. Zoals gezegd biedt Visual Studio, naast de uitbreidingsmogelijkheden voor templates, ook ruimte om tooling in de IDE te integreren. Hierna licht ik de verschillende mogelijkheden toe.

External Tools

Via het 'Tools'-menu komen we terecht bij de 'External Tools'-dialoog waarmee we, zoals de naam al laat raden, tooling buiten de IDE kunnen opstarten. Met deze dialoog, getoond in afbeelding 2, kun je configureren welke externe tools er zijn en hoe deze zijn aan te roepen. Het is mogelijk een aantal eigenschappen van de solution mee te geven bij het opstarten van de tool. Zo is te zien in afbeelding 2 dat de target-dir van het huidige project wordt meegegeven als opstartfolder. Op deze manier wordt het configureren van externe applicaties eenvoudig en heb je de flexibiliteit om tooling buiten de IDE aan te roepen. Er is echter geen sprake van een geïntegreerde ontwikkelomgeving, zoals wel het geval is bij de voorbeelden uit de volgende artikelonderdelen.

Macro's

Visual Studio-macro's zijn stukjes Visual Basic .NET-code, die in de IDE zelf uitgevoerd worden. Een macro kan de interne structuur van Visual Studio benaderen, het Visual Studio .NET IDE automation-objectmodel. Hiermee kunnen de menu's, de solutionstructuur en de editor worden bestuurd. Zo kan een macro bijvoorbeeld alle vensters sluiten en de projecthiërarchie in de Solution Explorer inklappen; zie codevoorbeeld 1.

Het aanmaken van een macro kan eenvoudigweg door het opnemen van toetsaanslagen en handelingen in de IDE. Hierna is de macro naar wens aan te passen in de Macro IDE. Macro's hebben echter ook nadelen. Zo kunnen ze alleen in Visual Basic .NET worden geschreven. Daarnaast zijn de userinterfaces van schermen alleen op een declaratieve manier aan te maken en dus niet met een designer. Het aanroepen van een macro kan vanuit de Macro Explorer of door de macro toe te kennen aan een button op een toolbar. Het behoort niet tot de standaard mogelijkheden om macro's te delen en centraal te beheren. In een team zullen ze dus met xcopy in de lokale macro-folder moeten worden geplaatst. Macro's zijn vooral geschikt voor kleine handelingen die geen userinterface vereisen en kunnen door het opnemen snel worden geschreven. Hoewel versie 2005 je een rijker IDE automation-objectmodel biedt dan versie 2003 is de werking en het beheer van macro's niet veranderd.

Add-ins

Wanneer ingewikkelde handelingen of userinterfaces nodig zijn voor een tool, is de macroaanpak niet toereikend. Visual Studio heeft dit opgelost door een add-in model te ondersteunen, waarmee je assemblies aan de IDE kunt toevoegen. Deze assemblies hebben toegang tot het automation-objectmodel, de EnvDTE namespace, en worden door implementatie van de 'IDTExtensibility2'-interface herkend als add-in voor Visual Studio. Met de EnvDTE namespace kun je de projectstructuur van de huidige solution, de menustructuur van de IDE en het Code Model benaderen en bewerken. Met dit Code Model is de broncode van een class in je huidige solution benaderbaar. Zo kan bijvoorbeeld van een projectitem worden opgevraagd welke classes deze implementeert en in welke overige projectitems deze partial class nog meer wordt gedefinieerd; zie hiervoor codevoorbeeld 2.

