



# Moving Java forward

*Lucas Jellema, Amis; Paul Bakker, Open Source Amdatu PaaS Platform; Bert Ertman, Java User Group Leader for NLJUG, Netherlands*

*Anfang Oktober war die Woche der Wahrheit für Java: Die JavaOne 2011 fand zum zweiten Mal unter der Regie von Oracle in San Francisco statt. Seit Oracle das Sagen hat, findet die JavaOne zeitgleich mit der Oracle OpenWorld statt. Das hat Vorteile wie zum Beispiel das Konzert von Sting, stellt aber auch große Herausforderungen an den Charakter dieser Java-Konferenz. In diesem Jahr sollte sich zeigen, ob unter der Verantwortung von Oracle die Java-Plattform und die Java-Community wieder zunehmen, nachdem die Zahlen in den letzten Jahren bestenfalls stagniert haben. In diesem Artikel beschreiben Paul Bakker, Bert Ertman und Lucas Jellema – alle drei waren Sprecher auf dem JavaOne – wie sie die Konferenz erlebt haben und was für sie die wichtigsten Ergebnisse hinsichtlich der Zukunft von Java sind.*



Gleich zu Beginn die größte Herausforderung: Eines der vorn anstehenden Probleme von Oracle nach der Übernahme von Sun ist der Umgang mit der Entwickler-Community. Keine andere Software-Entwicklungsplattform kennt eine derart bestimmende Community wie Java. Die Teilnahme an dieser Community ist etwas Besonderes. Für den Lieferanten hingegen macht das die Arbeit schwierig. Doch hat der Erfolg von Java viel damit zu tun, wie Sun in den vergangenen sechzehn Jahren mit der Community zusammengearbeitet hat. Ohne Community würde es heute kein Java geben. Nach der Übernahme von Sun will Oracle diesen Prozess steuern – ähnlich wie der Hersteller auch versucht, die Usergroups seiner anderen Produkte zu beeinflussen. Nicht zu vergessen: Die Java-Benutzer sind (potenzielle) Kunden für Oracle. Die letzten eineinhalb Jahre haben jedoch deutlich gezeigt, dass sich die Entwickler-Community nicht wie eine Usergroup steuern lässt. Hinzu kommt, dass Oracle, ein Unternehmen mit vielen Java-Entwicklern, den Ruf des „Abzockers“ hat. Es sollte daher ein anderer Weg eingeschlagen werden, der schnell zum Erfolg führen sollte, sonst könnte Oracle die letzte Glaubwürdigkeit als Hüter von Java verlieren.

Nach dem Aufruhr im Vorfeld war klar, dass etwas geschehen musste. Oracle hat daher beschlossen, auf verschiedenen Gebieten aktiv zu werden und hat die Verantwortlichen der Java-Community (Java-User-Group-Leader und Java-Champions) zu strategischen Briefings eingeladen. Zudem hat Oracle monatliche Telefonkonferenzen organisiert, um nahe bei den Entwicklern zu sein und um zu erfahren, was die Community bewegt. Diese Art von Offenheit war neu für Oracle. Es war deutlich zu sehen, wie unbequem es für das Unternehmen war. Trotzdem wurde teilweise

das verlorene Vertrauen wiedergewonnen. Die Anerkennung der Entwickler-Community ist ein ganz wichtiger Punkt und ein Gewinn für die Java-Entwickler. Java ist ein technologisches Ökosystem, in dem neben Oracle auch andere Firmen und Personen eine Daseinsberechtigung haben. Oracle ist der Verwalter von Java, hat aber nicht das alleinige Sagen. Das Einbringen aller ist sehr wichtig für die Existenz dieser Plattform.

Das letzte Jahr stand im Zeichen der Klage gegen Android von Google und des Verlassens des JPC unter anderem durch



*Abbildung 1: Adam Messinger, Vice President Development für Oracle Fusion Middleware (links), sowie Cameron Purdy, Oracle Vice President Development, auf der JavaOne 2011*



Apache. Beide Themen wurden übrigens von den Offiziellen der JavaOne-Konferenz geschickt umgangen. Trotzdem sieht man, wie hinter den Kulissen die Rechtsabteilungen krampfhaft alles kontrollieren und nicht aus den Händen geben wollen.

Es gibt aber auch positive Entwicklungen: Mit „JCP.next“ ist der JCP transparenter geworden. Es gibt wieder neue Versionen der Java-Plattform, die durch den JCP ratifiziert werden. Neue Gruppierungen nehmen am OpenJDK und beim JCP teil, das Ökosystem um Java braucht schließlich eine richtige Wiederbelebung. Noch ist nicht klar, wie groß der zuvor entstandene Schaden ist, aber das Wichtigste ist, der „Patient“ lebt noch und kann weiterentwickelt werden. Java bewegt sich wieder vorwärts, das war ein guter Start in die JavaOne.

