Migratie Designer naar JDeveloper

Project bij Centraal Boekhuis

Recentelijk heeft Oracle aangegeven de functionaliteit van ontwikkeltool Designer te bevriezen en positioneert JDeveloper nu als de ontwikkelomgeving voor de toekomst. Deze annoncering is voor Centraal Boekhuis aanleiding geweest een migratie voor te bereiden van Designer naar JDeveloper. In dit artikel wordt ingegaan op de motivaties voor dit 'voorhoedeproject' en de wijze waarop Centraal Boekhuis in samenwerking met IT-eye dit project heeft uitgevoerd. Hierbij heeft de mogelijkheid om JDeveloper te voorzien van extra functionaliteit een belangrijke rol gespeeld.

Oracle biedt op dit moment een tweetal producten voor het ontwikkelen van software: Designer en JDeveloper. Designer bestaat al sinds 1995 en is vooral gericht op het genereren van Oracle*Forms-applicaties. JDeveloper is een ontwikkelomgeving voor Java/J2EE-applicaties. Medio 2004 heeft Oracle in een Statement of Direction (SoD) aangegeven dat er geen nieuwe functionaliteit meer wordt toegevoegd aan Designer. Deze stap werd alom geïnterpreteerd als de aankondiging van het einde van Designer, en heeft ertoe geleid dat veel organisaties zich afvroegen[1]:

- "Is JDeveloper de opvolger van Designer?"
- "Biedt JDeveloper op termijn dezelfde mogelijkheden als Designer?"
- "Word ik geacht twee ontwikkelomgevingen naast elkaar operationeel te houden?"
- "Moet ik al mijn Forms-applicaties gaan migreren naar Java, en zo ja, komt daar dan een migratie-tool voor?"

De bovenstaande vragen zijn niet helemaal terecht binnen de context van het afgegeven SoD: "het einde van Designer is in zicht". Want hoewel Designer veel informatie bevat over de applicaties die ermee zijn gebouwd, zijn die applicaties zelf voornamelijk in Forms ontwikkeld. De genoemde vragen zouden slechts terecht zijn geweest als Oracle het einde van Forms had aangekondigd. Een vraag die de revue had moeten passeren, zou kunnen luiden: "wat moet ik doen met de *administratie* van mijn Oracle (Forms)-software?"

Achtergrond

Centraal Boekhuis (CB) maakt gebruik van Designer voor het ontwikkelen van (webgebaseerde) Forms-applicaties voor intern gebruik en van JDeveloper voor de ontwikkeling van Java/J2EE-applicaties voor gebruik door haar klanten. De aankondiging van het einde van Designer heeft CB aangezet om de volledige administratie van Designer naar JDeveloper te willen migreren. Daarvóór echter waren er ook al redenen voor CB om de positie van Designer voorzichtig ter discussie te stellen. De belangrijkste hierin zijn het ontbreken van een goede voorziening voor versiebeheer en (in combinatie met de geldende werkprocedures binnen CB) de op onderdelen gebrekkige productiviteit.

Ten aanzien van versiebeheer kondigde Oracle eveneens aan niet langer in SCM te investeren. SCM is een door Oracle ontwikkeld versiebeheersysteem gebaseerd op de Oracle database. Oracle richt zich in plaats daarvan volledig op het meest gebruikte versiebeheersysteem CVS[2].

Een poging van CB in 2001 om SCM te gaan inzetten voor versiebeheer is destijds gestrand op de hoge mate van complexiteit van SCM en de schier onuitputtelijke lijst randvoorwaarden die moet zijn ingevuld alvorens er eenvoudig (productief, rendabel) mee kon worden gewerkt. Daarentegen biedt JDeveloper een prima integratie met CVS. CB gebruikt dat al bij de ontwikkeling van Java/J2EE applicaties. Een consequentie van het willen werken met CVS voor versiebeheer is echter wel dat dan alle objecten als bestand (ASCII of binair) beschikbaar moeten zijn. Een combinatie met Designer is daarin geen alternatief.

