

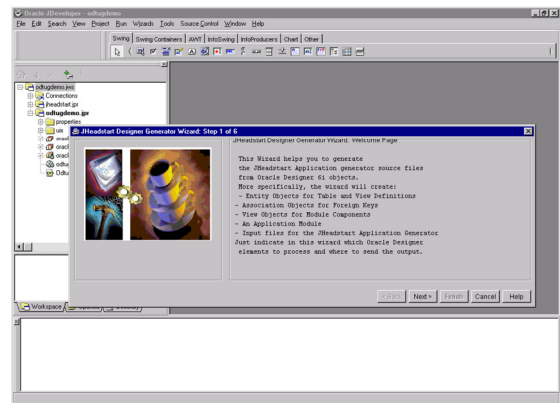
# JHeadstart: De logische weg

## Van Oracle Designer naar Java

**De laatste tijd bestaat er bij veel klanten onduidelijkheid over de koers die Oracle vaart met betrekking tot de Designer/Developer toolset in relatie tot Java. In dit artikel licht de auteur de toekomstvisie van Oracle toe en beschrijft hij hoe bedrijven die plannen hebben om te beginnen met Java, of willen migreren naar Java, op weg worden geholpen met JHeadstart.**

Menig Oracle-ontwikkelaar in Nederland bekruipt de laatste één à twee jaar regelmatig het gevoel 'loop ik nu zo achter of loopt Oracle nu zo voor?'. En het behoeft geen toelichting waar we het dan over hebben: Java. We kunnen geen Oracle seminar volgen en geen Oracle magazine lezen, of we worden met Java overstelpt. Sterker nog: als we alle Java gerelateerde onderwerpen zouden negeren blijft er vaak niets over. Voor het IT-speelveld in Nederland een heel bizarre situatie: de bedrijven waar nu al met Java wordt gewerkt zijn niet de traditionele klanten van Oracle, en bij het begrip Java denken deze bedrijven vaak aan andere productleveranciers dan Oracle. Daarentegen willen de traditionele klanten van Oracle juist van geen Java weten. Deels is dat koudwatervrees natuurlijk, maar geef ze eens ongelijk: jarenlang leek er voor Nederlandse Oracle klanten geen andere serieuze productfamilie te bestaan in het Oracle portfolio dan de combinatie van Designer en Developer. Veel klanten baseren al zeker tien tot vijftien jaar - en soms nog langer - hun systeemontwikkeling op de Designer/Developer toolset. Op zich geen verkeerde keus. Hoewel deze toolset best complex genoemd mag worden en dan ook een vrij steile leercurve kent, wordt dit meer dan gecompenseerd door de hoge productiviteit van de tools. Klanten die deze producten toepassen zijn over het algemeen dan ook meer dan enthousiast. Het nadeel voor deze klanten is echter dat zij een relatief gering deel uitmaken van het wereldwijde klantenbestand van Oracle. Uiteraard stemt Oracle haar strategie af op het grootste deel van haar klantenbestand, de VS. En van de VS is algemeen bekend dat Java daar jaren geleden al een grote vlucht heeft genomen. Nu kunnen we Oracle gemakkelijk betichten van het denken 'dat de VS de hele wereld is', maar wij zijn op onze beurt geneigd te denken 'dat Nederland de hele wereld is'.

Beide denkwijzen zijn uiteraard verkeerd, maar we mogen best aannemen dat Oracle als geen ander haar klantenbestand kent. Daarbij konden we tot voor kort alleen maar hopen dat Oracle's 'move to Java' ook voor de traditionele Designer/Developer klanten een soepele transitie zou zijn. Inmiddels is er eindelijk duidelijkheid over de status van Designer/Developer en is er zelfs een heus migratietool beschikbaar voor een vloeiende overgang van Designer/Developer naar Java/XML: JHeadstart. Maar wellicht is het nog te vroeg om nu al een overstap naar Java te maken.