## Cloud

Der wichtigste Trend in diesem Jahr hieß eindeutig „Cloud“. Obwohl das Thema bereits seit ein paar Jahren wichtig ist, sieht man jetzt, dass tatsächlich etwas entsteht, statt nur darüber zu reden. Ein intensiver Track mit Cloud-Themen, Best-Practice-Sessions von Early Adopters sowie von Cloud-Providern hat PaaS-Lösungen vorgestellt und in Java-EE-orientierten Sessions demonstriert, wie Cloud zu handhaben ist. Inzwischen ist klar, dass eine Cloud nicht nur für Benutzer mit extrem vielen Daten interessant ist. Auch die Skalierbarkeit ist ein wichtiges Thema. Mit PaaS verschwindet der Application-Server in der darunterliegenden Infrastruktur, was wichtige Voraussetzungen dafür schafft, wie Applikationen entwickelt und ausgerollt werden können. Es geht darum, das „ease-of-development“ zu gewinnen. Das reduzierte, POJO-basierte Programmier-Modell mit vielen Java-APIs und Frameworks sowie die DevOps-Bewegung (kommt von Development und Operations) lieferten dazu einen wichtigen Beitrag.

## Modularität

Nachdem es in den letzten Jahren den Anschein hatte, OSGi-Technologie sei auf der JavaOne verboten, gab es in diesem Jahr eine gute Sichtbarkeit. Neben reinen OSGi-Themen standen mehrere Sessions über Modularität im Allgemeinen auf dem Plan. Jeder scheint inzwischen damit ein-



Abbildung 2: Treffpunkt der Community unter freiem Himmel in der Mason Street

verstanden zu sein, dass Modularität gewährleistet sein muss, insbesondere wenn Applikationen in einer Cloud laufen sollen. Für die genaue Modularitäts-Lösung ist aber noch einiges zu klären. Alle Augen sind dabei auf das Jigsaw-Projekt gerichtet, das Teil von Java 8 ist.

Jigsaw startete vor ein paar Jahren, um den JDK selbst modular zu machen. Der JDK ist durch Hinzufügen von immer mehr Features und APIs in den letzten zehn Jahren stark gewachsen und hat mittlerweile einen bedeutenden Umfang.

Bei manchen APIs stellt sich allerdings die Frage, wie sinnvoll diese noch sind, weil sie in modernen Applikationen fast nicht mehr verwendet werden. Bestes Beispiel dafür ist die Unterstützung von Corba. Andererseits ist es unmöglich, diese APIs einfach zu entfernen, weil dann die Rückwärtskompatibilität fehlschlägt, und die Applikationen, die abhängig davon sind, werden dann zur ewigen Anwendung einer der älteren Java-Versionen verurteilt.

Die einzige Lösung ist, das JDK modular zu gestalten, damit der Anwender nur die JDK-Teile herunterladen muss, die zum Starten einer Applikation wichtig sind. Auf JDK-Ebene ist OSGi aber keine Lösung, weil OSGi eine Ebene darüber steht. Die jetzigen Pläne für Jigsaw gehen aber viel weiter als bis zum Aufteilen von JDK. Die Unterstützung der Modularität wird auch

für Java-Entwickler nützlich sein, um eigene Applikationen modular zu machen.

Jigsaw löst das Problem mit einem „bundle“ mit der OSGi-Ebene. Wo OSGi dafür ein externes Framework und ein extra Metadaten-Format benötigt, bekommt Jigsaw Unterstützung aus der Virtual-Machine-Sprache. Zurzeit wird heftig diskutiert, ob die Art, wie Jigsaw das angeht, die richtige ist. Momentan sieht es danach aus, dass Jigsaw und OSGi zwei separate Lösungen werden, die Zusammenarbeit dieser beiden jedoch in Zukunft möglich sein wird. Die Meinungen darüber, was dies künftig für OSGi bedeutet, laufen auseinander. In jedem Fall bietet Jigsaw keine Lösung auf der Services-Ebene in OSGi. Das ist sicherlich der OSGi-Bereich, der die meisten Vorteile für das richtige modulare Gestalten von Applikationen bietet. Eine Kombination aus Modulen mit Basis-Sprachunterstützung und darauf aufsetzenden OSGi-Services könnte sicherlich eine sehr interessante Sache sein. Das letzte Wort dazu ist sicher noch nicht gesprochen.

## Java SE 8

Aus verschiedenen Gründen hat sich das Release von Java SE 7 erheblich verzögert. Nachdem seit der Version 6 mehr als fünf Jahre vergangen sind, wurde Java SE 7 mit nur wenigen Erneuerungen herausgebracht, während die großen Änderungen