Deze add-ins zijn niet nieuw voor Visual Studio 2005, maar worden al sinds Visual Studio .NET 2001 ondersteund. De EnvDTE namespace in versie 2005 is echter wel uitgebreid met een EnvDTE80 namespace, waarmee een rijker add-in programmeermodel voorhanden is. Voorts biedt Visual Studio 2005 een managed assembly voor beheer van de toolbars, menu's en commands, waarvoor in Visual Studio .NET 2003 nog een reference naar een Office-COM-assembly nodig was. Door het projecttype 'Visual Studio Add-in' wordt een wizard gestart, die een basis legt voor een eigen add-in implementatie. Om je add-in aan anderen beschikbaar te stellen, kun je een msi maken met behulp van het door de wizard aangemaakte Setup-project. Op de extensibility-webpagina's zijn voldoende standaard voorbeelden van add-ins te vinden; zie bijvoorbeeld http://msdn.microsoft.com/vstudio/extend/. Ook een zoektocht op internet levert veel varianten van add-ins op. Het betreft dan vaak add-ins voor code-snippets en refactoring. In Visual Studio 2005 zijn deze al standaard geïmplementeerd en zelfs beter en completer dan voorheen.

De complexiteit van add-ins is te overzien. De beide EnvDTE namespaces zijn in Visual Studio 2005 wel gedocumenteerd, maar heel veel meer dan de beschikbare methoden en properties vind je er niet. Gebruik daarom de macrorecorder om te kijken wat de IDE bij een bepaalde functie doet en bestudeer enkele op het internet beschikbare voorbeelden.

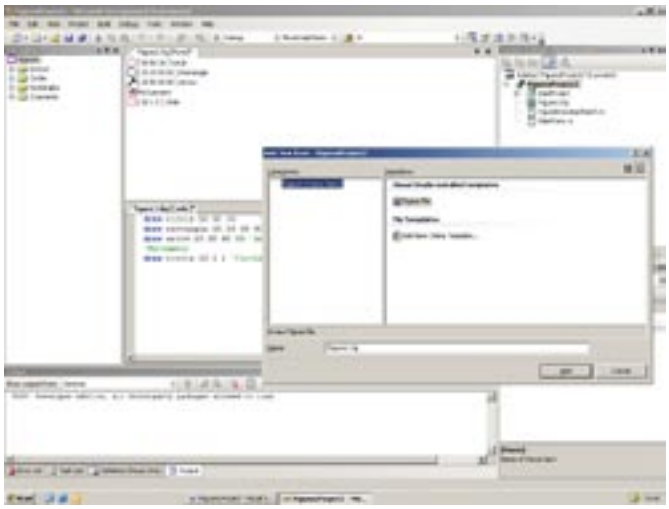
Wizards

De IDE kan wizards opstarten na het toevoegen van een item of project aan de solution. Dit gebeurt niet automatisch bij alle typen projecten; je kunt hiervoor bijvoorbeeld wel de eerder genoemde 'Visual Studio Add-in' kiezen. De in de wizard gemaakte keuzes bepalen hoe het project vervolgens wordt ingericht met het gedefinieerde stappenplan. Veel verschil met het schrijven van een add-in is er niet, alleen moet nu de interface 'IDTWizard' geïmplementeerd worden. Vervolgens moet je ook nog een vsz- en vsdir-bestand configureren om een gekozen item aan je wizard te koppelen. Naar mijn mening kost het schrijven van een wizard in veel gevallen meer moeite en tijd dan dat het je oplevert. Wil je bijvoorbeeld alleen stukken code wel of niet in het project plaatsen, dan ben je sneller uit wanneer je van het eerdergenoemde project projectitem-templates inzet. Deze zijn gemakkelijker aan te passen en eenvoudiger te verspreiden. Schrijf daarom alleen een wizard als de complexiteit van de projectinrichting je tot deze keuze dwingt.

VSIP en VSPackages

Add-ins zijn krachtig, maar de voor een add-in beschikbare namespaces zijn niet voor alles toereikend. Zo kun je wel de items in een project opvragen, maar is het niet mogelijk om een eigen type project te maken. Toch zien we soms na installatie van een softwarepakket, zoals BizTalk, dat de Visual Studio IDE verrijkt is met een nieuw type project, een nieuwe explorer en nieuwe editors. Ook de IDE van Visual Studio 2005 zelf is voorzien van een aantal nieuwe editors, designers en toolvensters, zoals de ClassDiagram-designer en het Document Outline-venster. Deze uitbreidingen, VSPackages genaamd, zijn gemaakt met de Visual Studio Industry Partner SDK's die ook beschikbaar zijn voor eigen gebruik.