Afbeelding 1. JHeadstart wizard in JDeveloper

### Status Oracle Designer/Developer

Mede door Oracle's onevenwichtige marketingbombardementen van de afgelopen jaren is er in de Oracle ontwikkelgemeenschap en bij klanten veel onrust ontstaan. Het dieptepunt wat dat betreft werd aan het begin van deze zomer bereikt. De productmanager van de Oracle Developer Tools schreef in het Oracle Magazine dat de projectnaam ('Project Cherokee') die werd gebruikt voor de ontwikkeling van de nieuwe versie van Oracle Developer, zou worden gereserveerd voor een ander project, nl. voor het maken van een nieuwe Integrated Development Environment (IDE) gebaseerd op JDeveloper. Die IDE zou de logische opvolger worden van Oracle Developer. Deze zinnen die ergens schijnbaar achteloos in een artikel waren verstopt betekenden in feite nogal wat.

Er had ook kunnen staan: de ontwikkeling van Oracle Developer is stopgezet en om met de volgende versie van Oracle's IDE te kunnen werken moet u eerst Java leren.

Inmiddels is de meeste stof gaan liggen: zoals we in de Optimize van september hebben kunnen lezen sprak Oracle zich tijdens de ODTUG 2001 voor het eerst sinds tijden expliciet uit over de (huidige inzichten van) de toekomst van Oracle Designer en Oracle Developer. Ook Oracle Nederland (zoals onlangs in een seminar gegeven door de Consulting afdeling) is nu bereid de productstrategie voor de toekomst iets harder te maken. Kortweg kunnen we deze als volgt schetsen:

1. Oracle blijft een dual-language strategie volgen, d.w.z. dat zowel PL/SQL als Java ondersteund zullen blijven in toekomstige producten.
2. De Designer/Developer combinatie is niet 'dood' verklaard, maar wel 'uitontwikkeld'. Dat wil zeggen dat er na de op handen zijnde opvolger van Designer 6i, Oracle9i Designer, geen functionaliteitsreleases meer uit zullen komen. In plaats daarvan komen uitsluitend maintenancereleases, met bugfixes of aanpassingen aan nieuwere versies van de Oracle Database.
3. De back-end van Oracle Designer (voorheen bekend onder de naam Oracle Repository) wordt als volwaardig product in de markt gezet onder de naam Oracle Software Configuration Manager (SCM). JDeveloper heeft bijvoorbeeld een volledig geïntegreerde link met Oracle SCM.
4. JDeveloper wordt verder uitgebreid tot de nieuwe standaard IDE van Oracle. Deze IDE is gebaseerd op Java, en zal in de loop der jaren worden uitgebreid met modelleringstools, te beginnen met een UML modelleringstool (Oracle9i JDeveloper in Q4).

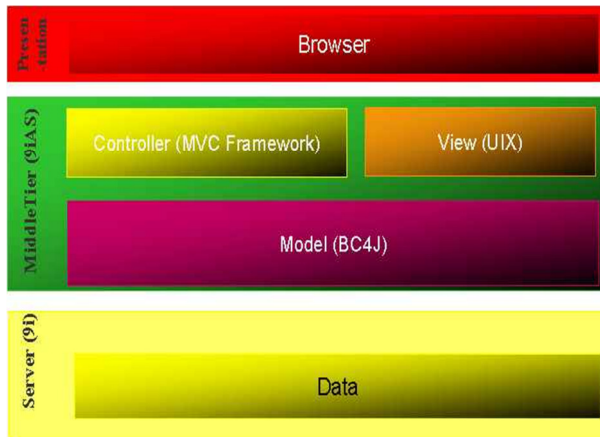
Met betrekking tot de status van Designer en Developer weet de Vice President van Oracle Development Tools, William Dwight, zelfs te melden dat deze gesupport zullen blijven 'for as long as i live'. Op zich een wel erg boude uitspraak. Maar als we het 'Statement of Direction Oracle Forms', gedateerd juli 2001, bekijken, valt daar te lezen dat Oracle Forms 6i tot 2006 gesupport zal zijn, waarbij zelfs de mogelijkheid wordt geboden de support tegen betaling te verlengen tot 2008. Hopelijk ligt het 'end-of-life' van de opvolgers van Forms 6i nog verder in de tijd. Hoe dan ook, deze statements geven veel vertrouwen voor de toekomst. Als we de boodschap achter de geformuleerde kernpunten in de strategie proberen te onderscheiden, dan kunnen we stellen dat 'op termijn' de combinatie van Designer/Developer uifaseert, en dat er een Java IDE voor in de plaats komt. En dat op dit moment de Designer/Developer combinatie als IDE eigenlijk volwassener is dan de huidige Java-gebaseerde IDE, JDeveloper, zeker wat de modelleringmogelijkheden betreft. En in

