

Agile Business Suite: een 4GL-omgeving voor .NET-ontwikkelaars

ONTWIKKELING, ONDERHOUD EN UITROLLEN VAN COMPLEXE BACKOFFICE-APPLICATIES

Om de beheerlast van de levenscyclus van applicaties te verlichten, worden componenten voor snelle applicatieontwikkeling (RAD), onderhoud en deployment tegenwoordig vaker in geïntegreerde ontwikkelomgevingen (IDE's) ingebouwd. Het ontwikkelen en het onderhouden van grote commerciële applicaties, waarbij het synchroniseren van databases en code noodzakelijk is, is zelfs met dit hulpmiddel niet bepaald gemakkelijk.

Agile Business Suite (AB Suite) van Unisys is een objectgeoriënteerde 4GL-omgeving die ontwikkeling, testen, versiebeheer en deployment-functies in Visual Studio .NET integreert. De generatie van de code, geautomatiseerde databasemi-gratie en integratie met een uitgebreide reeks .NET-technologieën vormen de basis voor een gecentreerd RAD-hulpmiddel om een ontwikkelteam grote, complexe backoffice-applicaties te laten bouwen en onderhouden.

De componenten van de omgeving

AB Suite heeft drie belangrijke componenten: de *Modeler*, de *Builder* en de *Runtime* zoals afgebeeld in afbeelding 1. De Modeler wordt uitgevoerd als een project-plug-in voor Visual Studio, zodat de ontwikkelomgeving vrij natuurlijk oogt en voelt voor .NET-ontwikkelaars. De Builder is ook in Visual Studio geïntegreerd, zodat je code van applicatiemodellen plaatselijk kunt produceren en compileren. Applicaties kunnen gedeployed worden naar Runtime-omgevingen die lokaal of op afstand zijn. De Runtime-component verleent bibliotheeksteun voor opgestelde databaseapplicaties die gebruikmaken van SQL Server of Oracle. De ontwikkelomgeving kan applicaties genereren en uitrollen voor verschillende operationele omgevingen, maar in dit artikel zullen wij ons concentreren op de mogelijkheden die .NET-gebaseerde technologieën bieden.

Het ontwikkelen van je applicatie

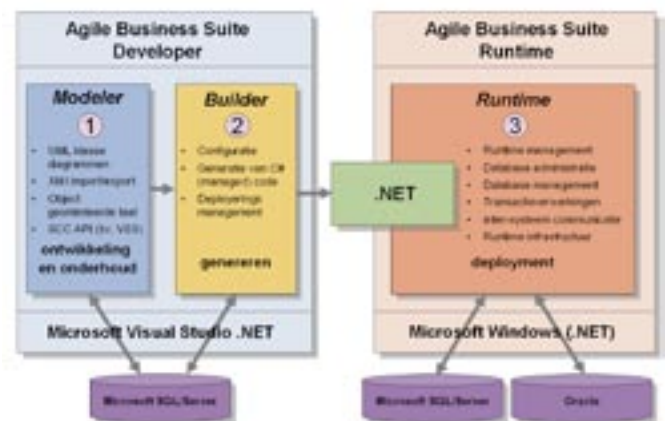
De Modeler gebruikt een SQL Server-database om het applicatiemodel en de broninformatie in op te slaan. Dit in tegenstelling tot de meeste 3GL-hulpmiddelen voor applicatieontwikkeling die reeksen tekstbestanden gebruiken. Hoewel het gebruik van een database - om toegang tot gedeelde middelen te controleren voor kleine teams van ontwikkelaars - zeer nuttig is, zal een versiebeheersysteem noodzakelijk, zijn zodra een team een bepaalde grootte bereikt of de onderlinge afhankelijkheid van de code complexer wordt. AB Suite ondersteunt de standaard SCC API voor integratie met versiebeheerssystemen zoals Visual Source Safe, zodat het werken in teamverband met meer ontwikkelteams aan grote backoffice-applicatieprojecten ongecompliceerd blijft. Om veranderingen in applicaties te handhaven, in het bijzonder bij grote ontwikkelteams, moeten de projectteams vaak volgens strikte ontwikkelnormen werken. Dit betekent niet alleen dat ze een reeks broncodenormen overnemen, waarbij gebruik wordt gemaakt van gemeenschappelijk vertrouwde codepatronen, maar ook dat ze

zoveel mogelijk betrouwbare code zullen hergebruiken. Tegelijk leidt het reduceren van de complexiteit meestal tot kleinere projectrisico's. Om ontwikkelteams te helpen deze doelstellingen voor applicatieontwikkeling te bereiken, heeft Unisys een kader- en datamodel geïntroduceerd.