Bij het doorspreken van het idee om de volledige inhoud van Designer naar JDeveloper te migreren werd al snel duidelijk

1 Zie ook http://www.oracle.com/technology/products/designer/FAQ_Schedule_2004.htm
2 Zie bijvoorbeeld http://www.nongnu.org/cvs/

hoe de eindsituatie er uit zou gaan zien. Maar dat maakte ook een aantal gevolgen zichtbaar:

- Alle software (code en documentatie) wordt uit Designer onttrokken waarbij per object een ASCII-bestand wordt aangemaakt waarin alle informatie over een object wordt samengevoegd. Deze bestanden worden vervolgens met JDeveloper onderhouden. Daarbij wordt CVS gehanteerd als versiebeheersysteem.
- Aan JDeveloper worden enkele functies toegevoegd om de *code* en *documentatie* goed te kunnen scheiden. Er wordt tevens een voorziening voor het modulenetwerk aangebracht. Hiervoor zal een extensie ontwikkeld worden (zie kader). Het modulenetwerk bij CB omvat meer dan het modulenetwerk van Designer. Het bestaat naast de relaties tussen de modules ook uit de relaties met en tussen alle andere objecten.
- Met JDeveloper wordt *dezelfde* software gemaakt en onderhouden als met Designer, dus voornamelijk SQL- en PL/SQL-code, Forms, Reports en een klein deel C-software.

IDE en Extensie SDK

De IDE Oracle JDeveloper is standaard voorzien van een complete verzameling ontwikkelfunctionaliteit. Toch blijft er altijd behoefte aan functionaliteit die (nog) niet in JDeveloper geboden wordt. Bij het ontwerp van JDeveloper is op voorhand rekening gehouden met deze onverzadigbare functionele honger door JDeveloper op te delen in een IDE-basisraamwerk en een Extensie Software Development Kit (SDK).
Vrijwel alle functionaliteit die de Java-ontwikkelaar ervaart als standaard JDeveloper is door Oracle ontwikkeld op basis van de SDK en toegevoegd aan het IDE-basisraamwerk. Dezelfde Extensie SDK is ook via OTN beschikbaar. Hiermee kan JDeveloper door iedereen op een vergelijkbare wijze worden voorzien van nieuwe functionaliteit door zelf een extensie te ontwikkelen.
Op dit moment is Oracle bezig om een standaard te ontwikkelen voor het toevoegen van functionaliteit binnen een willekeurige Java-IDE op basis van extensies.

De gevolgen zijn dat in de eindsituatie er geen gebruik meer kan worden gemaakt van upper-case voorzieningen (bijvoorbeeld ERD) en er geen Forms meer kunnen worden gegenereerd, of hergegenereerd. Er wordt uiteraard wel gebruik gemaakt van Forms Builder et cetera.
Het begrip verhuizing als metafoor voor de migratie van Designer naar JDeveloper komt goed van pas. Veel van de activiteiten die vooraf en tijdens de migratie een rol spelen, komen overeen met zaken die bij een normale verhuizing ook aan de orde komen, zoals inventariseren wat je allemaal hebt, inpakken van spullen die je mee wilt nemen, weggooien van zaken die je toch niet meer gebruikt, en oppoetsen van alles dat je handen passeert. In dit rijtje mag het belangrijkste aspect van een verhuizing niet ontbreken: je nieuwe huis is mooier dan het oude en je voelt je er ook direct in thuis. In de volgende paragrafen passeren de startsituatie, de voorbereiding en de uitvoering de revue.

