Jan-Willem Overbeek is software architect en ontwikkelaar bij Capgemini.

Wat is er nieuw in Visual Studio 'Whidbey' ?

MIJLPALEN IN DE ONTWIKKELING VAN HET .NET-PLATFORM

Onder de codenaam 'Whidbey' werkt Microsoft aan nieuwe versies van het .NET Framework en Visual Studio. Vroege versies hiervan zijn vorig jaar aan bezoekers van de Microsoft Professional Developer Conference (PDC) ter beschikking gesteld, en in de loop van 2004 zullen er ook bèta's voor een groter publiek beschikbaar komen. De uiteindelijke release staat gepland voor de eerste helft van 2005.

Inmiddels is ook de definitieve naam voor Visual Studio 'Whidbey' bekend: Visual Studio 2005. Oplettende lezers zullen opmerken dat ook bij dit product de toevoeging .NET weer verdwenen is. Het .NET Framework Whidbey heeft alleen een nieuw versienummer gekregen, en heet dus gewoon .NET Framework 2.0.
In dit artikel doe ik een poging een overzicht te geven van de voornaamste nieuwe ontwikkelingen die met Whidbey geïntroduceerd zullen worden. Hierbij moet ik opmerken dat de ontwikkeling van Whidbey nog in volle gang is, en dit overzicht daarom nog niet 100% volledig kan zijn. In toekomstige nummers van .NET magazine zal eventuele ontbrekende informatie uiteraard worden aangevuld. Lezers die meer over specifieke onderwerpen willen weten, kunnen het beste kijken naar de presentaties van de PDC [1], of een bezoek brengen aan de Tech-Ed 2004.

.NET Framework 2.0 en Visual Studio 2005

Whidbey valt uiteen in een tweetal producten: .NET Framework 2.0 en Visual Studio 2005. Het merendeel van de nieuwe functionaliteit is onderdeel van het .NET Framework 2.0. Visual Studio 2005 biedt vooral ondersteuning om deze nieuwe functionaliteit eenvoudig te kunnen toepassen. Denk hierbij aan nieuwe en verbeterde designers, wizards, editors en project templates. Maar Visual Studio 2005 zal echter ook nieuwe functionaliteit gaan bieden, die niet direct is gerelateerd aan de verbeteringen in .NET Framework 2.0. Een voorbeeld hiervan is een nieuwe set van visuele modelleringtools, die op dit moment alleen nog bekend zijn onder de codenaam 'Whitehorse'. Verderop in dit artikel zal ook deze nieuwe functionaliteit van Visual Studio 2005 nader worden toegelicht.

.NET Framework 2.0

Zoals het versienummer 2.0 al aangeeft, is deze versie de eerste major release sinds de introductie van het .NET Framework in 2000. De verbeteringen in deze versie zijn vooral terug te vinden in de Framework Class Library (FCL), maar ook in de Common Language Runtime (CLR) zelf is een aantal belangrijke verbeteringen doorgevoerd.

Verbeteringen in de CLR

De CLR van .NET Framework Whidbey biedt een aantal verbeteringen, zoals:

- Generics
- 64 bit support
- Performanceverbeteringen
- 'Edit and Continue' [2]

Vooral generics is een belangrijke nieuwe feature voor ontwikkelaars. Hierover is inmiddels al vrij veel informatie gepubliceerd [3]: wat generics precies zijn, hoe de CLR deze technologie ondersteunt, en de syntax waarmee generics kunnen worden toegepast in de diverse .NET-programmeertalen (C#, VB.NET, MC++). Om die reden zal ik daar hier niet verder op in gaan. Toch lijkt het me nuttig om het belang van generics, en het effect dat ze zullen gaan hebben op het .NET Framework, nog een keer expliciet te benadrukken. De kans is namelijk groot, dat generics voor een groot deel van de .NET-onwikkelaars een nieuw fenomeen zal zijn. C++-ontwikkelaars zijn doorgaans al wel bekend met het fenomeen; in C++ wordt deze functionaliteit doorgaans 'templates' of 'geparametriseerde types' genoemd. Maar voor VB-, C#- en Java-ontwikkelaars[4] is het belang van generics wellicht nog een mysterie.

Ik durf echter te stellen dat de invloed van generics op de Framework-classes groot zal zijn. Dit zal nog niet zo sterk merkbaar zijn in versie 2.0, maar in volgende versies zal er intensief gebruik gemaakt van gaan worden - in vrijwel alle onderdelen van de FCL. Het is dus belangrijk voor ontwikkelaars om alvast te wennen aan de concepten van generic programming, want het zal in de toekomst vrijwel onmogelijk zijn om met het .NET Framework te ontwikkelen zonder deze kennis. En ook voor (ex) C++-ontwikkelaars, die al ervaring hebben met templates, is het nuttig om zich alvast wat in de .NET-implementatie van generics te verdiepen. Want er zijn toch wel wat verschillen tussen templates in C++ en generics in .NET. Het kan dus zeker geen kwaad om je als ontwikkelaar vroegtijdig te verdiepen in de toepassing van generics. Te meer omdat het concreet toepassen van deze technologie toch een periode van gewenning vereist. Het kost de meeste ontwikkelaars doorgaans een tijdje om echt thuis te raken in het gebruik van generic classes, en zeker in het zelf ontwikkelen van generic classes, want:

- Het gebruik van generic classes brengt een nieuwe syntax met zich mee - afhankelijk van de gebruikte programmeertaal - die niet altijd even eenvoudig te lezen is en in één oogopslag te doorgronden. De meeste ontwikkelaars vinden dat het gebruik van generics de leesbaarheid van de source-code in eerste instantie nogal vertroebelt, totdat men helemaal gewend is aan de nieuwe syntax.
- Het ontwikkelen van een generic class dwingt de ontwikkelaar om te denken op een hoger abstractieniveau dan bij het ontwikkelen van een 'gewone' class. Het kan dus enige tijd vergen om hier productief in te worden. Ook het debuggen van generic classes kan in sommige gevallen iets lastiger zijn dan bij 'normale' classes.

