Hoe kun je op een productieve wijze kwalitatief hoogwaardige applicaties realiseren? In dit artikel beschrijft Jan Vissers de wijze waarop, vanuit een ontwikkelde expertise op het gebied van het Oracle RDBMS en Java/J2EE, applicatieontwikkeling zou kunnen worden ingevuld. De kernelementen hierin vormen de erkenning van het RDBMS als belangrijk architectuurcomponent, selectie van 'best-of-breed'-frameworks en pragmatische inrichting van ontwikkelstraat.

Een open blik op Java/J2EE-applicatieontwikkeling

Database als architectuurcomponent

Toen het bedrijf waar ik werkzaam ben ongeveer vijf jaar geleden begonnen met Java/J2EE-applicatieontwikkeling, hadden we daarvoor eigenlijk alleen echt serieus met OracleForms en PL/SQL-applicaties gebouwd. We schrokken in eerste instantie van het feit dat onze productiviteit met Java/J2EE veel lager lag dan dat we gewend waren, vooral bij webapplicaties. Natuurlijk moesten we de taal nog leren en waren nog niet bekend met de essentiële onderdelen van J2EE. Toch hadden we niet gedacht dat de 'overgang' zo lastig zou zijn.

Er is vanaf dat moment veel veranderd. Zo is Java/J2EE volwassen geworden en zijn er ter ondersteuning van het ontwikkelproces diverse bouwstenen, tools en methodieken beschikbaar gekomen die productief en efficiënt werken mogelijk maken. De rol die het Oracle-RDBMS speelt, is daarbij niet veranderd. Zowel in projecten waarin wij actief zijn, als voor onze eigen Tangelo-productsuite kunnen wij nog altijd 'bouwen' op het fundament dat de database ons biedt. Vanuit de pure Java/J2EE-gedachte is dit wellicht een gevaarlijke uitspraak, maar wij ondervinden dagelijks dat het verstandig is om de database niet als gemeengoed te beschouwen – maar om het een wezenlijk onderdeel te laten zijn van de architectuur. Dit artikel geeft een indruk hoe J2EE-applicatieontwikkeling zou kunnen worden ingevuld, waarbij de nadruk ligt op:

- Verstandig gebruik van het RDBMS.
- Duidelijk afgebakende framework stacks.
- Pragmatische inrichting van ontwikkelstraat.

VERSTANDIG GEBRUIK VAN HET RDBMS Vanuit de pure Java/J2EE-gedachte, behoort toegang tot en opslag van data in een 'persistent store' niet tot het meest prominente onderdeel van de totale architectuur. Dit is een potentieel gevaarlijke aanpak – die vaak niet in relatie staat tot de praktijk. In veel gevallen zullen applicaties ontwikkeld moeten worden op een al bestaande databasestructuur, die mogelijk niet alleen op basis van Java/J2EE-technologie benaderd worden. Er zijn daarnaast organisaties, die veel tijd en geld geïnvesteerd hebben in een robuuste database. Dit komt dan tot uiting in een grote hoeveelheid code in de database. Er zullen niet veel organisaties zijn die, omwille van wat de J2EE-specificatie of blueprints voorschrijven, besluiten om dit alles over boord te zetten.

Een ander punt is, dat uiteindelijk de data die in een Java/J2EE-applicatie omgaan,opgeslagen zullen moeten worden. Natuurlijk hoeft deze opslag niet RDBMS-gebaseerd te zijn, maar in de praktijk zal dit meestal wel het geval zijn. We gaan hierbij bewust voorbij aan de discussie die vervolgens ontstaat rondom de 'impedance mismatch' – het min of meer onverenigbaar zijn van OO en relationele principes – omdat deze discussie zelden ergens toe leidt. Het is verstandiger om de database, als geoptimaliseerd medium om gegevens in op te slaan en op te halen, zijn werk te laten doen op een manier die daarbij het beste aansluit – op relationele wijze dus. Dit biedt meer zekerheid voor een zo optimaal mogelijk presterend systeem.

Nagenoeg alle projecten en producten waar ik mee te maken krijg, hebben op één of andere manier te maken met het Oracle-RDBMS. Dit product biedt tal van interessante, technische uitbreidingen die waarde kunnen toevoegen aan een oplossing. We noemen hierbij vooral

de ingebedde XML-ondersteuning, waar wij vaak dankbaar gebruik van maken.