Calvinistisch Nederland slaan we modellerend ontwikkelen nu eenmaal hoger aan dan flexibel programmeren zoals in de VS. Het is alleen wel even wennen dat er geen separate JDesigner komt als tegenhanger van Designer. In plaats daarvan zal JDeveloper worden uitgebreid met modelleringstools. Als we echter nagaan hoeveel effort er de afgelopen tien jaar is gestoken in het integreren van Designer en Developer is het meer dan logisch dat een omgeving die van scratch af aan wordt ontwikkeld al bij aanvang uitgaat van één geïntegreerd product.

### iDevCoE en Java

Het iDevelopment Center of Excellence (iDevCoE) van Oracle, bij de meesten bekend als de afdeling die verantwoordelijk is voor producten als CDM, ODWA en Headstart, is binnen Oracle in het verleden (naast uiteraard Oracle UK, de makers van Oracle Designer) wellicht de grootste lobbyist geweest van de Designer/Developer combinatie. En zij hebben als geen ander ingezien dat de move die Oracle Corporation maakt richting Java in feite onomkeerbaar is. Daarom zijn zij zich zo'n twee jaar geleden al bezig gaan houden met het nadenken over gestructureerde softwareontwikkeling met Java. Het meest voor de hand liggend zou natuurlijk zijn om een Java variant van haar producten uit te brengen, zoals een soort JCDM waarin standaarden en richtlijnen staan beschreven voor maatwerk systeemontwikkeling met behulp van Java en een soort JHeadstart die als Javacomponenten en Java-utilities de ontwikkeling met Java zou moeten versnellen. Maar als je hier even bij stilstaat zie je al snel dat deze gedachte iets te simpel is. Het ontwikkelen met Designer en Developer gebeurt met de leverancier-specifieke taal PL/SQL, en daarvoor waren -voordat CDM en Headstart uitkwamen- niet echt componenten beschikbaar die wereldwijd bekend waren en werden toegepast. Java is echter een open (geen vendor lock-in) platform met raakvlakken aan vele andere technologieën en standaarden. Als aanvulling op de Java specificatie heeft Sun een schat aan informatie op de Java-website gezet, waaronder een aantal standaarden van componentmodellen, en suggesties voor gestructureerde ontwerpen van applicaties, de zogenaamde J2EE Design Patterns. Zo is wat betreft standaarden iedereen (zonder wellicht de betekenis te kennen) wel eens de term Enterprise Java Beans (EJB), Servlets, of Java Server Pages (JSP) tegengekomen. Bij het ontwikkelen van een Headstart variant in de Java wereld is het daarom niet zozeer het van de grond af aan ontwikkelen van eigen componenten, maar meer het zinvol gebruik maken van bestaande componenten. En het beschrijven van standaarden en richtlijnen is eveneens niet het van de grond af aan bedenken, of het door ervaring inzicht verwerven in de beste manieren van ontwikkelen. J2EE Design patterns zijn immers reeds beschikbaar in de Java-wereld. Nee, een Headstart voor Java die toegevoegde waarde heeft zal het meer moeten hebben van slim gebruik van bestaande componentmodellen, en het waarborgen van de reeds gedane investeringen in Designer/Developer, bij de overstap naar Java. En

precies dát is het product geworden dat het iDevCoE de Oracle gemeenschap te bieden heeft.