Ondanks deze leercurve is generic programming een zeer krachtige techniek, en het leren zeker de moeite waard. Het maakt het mogelijk om zeer compacte, flexibele, typesafe code te schrijven. Generic programming zal het aantal regels code in een applicatie flink terugbrengen, wat de onderhoudbaarheid en performance weer ten goede komt.

Om een beeld te krijgen hoe een class library die gebruik maakt van generics er uitziet, is de Standard Template Library (STL) een mooi voorbeeld. Deze library is een onderdeel van de C++ ISO standaard, en wordt al jaren meegeleverd met Visual C++. In MSDN is hier voldoende informatie over te vinden. Al met al is de ondersteuning van generics een belangrijke nieuwe functionaliteit van de CLR. En hoewel het gebruik ervan in de Framework Class Library in Whidbey nog beperkt is, is er toch al wel een aantal te vinden. Hieronder volgt een aantal interessante voorbeelden.

Verbeteringen in de Base Class Library (BCL)

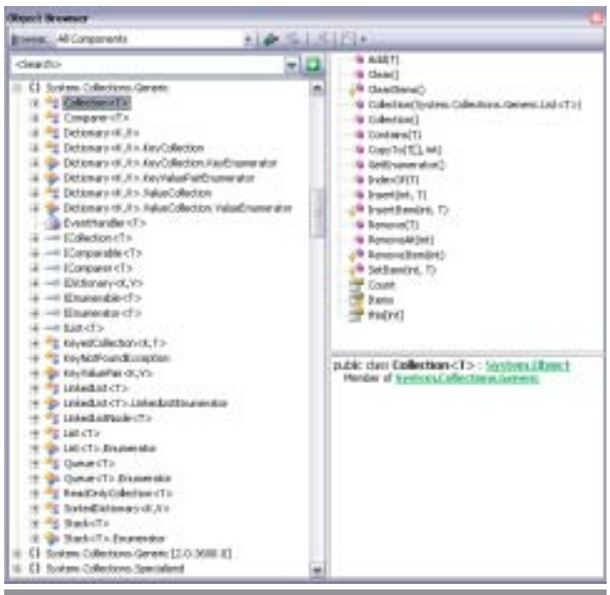
De Base Class Library is een subset van de Framework Class Library, waartoe een aantal [5] van de classes in de "System.*" namespace behoren. In deze classes is een aantal kleine en grote verbeteringen aangebracht. In de BCL zien we ook de eerste toepassing van generics terug, en wel in de System.Collections.Generics namespace. Niet verwonderlijk, want dit soort classes leent zich uitermate goed om met behulp van generics te worden geïmplementeerd.

Deze nieuwe namespace bevat een aantal generic collection classes, zoals Collection<T>, Queue<T>, Stack<T>, enzovoort. De oude collection classes (System.Collection) zijn uiteraard ook nog steeds beschikbaar, maar zijn grotendeels overbodig geworden door de komst van de nieuwe generic collection classes.Een andere interessante toepassing van generics in de BCL is structure Nullable<T>. Hiermee wordt het mogelijk om iedere bestaande value type nullable te maken, zoals een standaard decimal.

Verder zijn er nog vele grote en kleine verbeteringen in de BCL. Hier volgen er enkele:

- Toegang tot seriële poorten
- Verbeterde tracing-functionaliteit
- Lichtgewicht transacties
- Stream-compressie

Voor een compleet overzicht zie de presentatie .NET Framework: Exploring What's New in the Base Class Library for 'Whidbey'.



Afbeelding 1. System.Collection.Generics in de Object Browser

```
Nullable<decimal> dec1 = Nullable<decimal>.default;
Nullable<decimal> dec2 = 1.23m;

if (dec1.HasValue)
{
     Console.WriteLine("Value = {0}", dec1.Value);
}
```

Codevoorbeeld 1

Deze is te vinden op de BCL community homepage (http://www.gotdotnet.com/team/clr/bcl/) en bij de PDC-presentaties.

Data Access

Data Access kent een aantal verbeteringen en nieuwe functionaliteit in .NET Framework 2.0. Een aantal zal hier worden besproken [6]:

- Algemene verbeteringen in ADO.NET
- Ondersteuning voor Microsoft SQL Server 2005 ('Yukon')
- ObjectSpaces

ADO.NET 2.0

De verbeteringen aan ADO.NET in .NET Framework 2.0 zijn vrij evolutionair. Gelukkig maar, want de Data Access roadmap van Microsoft is de afgelopen 10 jaar nogal grillig geweest.

De voornaamste verbeteringen in ADO.NET zijn:

- Performance: de performance van DataSets is sterk verbeterd. DataSets ondersteunen nu zeer snelle binaire serialization, wat met name de performance in remoting scenario's ten goede komt.
- Deployment: ADO.NET is niet meer afhankelijk van MDAC als er gebruik wordt gemaakt van de managed providers.
- Common Provider Model: ADO.NET 2.0 heeft een nieuw, provideronafhankelijk objectmodel. Voorheen was er een afzonderlijk objectmodel per provider, wat het lastig maakte om provideronafhankelijke code te schrijven.

Microsoft SQL Server 2005

Het zal de meeste ontwikkelaars niet zijn ontgaan dat in de volgende versie van SQL Server een diepgaande integratie met het .NET Framework zal worden geïntroduceerd. Hierdoor wordt het mogelijk om .NET-classes als Yukon User Defined Types (UDT's) te gebruiken, en zo .NET-objecten rechtstreeks in SQL Server 2005 op te slaan. Om een .NET-class geschikt te maken om als SQL Server 2005 UDT te gebruiken, moet deze aan een aantal voorwaarden voldoen. De voornaamste eisen zijn:

- De class moet worden voorzien van (minimaal) twee attributen, te weten 'Serializable' en 'SqlUserDefinedType'.
- De class moet nullable zijn - door de INullable interface te implementeren.
- De class moet minimaal een parameterloze publieke constructor bevatten.
- De class moet conversie van en naar een string ondersteunen.