Kader- en datamodel

Werken met AB Suite-projecten is vrij simpel voor ontwikkelaars met ervaring in Visual Studio .NET en zelfs nog meer voor diegenen met inzicht in objectgeoriënteerde programmeertechnieken. Een project is een model dat één of meer *Segmenten* bevat, waarvan elk een logische afspiegeling van een applicatie is. Het is als model veel gemakkelijker te veranderen dan een systeem dat met traditionele 3GL-programmeertalen is gemaakt. Het veranderen van een logisch element van het model zorgt ervoor dat de Modeler de aangewezen updates aan andere applicatie-entiteiten zoals klassen, interfaces en databasetabellen maakt.

Woordenboeken ('data dictionaries') worden verstrekt, zodat ontwikkelaars gemeenschappelijke datatypes voor systemen gemakkelijk kunnen beheren. Bijvoorbeeld, een datatype 'Saldo' zou een nummer met 8 cijfers en 2 decimalen achter de komma kunnen vertegenwoordigen. Telkens als je met een 'Saldo' in je model moet werken, zou je van deze woordenboekdefinitie gebruik kunnen maken. Naast het definiëren van de eigenschappen van een datatype, kun je zijn waarde ook automatisch controleren door extra



Afbeelding 1. AB Suite-componenten voor .NET

criteria te specificeren. Bijvoorbeeld, de voorwaarde 'this > 0' zorgt ervoor dat slechts waarden groter dan nul geldig zijn. De veranderingen die je in een woordenboekdatatype aanbrengt vloeien door naar alle andere elementen die ervan afhangen, inclusief de databasedefinitie van attributen die op dit datatype worden gebaseerd. *Globale woordenboeken* kunnen worden gebruikt om datatypes met meer applicaties te delen. *Lokale woordenboeken* zijn handig om datatypes binnen een *Segment* toe te passen.

In de Modeler is de klasse de basisbouwsteen van een applicatie. Naast gewone klassen, is een reeks stereotype klassen beschikbaar die krachtiger gedrag vertonen. Met deze klassen zijn de mogelijkheden van de *Runtime*-component te gebruiken. Door veel van de kerninfrastructuur van databaseapplicaties voor transactieverwerkingen (OLTP) te systematiseren, kun je je concentreren op het coderen van de bedrijfsregels. Je hoeft je dus niet bezig te houden met het coderen van het loodgieterwerk. Als je begrijpt hoe elke stereotype klasse werkt, dan kun je zoveel mogelijk van je systeem creëren zonder te veel code te schrijven. Een functioneel prototype van het systeem inclusief een gebruikersinterface kan daarom zeer snel ontwikkeld worden.

Elke klasse kan *Attributen* en *Methodes* bevatten. Attributen zijn variabelen die aan een klasse hangen. Voor elk Attribuut kan je persistentie en/of presentatie definiëren. Attributen met persistentie hebben een vertegenwoordiging in de database. Het databaseschema wordt bepaald op het moment dat het systeem wordt gebouwd door samenhangende Attributen te groeperen in een reeks tabellen. Attributen die een presentatie bevatten, maken deel uit van een externe interface. Deze Attributen kunnen afgebeeld worden in een grafische vormeditor (in Visual Studio). Uiteindelijk kunnen deze vormen in één of meer grafische gebruikersinterfaces (GUI) worden gegenereerd zoals WinForms.

Methoden zijn codefragmenten die deel uitmaken van een klasse en die door het kader of door applicatiecodes worden opgeroepen. Je schrijft je code in een scripttaal genaamd LDL+ (zie afbeelding 2) die de volledige mogelijkheden van objectgeoriënteerde concepten tot zijn beschikking heeft, inclusief inkapseling, samenstelling, overerving en polymorfisme.

Stereotype klassen

De belangrijkste stereotype klassen die voor de bouw van applicaties worden verstrekt, worden *Segmenten*, *Folders*, *Ispecs* (interfacespecificaties) en *Rapporten* genoemd. Zoals eerder is vermeld, is een *Segment* een logische vertegenwoordiging of een model van de applicatie. Een *Folder* is een nuttige constructie voor de groepering van samenhangende klassen, op welke manier dan ook. Je zou de applicatie bijvoorbeeld kunnen onderverdelen in een reeks

functionele gebieden. *Ispec*- en *Rapport*-klassen zijn opgeslagen in *Segmenten* en *Folders*.