Afbeelding 2. Java/XML applicatie architectuur

## Achtergrond

Na de voorgaande schets van de ontwikkelingen rondom Java in de Oracle gemeenschap is het tijd geworden om JHeadstart in wat meer detail te gaan bekijken. Om een goed beeld te krijgen van de mogelijkheden van JHeadstart is het echter noodzakelijk enige voorkennis te hebben van de (Java-) componentmodellen en J2EE Design Patterns waar met JHeadstart ontwikkelde applicaties gebruik van maken. In afbeelding 2 is een schematisch overzicht gegeven van de architectuur van een Java/XML applicatie als we JHeadstart toepassen.

Allereerst is daar Business Components for Java (BC4J), een door Oracle ontwikkeld componentenmodel waarvan de componenten met JDeveloper 3.0 en hoger kunnen worden onderhouden. Merk hierbij op dat gebruik van een door Oracle ontwikkeld componentenmodel niet betekent dat de op basis daarvan ontwikkelde applicaties niet 'open' meer zijn. De componenten zijn zelf immers ook geprogrammeerd in Java, en worden mee-gedeployed. Ontwikkelde applicaties kunnen daarom op elke J2EE (Java 2 Enterprise Edition) compatibele application server worden uitgerold. Door de native ondersteuning van BC4J in JDeveloper is het wel zuiver te zeggen dat Oracle probeert ontwikkelaars op deze manier aan JDeveloper te binden. Maar verplicht is het niet: theoretisch is het zelfs mogelijk met een willekeurige tekst-editor een BC4J applicatie te ontwikkelen. Maar dan moet je wel tamelijk verknijpt zijn.

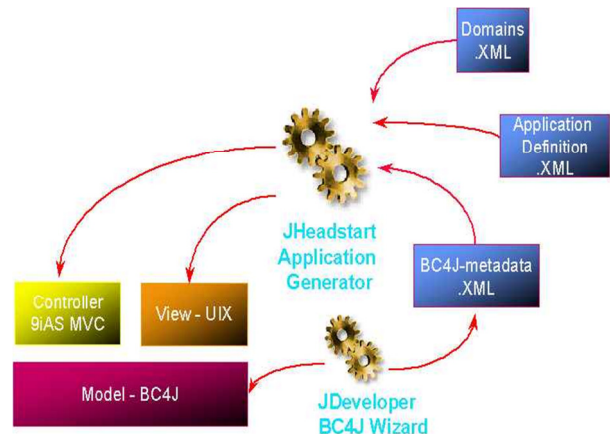
Zonder dieper in te gaan op de verschillende soorten componenten binnen BC4J kun je stellen dat BC4J componenten de applicatiedata toegankelijk maakt en mogelijkheden biedt business logica te implementeren, conceptueel vergelijkbaar met bijvoorbeeld de Table API en Module Component API zoals die zijn te genereren met Oracle Designer.

Een ander framework dat wordt gebruikt door JHeadstart is UIX. Dit staat voor User Interface XML. Het is een technologie die het mogelijk maakt de complexe HTML user interfaces te maken die

ook worden toegepast door de Oracle Applications Developer Group, user interfaces met de zogenaamde Oracle Browser Look and Feel. Voorbeelden van deze UI zijn op het net terug te vinden in de meeste Oracle sites, gekenmerkt door de tabs waarmee andere pagina's kunnen worden gekozen. De uitdaging is hier dat we alles uit de kast moeten halen op HTML gebied om deze UI te kunnen maken. UIX maakt het echter mogelijk om declaratief (in XML) de layout en content van de pagina's te definiëren. Er is native ondersteuning gepland voor UIX in Oracle9i JDeveloper, door middel van een Visual UIX Layout Editor.