Als aan deze voorwaarden is voldaan kan de class worden gebruikt als een willekeurig ander SQL-type. Houdt er echter rekening mee dat dit een feature is die specifiek is voor SQL Server 2005, dus applicaties die er gebruik van maken werken op geen enkele andere database.

```
[Serializable]
[SqlUserDefinedType (Format.Native)]
Class SqlUdt : INullable
{
     public SqlUdt() {…}
     public bool IsNull {…}
     public static SqlUdt Parse(SqlString s) {…}
     public override string ToString() {…}
}
```

Codevoorbeeld 2

ObjectSpaces

Een geheel nieuwe technologie in .NET Framework 2.0 is het objectrelationele mapping framework genaamd ObjectSpaces. Het doel van deze technologie zal ik illustreren met behulp van het volgende voorbeeld:

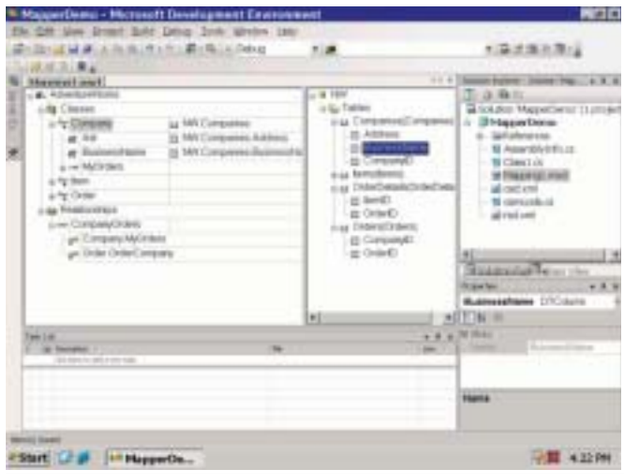
Iedereen die wel eens businessobjecten heeft ontwikkeld voor een applicatie die gebruik maakt van een relationele database, herkent het volgende patroon of een variatie erop.

1. Haal de data op met behulp van ADO.NET.
3. Instantieer een businessobject.
4. Haal de data uit een ADO.Net DataSet (of SqlDataReader) en stop deze in de properties van het object.
5. Voor een manipulatie van het object uit.
6. Haal de veranderde properties uit het object en stop deze in een DataSet.
7. Sla de data op in de database met behulp van ADO.NET.

Dit is een typisch voorbeeld van een handmatige vorm van object-relationele mapping, dat in tal van applicaties wordt toegepast. Het patroon is zo veel voorkomend dat vele ontwikkelaars hiervoor inmiddels een eigen (min of meer) generiek framework geschreven hebben. Ook zijn er al tal van third party frameworks verkrijgbaar, zowel commercieel als Open Source. Met ObjectSpaces introduceert Microsoft een nieuw framework voor het mappen van relationele data op objecten. En mijn verwachting is dat vele ontwikkelaars hier dankbaar gebruik van zullen maken, want het is geen pretje om deze saaie code zelf te moeten schrijven. Om de mapping te kunnen doen, maakt ObjectSpaces gebruik van een aantal mapping-files. Dit zijn (hoe verrassend) XML-files die de relaties beschrijven tussen de classes en properties aan de ene (objectgeoriënteerde) kant, en tabellen en kolommen aan de andere (relationele) kant van de applicatie. Zodra deze mapping-files zijn gemaakt, kunnen relationele data op een natuurlijke, objectgeoriënteerde manier worden gemanipuleerd. Codevoorbeeld 3 laat zien hoe dit er in code ongeveer uitziet.

```
public class Customer
{
     public string ID;
     public string Name;
     public string Company;
     // etc …
     public ObjectList<Order> Orders = new ObjectList<Order>();
}

public class Order
{
     public int ID;
     public DateTime OrderDate;
     // etc …
}

public static void Main()
{
     SqlConnection conn = new SqlConnection(…);
     ObjectSpace os = new ObjectSpace("map.xml", conn);
     ObjectSet<Customer> cSet = os.GetObjectSet<Customer>("Company =
          'SomeName'");

   foreach (Customer c in cSet)
   {
        foreach (Order o in c.Orders)
          {
               // Doe iets interessants met de order
          }
     }
}
```

Codevoorbeeld 3



Afbeelding 2. Zo zou de ObjectSpaces mapping-editor er uit kunnen gaan zien

Wat in dit voorbeeld opvalt is de afwezigheid van SQL om de gewenste objecten te selecteren. In plaats van SQL wordt een nieuw query-mechanisme toegepast, OPath genaamd. Dit is een nieuwe technologie, die naast ObjectSpaces ook in WinFS, het nieuwe Windows 'Longhorn' filesysteem, wordt toegepast [7]. Overigens wordt op het laagste niveau nog steeds SQL gebruikt om de data in en uit de database te krijgen, maar deze queries worden door de ObjectSpaces-engine gegenereerd aan de hand van de mapping en OPath-queries. Nu ziet dit er allemaal prima uit in een simpel voorbeeld, maar meestal is de praktijk vele malen complexer. Vaak is het zo dat men al snel tegen allerlei beperkingen van het framework aanloopt. En het omzeilen van dergelijke beperkingen kan dusdanig tijdrovend zijn, dat de initiële productiviteitswinst van het framework weer teniet wordt gedaan. Het lijkt er echter op dat ObjectSpaces voldoende flexibiliteit biedt om zelfs de meest complexe scenario's het hoofd te kunnen bieden. Zo is het zelfs mogelijk om zelf de data op te halen - als bijvoorbeeld een zeer complexe, geoptimaliseerde query noodzakelijk is - in een standaard ADO.NET DataReader, en vervolgens hieruit de businessobjecten te vullen. Op dit moment zijn er helaas nog geen tools beschikbaar om de mapping-files te creëren. Het is de bedoeling dat in de definitieve versie van Visual Studio 2005 de mapping-file op een eenvoudige wijze te bewerken is. Afbeelding 2 laat zien hoe een dergelijke editor er uit zou kunnen gaan zien.

Ook is op dit moment nog niet helemaal duidelijk welke databases ondersteund zullen gaan worden door ObjectSpaces. Op dit moment worden alleen Microsoft SQL Server 2000 en 2005 nog ondersteund, de vraag is of dit er nog meer zullen worden.