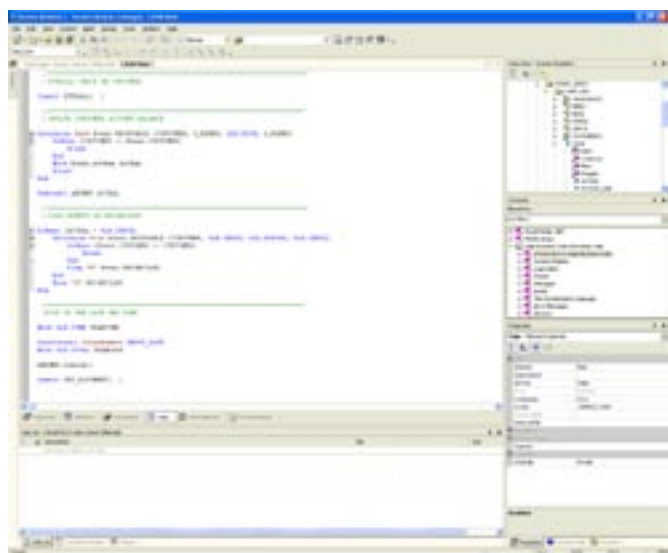
Ispecs zijn klassen die databasepersistentie en/of externe interfaces bepalen. Een *Ispec* met attributen met persistentie heeft een onderliggende databasetabel om deze attributen in op te slaan. Een *Ispec* die attributen met presentatie-eigenschappen heeft zal een externe interface definiëren die ook een GUI-vertegenwoordiging zou kunnen hebben. *Ispecs* die attributen met persistentie en presentatie-eigenschappen hebben, zullen beide een databasetabel hebben en een externe interface. Alle gegenereerde *Ispecs* maken deel uit van de OLTP-componenten van de applicatie. *Ispecs* met attributen die presentatie-eigenschappen hebben, vertegenwoordigen een uitvoerbaar transactietype.

Om ontwikkelen in AB Suite eenvoudig te maken, verleent de *Runtime*-component ondersteuning voor een kader dat de levenscyclus van de *Ispec*-transactie bestuurt. Belangrijk voor ontwikkelaars is dat het gebruik van dit kader garandeert dat data-integriteit automatisch wordt gehandhaafd, zonder expliciet de grenzen van de transactiecode te moeten aangeven en zonder je druk te maken over andere gelijktijdig lopende processen. Om zo'n garantie te kunnen geven beschikt *Ispecs* over drie vooraf gedefinieerde methoden die door het kader op strategische punten in de transactielevenscyclus worden opgeroepen (zie afbeelding 3).

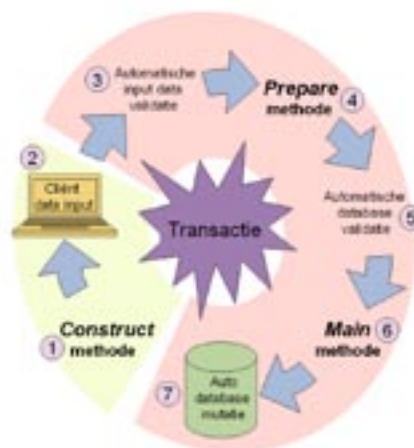
Deze methoden worden *Construct*, *Prepare* en *Main* genoemd. Je kunt deze methoden eenvoudig invullen met applicatielogica en het kader zal automatisch de code op het correcte moment tijdens de uitvoering van de transactie oproepen. De *Construct*-methode wordt opgeroepen door het kader om een objectinstantie te initialiseren. De *Prepare*-methode wordt opgeroepen nadat de gebruikersinput ontvangen is en dient om de inputgegevens te controleren. Hier kun je complexere controles uitvoeren. Onder bepaalde voorwaarden roept het kader de *Main*-methode op die normaal gesproken het grootste deel van het transactiewerk bevat. Je bent niet beperkt tot het gebruiken van deze vooraf bepaalde methoden alleen, je kunt zo veel van je eigen methoden schrijven als je wilt.