Een derde concept waar JHeadstart zich op baseert is het design pattern 'Model-View-Controller'. Dit 'best practices' advies dat ook door Sun wordt gepropageerd, komt neer op het bouwen van applicaties waarin drie taakgebieden separaat zijn geprogrammeerd: Model (alles wat samenhangt met de applicatiedata), View (alles wat samenhangt met het weergeven van applicatiedata, de user interface) en Controller (alles wat de samenhang vormt tussen het Model en de View, de zogenaamde process flow engine). Nu is een J2EE design pattern niets meer dan een ontwerpsuggestie, en zeker geen classlibrary (een Java programmalibrary) waar je op voort kunt borduren.

Oracle Consulting in de VS heeft dit design pattern echter in een van haar projecten volledig geïmplementeerd, en heeft deze MVC implementatie dusdanig open opgezet dat dit als framework voor toekomstige projecten kan worden gebruikt. Dit framework staat (ten tijde van het schrijven van dit artikel) nog bekend als 'Project Cleveland', en is terug te vinden op OTN<sup>1</sup>. In de toekomst zal het echter onderdeel uit gaan maken van de Oracle 9i Application Server 2.0, onder de naam MVC Framework for J2EE, en zal gratis te downloaden zijn van OTN als een developer kit.



Afbeelding 3. JHeadstart applicatie ontwikkeling

## JHeadstart

Nu we genoemde frameworks enigszins kunnen plaatsen is het ook wat eenvoudiger geworden om JHeadstart te positioneren. In afbeelding 3 is de wijze van ontwikkelen te zien als we gebruikmaken van JHeadstart en JDeveloper.

```

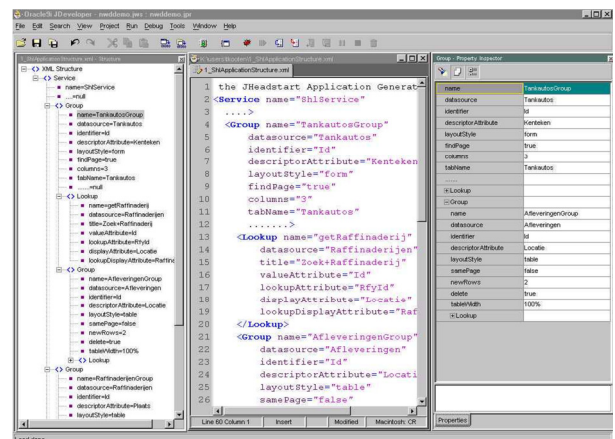
<Service name="ShService">
<!-- In de eerste groep kunnen tankauto's en hun aflevering
worden onderhouden. Door het nesten van de groepen ontstaat
een master-detail relatie. De layout style van de master is
'form' en die van de detail is 'table'. Beide groepen bevatten
look ups de tankauto naar de raffinaderij en de aflevering naar
het pompstation. De tankauto groep bevat ook een Find Pagina die
vergelijkbaar is met enter-query mode in Forms.>
<Group name="TankautosGroup"
  datasource="Tankautos"
  identifier="Id"
  descriptorAttribute="Kenteken"
  layoutStyle="form"
  findPage="true"
  columns="3"
  tabName="Tankautos"
  .....>
<Lookup name="getRaffinaderij"
  datasource="Raffinaderijen"
  title="Zoek+Raffinaderij"
  valueAttribute="Id"
  lookupAttribute="RfId"
  displayAttribute="Locatie"
  lookupDisplayAttribute="Raffinaderij">
</Lookup>
<Group name="AflleveringenGroup"
  datasource="Aflleveringen"
  identifier="Id"
  descriptorAttribute="Locatie"
  layoutStyle="table"
  samePage="false"
  newRows="2"
  delete="true"
  tableWidth="100%">
<Lookup name="getPompstation"
  datasource="Pompstations"
  title="Zoek+Pompstation"
  valueAttribute="Id"
  lookupAttribute="Psnld"
  displayAttribute="Naam"
  lookupDisplayAttribute="Pompstation">
</Lookup>
</Group>
</Group>
<!-- Onderhoudsscherm voor raffinaderijen met layoutStyle is
table. Aangezien sortable 'aan' staat kan er gesorteerd worden op
alle velden die in the tabel voorkomen>
<Group name="RaffinaderijenGroup"
  datasource="Raffinaderijen"
  identifier="Id"
  descriptorAttribute="Plaats"
  layoutStyle="table"
  newRows="4"
  sortable="true"
  tabName="Raffinaderijen"
  ....>
</Group>
<!-- Onderhoudsscherm voor pompstations met layoutStyle is
table-form. Eerst wordt een tabel gepresenteerd met de
belangrijkste attributen. Nadat de gebruiker een pompstation
gekozen heeft kan ingezoomd worden op alle attributen van
pompstation in een Form Layout-style>
<Group name="PompstationsGroup"
  datasource="Pompstations"
  identifier="Id"
  descriptorAttribute="Naam"
  layoutStyle="table-form"
  sortable="true"
  tabName="Pompstations"
  ....>
</Group>
</Service>