Met andere woorden, de database zeer zeker geen sluitstuk. Gebruik van de database, eventueel met zijn specifieke features, hoeft zeker niet ten koste te gaan van een nette, goed uitgebalanceerde architectuur. Het benutten van de kracht van onder andere het relationele model, draagt bij tot een optimaal systeem. Bovendien sluit deze aanpak vaak beter aan bij de praktijk. Bijvoorbeeld in organisaties waarbij geïnvesteerd is in zaken als het afdwingen van data-integriteit in de database. Het uiteindelijke doel ligt besloten in het bij elkaar brengen van het beste van beide werelden: Java/J2EE en RDBMS. Deze overtuiging draagt mede bij aan de wijze waarop binnen Cumquat de diverse framework-stacks zijn samengesteld.

Duidelijk afgebakende framework stacks

Aanvankelijk betekende het gebruik van Java/J2EE een zekere mate van productiviteitsverlies. Een belangrijk deel hiervan laat zich verklaren door de normale cyclus die nieuwe technologie doorloopt van 'ontstaan' tot 'volwassenheid'. Hierbij blijft de basis redelijk constant: de programmeertaal en de specificaties. Wat vaak wat langer duurt, is de ontwikkeling van tools, bouwstenen, methodieken en best practices rondom de technologie. Het zijn deze onderdelen die het, op een productieve manier applicaties ontwikkelen voor een groter publiek, mogelijk maken. Een onmiskenbaar grote rol hierbij is weggelegd voor zogenaamde frameworks.

Een framework is een verzameling van één of meer concreet geïmplementeerde design patterns in de vorm van een library, eventueel aangevuld met documentatie en tool-ondersteuning in de vorm van een IDE plugin. Een design pattern is hierin een geformaliseerde oplossingsstrategie voor een vaker terugkerend softwareontwikkelingsprobleem. Een framework biedt een bepaalde abstractie over een functioneel probleem enerzijds en de technisch onderliggende laag anderzijds. Niets weerhoudt mij om bijvoorbeeld het *FrontController*-pattern zelf te implementeren met behulp van de Java Servletspecificatie, maar het is meestal productiever en efficiënter om een framework te selecteren wat dit doet. Frameworks kunnen ons dus ondersteunen om productiever te zijn. Maar welk framework ga je kiezen?

Er gaat bijna geen maand voorbij in Java/J2EE-land, of er wordt wel een nieuw framework aangekondigd. Alle goede bedoelingen ten spijt maakt dat het er voor wat betreft applicatieontwikkeling niet gemakkelijker op, want 'wat als ik nu het verkeerde framework kies'? Een belangrijk onderdeel van een Java/J2EE-project wordt dan ook gevormd door een selectie van frameworks. Hierbij kunnen tal van criteria gehanteerd worden:

- Volwassenheid
- Stabiliteit
- Documentatie
- Afbakening
- Omvang
- Community
- Acceptatie
- Support
- Leercurve
- Toekomstvastheid
- Schaalbaarheid
- IDE ondersteuning
- Prijs
- Toepasbaarheid
- Productiviteit

Ook Cumquat heeft aan de hand van bovenstaande selectiecriteria een afweging gemaakt ten aanzien van welke frameworks tot 'standaard' verheven zouden kunnen worden. Dit had tot doel om enerzijds Tangeloproductdevelopment bepaalde richtlijnen mee te kunnen geven en anderzijds om bestaande en nieuwe Java/J2EE-projecten te kunnen ondersteunen en uitvoeren. Als aanvullende criteria zijn hierbij de volgende observaties en overwegingen meegenomen:

- Java/J2EE-projecten waarbij Cumquat betrokken wordt hebben in 98% van de gevallen het Oracle-RDBMS als persistent store.
- 75% van de medewerkers van Cumquat hebben ruime ervaring op het gebied van het Oracle RDBMS.
- Uit ervaring blijkt dat Java/J2EE-ontwikkelaars vaak onevenredig veel tijd moeten besteden aan de 'look & feel' van een applicatie.
- De gekozen frameworks dienen geschikt te zijn om naast elkaar ingezet te worden, in parallelle ontwikkelteams ter verkorting van de doorlooptijd.
- Al gerealiseerde applicaties die bij klanten draaien, zullen ondersteund moeten worden.
- Bij voorkeur open source.