Startsituatie

Bij een verhuizing zijn eigenschappen van de bestaande woning vaak bepalend voor de keuze van de nieuwe woning. Altijd is de bestaande situatie mede bepalend voor de argumenten voor een gemaakte keuze. Naar analogie hiervan, kan het zijn dat de migratie zoals CB deze heeft ingezet niet voor iedere organisatie toepasbaar is. Daarom besteden we eerst kort aandacht aan de startsituatie bij Centraal Boekhuis.
De 'softwareplas' van CB omvat in tachtig applicaties onder anderen zo'n 4.000 bedrijfsprocedures, 3.000 entiteiten, 1.500 structural rules, 2.000 tabellen, 3.000 views, 2.000 Forms en Reports en meer dan 11.000 PL/SQL-sources. Relatief gezien weinig Forms en veel back-end software in PL/SQL. De Forms-applicaties zijn dan ook erg 'dun' en bevatten bijna geen code. Tevens geldt dat Centraal Boekhuis wat betreft software-ontwikkeling voornamelijk in een beheerfase zit. In 2001 is een omvangrijke migratie van mainframe-programmatuur naar Oracle-programmatuur afgerond. Sindsdien worden jaarlijks voornamelijk projecten uitgevoerd waarin omvangrijke en minder omvangrijke aanpassingen aan de bestaande systemen worden gerealiseerd. Voor alle taken (architectuur, informatie-analyse, projecten en beheer) is een team van ruim twintig medewerkers actief.
Bij het gebruik van Designer is al enige tijd geen sprake meer van hergenereren van Forms-applicaties. In een eerder stadium is vastgesteld dat in de CB situatie de kosten van het hergenereerbaar onderhouden van de Forms-applicaties niet lager maar zelfs hoger zijn dan de kosten van het handmatig aanbrengen van wijzigingen. Dit komt met name door de wisselingen in benodigde property's die vastgelegd moeten worden in de opeenvolgende versies van Designer. Hierbij moet worden opgemerkt dat CB relatief laat overstapte van Designer 1.3.2 naar 6i en inmiddels werkt met Designer 10g. Headstart wordt wel toegepast, maar dan wel de oudere versies 3.2.1 en 3.4.1. Het loslaten van Designer zal hier derhalve alleen een 'verliespunt' opleveren ten aanzien van het *initieel* genereren van Forms.

Er is door CB aandacht besteed aan alternatieven waarbij het migreren vanuit Designer wel het meest aantrekkelijke blijkt. Alternatieven zouden zijn (1) om niets te doen, en te wachten tot Oracle komt met een opvolgingsstrategie voor Designer,

(2) het migreren van alle Forms-applicaties naar J2EE of (3) het kiezen voor een ander front-end tool in plaats van Forms. Alternatieven 2 en 3 zijn oplossingen voor een ander probleem dan het einde van het Designer tijdperk en alternatief één geeft op geen enkele wijze invulling aan de al eerder genoemde ambities van CB om over versiebeheer en een meer productief tool te beschikken.
Tot slot zijn er een aantal uitgangspunten benoemd die bij het voorbereiden van de migratie in acht dienen te worden genomen. Zonder al te veel uitleg sommen we die hier kort op:

- De migratie vindt plaats naar het Oracle-product JDeveloper, er vindt géén selectieproces plaats naar wat de beste IDE is om naar te migreren.
- We gaan werken met CVS en accepteren de consequenties van het werken met bestanden en zoeken zonodig oplossingen binnen de context van JDeveloper.
- Indien nodig breiden we de functionaliteit van JDeveloper uit, maar we gaan onder geen beding Designer nabouwen (hetgeen theoretisch wel mogelijk is.....)
- De minimale functionaliteit moet invulling geven aan het adequaat kunnen beheren van alle noodzakelijke property's van objecten uit Designer plus een adequaat onderhoud van het modulenetwerk.

Voorbereiding

Een goede verhuizing staat of valt met een goede voorbereiding. Elke verhuizing is een prima gelegenheid om eigendommen nog eens te toetsen op hun waarde. Sommige zul je weggooien en sommige zul je niet meer in de kamer plaatsen maar ergens bovenop zolder. In de voorbereidingsfase van de migratie zijn een drietal trajecten gevolgd: (1) Inventarisatie van de data en het ontwikkelen van migratiescripts, (2) toevoegen van gewenste functionaliteit in JDeveloper, en (3) het bepalen van de impact op de werkprocessen. Een reeks testen rondde deze fase af. Hierdoor moet de uitvoeringsfase kunnen worden gereduceerd tot een zuiver omschakelmoment.