In tegenstelling tot *Ispec*-transacties is een gegenereerd *Rapport* een zelfstandig uitvoerbaar proces dat geen deel uitmaakt van het online systeem, hoewel het wel dezelfde database deelt. Het is een batchproces dat kan worden gebruikt om de database te lezen en bij te werken. Rapporten kunnen verscheidene expliciete database-transacties bevatten die grotere flexibiliteit toestaan voor bulk-leesoperaties en updateverrichtingen. In tegenstelling tot *Ispec*-transacties die een bepaalde levenscyclus hebben, begint een *Rapport* eenvoudig zijn uitvoering in de *Main*-methode.

De *Runtime*-component verleent diensten zoals automatisch herstarten en herstellen voor Rapporten. Je hoeft je niet bezig te houden met het coderen hiervan, dat doet de infrastructuur



Afbeelding 2. Het schrijven van code in LDL+



Afbeelding 3. Transactielevenscyclus

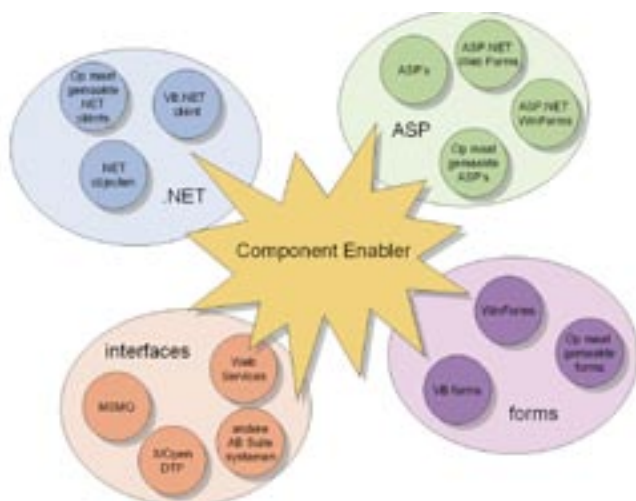
van AB Suite zelf. Nogmaals, omdat je het loodgieterwerk niet hoeft te coderen, kun jij je concentreren op het coderen van de bedrijfsregels. Het automatisch herstellen van gefaalde langlopende processen betekent dat een opnieuw gestart Rapport zijn database en bestandsverwerking vanaf het punt van de laatste succesvolle transactie kan beginnen in plaats van het allereerste begin.

Het bouwen en uitrollen van je applicatie

Zodra de applicatie is ontwikkeld, zal het op een server moeten worden uitgerold. De ontwikkelomgeving houdt deze informatie met de logische modeldefinitie in zijn SQL Server-database. De deployment-informatie wordt onderhouden door middel van de Visual Studio .NET-omgeving. Je kunt veelvoudige reeksen eigenschappen van de configuratie van de deployment opslaan, waarbij op het punt van deployment één reeks kan worden geselecteerd. AB Suite verstrekt een één-klik mechanisme voor het uitrollen van applicaties, waarbij de eerste fase wordt geleid door de Builder-component.

De Builder vertaalt de informatie van het modelontwerp in een lopende databaseapplicatie en maakt daarnaast clients om toegang te krijgen tot die applicatie. Tijdens dit proces combineert de Builder configuratie-informatie met structurele informatie en logica in het model. De codegeneratie gebruikt een reeks templates, die een standaard codestructuur voor elke gestereotypeerde klasse verstrekt. De templates bevatten triggers die door de Builder gebruikt worden om specifieke codefragmenten te genereren die corresponderen met de logica en eigenschappen van ontwikkelde applicatieklassen. De bouw van grote, complexe applicaties vereist vaak dat de tijden voor het genereren en uitrollen worden geminimaliseerd. Daartoe verstrekt de ontwikkelomgeving een veranderinganalysemechanisme. Dit vergelijkt de huidige staat van het model met de bestanden uit de vorige versie van de applicatie om zo te bepalen welke elementen moeten worden gegenereerd. De Builder slaat de recentste gegenereerde bestanden voor elke configuratie op, zodat hij slechts de veranderde delen van het model opnieuw moet genereren. De ontwikkelomgeving genereert (managed) C#-code voor elke Ispec en Rapport. Het Runtime-componentkader vormt de basis voor een service oriented architecture (SOA) waarin de Ispec-componenten zijn gestopt. Het toepassen van het optionele product Component Enabler, maakt het mogelijk gebruik te maken van uitgebreide .NET-presentatie- en integratietechnologieën; zie afbeelding 4. Er wordt bijvoorbeeld automatisch code gegenereerd voor onder andere ASP.NET, webservices en MSMQ voor gekozen delen van de applicatie. Aangezien de interfaces automatisch gegenereerd worden uit het model, is er verder geen onderhoud te verrichten wanneer het model verandert.