Windows Forms

De verbeteringen van Windows Forms staan vooral in het kader van compleetheid en performance. In versie 1.0 en 1.1 ontbrak nog wat functionaliteit wat het moeilijk maakte om echte 'XP-achtige' applicaties te ontwikkelen. Hier is met 2.0 duidelijk verandering in gekomen. Er is een aantal nieuwe controls, en vele verbeteringen aan de bestaande controls. Nieuw zijn onder meer:

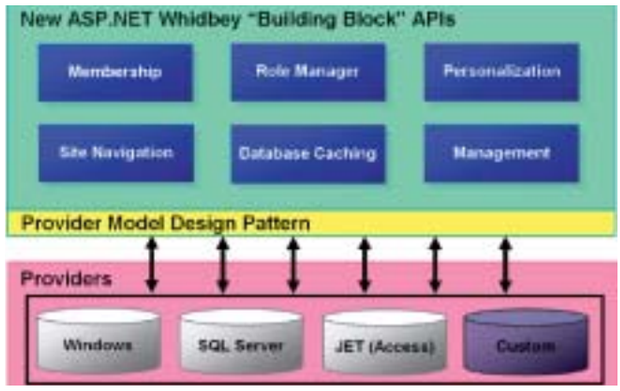
- DataGridView control

Dit control vervangt het DataGrid control uit V1.x. Het is compleet opnieuw geschreven en ontworpen om de vele problemen van zijn voorganger op te lossen. Overigens is het DataGridView-control niet meer hiërarchisch zoals z'n voorganger.

- WebBrowser control

Dit is een wrapper om de COM WebBrowser-control, die stukken

Afbeelding 3. DataGridView en ToolStrips in actie



Afbeelding 4. De Building Blocks kunnen gebruik maken van verscheidene providers

vriendelijker is om vanuit managed code te gebruiken. Het biedt tevens managed HTML DOM-functionaliteit.

- ToolStrip control

Dit is een generieke container voor controls, waar ondermeer de MenuStrip en StatusStrip van zijn afgeleid. De strips zijn runtime te positioneren binnen hun frame; zoals in Internet Explorer. Deze controls vervangen de Toolbar-, MainMenu- en StatusBar-controls.

Verder zijn er vele grotere en kleinere verbeteringen die Windows Forms een 'first class citizen' maken voor het ontwikkelen van Windows client-applicaties. Een beknopte opsomming:

- Ondersteuning voor XP Visual Styles.
- SplitContainer, SoundPlayer, MaskedTextBox classes.
- Verbeterde owner drawing functionaliteit.
- Ondersteuning voor double buffering.
- Verbeterde forms layout engine (Grid, Flow).

ASP.NET

Een aantal van de grootste verbeteringen in Whidbey is te vinden in ASP.NET [8]. Een van de voornaamste doelstellingen van ASP.NET 2.0 is het dramatisch verminderen van de benodigde hoeveelheid code. Om dit te bewerkstelligen is er een aantal nieuwe 'Building Block' API's. Deze API's zijn flexibel en uitbreidbaar. Hiervoor wordt van een 'Provider' design-pattern gebruikt. Een aantal van deze providers is standaard in ASP.NET 2.0 beschikbaar, om de integratie met andere Microsoft-producten zo eenvoudig mogelijk te maken. Maar het is ook mogelijk om zelf een provider te ontwikkelen, bijvoorbeeld om gebruik te maken van reeds aanwezige infrastructuur zoals een Oracle-database of LDAP-server.

Verder is er een grote hoeveelheid nieuwe functionaliteit in ASP.NET 2.0, die weer van deze API's gebruik maakt. Hiermee wordt het mogelijk om geavanceerde features aan een ASP.NET-site toe te voegen met slechts enkele regels code.

Hier volgt een overzicht van de voornaamste nieuwe features.

Web Parts Framework

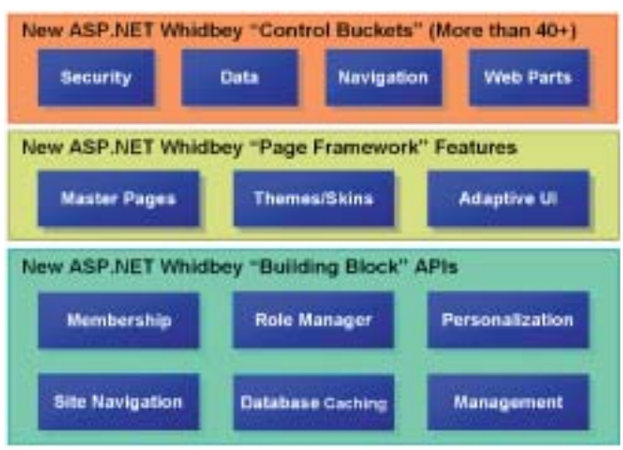
Iedereen die wel eens met Windows SharePoint Services site heeft gewerkt, kent het fenomeen Web Parts. Web Parts zijn componenten, die door de eindgebruiker zelf te configureren en positioneren zijn op een 'gepersonifieerde' webpagina. Het goede nieuws is dat deze krachtige technologie nu standaard in ASP.NET beschikbaar is. Een standaard ASP.NET-site kan dus op dezelfde manier door de gebruiker worden aangepast als een WSS-site.

Toekomstige versies van WSS zullen zelfs verder gaan bouwen op deze basisfunctionaliteit van ASP.NET 2.0. Het is daardoor in de toekomst ook mogelijk om een standaard ASP.NET 2.0 control als Web Part te gebruiken, dus zonder de rompslomp die tot nu toe kwam kijken bij de ontwikkeling van een Web Part. Dit zal er ongetwijfeld toe leiden dat de hoeveelheid beschikbare third party Web Parts in de komende jaren sterk zal gaan groeien.