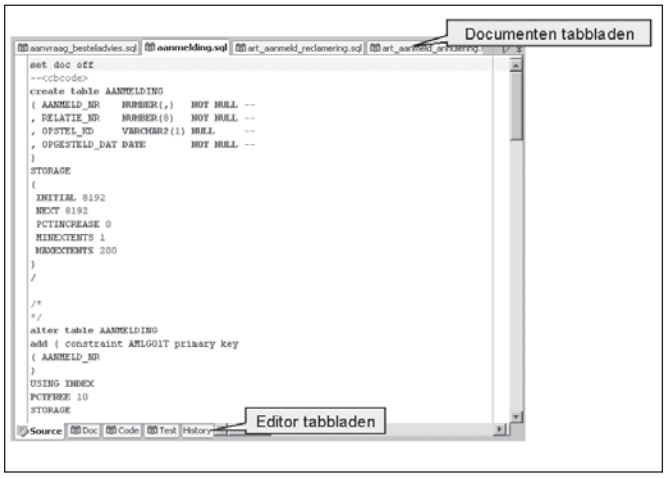
Inventarisatie van de data en het ontwikkelen van migratiescripts
Het samenstellen van de migratiescripts is het aandachtsgebied dat het meest tot de verbeelding spreekt. Iedereen heeft een eigen beeld van wat er in Designer zit en wat daarvan actief wordt gebruikt en onderhouden. Daarnaast moet ook worden bedacht hoe de informatie in ASCII-bestanden kan worden opgenomen.
De eerste actie is het inventariseren van de te migreren data uit Designer. Zo is per objecttype bepaald welke relaties er zijn met andere objecttypen en tevens welke property's daadwerkelijk gemigreerd moeten worden. In dit stadium is veel aandacht besteed aan de integratie van entiteiten en tabellen.
De volgende stap is het opstellen van voorbeelden voor de nieuwe vormgeving in losse bestanden. Hierbij is veel gediscussieerd over het al of niet migreren naar een volledige XML-notatie. Daarbij is de volgende vraag relevant: "Waar halen we de nul-stand vandaan?" Van de productie-, van de test- of van de ontwikkelomgeving? Er is gekozen om de productieomgeving als uitgangspunt te hanteren en aan te vullen met de documentatie uit Designer. Op deze manier kan een mooie schone startsituatie worden gecreëerd voor het werken in een omgeving met versiebeheer! Deze keuze heeft wel impact op de complexiteit van de migratiescripts.
De migratie van de inhoud van Designer lijkt behoorlijk overzichtelijk. In Designer zijn alle objecten netjes geordend en de property's zijn allemaal eenduidig in hun betekenis. In de praktijk doen zich toch wat problemen voor. Zo heeft CB bijvoorbeeld de afgelopen jaren onder het Designer-label *Business Functions* een vijftal verschillende objecttypen ondergebracht. Deze willen we nu weer van elkaar scheiden. Maar naamgevingsconventies of afspraken van wel en niet gevulde property's blijken niet sluitend om het onderscheid te maken. Het vereist enig handwerk om de inhoudelijke migratie volledig en juist uit te voeren. Bij het uitwerken van de migratiescripts zijn oplossingen gecreëerd voor specifieke Designer-issues zoals het delen (sharen) van objecten, domeinreferentie, en het kunnen vermelden in welke schema's een object is geïnstalleerd.

Toevoegen van gewenste functionaliteit in JDeveloper
Ondanks het feit dat Oracle JDeveloper standaard voorzien is van een complete verzameling ontwikkelfunctionaliteit blijft er altijd behoefte aan functionaliteit die (nog) niet in Oracle JDeveloper geboden wordt. Deze behoefte kan bijvoorbeeld komen vanuit softwareleveranciers die hun productspecifieke functionaliteit geïntegreerd beschikbaar willen stellen binnen JDeveloper. Maar ook van ontwikkelaars die bepaalde ontwikkelfunctionaliteit missen of anders willen hebben. Of van organisaties die specifieke ondersteuning van een werkproces beschikbaar willen stellen binnen Oracle JDeveloper. Tot deze laatste categorie behoort Centraal Boekhuis. Voor Centraal Boekhuis heeft IT-eye een extensie ontwikkeld waarmee een geselecteerd aantal gebruikersfuncties uit Designer overgenomen wordt binnen Oracle JDeveloper.
Twee belangrijke ontwerpbeslissingen die genomen zijn voor de verhuizing van Designer, zijn dat de objectinformatie uit de Designer Repository in ASCII-bestanden zal worden vastgelegd, per object een bestand, en dat het modulenetwerk, inclusief CRUD informatie uit het functioneel model, vanwege het grote aantal relaties in de database zal worden overgenomen.
Hiervoor is een specifiek datamodel ontworpen die door de migratiescripts gevuld wordt.