De gegenereerde C#-projectbestanden worden gecompileerd en



Afbeelding 4. Component Enabler

verbonden met de bibliotheken van de Runtime-componenten en worden aan het deploymentproject toegevoegd. Het deploymentproject wordt omgebouwd tot een standaard MSI- deploymentpakket. AB Suite *Oplossingen* kan een aantal projecten bevatten, waarbij het niet alleen AB Suite-projecten hoeven te zijn. Een AB Suite Oplossing kan bijvoorbeeld een C#-project en een AB Suite-project bevatten en deze over meer serversystemen uitrollen. Je zou ook moeten opmerken dat de toegang tot AB Suite van andere .NET-talen (en andersom) wordt ondersteund, wat betekent dat je het juiste hulpmiddel voor het juiste traject kunt kiezen.

Het synchron uittrollen van code- en database-wijzigingen

Eén van de moeilijkste taken van de bouw van grote, complexe transactieapplicaties is het handhaven en uitbreiden van de applicatie, vooral wanneer de wijzigingen aan het databaseschema verrijkt moeten worden. Het handhaven van synchronisatie tussen de datastructuren en het databaseschema die door de transactieprogramma's worden gebruikt, kan problemen geven die gewoonlijk pas bij het testen van applicaties worden geïdentificeerd. Hoewel er hulpmiddelen in IDE's bestaan om afhankelijkheden van programma's te ontdekken, is het beheren van veranderingen in grote databaseapplicaties vaak een taak waarbij gemakkelijk fouten kunnen ontstaan. Nadat bepaald is hoe het databaseschema veranderd moet worden, vereist de deployment meestal het handmatig veranderen van het schema en de datamigratie. In grote productieomgevingen zijn dit vaak procedures waarbij gemakkelijk zeer ernstige fouten kunnen optreden.

Dit is waar deze 4GL-technologie in uitblinkt. Je hoeft je niet druk te maken om de afhankelijkheden tussen de code en de database, en het productieteam hoeft zich ook geen zorgen te maken over de wijzigingen van het databaseschema en de datamigratie. AB Suite regelt dit alles automatisch door op de hoogte te blijven van ieder attribuut, methode en klasse die gebruikt of veranderd is tussen de opeenvolgende deployments van de applicatie. De ontwikkelomgeving weet welke applicatiecomponenten opnieuw moeten worden gegenereerd, gecompileerd en in gebruik genomen. Door het huidige deployment-databaseschema te vergelijken met het nieuw op te stellen databaseschema, kan AB Suite de kolommen en de tabellen veranderen, toevoegen of verwijderen. De ontwikkelomgeving beheert ook automatisch de datamigratie, waarbij tabellen opnieuw geladen worden indien dit noodzakelijk is. Van tevoren wordt ook aangegeven wat het effect van de deploymentactie zal zijn, zodat je begrijpt welke activiteiten moeten gebeuren.

Een fluitje van een cent

Het ontwikkelen, handhaven en opstellen van grote, complexe backoffice-applicaties is geen gemakkelijke taak. AB Suite biedt een objectgeoriënteerde 4GL-omgeving die deze taak vereenvoudigt. Hierbij kunnen projectteams zich concentreren op het schrijven van de bedrijfsregels in plaats van zich druk te moeten maken over het loodgieterwerk. Dit vermindert de omvang van het project aanzienlijk en daarbij ook de projectrisico's. De applicatielevenscyclus van AB Suite, gekoppeld aan de faciliteiten van de automatische codegeneratie en integratie van de .NET-technologieën zoals ASP.NET, webservices en MSMQ, maakt het bouwen en handhaven van complexe backoffice op .NET gebaseerde applicaties een fluitje van een cent.

Michael P. Leznar is directeur en hoofdconsultant bij Leznar Consulting (www.leznarconsulting.com). Hij heeft zich beziggehouden met alle facetten van software-engineering (ontwikkeling, ondersteuning en management) inclusief product- en projectmanagement en het verlenen van technisch advies. Zijn e-mailadres is mike@leznarconsulting.com

Referenties

www.agilebusinessuite.com