Master pages

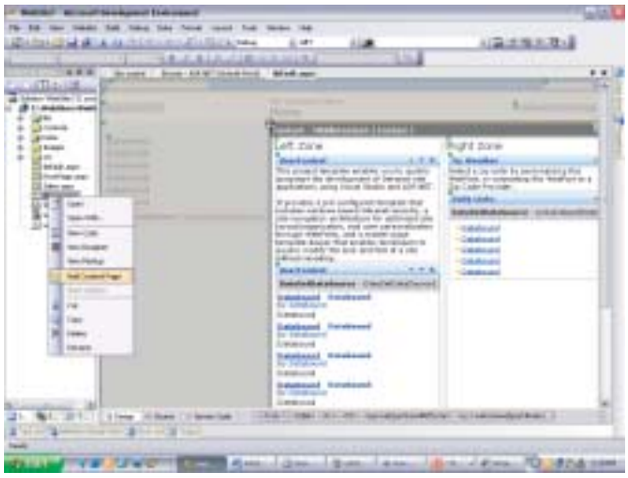
ASP.NET 2.0 heeft ook een feature die 'Master Pages' wordt genoemd. Deze nieuwe functionaliteit maakt het mogelijk een gemeenschappelijke lay-out voor de pagina's van een site vast te leggen. Vervolgens hoeven er alleen nog content-pagina's gemaakt te worden die de gemeenschappelijke eigenschappen van de Master-pagina automatisch erven.

Hierdoor wordt de benodigde code per pagina sterk teruggebracht, en zal de consistentie van de site sterk toenemen.



Afbeelding 5. Een architectureel overzicht van de nieuwe functionaliteit in ASP.NET 2.0

Afbeelding 6. Web Parts kunnen door de gebruiker worden verplaatst



Afbeelding 7. Visual Studio 2005 heeft indrukwekkende ondersteuning voor Master Pages

Membership

De nieuwe Membership-functionaliteit van ASP.NET 2.0 maakt het mogelijk om met een minimum aan code een authenticatiemechanisme aan een site toe te voegen, compleet met de fysieke opslag van de gebruikersgegevens, password recovery, login status, rollen, en meer van dat soort zaken. De eenvoudigste manier om deze settings te configureren is met behulp van een wizard in de ASP.NET configuratietool. Deze wizard doet alle benodigde aanpassingen in de web.config-file aan de hand van een aantal stappen.

Zo helpt de wizard met het kiezen van een authenticatiemechanisme, afhankelijk van het type site. Voor een intranet-site wordt Windows-authenticatie gebruikt, voor een Internet-site het e-mail-adres. Men kan ook kiezen welke provider gebruikt moet worden om deze informatie op te slaan. Er zijn standaard membership-providers voor SQL Server en Access, maar aangezien het provider-mechnisme uitbreidbaar is kan er ook een custom provider worden gebruikt. Vervolgens kunnen gebruikers worden toegevoegd, rollen worden gedefinieerd, en rechten worden toegekend. Dit alles kan vanuit één configuratietool worden geregeld, de security-settings zijn dus niet meer over een aantal locaties verspreid.

Personalisation

Vrijwel iedere site van enige omvang ondersteunt wel een zekere vorm van personalisatie. Helaas bestond hier geen standaard mechanisme voor, en werd voor iedere site het wiel weer opnieuw uitgevonden. Gelukkig is hier met de komst van ASP.NET 2.0 een einde aangekomen. Met de nieuwe personalization-features die ASP.NET biedt is het mogelijk om met een minimum aan code rijke personalization-functionaliteit aan een site toe te voegen. Personalization kan worden geconfigureerd met behulp van de ASP.NET configuratietool. Hiermee kan worden ingesteld welke provider gebruikt zal worden om de gebruikersprofielen op te slaan. Ook hier is er standaard ondersteuning voor een SQL Server of Access personalisatie-provi-

Afbeelding 9. Security- en managementsettings worden hier geregeld

Afbeelding 8. ASP .NET configuratietool

der, maar het is ook mogelijk om een custom provider te implementeren. Vervolgens kan er binnen de site op eenvoudige wijze gebruik van worden gemaakt. In een configuratiefile kan worden gedefinieerd welke properties onderdeel uitmaken van een gebruikersprofiel. Vervolgens kan met behulp van het Profile-object - beschikbaar op de Page-class - ieder willekeurig serializable .NET-object in een property van het gebruikersprofiel worden opgeslagen.

Overige ASP.NET 2.0 verbeteringen

Zoals al eerder gezegd is ASP.NET 2.0 het onderdeel van Whidbey waarin de grootste functionele verbeteringen zijn doorgevoerd. Helaas is dit blad niet dik genoeg om ze allemaal in detail te vermelden, zodat ik helaas nu niet in kan gaan op zaken als:

- Themes en skins
- Verbeterd codecompilatiemodel
- Eén set controls voor alle devices
- Betere tracing en monitoring

Gelukkig is er op MSDN inmiddels redelijk veel informatie beschikbaar over ASP.NET Whidbey. Kijk op http://msdn.microsoft.com/asp.net/whidbey/aspnetwhidbey.aspx voor een overzicht.

ClickOnce

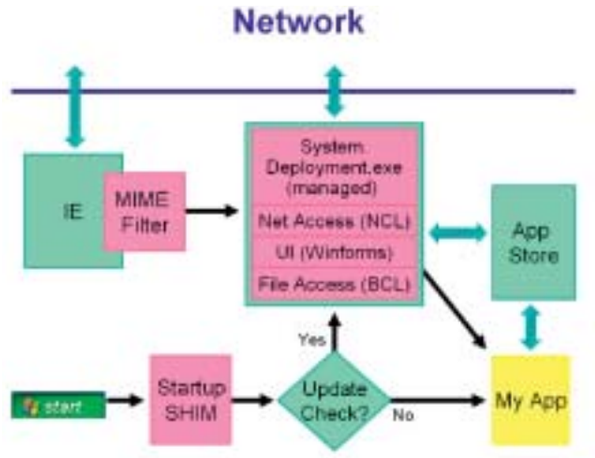
Een andere interessante nieuwe technologie van het .NET Framework 2.0 is ClickOnce. Deze technologie is er op gericht om deployment van Windows Forms-applicaties te vereenvoudigen. Deployment is meestal de reden om een Web-client te verkiezen boven een Windows-client, en ClickOnce heeft tot doelstelling om (op termijn) het gemak van uitrol van een webapplicatie zo dicht mogelijk te benaderen. ClickOnce maakt het mogelijk om een .NET-applicatie eenvoudig en veilig te starten vanaf een Web Server, FTP Server, File Share of een filelocatie - zoals een CD/DVD - zonder dat een ingewikkelde installatieprocedure hoeft te worden doorlopen. Hierbij wordt er in ClickOnce onderscheid gemaakt tussen twee soorten applicaties:

- Applicaties die zowel online als offline beschikbaar zijn.