De wijze waarop de onderhoudsfuncties aan de eindgebruiker beschikbaar worden gesteld sluit naadloos aan op de look-and-

Extensies

Het ontwikkelen van een extensie is vergelijkbaar met het ontwikkelen van een willekeurige Java applicatie. Het ontwikkelen van een extensie kent bijvoorbeeld evenals het ontwikkelen van een J2EE-applicatie wel een specifiek programmeermodel. Dit programmeermodel is beschreven in de Extensie SDK-documentatie. Tevens zijn een groot aantal voorbeeldextensies (inclusief broncode) opgenomen in de Extensie SDK.
Een ervaren Java-ontwikkelaar kan uit de API-beschrijving van de Extensie SDK het programmeermodel al afleiden uit de verschillende ontwerppatronen die toegepast zijn, zoals Singleton, CoR, Subject/Observer of Factory.



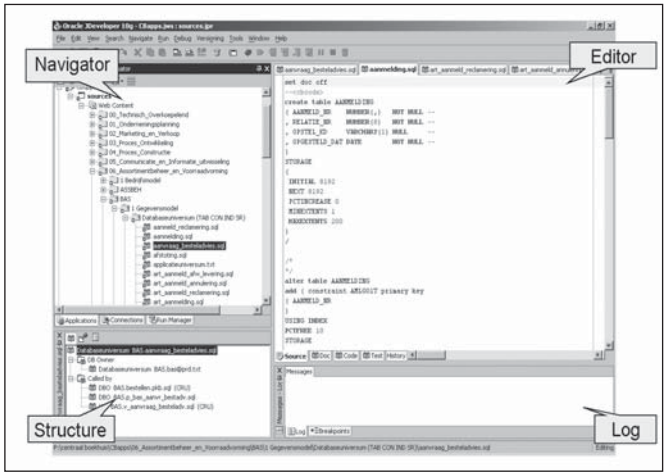
Een object dat vanuit de Navigator geopend wordt in de Editor.

feel van JDeveloper. Hierdoor is een intuïtieve en uniforme interface verkregen. Het is de taak van de extensie deze informatie geïntegreerd binnen JDeveloper te presenteren en de integriteit tussen beide informatiebronnen zoveel mogelijk te waarborgen. De extensie maakt voor de interface gebruik van de volgende IDE vensters:

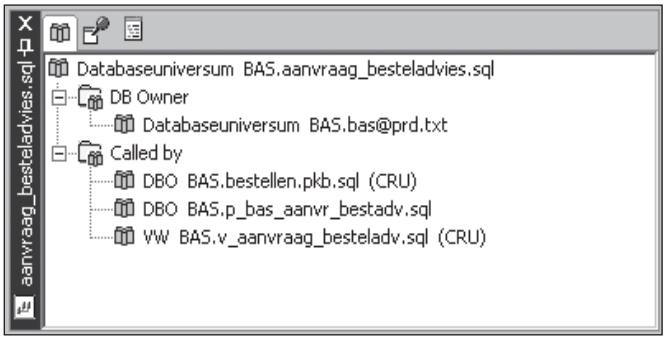
- Application Navigator Venster
- Editor Venster
- Structure Venster
- Log Venster

Binnen de Navigator worden de objecten aangeboden die zich op het bestandsysteem bevinden. De objecten zijn volgens een CB-structuur ondergebracht. Om deze structuur ongewijzigd over te nemen in de Navigator worden de objectbestanden in de logische projectmap 'webcontent' aangeboden aan de eindgebruiker (zie afbeelding 1). Een object dat vanuit de Navigator geopend wordt in de Editor, wordt via een documenten-tabblad en een vijftal editor-tabs getoond aan de eindgebruiker (zie afbeelding 2).