Op basis van dit alles zijn hebben we de volgende onderverdeling van (web) applicatieontwikkeling stacks vastgesteld:

- Traditional – JSP/JSTL/Tiles, Struts, iBatis
- JCP/JSR Driven – JSP/Tiles, JSF, Spring, iBatis
- Advanced – XHTML/CSS, Tapestry, Hivemind, iBatis

Kenmerkend voor deze stacks is dat iBatis in alle gevallen gebruikt wordt. iBatis (http://ibatis.apache.org/) bestaat uit het sqlMap framework en een DAO-framework. Deze onderdelen stellen ons samen in staat om het maximale uit de (Oracle) database te halen zon-

der dat we daarbij zelf JDBC moeten coderen. Daarnaast is het framework bijzonder lichtgewicht en simpel in het gebruik. Ten aanzien van 'Traditional' kan worden opgemerkt dat deze stack vooral gebruikt wordt voor het ondersteunen van applicaties die al bij klanten in gebruik zijn.

De overeenkomst tussen 'JCP/JSR Driven' en 'Advanced' is dat ze beide een nieuwe generatie vertegenwoordigen voor de manier waarop webapplicaties gerealiseerd worden. Ingevuld door JSF (http://myfaces.apache.org/) en Tapestry (http://jakarta.apache.org/tapestry/).

JSF kenmerkt zich op dit moment door het feit dat er behoorlijk veel lawaai rondom heen gemaakt wordt. Nagenoeg alle grotere leveranciers (partijen als Oracle, Sun en IBM) geven aan dat ze deze standaard gaan ondersteunen. Deze standaardisatie kan echter ook een potentieel nadeel vormen, waarbij innovatie afgeremd wordt door het relatief logge standaardisatie proces rondom JCP en JSR's. Belangrijke minpunten zijn dat JSF nog behoorlijk in ontwikkeling is en op dit moment leunt op JSP (JavaServer Pages). Toch hebben wij de verwachting dat mede door het commitment dat is uitgesproken door de grote partijen en het belang dat bepaalde organisaties hechten aan JCP/JSR gedreven ontwikkeling, JavaServer Faces een rol zal blijven spelen.

Wat ons betreft is Tapestry op dit moment het meest geavanceerde framework met enorm veel potentie. Anders dan dat bij JSF het geval is, gaat Tapestry uit van HTML als view technologie en dit opent een aantal interessante voordelen. Onze ervaring leert bijvoorbeeld dat een groot deel van de tijd verloren gaat aan Java/J2EE-ontwikkelaars die zich bezig houden met het maken van de user interface. Het is maar zeer de vraag in hoeverre deze mensen daarvoor voldoende gekwalificeerd zijn. In de meeste gevallen is het verstandiger om dit te beleggen bij de web designer/vormgever.

Een ander voordeel van de op Tapestry gebaseerde aanpak is het feit dat de disciplines webdesigner en Java/J2EE-ontwikkelaar ook daadwerkelijk uit elkaar getrokken kunnen worden. Dit komt de productiviteit en doorloopsnelheid ten goede. Naast een andere view technologie kenmerkt Tapestry zich door een uitgebreid palet aan standaardfunctionaliteit, zoals: validatie van HTML-formulieren, internationalisatie, ondersteuning voor clustering, portal ondersteuning en component based development. Voor een deel worden deze ook door JSF geboden.

In de 'JCP/JSR Driven' en 'Advanced'-stacks zien we nog twee onderdelen die we graag toelichten. Het gaat hierbij om Spring (http://www.springframework.org/) en Hivemind (http://jakarta.apache.org/hivemind/). Beide frameworks zijn gebaseerd op het 'Inversion of Control'-principe. Er zijn tal van definities voor dit principe, maar ruwweg komt het erop neer dat gedeclareerd wordt hoe een Java-object aangemaakt dient te worden in plaats van dat het object vanuit de code aangemaakt wordt. De voordelen die hiermee te behalen zijn, zijn nagenoeg allemaal een resultaat van verdere ontkoppeling van de diverse architectuurlagen.