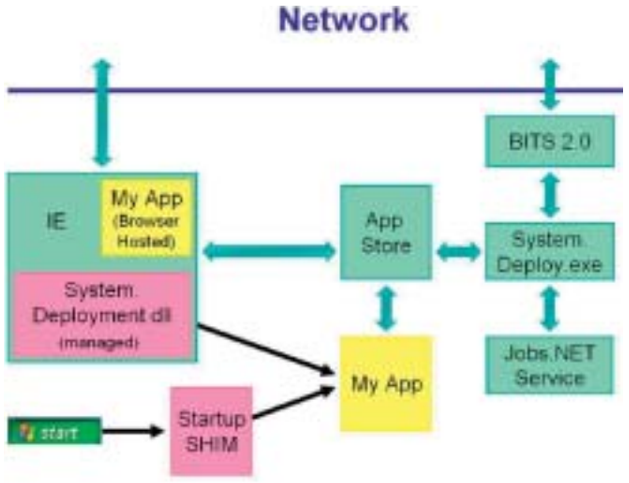
Dit type applicaties wordt op de clientmachine geïnstalleerd, en

```
<personalization enabled="true" defaultProvider="SQL">
     <profile>
     <property name="SomeValue" type="int" defaultValue="42" />
     </profile>
</personalization>

…

void Page_Load(object sender, EventArgs e)
{
     int i = Profile.SomeValue;
}
```

Codevoorbeeld 4



Afbeelding 10. geeft een overzicht van de ClickOnce-architectuur zoals deze in .NET Framework 2.0 geïmplementeerd is.



Afbeelding 11. ClickOnce-architectuur in LongHorn

krijgt ook een shortcut in het Startmenu. De applicatie is ook gewoon te deïnstalleren met behulp van 'Add/Remove Programs'.

- Applicaties die alleen online beschikbaar zijn.

Dit type applicaties wordt direct vanaf de oorspronkelijke locatie gestart. De applicatie wordt op de client gecached, maar er wordt niets permanent geïnstalleerd.

Ook is het mogelijk om er voor te zorgen dat de client automatisch altijd gebruik maakt van de meest recente versie van de applicatie. Hiervoor is er de keuze uit diverse opties, zoals een controle bij het opstarten van de applicatie, een periodieke controle, 'handmatige' controle door de applicatie zelf, enzovoort. En mocht een nieuwe versie om een of andere reden niet in orde zijn, dan biedt ClickOnce de mogelijkheid om op eenvoudige wijze terug te gaan naar de vorige versie van de applicatie. Dit kan zelfs bij clients die na de foutieve update weer offline zijn gegaan.

Een ander belangrijk element van de ClickOnce-technologie is security. Applicaties die via ClickOnce worden gedeployed draaien binnen een security 'sandbox', gebaseerd op het standaard CLR Code Access Security-model. Deze sandbox legt een aantal beperkingen op aan de activiteiten die een applicatie mag uitvoeren op de clientmachine. Hierdoor kan een gebruiker met een gerust gevoel een ClickOnce-applicatie starten, zonder bang te hoeven zijn dat de applicatie schade aan kan richten. Als een applicatie meer rechten nodig heeft dan hij tot zijn beschikking heeft, kan de applicatie de gebruiker hierom vragen met behulp van een dialoog die getoond wordt bij het opstarten van de applicatie. Het is echter ook mogelijk om een applicatie administratief meer rechten te geven, iets wat nuttig is in een intranetomgeving.

Afbeelding 12. ClickOnce updateconfiguratie

ClickOnce manifestfiles

Om al deze features te kunnen bieden, maakt ClickOnce gebruik van een tweetal nieuwe configuratiefiles. Dit zijn het deployment-manifest en het application-manifest. Dit zijn standaard .NET XML configuratiefiles, die herkenbaar zijn aan de extensie .deploy en .manifest. Het deployment-manifest beschrijft de eigenschappen van het deploymentproces, en verwijst naar een application-manifest dat de exacte samenstelling van de applicatie beschrijft. Een ClickOnce-deployment wordt gestart met behulp van de .deploy-file. In een browser zal een link naar een ClickOnce-deployment er dus uitzien als http://myserver.net/myapp/app.deploy. Als een gebruiker op een dergelijke link klikt, wordt dit afgehandeld door een MIME-filter dat door het .NET Framework bij de browser is geregistreerd. Dit MIME-filter zorgt er voor dat er een managed proces wordt gestart dat de .deploy-file verwerkt en zorgt dat de juiste versie van de applicatie wordt gestart. Als de applicatie vanuit het Startmenu wordt gestart, wordt eerst een tussenliggende

Afbeelding 13. ClickOnce Deployment Wizard

DLL aangeroepen die controleert of er een update van de applicatie beschikbaar is.

ClickOnce in Windows Longhorn

Het moge duidelijk zijn dat de ClickOnce-technologie in .NET Framework 2.0 al behoorlijk ver gevorderd is. In Windows 'Longhorn' zal de technologie echter nog verder worden uitgebreid, en dieper in het besturingssysteem worden geïntegreerd (zie afbeelding 9). Tevens zal in Longhorn de sandbox ruimer worden, zodat ClickOnce-applicaties meer secure kunnen doen zonder om extra rechten te hoeven vragen. Hierdoor zal de drempel om voor een Windows-client te kiezen nog weer een stuk lager worden. Ook zal ClickOnce-deployment interessante mogelijkheden bieden in combinatie met de nieuwe Avalon 'Web Applicaties'. Hierdoor ontstaat de mogelijkheid om rich en thin client applicaties naadloos in elkaar over te laten gaan, omdat deze beide naast elkaar in de Longhorn-browser kunnen draaien.