Het Visual Studio Integration Package (VSIP) bestaat al sinds de eerste versie van Visual Studio, alleen was het voor 3rd-party developers pas beschikbaar na aanschaf van een licentie. Deze licentieovereenkomst gaf recht op deelname aan een speciaal programma, bestaande uit de VSIP SDK's, toegang tot cursussen, support en gezamenlijke marketinginspanningen. Toen VS2003 uitkwam is besloten het licentiemodel te herzien en is aan Visual Studio een gratis Affiliate-variant toegevoegd, waarbij de VSIP SDK's vrijelijk gebruikt mogen worden na registratie via de website.



Afbeelding 3. De sample 'FigPkgs' met een editor, een designer en een projecttype met hiërarchie van projecten

Met de uitgave van Visual Studio 2005 in het vooruitzicht is nu alvast de VSIP 2005 bèta 1 beschikbaar gesteld. Het opmerkelijkste verschil met de vorige VSIP-versies is, dat er nu ook managed code gebruikt kan worden voor implementatie van VSPackages. Hierdoor kunnen alle mogelijkheden van het .NET Framework benut worden, hetgeen een stabieler platform oplevert en debugging van de te ontwikkelen packages vereenvoudigt. Ik verwacht niet dat er veel verschillen zullen zijn tussen VSIP 2005 bèta 1 en de definitieve versie. De meeste uitbreidingen in versie 2005 zelf zijn ook met de VSIP geschreven. We kunnen er dus van uitgaan dat de meeste onvolkomenheden in het objectmodel zijn ontdekt en opgelost.

VSIP en geavanceerde projecten

Bij het ontwikkelen van VSPackages kun je zowel gebruik maken van de EnvDTE namespaces als van een aantal specifieke VSIP namespaces. Met de combinatie van deze namespaces kunnen onder andere de volgende soorten VSPackages worden geschreven:

- *Custom Tools*: ook wel generators genoemd. Hiermee kan de inhoud van een item geconverteerd worden naar een andere vorm. Visual Studio heeft standaard al een aantal generators, zoals de MSDataSetGenerator voor het genereren van de typed classes van een dataset.
- *Tool windows*: dit zijn de 'snappable' vensters in de IDE, zoals de Solution Explorer, Task List en het Property-venster. Met add-ins zijn ook toolwindows te maken, maar de VSIP-variant verschaft meer controle; Zie het meegeleverde voorbeeld 'ToolWindow'.
- *Editors of designers*: zoals bijvoorbeeld de designer voor een typed dataset of de C#-editor. Zie het meegeleverde voorbeeld 'BasicEditor' of 'FigPkgs'.
- *Projecttypen*: zoals bijvoorbeeld het databaseproject of een console-applicatie. Zie het meegeleverde sample 'Project' of 'FigPkgs' voor een voorbeeldimplementatie.
- *Programmeertalen*: zoals bijvoorbeeld C# ook als taal binnen de IDE wordt ondersteund door een editor en de debugger en de Object Browser. Zie het meegeleverde voorbeeld 'Babel'.
- *Services*: dit zijn VSPackages die worden geïmplementeerd om functionaliteit aan andere VSPackages beschikbaar te stellen, zoals de SourceSafe-service gebruikt kan worden voor SourceSafe-integratie. Zie het meegeleverde voorbeeld 'Babel' voor implementatie van een service of 'FigPkgs' voor het gebruik van de SourceSafe-service.