```

Afbeelding 4. Fragment van een JHeadstart Application Definition XML file

Hierbij zien we links onderin de applicatiecomponenten die tezamen de te bouwen applicatie vormen. Allereerst de Viewcomponenten die zijn gebaseerd op UIX en worden gegenereerd vanuit JHeadstart. De Controllercomponenten zijn gebaseerd op het 9iAS MVC framework en worden eveneens gegenereerd vanuit JHeadstart. De Modelcomponenten tot slot worden met de BC4J-wizard gegenereerd vanuit JDeveloper. Bij laatstgenoemde generatie wordt informatie over de gegenereerde componenten vastgelegd in een zogenaamde BC4J-metafile. Deze file is op zijn beurt weer één van de drie XML files die door JHeadstart worden gebruikt om de Controller- en de View-componenten te genereren. De andere XML files zijn voor het vastleggen van de zogenaamde domains en de application structure.

In afbeelding 4 wordt een fragment getoond van de application structure XML file. In dit fragment is te zien dat deze file properties bevat die ook in Designer of Developer terug te vinden zijn, zoals bijvoorbeeld Module Component properties of Block properties. Het lijkt daarom of we van geavanceerde tools weer teruggaan naar softwareontwikkeling met behulp van tekst-editors, met alle fouten van dien. Een in XML specificerde file kan echter uitstekend geautomatiseerd gevalideerd worden.

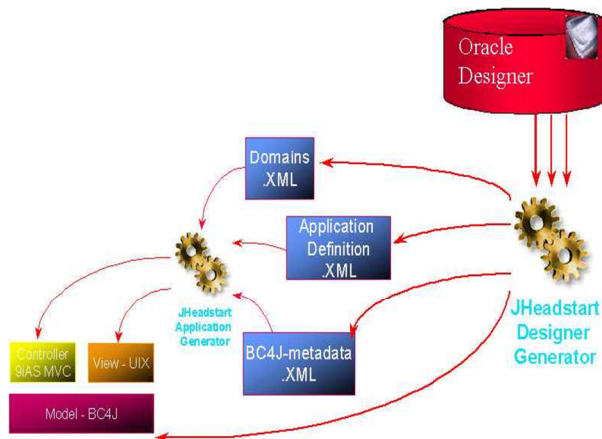


Afbeelding 5. Onderhouden van XML files in Oracle9i JDeveloper

Afbeelding 5 toont hoe deze file binnen Oracle9i JDeveloper kan worden onderhouden. Links is de structuur te zien van de XML file en daarmee (in het geval van de application structure XML file) ook de structuur van de applicatie. In het midden bevindt zich de XML file zelf en aan de rechter kant de zogenaamde Property Inspector. Wijzigingen kunnen zowel in de Property Inspector als in de XML file gemaakt worden, en zijn meteen te zien zijn in alle windows. Met één druk op de rechter muisknop kan de XML syntax gecontroleerd worden op eventuele fouten.

In afbeelding 6a t/m 6c zijn verschillende voorbeelden te zien van een demo-applicatie die is ontwikkeld met JHeadstart<sup>2</sup>.