ClickOnce in Visual Studio 2005

In Visual Studio 2005 is brede ondersteuning voor ClickOnce aanwezig. De configuratie van ClickOnce is ingebouwd in de projectsettings, en maakt ook onderdeel uit van het bouwproces. Het is dus niet iets wat achteraf pas gebeurt, zoals tot nu toe het geval was. In de projectsettings kan er bijvoorbeeld worden ingesteld hoe de ClickOnce-applicatie met updates om moet gaan.

Om het configureren van de ClickOnce-opties nog eenvoudiger te maken, is er een nieuwe wizard. Deze geeft de keuze om een project te publiceren op een Web Server, FTP Server, een File Share of in het file-systeem (zie afbeelding 12). Als alle stappen van de wizard zijn doorlopen, wordt er gelijk een handige pagina in Internet Explorer getoond waarmee de applicatie direct gestart en getest kan worden.

Verder is er nog een aantal handige features dat het ontwikkelen van een ClickOnce-applicatie een stuk vereenvoudigt. Zo is er een 'Debug in Sandbox' mogelijkheid. Dit stelt de ontwikkelaar in staat de applicatie te debuggen onder dezelfde omstandigheden waaronder de applicatie zal draaien op een clientmachine; dus met dezelfde restricties wat betreft rechten enzovoort. Ook handig is de Permission Calculator. Deze tool bepaalt aan de hand van de code in de applicatie welke rechten een applicatie nodig heeft om te kunnen functioneren. De ontwikkelaar hoeft dit dan niet meer 'handmatig' vast te stellen.

MSBuild

Het bouwproces is in Visual Studio.NET 2002/2003 een gesloten mechanisme, waar de ontwikkelaar weinig greep op heeft. Met de komst van .NET Framework 2.0 komt hier grote verandering in. Het bouwen van projecten gebeurt dan namelijk niet meer door Visual Studio zelf, maar door een nieuwe build-engine die onderdeel is van het .NET Framework. Het is daardoor dus mogelijk om ook projecten te bouwen op machines waar geen Visual Studio aanwezig is. Deze nieuwe engine heet MSBuild, en verschaft ontwikkelaars ongekende controle over het bouwproces. De engine is namelijk geheel generiek, en de individuele stappen - tasks - die tijdens het bouwproces worden uitgevoerd zijn vastgelegd in .NET- classes. MSBuild biedt een aantal voorgedefinieerde tasks, zoals het compileren van een source-file - Task CSC" - maar het is mogelijk om zelf iedere gewenste stap aan het bouwproces toe te voegen. Een belangrijk gegeven van MSBuild is dat het schema van nieuwe projectfiles in Whidbey volledig gedocumenteerd is, en dus ook met een gewone editor kan worden aangepast. Voorheen mocht officieel alleen Visual Studio zelf deze files bewerken.

```
<Project>
   <Property BinDir="Bin" />
   <Item Type="CSFiles" Include="MSBuildSample.cs" />

   <Target Name="Build" >
      <Task Name="MakeDir" Directories="$(BinDir)"
           Condition="!Exists('$(BinDir)')" />
      <Task Name="CSC" Sources="@(CSFiles)"
                  OutputAssembly="@(CSFiles->'$(BinDir)\%n.exe')" />
   </Target>
</Project>
```

Codevoorbeeld 5

toont een simpel MSBuild-project er uit.

Het toevoegen van een nieuwe task is eenvoudig.Maak een class en implementeer hierop de ITask-nterface. Vervolgens kan deze in de projectfile worden gebruikt, en zal de build-engine de class instantiëren, eventuele properties zetten met de waarden zoals die in de projectfile zijn aangegeven, en vervolgens de 'Execute'-method aanroepen.

Vooral voor grote projecten zal de nieuwe MSBuild-engine een uitkomst zijn. Het bouwproces wordt veel beheersbaarder, zonder dat er gebruik hoeft te worden gemaakt van third party tools zoals NAnt of Visual Build.

Whitehorse

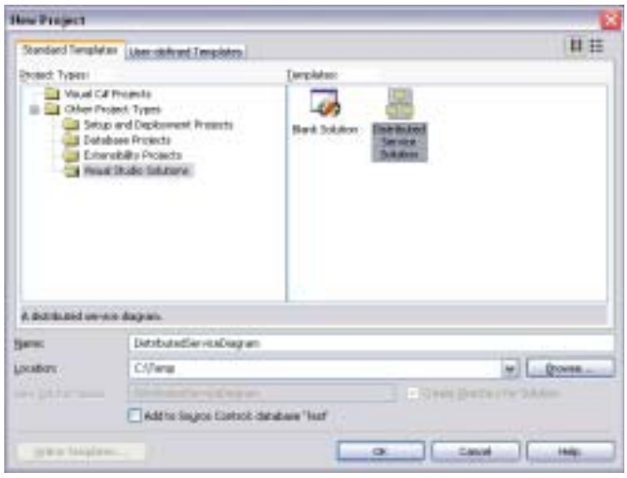
Een intrigerende nieuwe technologie die in Visual Studio 2005 zijn intrede zal doen, is 'Whitehorse'. Er is nog weinig over bekend, afgezien van een aflevering op MSDN TV en een presentatie op de PDC [9]. Whitehorse is een modelleringtool dat gericht is op het ontwerpen van gedistribueerde applicaties. Het bijzondere hierbij is dat Whitehorse zowel de applicatiearchitect als de infrastructuurarchitect in staat stelt gezamenlijk aan het ontwerp te werken.