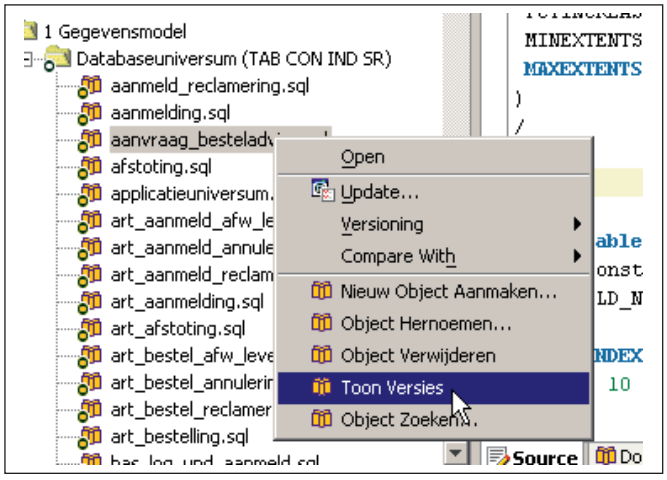
Ieder Editor-tabblad toont informatie over één en hetzelfde object vanuit verschillende perspectieven: documentatie, definitie, test en versiebeheer. Tevens is het volledige document als extra tabblad (source) toegevoegd. Aanpassingen in één van editors worden automatisch gesynchroniseerd met de andere editors. In elk objectbestand is middels markeringen (zoals -<cbcode>) vastgelegd welke informatie in welk Editor-tabblad moet worden getoond.
Van het modulenetwerk worden alleen de directe relaties van het geselecteerde object getoond in het Structure-venster. In zekere zin is dit venster ook weer een perspectief voor het geselecteerde object. Vanuit dit venster kan vervolgens een direct gerelateerd object worden geselecteerd. Op deze wijze kan door het modulenetwerk worden genavigeerd. Het terugnavigeren kan zowel vanuit het Structure-venster als de Navigator. Tot slot worden in het logvenster eventuele (fout)meldingen getoond die betrekking hebben op het gebruik van de extensie.



De extensie maakt voor de interface gebruik van vier IDE-vensters.



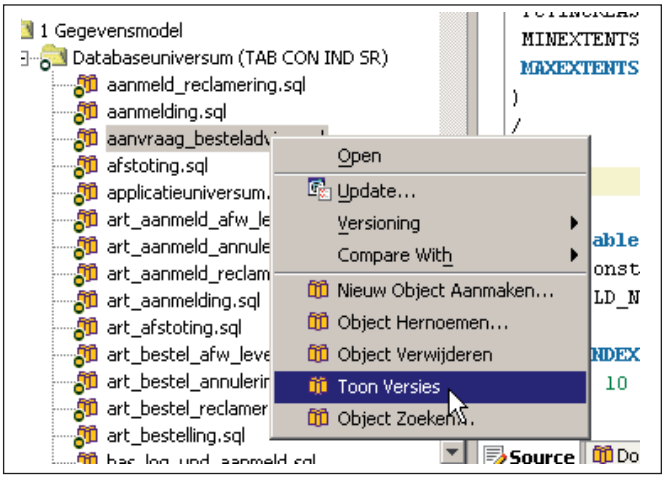
De directe relaties van het geselecteerde object worden getoond in het Structure-venster



In de navigator kan de versie van een object in de verschillende omgevingen worden opgevraagd

Een object bevindt zich in de productie-, test- en ontwikkelomgeving. Vanwege het evolutionaire karakter van objecten kan de versie van een object in de genoemde omgevingen verschillen. In de navigator kan de versie van een object in de verschillende omgevingen worden opgevraagd middels het contextmenu van een object. In een dialoog worden vervolgens de actuele versies getoond in de verschillende omgevingen. Deze informatie wordt rechtstreeks uit de database gehaald op basis van de versie-informatie die door CVS aan het object is toegevoegd. Met deze functie is het, in vergelijking met Designer, erg makkelijk geworden om op ieder moment inzicht te krijgen welke versie zich in de database bevindt.

Binnen JDeveloper is het mogelijk nieuwe objecten en relaties te definiëren, bestaande objecten te verwijderen en bestaande relaties te wijzigen of verwijderen. Deze functies worden de onderhoudsfuncties genoemd. Het onderhouden van objecten gebeurt binnen de Navigator en het onderhouden van relaties binnen het Structure-venster.

