Als je de titels en abstracts van de bijna 400 presentaties die er op JavaOne 2008 werden gegeven doorloopt, is er eigenlijk maar één conclusie mogelijk: "UI is hot, persistence… not". Hoewel de allereerste Java frameworks van enigerlei betekenis zich op het ophalen en persisteren van gegevens uit en naar (relationele) databases richtten, zijn het nu dingen als Web 2.0, AJAX, "server push" en mobile applications waar de aandacht van de Java community naar uitgaat. Swing en zelfs Applets zijn, hoewel vaak een beetje verstopt achter scripting talen en frameworks, weer helemaal terug.

Persistence op JavaOne 2008

Het aantal sessies dat zich met persistence bezig hield was op de vingers van twee handen te tellen. Het merendeel daarvan betrof, zoals te verwachten viel, het werken met de Java Persistence API (JPA/EBJ3.0). Hier en daar hoorde je geluiden over andere vormen van persistence dan via een relationele database, bijvoorbeeld in "data grids" waar gegevens "in memory" worden opgeslagen in grote grids (of "clouds" van computers. Maar in een tijd van globalisering waarin de hoeveelheid data die door sytemen verwerkt wordt enorm toeneemt, verbaast het mij dat er niet meer aandacht was voor producten als Oracle Coherence (voorheen Tangosol), JBoss Cache, en andere gridstorage-oplossingen. Maar zelfs de twee grootste open source JPA-implementaties, Hibernate en EclipseLink, waren slechts zeer bescheiden aanwezig. Ook over objectgeoriënteerde databases, waarbij de overhead van het definiëren van object-relational mappings overbodig is en die volgens voorstanders al jaren vergelijkbare of zelfs betere performance bieden dan relationele databases in combinatie met ORM frameworks, was er zeer weinig te vinden. Desondanks heb ik een aantal interessante sessies rondom persistence bijgewoond, waarvan hieronder een korte beschrijving.

Java™ Persistence API 2.0

Wellicht de meest veelbelovende sessie over de toekomst van Java persistence was de "Java Persistence API 2.0" presentatie die door Linda DeMichiel en Marina Vatkina, beiden van Sun Microsystems, werd verzorgd. Deze sessie had tot doel een "sneak preview" te bieden van het werk dat er momenteel gedaan wordt om de 2.0 versie van de Java Persistence API die in Java EE5 werd geïntroduceerd tot stand te laten komen.
Het doel van de Java Persistence 2.0 API is vooral om de scope van de API te vergroten en features in te bouwen waar de Java community in de afgelopen jaren om heeft gevraagd. Dit zijn onder andere:

- Uitgebreidere modelling- en mapping-mogelijkheden, zoals ondersteuning voor collecties van embedded objects, ordered listst, etc.
- Grotere flexibiliteit in het combineren van bestaande mapping-opties.
- Uitbreidingen aan de JPA query language
- Standaardisering van "hints" voor de configuratie van Entity Managers en queries
- Standaardisering van mogelijkheden voor het omgaan met "detached objects"

Soms kom je uit een "sneak preview" sessie met een gevoel van anticipatie, van "ik kan niet wachten tot dit er is". Dat gevoel had ik hier niet. Weliswaar worden er een aantal zeer nuttige zaken aan de standaard toegevoegd die nu niet of alleen via vendor-specifieke uitbreidingen te implementeren zijn, maar het gaat om verfijningen meer dan om forse uitbreidingen. Ik heb niet het gevoel dat de bekende JPA frameworks veel problemen zullen hebben om de 2.0 standaard te gaan ondersteunen.

Developing Java™ Persistence API Applications with the NetBeans™ IDE and EclipseLink

Zoals waarschijnlijk bekend, heeft Oracle begin vorig jaar het persistence framework "Toplink" gedoneerd aan de Eclipse Foundation, waar het als het ambitieuze open source project "EclipseLink" verder gaat. Toplink is het oudste Object Relational Mapping (ORM) framework (oorspronkelijk ontwikkeld in SmallTalk bestaat het al langer dan Java zelf), maar kon als gelicenceerd product de concurrentie met het in features minder rijke Hibernate niet aan. De ontwikkelaars van Toplink zijn echter de drijvende kracht achter de JPA specificatie geweest, leverden met "Toplink Essentials" de eerste (gratis) referentie implementatie van JPA, en nu het open source product EclipseLink de JPA 2.0/EJB 3.1 referentie implementatie wordt en onder andere in de GlassFish v3 applicatie server gebruikt zal worden, bestaat daarmee de mogelijkheid dat het op termijn wel een serieuze concurrent kan zijn.