De infrastructuurarchitect kan een logische weergave maken van de architectuur van het Datacenter, en de softwarearchitect een logische weergave van de architectuur van de Services. Vervolgens kunnen de services worden gemapped op de (logische) servers waarop ze zullen worden gedeployed, en kan Whitehorse controleren of dit gebeurt op een manier die past binnen de restricties van de infrastructuur. Zo zal Whitehorse bijvoorbeeld

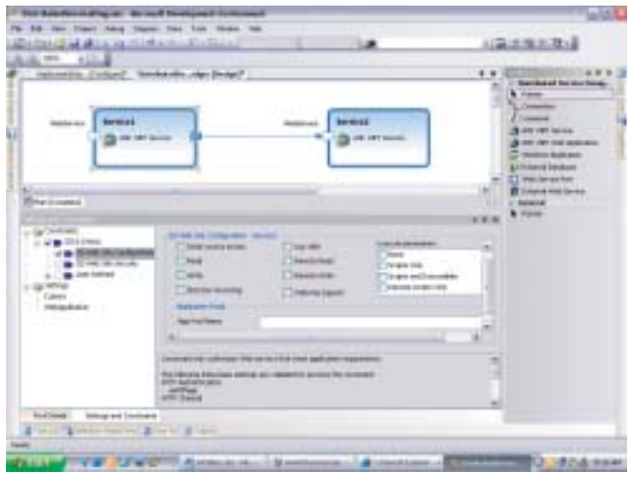
```
// Implementatie van een simpele task
public class SimpleTask : Task
{
   private string _p;

   public string SimpleProperty
      {
        get { return _p; }
        set { _p = value; }
      }

      public void Execute ()
      {
      // Doe iets leuks met de waarde van SimpleProperty
      }
}

// Voorbeeld van het gebruik van deze task.
<Project>
      <Target Name="Build">
      <Task Name="SimpleTask" SimpleProperty="SomeValue" />
      </Target>
</Project>
```

Codevoorbeeld 6

laat een voorbeeld van een simpele task zien.



Afbeelding 14. Het aanmaken van een Distibuted Service Solution



Afbeelding 15. Een diagram van een gedistribueerde applicatie

constateren dat twee services niet kunnen communiceren als het gebruikte communicatieprotocol niet kan worden toegepast, vanwege de configuratie van een firewall in de infrastructuur. Of dat de deployment-eisen van een service niet passen binnen de configuratierestricties van een server. Op deze manier kunnen eventuele problemen in het design in een vroeg stadium worden opgespoord, zodat er tijdens deployment minder onaangename verrassingen zijn.

Mijlpalen

.NET Framework 2.0 en Visual Studio 2005 zijn indrukwekkende mijlpalen in de ontwikkeling van het .NET-platform. De talloze verbeteringen zijn er vooral op gericht om de productiviteit van ontwikkelaars te verhogen. En dat is een thema dat in het huidige tijdperk velen zal aanspreken. In ieder geval hebben ontwikkelaars met deze release weer voldoende gereedschap in huis om de periode te overbruggen tot de volgende grote technologische omwenteling die Longhorn met zich mee zal brengen.

Jan-Willem Overbeek is werkzaam bij Capgemini als software architect en ontwikkelaar. Hij is gespecialiseerd in het ontwikkelen van software voor het .NET platform.
E-mail: janwillem.overbeek@capgemini.com

Noten

[1] Deze zijn te vinden op http://microsoft.sitestream.com/PDC2003/Default.htm.

[2] Het feit dat ondersteuning voor Edit and Continue in de CLR beschikbaar is betekent overigens niet dat het ook in alle .NET talen toegepast wordt. Tot nu toe is alleen van VB.NET bekend dat het Edit en Continue in Visual Studio 2005 ondersteund.

[3] Zie http://msdn.microsoft.com/msdnmag/issues/03/09/NET/ en http://msdn.microsoft.com/msdnmag/issues/03/10/NET/

[4] Generics komen in Java ergens in de loop van 2004 beschikbaar (J2SE V1.5). Boze tongen beweren echter dat de implementatie van Generics in Java niet zo elegant is als die in .NET.

[5] Voor een volledig overzicht, zie de BCL Community Homepage: http://www.gotdotnet.com/team/clr/bcl/

[6] Voor een compleet overzicht, zie http://www.gotdotnet.com/team/mfussell/DataAccessinWhidbey.htm

[7] Zie http://longhorn.msdn.microsoft.com/lhsdk/ndp/daconopathreference.aspx voor meer details.

[8] De nieuwe features van ASP.NET zijn inmiddels al vrij uitgebreid gedocumenteerd, bijvoorbeeld op http://www.asp.net/whidbey.

[9] Zie: http://microsoft.sitestream.com/PDC2003/TLS/TLS345_files/Default.htm en http://msdn.microsoft.com/msdntv/episode.aspx?xml=episodes/en/0040129VSTUDIOAT/manifest.xml

MICROSOFT FXCOP VERSIE 1.30

Laat uw .NET code door FxCop inspecteren. Het maakt niet uit of uw .NET applicatie is geschreven in C#, Visual Basic, C++, Cobol, perl of een andere .NET-taal, FXCop toets uw code aan ontwerprichtlijnen van Microsoft en eventueel uw eigen ontwerprichtlijnen.

Nog niet bekend met FxCop?

FxCop is een tool die door middel van refection, IL-Parsing en callgraph uw .NET assemblies inspecteert op foutieve programmeerpatronen. Voorbeelden van deze patronen zijn:

- Het niet volgen van Design Guidelines (niet volgen van naming conventions)
- Detecteren van globalizatie problemen (bijvoorbeeld vergeten gebruik van CultureInfo)
- Detecteren van vergeten security attributen
- Controle op juist gebruik van Exceptions
- Detecteren van ongebruikte code

In totaal bevat FxCop meer dan 200 regels. Een SDK wordt meegeleverd om uw eigen regels te schrijven. Doordat FxCop op assembly niveau uw programmacode analyseert kunnen alle .NET talen worden geïnspecteerd met behulp van FxCop. Er is zowel een grafische user interface als een command line interface, zodat u FxCop kunt meenemen in het build proces.

Hoe te verkrijgen?

FxCop is downloaden vanaf http://www.gotdotnet.com/team/fxcop/ via deze site kunt u ook op een forum (engels) hulp krijgen bij het gebruik van FxCop en discussieren over FxCop regels