Afbeelding 7. JHeadstart applicatie migratie

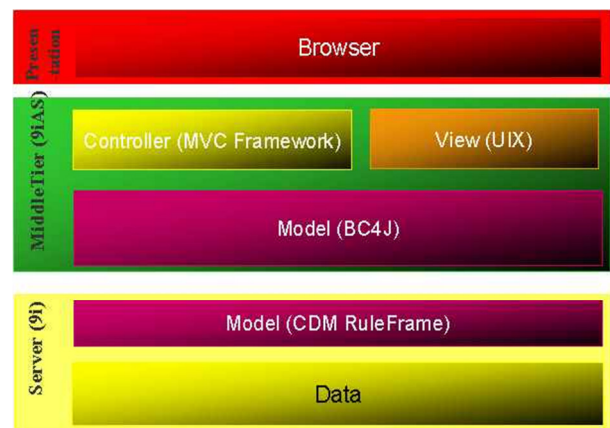
### Generator

Hoe 'kaler' de Forms Module-definities in Designer (de vanuit Designer gegenereerde Oracle Forms waren in feite pure User Interfaces), des te groter de kans dat JHeadstart als een heuse generator kan worden gebruikt. Oplettende lezers zullen direct uitroepen: maar m'n business logica dan? Inderdaad. In een pure User Interface zit geen business logica. Hoewel verwacht mag worden dat in toekomstige versies van JHeadstart steeds meer business logica die in Designer is vastgelegd gebruikt zal worden, staat dit nu nog in de kinderschoenen. Was de logica daarentegen niet in de Forms maar in de Database Server geïmplementeerd, dan werkt de door JHeadstart gegenereerde applicatie (nu eveneens in de rol van User Interface) daar prima bovenop, en is er 100% generatie te bewerkstelligen. Wat dit betreft worden klanten die al eerder zijn gemigreerd naar het CDM RuleFrame beloofd: alle logica die in het RuleFrame is geïmplementeerd wordt aangesproken via de gegenereerde Java-applicatie, en alle fouten worden transparant afgehandeld, alsof de business logica in de Java-applicatie zelf is gecodeerd. Een schematische voorstelling van deze applicatie architectuur is weergegeven in afbeelding 8. Voor alle duidelijkheid: in deze configuratie blijft Oracle Designer een centrale rol spelen. De business rules worden onderhouden in PL/SQL, gegenereerd vanuit Designer, en de user interfaces worden gegenereerd met JHeadstart, eveneens vanuit Designer. Het woord 'vanuit' moet hier overigens worden gelezen als: 'op basis van definities in'. De JHeadstart Designer Generator is conceptueel dan wel een uitbreiding van Designer, het is fysiek echter geïmplementeerd als een wizard in JDeveloper (zie afbeelding 1). Het behoeft geen betoog dat deze mix van technologie niet gezien moet worden als een eindscenario. Het is echter een zeer laagdrempelig doorgroeiscenario naar de Javawereld.

### Migratietool

In de situatie dat er wel degelijk veel business logica in de client is geprogrammeerd en er geen CDM RuleFrame is toegepast, is JHeadstart zoals gezegd niet in staat om op basis van de Designerdefinities een 100% generatieslag te doen naar Java/XML.

Maar welbeschouwd is de situatie van de Fat Client in het huidige internettijdperk ook niet meer van deze tijd. En als we op enig moment toch besluiten te migreren, dan kan zelfs de in dat geval wat beperktere werking van JHeadstart uitkomst bieden. In dit scenario gebruiken we JHeadstart om de definities uit Designer om te zetten naar een applicatie volgens de architectuur zoals eerder weergegeven in afbeelding 2. Vervolgens moet in een verrijkingsslag de business logica die voorheen in Designer en/of Developer was gedefinieerd, ditmaal in Java worden gecodeerd in de BC4J laag. Is deze migratie achter de rug, dan kan theoretisch de stekker uit Designer. In de praktijk zullen we Designer voorlopig nog wel even blijven gebruiken omdat het toch wel erg geschikt is voor het genereren van het Server Model in de database, de tabellen en andere objecten. Overigens is de JHeadstart Designer Generator ontworpen voor het uitlezen van een Oracle Designer 6i Repository. Dat betekent dat om deze generator te gebruiken de repository in ieder geval gemigreerd dient te worden naar 6i. Dit lijkt veel werk, maar bij repositorymigraties zit het meeste werk in het verrijken van de repositorygegevens na het overzetten van de pre-6i naar de 6i repository. En die verrijkingsslag is voor gebruik van de JHeadstart Designer Generator niet noodzakelijk.