Een vaak gebruikt argument voor ontkoppeling van diverse lagen is het feit dat je op die manier een of meerdere frameworks kunt vervangen door een ander, zonder dat daar het systeem last van heeft. Eerlijk gezegd vinden wij dit geen sterk argument, omdat het in de praktijk zelden zal voorkomen. Waar ontkoppeling van de lagen wel voor zorgt, is de mogelijkheid tot het afsplitsen van het projectteam naar architectuurlaag en eventueel expertiseniveau. Dit kan een positief effect hebben op de totale doorloopsnelheid, omdat de lagen parallel in teams ontwikkeld kunnen worden. De afstemming tussen de diverse teams geschiedt dan op basis van 'een contract'. De ontkoppelde lagen kunnen onafhankelijk van elkaar gerealiseerd en getest worden en worden uiteindelijk weer aan elkaar geknoopt met behulp van een 'Inversion of Control'-framework. Overigens geldt dat Hivemind – uit de 'Advanced'-stack – ook kan worden ingezet in de 'JCP/JSR Driven'-stack in plaats van Spring en omgekeerd.

Het kiezen van de juiste frameworks is geen eenvoudige klus, juist omdat dit inhoudt dat bepaalde frameworks zullen moeten afvallen. Aan de andere kant is dit uiteindelijk wel een rustiger idee, want niets is zo vervelend als het niet kunnen of durven maken van keuzes.

JSF en Tapestry: een nieuwe generatie web framework

JSF en Tapestry kenmerken zich op hoofdlijnen door het feit dat het onderliggende HTTP protocol minder zichtbaar is. Bij de meer traditionelere frameworks zoals Struts is dit wel het geval. Dat framework beschrijft acties in termen van een HTTP request/response. Een HTTP request moet door een Struts action bijvoorbeeld uitgeplozen worden, de waarden uit het request vervolgens samengebracht worden naar een object om vervolgens de onderliggende service aan te kunnen roepen.

Anders is dat bij JSF en Tapestry. Deze frameworks lijken qua werking veel meer op de C/S applicaties zoals Delphi, Swing, et cetera. Het minutieus doorspitten van het HTTP request is niet meer nodig, het framework zorgt ervoor dat de diverse samenstellende objecten 'klaar' staan voor gebruik.

Zie ook:
http://www.theserverside.com/articles/article.tss?l=JSFTapestry

Het uiteindelijke doel is immers niet de gekozen frameworks, maar hoe de frameworks gaan bijdragen aan productiever Java/J2EE-applicatieontwikkeling. Naast frameworks is hierbij uiteraard ook de overige invulling van de ontwikkelstraat van belang.

PRAGMATISCHE INRICHTING VAN ONTWIKKELSTRAAT Onmisbaar bij het efficiënt, productief en voorspelbaar ontwikkelen van applicaties is de inrichting van een ontwikkelstraat. Een geselecteerde framework-stack maakt hiervan onder andere deel uit. Daarnaast zijn er nog een aantal belangrijke onderdelen die hierin ondergebracht worden. Daarbij is het verstandig om een pragmatische instelling te blijven nastreven, want ook de ontwikkelstraat is slechts een middel en geen doel. We moeten oppassen dat we niet vervallen in het alsmaar toevoegen van aardige frameworks en hulpmiddelen aan de ontwikkelstraat. Dit gaat op den duur ten koste van tijd en geld dat gemoeid is met het configureren en beheren ervan. De volgende onderdelen zijn essentieel in een ontwikkelstraat:

- Infrastructure
- Standards & guidelines
- Test tool
- Build tool
- Version control
- Dependency management
- Continuous integration
- Issue management

Infrastructure
Dit spreekt eigenlijk voor zich. De software wordt ontwikkeld binnen een bepaalde infrastructuur die bestaat uit onderdelen als de werkstations van de diverse projectleden, maar ook één of meerdere applicatieservers en databaseservers. Het is van belang om de infrastructuur op orde te hebben alvorens een project gestart kan worden.

Standards & guidelines
Dit hoeft niet heel erg zwaar aangezet te worden. Het gaat meer om bijvoorbeeld een standaard-template om technische en functionele documenten in op te stellen. Onder deze noemer vallen ook aspecten als een *code standards* en tijdens het project vergaarde best practices.