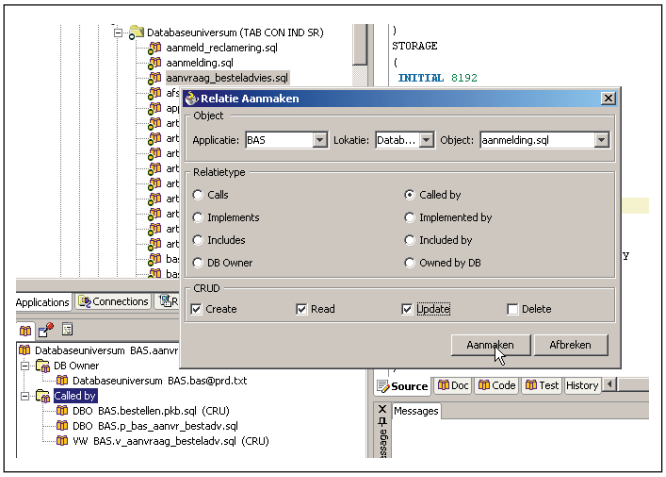


Het aanmaken van een object

Het aanmaken van een object bestaat uit het aanmaken van het object in de database ten behoeve van het modulenetwerk, het aanmaken van een bestand op het filesysteem op basis van een voorgedefineerde template en het opnemen van het object in het versiebeheersysteem. Deze operaties worden voor de eindgebruiker als één transactie uitgevoerd door de extensie. Het aanmaken, wijzigen en verwijderen van een relatie gebeurt binnen het Structure-venster. Voor het aanmaken en wijzigen van een relatie wordt een dialoog getoond waarbinnen de eindgebruiker kan aangeven met welk object een relatie moet worden gelegd, het type relatie dat van toepassing is eventueel aangevuld met CRUD-informatie. Het aanmaken van een relatie vindt alleen plaats in de database.

Impact op werkprocessen

Vóórdat de migratie kan worden uitgevoerd zal ook de impact op de werkprocessen moeten worden bepaald en verwerkt. De beoogde productiviteitswinst zal voor een deel worden geboekt door werkprocessen zoals het proces van kwaliteitsreview of het proces van opleveren te kunnen vereenvoudigen. De migratie naar JDeveloper als IDE zal sowieso een productiviteitswinst creëren of zoals een ontwikkelaar het zei "na een dagje werken bleken er heel veel handigheden in JDeveloper te zitten die het werk wel erg plezierig maken". Het gaan werken met versiebeheer heeft de grootste impact op alle werkprocessen. Het is van belang om vooraf na te denken over de manier waarop alle sources als losse file worden georganiseerd. En ook over de toepassing van een optimistisch of pessimistisch locking-scenario. Omdat CVS binnen CB al werd ingezet bij de J2EE-softwareontwikkeling was al veel bekend over de mogelijkheden. Nu een veel grotere groep ontwikkelaars er mee aan



Het aanmaken, wijzigen en verwijderen van een relatie gebeurt binnen het Structure-venster.

de slag gaat, is een verdieping van die kennis wel noodzakelijk. Een tweede belangrijk issue is het creëren van een alternatief voor het initieel kunnen genereren van Forms. Als uiteindelijk Designer niet meer beschikbaar is, zullen Forms op basis van een brede set templates worden ontwikkeld. Voor zeer complexe Forms zijn deze templates niet toereikend, maar binnen de context van CB komen die weinig voor en rechtvaardigen die op dat moment een toename van de ontwikkelinspanning. Tot slot heeft de migratie impact op alle kwaliteitshandboeken en standaarden. Daarin staan veel verwijzingen naar Designer en deze moeten worden aangepast aan de nieuwe omgeving. Deze klus is niet moeilijk, uitsluitend tijdrovend.