De genoemde voorbeelden zijn na installatie van VSIP 2005 te vinden in de EnvSDK-folder, waarbij direct opvalt dat er veel meer samples, managed en unmanaged, wordt meegeleverd. In de meegeleverde documentatie van Visual Studio 2005 staat beschreven

```
// rootElement is de namespace, deze bevat classes
foreach (CodeElement2 Element in rootElement.Children)
{
  if (childElement.Kind == vsCMElement.vsCMElementClass)
  {
    CodeClass2 codeClass = (CodeClass2)childElement;

    string className = codeClass.Name;
    string projectItems = string.Empty;

    foreach (CodeElement2 partElem in codeClass.PartialClasses)
    {
    projectItems += "\r\n - " + partElem.ProjectItem.Name;
    }

    MessageBox.Show("class [" + className
    + "] is implemented in: " + projectItems);
  }
}
```

Codevoorbeeld 2.

hoe deze voorbeelden zijn opgebouwd en hoe ze geïnstalleerd kunnen worden. Deze documentatie is zeker nog niet volledig, maar bevat wel al duidelijke stappenplannen voor de implementatie en deployment van eigen VSPackages. Ook is na installatie een nieuw projecttype 'Visual Studio Integration Package' beschikbaar, waarmee de wizard wordt opgestart voor het opzetten van een VSPackage. Afbeelding 3 toont de sample 'FigPkgs' met een editor, een designer en een projecttype met hiërarchie van projecten.

Het gebruik van VSIP 2005 biedt veel voordelen. Je kunt er geavanceerde projecten mee realiseren, die zeer hecht geïntegreerd zijn met de ontwikkelomgeving. Daarnaast heb je bij VSPackages het voordeel dat zaken die buiten Visual Studio vaak veel inspanning vergen, nu eenvoudig in de IDE in te passen zijn, zoals de genoemde SourceSafe-integratie. Natuurlijk gaat zo'n positief oordeel vergezeld van een kanttekening: de implementatie van VSPackages mag niet onderschat worden. Ter illustratie: het FigPkgs sample van afbeelding 3 implementeert ongeveer 26 (!) interfaces en gebruikt nog eens 14 interfaces bij communicatie met andere VSPackages. Nu moet ik eerlijkheidshalve zeggen dat andere samples minder interfaces uit de VSIP SDK gebruiken. Afhankelijk van de keuzes die je maakt, zul je dus een behoorlijke investering moeten doen, maar het levert je dan wel een mooie, geïntegreerde oplossing op.

Leerzaam en uitdagend

De opeenvolgende versies van Visual Studio vertonen een duidelijke ontwikkeling naar meer integratie van de tooling met de ontwikkelomgeving. De tooling van nieuwe serverproducten van Microsoft wordt meer en meer in de IDE opgenomen en die trend zal doorzetten in Visual Studio 2005 Team Systems. Uit de toon van dit artikel mag blijken dat ik die ontwikkeling toejuich. Niet alleen omdat ik van mening ben dat het leidt tot een efficiënte werkwijze en betere applicaties, maar ook omdat ik als ontwikkelaar heb ervaren hoe leerzaam en uitdagend het kan zijn om de IDE uit te breiden met de mogelijkheden die Visual Studio ons standaard biedt.

Zoals gezegd zal de implementatie van add-ins en VSPackages weliswaar een substantiële investering vergen, maar er kan hierbij gelukkig gebruik worden gemaakt van een aantal bewezen, goed werkende onderdelen van de IDE. Met de overige van de hier behandelde uitbreidingen kun je zeer snel zelf of binnen het ontwikkelteam aan de slag. Laat de tips en de toelichting in dit artikel een aanleiding zijn om er optimaal gebruik van te gaan maken.

Bart Janson is werkzaam als Professional Software Engineer bij Qurius ETX te Rijswijk (http://www.quriusetx.nl/). Hij is nauw betrokken bij ontwikkeling van de tooling voor de SoftwareStraat.NET van Qurius ETX en is te bereiken op bartj@quriusetx.nl.

Referenties

Visual Studio Extensibility Center: http://msdn.microsoft.com/vstudio/extend/ en http://www.vsipdev.com/

GotDotNet Developer PowerToys: http://www.gotdotnet.com/team/ide/

Project Item Templates: http://www.code-magazine.com/article.aspx?quickid=0411091

Inside Microsoft® Visual Studio® .NET 2003: http://www.microsoft.com/mspress/books/companion/6425.asp

Nieuwsgroepen: microsoft.public.vsnet.ide en microsoft.public.vstudio.extensibility