Afbeelding 8. Java/XML applicatie architectuur met CDM RuleFrame.

### Resumerend

De strekking van dit artikel moge duidelijk zijn. Hoewel de marketingcampagne van Oracle ons anders doet vermoeden, hoeven we zeker niet in allerijl de overstap naar Java te maken. De Designer/Developer toolset blijkt nog een heel leven voor zich te hebben. Niettemin is het duidelijk dat Oracle op de lange termijn de plaats van de traditionele tools ingenomen wil zien door Java en de bijbehorende IDE. In het licht daarvan is het zeker verstandig voorzichtig initiatieven te ontplooiën op het gebied van Java. Te denken valt hierbij aan een pilotproject, waarbij gebruik van JHeadstart een laagdrempelige manier is om met de Javawereld kennis te maken. Op de middellange termijn zou dan een deel van de applicatie-ontwikkeling in Java plaats kunnen vinden. Verwacht mag worden dat JDeveloper daar tegen

die tijd (vergelijkbaar met Oracle Designer nu) alle modelleringsmogelijkheden voor zal hebben. Maar zover is het nog niet. Klanten die voorlopig graag met de traditionele tools blijven ontwikkelen, doen er echter wel verstandig aan een consolidatie van de toolstack door te voeren. Hierbij zouden eigenlijk alle gebruikte ontwikkelomgevingen naar Designer6i, Developer6i en eventueel Headstart6i opgewaardeerd moeten worden. Enerzijds is dit verstandig qua support door Oracle: support op pre-6i releases verstrijkt eind dit jaar en door gebruik te maken van 6i is support tot 2008 gewaarborgd. Anderzijds biedt dit mogelijkheden om het CDM RuleFrame toe te gaan passen, zodat investeringen in business logica volledig beschermd blijven als er op enig moment met behulp van JHeadstart wordt gemigreerd naar Java. Want hoe verknocht we ook zijn aan de traditionele tools, dat we uiteindelijk op Java uit zullen komen, dat staat vast.

---

<sup>1</sup>Meer informatie over het MVC framework is op OTN terug te vinden onder de naam Project Cleveland in de sectie over Oracle Containers for Java (OC4J) met de volgende URL:  
[http://otn.oracle.com/sample\\_code/tech/java/oc4j/](http://otn.oracle.com/sample_code/tech/java/oc4j/)

<sup>2</sup>Het in de demo gebruikte Shell-logo en de Shell merknaam zijn geregistreerde handelsmerken van de Shell-organisatie. Deze merkuitingen zijn slechts bedoeld om de demo te verlevendigen. Shell is in geen enkele vorm gelieerd aan JHeadstart.

Meer info over de in dit artikel beschreven onderwerpen is te vinden op de volgende plaatsen:

Oracle Browser Look and Feel (BLAF)  
<http://otn.oracle.com/tech/blaf/index.html>

Business Components for Java (BC4J) white paper  
<http://otn.oracle.com/products/jdev/info/techwp20/wp.html>

MVC design pattern op Sun website  
[http://java.sun.com/j2ee/blueprints/design\\_patterns/model\\_view\\_controller/index.html](http://java.sun.com/j2ee/blueprints/design_patterns/model_view_controller/index.html)

User Interface XML (UIX)  
- online documentatie van Oracle9i JDeveloper Beta

Kijk voor meer publicaties op  
<http://www.anewlink.nl/ict/nl/publicaties/>.

(c) Copyright 2001, A New Link bv