Test tool
Een uitermate belangrijke activiteit in een ontwikkelproces is het testen van een systeem. Voor wat betreft Java/J2EE-applicatieontwikkeling onderkennen we TDD, Test Driven Development, als bruikbare methodiek. We plaatsen er wel een kanttekening bij. Het lijkt soms dat testen op zich een doel wordt voor een project, terwijl het uiteindelijk natuurlijk gaat om het op tijd en binnen budget opleveren van de gevraagde software. Dit is iets om rekening mee te houden. Het kan wat ons betreft niet zo zijn dat door de testinspanning de grenzen van tijd en geld zonder meer opgerekt worden. Het gebruik van in ieder geval JUnit wordt in onze aanpak verondersteld (http://www.junit.org/index.htm).

Build tool
Kunnen we ook kort over zijn: gebruik ANT en val niet standaard terug op de faciliteiten die door een ontwikkelomgeving geboden worden om eenvoudige deployment te doen. Met ANT kan software ongeacht de IDE gepackaged en gedeployed worden (http://ant.apache.org/).

Version control
Uiteraard is goed versiebeheer uitermate belangrijk bij het ontwikkelen van software, zeker op het moment dat het gaat om grote(re) ontwikkelteams. Met versiebeheer wordt het mogelijk om op een goede manier softwarereleases uit te voeren en support te leveren op uitgeleverde releases. Wij gebruiken in de meeste gevallen CVS (http://ximbiot.com/cvs/wiki/index.php?title=Main_Page).

Dependency management
Het kunnen vastleggen en bewaken van afhankelijkheden van je software op enerzijds componenten die buiten je bereik vallen, bijvoorbeeld Java-library's en anderzijds eigen gebouwde componenten en projecten is uitermate zinvol. Hiermee zijn onder andere potentiële versieverschillen tussen afhankelijke bouwstenen te voorkomen. Cumquat gebruikt hiervoor Ivy, een tool die naadloos aansluit bij ANT en ons continuous integration component (http://www.jayasoft.org/).

Continuous integration
Het voordeel van continuous integration is dat op één centrale plaats, op een gecontroleerde wijze en tijdstip(pen) een complete build cyclus doorlopen kan worden. Onderdelen hiervan kunnen onder meer zijn, het ophalen van de broncode uit het versiebeheer systeem, het uitvoeren van JUnit integratietesten en het packagen van de applicatie. Hierdoor kan op een proactieve manier een potentieel probleem gedetecteerd worden, voor dat het 'te laat' is. In onze aanpak maken we gebruik van Cruisecontrol (http://cruisecontrol.sourceforge.net/).

Issue management
Het kunnen bijhouden en bewaken van geconstateerde problemen of nieuwe wensen en eisen aan de software is van groot belang. In de meeste gevallen kunnen dergelijke systemen gekoppeld worden aan het

versiebeheer systeem, waarmee een nog efficiëntere wijze van werken mogelijk wordt. Gebruik bij voorkeur bijvoorbeeld geen Excel-spreadsheet, maar stap over op een alternatief als Bugzilla (http://www.bugzilla.org/). Afhankelijk van het budget zou overwogen kunnen worden om JIRA (http://atlassian.com/software/jira/) aan te schaffen.

EEN OPEN BLIK We hebben in dit artikel gezien hoe er kan worden aangekeken tegen de rol van het (Oracle) RDBMS als onderdeel van Java/J2EE-applicatieontwikkeling. Daarnaast hebben we beschreven op welke manier frameworks een bepalende rol spelen bij het productiever kunnen ontwikkelen en hebben tevens een aantal framework-stacks gepresenteerd en beargumenteerd. Tenslotte zijn een aantal andere belangrijke onderdelen van de ontwikkelstraat beschreven.

Jan Vissers (e-mail: Jan.Vissers@cumquat.nl) is technology manager bij Cumquat Information Technology, een bedrijf dat zich bezighoudt met dienstverlening en product-development rondom Oracle, XML en Java/J2EE-technologie.

PATCHES Patches PATCHES Patches PATCHES Patches PATCHES

Artikelen over onderwerpen als software-ontwikkeling, Java, UML, eXteme Programming en nog veel meer vindt u in het Online Archief van Array Publications. Vaktijdschriften als Software Release, Java Magazine, Database Magazine en ons Oracle vakblad Optimize hebben hun artikelenarchief online gezet. Dankzij de heldere zoekstructuur vindt u snel wat u zoekt op www.release.nl.