Een sessie over de integratie van EclipseLink met de steeds populairder wordende NetBeans IDE van Sun zou hiervan, zo verwachtte ik in ieder geval, een van de voortekenen kunnen zijn. Alle ORM-frameworks maken gebruik van meta-data, die de manier beschrijft waarop de vertaling tussen het Object model

en het relationele database model moet plaatsvinden. In JPA wordt deze informatie vastgelegd in XML files en/of annotaties. TopLink heeft zich al in een vroeg stadium gerealiseerd dat deze meta-data enerzijds cruciaal is voor de applicatie, en anderzijds dat het onderhouden van deze informatie lastig en zeer foutgevoelig werk kan zijn. Daarom had Toplink behalve een runtime omgeving ook altijd de "Mapping Workbench", een krachtige, standalone IDE die het zeer eenvoudig maakt om deze meta-data aan te maken, te bekijken, te modificeren en (heel belangrijk) te valideren. Deze "Mapping Workbench" is beschikbaar als standalone product, maar is ook in Oracle's eigen Java IDE, JDeveloper, geïntegreerd zodat je binnen één IDE het Object model, database ontwerp én de mapping ertussen kan onderhouden op een declaratieve manier. Mijn verwachting bij het begin van deze sessie was dus dat ik te zien zou krijgen hoe de "Mapping Workbench" nu ook in de NetBeans IDE verwerkt zou zijn.

Helaas bleek hier geen sprake van. Tijdens de sessie, verzorgd door Toplink product manager Doug Clarke van Oracle en Andrei Badea, NetBeans Software Engineer van Sun Microsystems, werd getoond hoe je door middel van wat properties in de persistence.xml file kon configureren dat EclipseLink als JPA provider gebruikt kon worden, dat je wat type-in support kreeg bij het aanmaken van annotaties, dat je automatisch database tabellen kon aanmaken op basis van Java classes, en andersom dat je Java classes kon gegeneren op basis van database tabellen. Weinig schokkend allemaal, zeker als je aan de Mapping Workbench of aan JDeveloper gewend bent.

The Java™ Persistence API on the Grid

Vorig jaar maakte ik op Oracle OpenWorld voor het eerst kennis met het "Oracle Coherence" product, dat Oracle kort daarvoor had overgenomen van Tangosol. Sindsdien ben ik bijzonder geïnteresseerd in het concept van in-memory data storage grids; vrijwel oneindig schaalbare, bijzonder robuuste en in vergelijking met disk-based opslag bloedsnelle opslag en toegang tot data. Het Oracle Toplink (nu EclipseLink) product ken ik uit eigen ervaring en ben er een groot fan van. Niet alleen de titel en abstract van deze sessie (waar volgens mij zeer bewust noch EclipseLink, noch Coherence in genoemd worden), maar vooral de sprekers, Doug Clarke en Mike Keith van Oracle, maakten dat ik deze BOF (Bird of a Feather) avondsessie niet wilde missen. Ik werd niet teleurgesteld: het ging hier inderdaad om een beschrijving van een extreem schaalbare architectuur met EclipseLink als JPA provider, gekoppeld aan een Coherence datagrid als gedistribueerde L2 cache. Doordat Coherence veel meer kan dan alleen het in-memory vasthouden van data, maar zelf ook weer op allerlei manieren van en naar databases kan lezen en schrijven, zijn er verschillende configuraties en niveaus van samenwerken tussen EclipseLink en Coherence mogelijk, die tijdens deze sessie aan bod kwamen. De sessie werd besloten met het benadrukken dat het hier niet persé een performance, maar vooral een schaalbaarheids oplossing betrof, en "dat dit volgend jaar een veel groter verhaal zou zijn". Ik ben zeer benieuwd!