Uitvoering

Verhuizen doe je niet zomaar. Je kiest weloverwogen het moment waarop je overgaat. Zo is ook de situatie bij Centraal Boekhuis. Medio augustus 2005 zijn alle testen afgerond en is de finale go/no-go beslissing genomen. De feitelijke migratie, die ten behoeve van de tests al enkele malen is uitgevoerd, wordt gepland en de ontwikkelaars worden voorbereid op de nieuwe ontwikkelomgeving.
Het migreren van Designer naar JDeveloper is ingrijpend voor de ontwikkelaars. Althans op het eerste gezicht. Het (voor velen al bijna tien jaar) vertrouwde uiterlijk en de gedragingen van Designer valt weg en daar komt het nog relatief onbekende JDeveloper voor terug. De makers van JDeveloper hebben niet voortgebouwd op wat de makers van Designer ooit gecreëerd hebben. Zij hebben autonoom een moderne en professionele ontwikkelomgeving opgezet. Zo bleken de Designer ontwikkelaars zeer gehecht aan de functietoetsen F12 en Shift-F12 (en die zijn er dus gekomen).
Met de volledige administratie van alle CB-software in JDeveloper wordt een eenduidige beheeromgeving voor alle software gecreëerd. Uitermate belangrijk is dat de software nu wordt ondergebracht in een toekomstvaste productlijn van Oracle: JDeveloper. Daarbij is geen afbreuk gedaan aan de betekenis (en voorziene continuïteit) van de feitelijke ontwikkeltools: PL/SQL, Forms en Reports. Oracle zelf heeft al aangegeven de 'J' in JDeveloper eerder staat voor 'Just anything' dan uitsluitend voor 'Java'. De mogelijkheden om JDeveloper met de Extensie SDK uit te breiden, blijken ruim toereikend om JDeveloper inderdaad in te zetten voor ontwikkeling en beheer van zowel conventionele software als Java/J2EE-software.
Bij veel nieuwe (Java)-projecten zien we een belangrijke stap in het opzetten en inrichten van een efficiënte ontwikkelstraat. Deze wordt afgestemd op de architectuur van de programmatuur en de gebruikte ontwikkeltools. Dit aspect lijkt nu met terugwerkende kracht ook mogelijk voor de traditionele PL/SQL- en Forms-omgeving. Enige aanpassing, ook in de functionaliteit van de IDE is daarbij niet vreemd, maar voor Designergebruikers wellicht nieuw.

De mogelijkheden om de ontwikkelstraat naar eigen inzichten in te richten, beperkt zich niet tot de functionele aspecten, maar omvat ook de keuze in objecttypen die worden onderscheiden binnen de software en uiteraard de relaties tussen die objecttypen. Veel meer dan in Designer kan de IDE een afspiegeling zijn van de werkelijkheid, zonder compromissen om verschillende typen software-objecten onder één noemer te hoeven administreren.

In dit overzicht mag de productiviteit van de nieuw ingerichte ontwikkelomgeving niet ontbreken. Hoewel de status is, dat alle voorbereidingen zijn afgerond en de finale migratie nog moet plaatsvinden, blijkt uit de testperioden dat de verwachtingen om productiviteitswinst te boeken terecht zijn. Met name door het versiebeheer worden op tal van momenten in de processen van softwareontwikkeling, softwarebeheer en oplevering winst geboekt. Denk hierbij aan minder benodigde tijd voor het vaststellen van de source die aanpassing behoeft bij een productie-incident of een testbevinding. Of de sterke reductie van de tijd die benodigd is voor het uitvoeren van kwaliteitsreviews. Of de tijdbesparing die kan worden geboekt door rechtstreeks van JDeveloper/CVS te kunnen opleveren naar de verschillende omgevingen (development en test). Nadat tijdens de startfase en de voorbereidingsfase voorzichtigere percentages zijn gehanteerd oordeelt Centraal Boekhuis momenteel dat een efficiencywinst van tien procent zonder meer haalbaar is.

Naschrift

Een presentatie naar aanleiding van dit project leidde in week 27 tot een vermelding op de homepage van OTN. Naar aanleiding daarvan ontvingen wij de volgende reactie van Oracle:

"There are many Designer customers who, like you, would use JDeveloper if it had a few extra pieces. I was very pleased to see that you had created extensions to the toolset where you saw and filled the gaps for your own purposes...I would like Designer developers to realize they can write extensions for JDeveloper using the APIs available"

Sue Harper, productmanager Designer

ing. Jules de Ruijter MBA
Manager Systeemontwikkeling, Centraal Boekhuis
e-mail: j.de.ruijter@centraalboekhuis.nl

Ir. Jeroen van Schaijk
Consultant, IT-eye
e-mail: jeroen.van.schaijk@it-eye.nl