Software AG ondersteunt Google maps met AJAX technologie

Composite Application Integrator (CAI), Software AG's krachtige ontwerp- en gebruiksomgeving om webservices te combineren met gebruikersvriendelijke internetapplicaties, ondersteunt nu Google maps. CAI is het eerste op JavaScript (AJAX) gebaseerde product dat een Google map control bevat. Bedrijven die bestaande bedrijfsprogrammatuur omzetten naar internet-toepassingen kunnen nu via CAI snel Google maps integreren. Zowel integratie van als het uitwisselen met Google maps is mogelijk. De communicatie met Google maps is slechts een van de meer dan 80 AJAX UI controls die beschikbaar zijn binnen CAI.

Met CAI kunnen snel rijke samengestelde Internet-toepassingen worden ontwikkeld op basis van bestaande bedrijfsprogrammatuur en -gegevens. Zo kunnen bestaande applicaties, ongeacht hun technologie, zonder veel inspanning worden opgenomen in een Service Oriented Architecture. CAI applicaties zijn volledig browser-gebaseerd, maar tegelijkertijd net zo gebruikersvriendelijk als traditionele PC-applicaties. De rijkdom van desktop toepassingen is beschikbaar binnen een webbrowser zonder dat ontwikkelaars complexe JavaScript codes moeten programmeren.

Toepassingen kunnen snel worden ontwikkeld voor een brede waaier van Java runtime omgevingen, inclusief alle belangrijke applicatieservers, waar ze met een browser of via een rich client (Java SWT client) kunnen worden benaderd. CAI-toepassingen kunnen ook in bestaande portalen worden geïntegreerd, zoals IBM WebSphere en SAP NetWeaver. PC gebruikersfaciliteiten zijn nu beschikbaar in een browser zonder de noodzaak voor ontwerpers om complexe JavaScript te gebruiken. De verzameling van AJAX UI controls die met CAI geleverd wordt, bevat ook de Google maps control. Composite Application Integrator is online verkrijgbaar, zowel als testversie of als productieversie via http://www1.softwareag.com/Corporate/products/cai/default.asp.

Conferentie JavaPolis 2005 in Antwerpen

De conferentie Javapolis vindt dit jaar voor de vierde achtereenvolgende keer plaats. JavaPolis is het grootste en belangrijkste evenement op Java-gebied in Europa en is een jaarlijks initiatief van 'the Belgian Java User Group' (BeJUG) of kortweg de Belgische Java-gebruikersgroep. Het evenement vindt plaats van 12 tot en met 16 december 2005 in Metropolis in Antwerpen. Er worden dit jaar ongeveer 2000 JavaPolians verwacht.

Java-developers, architecten en IT-managers kunnen er kennismaken met de laatste nieuwtjes op het gebied van Java-technologie, tools en toepassingen. Daarnaast kunnen ze deelnemen aan diverse cases, technische breakout sessies en diepgaande presentaties. Ook dit jaar staan er opnieuw grote namen uit de Java-wereld op het programma en zullen er meer dan 80 uitstekende sprekers aanwezig zijn. 12 en 13 december zijn University-dagen met ondermeer praktijkgerichte sessies en diepgaande presentaties over EJB 3.0, AJAX, Groovy, JSF, Desktop Java, XML, Agile Development en WebServices. 14,15 en 16 december zijn de zogenaamde Conferentie-dagen met meer dan 60 presentaties en cases over J2ME, J2SE, J2EE, Open-Source, Methodology, Architecture, Security & Messaging.

Op dinsdag 13 december zal er ook een Rapid Application Development (RAD) Race plaatsvinden. Vijftien internationale teams van telkens 2 ervaren Java-ontwikkelaars moeten binnen een tijdslimiet van 12 uur een levensechte business case oplossen en een programma aanleveren. De winnaar wordt verkozen door een onafhankelijke groep van IT-specialisten en zal bekend gemaakt worden tijdens de conference keynote. Op 15 december kan men zoals gewoonlijk ook genieten van een mooie film. Voor meer informatie: www.javapolis.com.