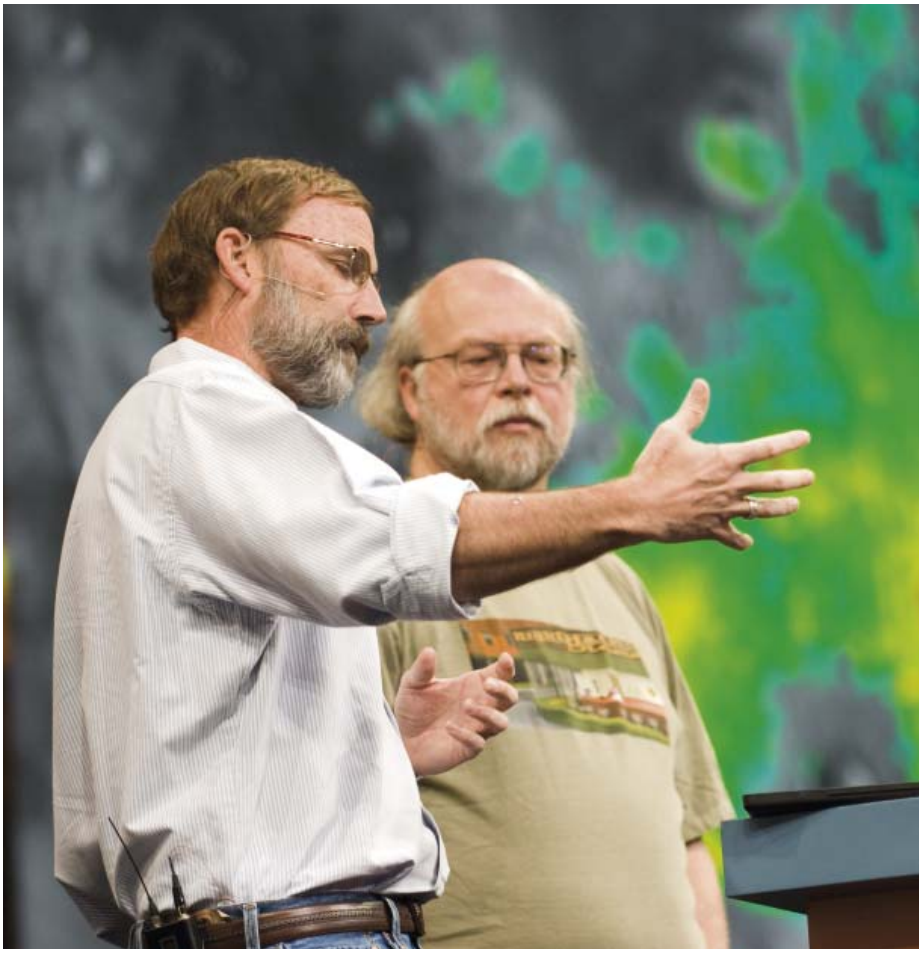

Foto: Dré de Man

Persistance was niet populair op JavaOne, maar niettemin in bijna alle applicaties aanwezig

Conclusie

Het was dit jaar stil op JavaOne rondom persistence; de User Interface domineerde het congres. Deels zal dit het gevolg zijn van de hoge mate van maturity die de grote persistence frameworks al bereikt hebben. Zelfs de presentatie over de aankomende JPA 2.0 standaard had een hoog "puntjes op de i" gehalte en bevatte weinig schokkende nieuwe functionaliteit. Toch zijn er mijns inziens genoeg interessante ontwikkelingen die wat (meer) aandacht hadden verdient. Zo is ORM maar één van de onderdelen uit het EclipseLink project, en zijn mappings naar XML, JCA, SDO en XML-aware databases, die ook binnen het totale EclipseLink project vallen, mijns inziens zeker relevant en vernieuwend. De echte eye-opener was voor mij de sessie over "JPA on the grid", die aangaf dat vastomlijnde grenzen tussen zaken als databases en applicatieservers, clusters en grids, geheugen en harddisks langzaam aan het vervagen zijn. «

Peter Ebell
AMIS Sevices