## EINIGE FAKTEN UND ZAHLEN

*Wie im letzten Jahr fand die JavaOne zusammen mit der Oracle OpenWorld statt. Beide Events sind an San Francisco nicht ganz spurlos vorbeigegangen – es wurden insgesamt rund 46.000 Besucher gezählt, die meisten (ca. 40.000) davon auf der OpenWorld. Im weiten Umkreis der Stadt war die ganze Woche kein Hotelbett mehr zu bekommen. Die zahlreichen Besucher kamen nicht nur zu beiden Konferenzen, sie brachten auch ziemlich viel Taschengeld mit. Nach Angaben der Stadt wurden in dieser Woche ca. 100 Millionen Dollar extra ausgegeben – ein bedeutender Anteil davon im lokalen Apple Store. Die OpenWorld konzentrierte sich auf das Moscone Convention Center und ein paar große Hotels in direkter Umgebung. Die JavaOne fand einige Blocks weiter nordwestlich statt. „The Zone“ nennt sich das Dreieck aus drei großen Hotels, wobei das Hilton im Mittelpunkt steht. Die Mason Street zwischen den Hotels war gesperrt. Dort war für die JavaOne-Besucher eine Lounge-Area aufgebaut, in der soziale Aktivitäten stattfinden. Oracle gibt die konkrete Anzahl der Besucher offiziell nicht bekannt, Insider sprechen von etwa 6.000. Damit hätte sich die Besucherzahl im Vergleich zum Vorjahr verdoppelt. Es gab rund 400 Vorträge, logistisch war einiges verbessert. Gleichartige Sessions fanden oft im gleichen Raum statt, in jedem Fall aber im gleichen Hotel. Wer an allen Aspekten von Java interessiert war, hatte einige Kilometer zurückzulegen.*

erst mit Java SE 8 kommen sollen. Java SE 7 steht mittlerweile seit Juli 2011 zur Verfügung. Während der JavaOne ist auch erwähnt worden, dass die offizielle Version für Mac OS X für Ende 2011 geplant ist und direkt darauf Anfang 2012 ein General-Availability-Release. Übrigens ist der Technology Compatibility Kit (TCK), mit dem bestimmt wird, ob ein JVM tatsächlich SE-7-zertifiziert wird, noch nicht fertig. Oracle nennt als Grund dafür das Fehlen von Anwälten und kündigt den TCK noch für das vierte Quartal 2011 an.

Weil Java SE 8 große Erneuerungen bringen soll, die zeitaufwändig sind, und weil große Firmen bei Oracle und beim OpenJDK-Projekt verkündet haben, dass eine Periode von ein bis zwei Jahren zwischen zwei wichtigen Java-Releases zu kurz ist, wird Java SE 8 bis zum Summer 2013 erwartet. Die wichtigsten Themen werden dabei die Modularität (Projekt Jigsaw) sowie das Projekt Lambda (closures und bulk parallel operation in Java collections sowie filter/map/reduce) sein.

Durch die Verschiebung des Release-Datums können in Java SE 8 auch Funktionalitäten integriert werden wie die neue Date-Time-API (JSR 310, auch bekannt als JodaTime), kleinere Verbesserungen in der Sprache, Security Updates, erneutes Net-

working, Type Annotations, eine neuere JavaScript-Engine (Projekt Nashorn) sowie Support für moderne Device-Funktionen wie Multi-Touch-User-Interface, Kompass-Integration-Tool sowie Kamera-Lokation-Bestimmung. Ein wichtiges Ziel bei Java SE 8 ist für Oracle die verbesserte Interaktion zwischen verschiedenen JVM-Sprachen. Die neue JavaScript-Engine soll außerdem ein gutes Beispiel sein für die Interaktion, die dann auch mit anderen JVM-Sprachen wie Groovy oder Scala nutzbar ist. Es sieht ferner danach aus, dass in Java SE 8 das Projekt Avatar (siehe unten) auch Funktionalitäten für HTML5 und WebSockets bringen wird. Mit dem OpenJDK-Projekt stehen Teile davon kurzfristig in Early-Access-Downloads zum Selbst-Entdecken zur Verfügung. Dies ist wiederum ein Zeichen der Community mit dem Ziel, Dinge zu ändern. Die gesamte Entwicklung von Java ist damit transparent und öffentlich – unfassbar für die übliche Produkt-Entwicklung innerhalb von Oracle.

Oracle hofft, kurzfristig nach Java SE 8 die JavaFX-3.0-API als Teil der SE-Spezifikation zum kurzfristigen Nachfolger der jetzigen Swing-API machen zu können. Das Hinzufügen von JavaFX (siehe unten) über die Erneuerungen auf diesem Gebiet wird ziemlich kontrovers diskutiert. Bisher wur-

de Java FX vollständig außerhalb von JCP und OpenJDK von Oracle selber entwickelt und ist zurzeit kein Bestandteil des Java-Standards. Wenn Java FX kurzfristig Swing als Standard für die UI-Entwicklung auf der Java-Plattform ersetzen soll, ist es wichtig, dass es Teil der Java-SE-Plattform wird. Dadurch muss es allerdings zuerst den JCP passieren, was nicht nur eine Formalität ist. Oracle ist zwar Verwalter von Java, hat aber nicht das alleinige Sagen innerhalb des JCP. Hier wird sich zeigen, wie gut der JCP funktioniert.

Es gibt auch schon Pläne für Java 9. Ein interessantes Thema dabei ist eine sich selbst tunende Virtual Machine, um die Abhängigkeiten von einigen (manchmal unklaren) Command-Line-Parametern zu verringern. Andere Themen, die in Gerüchten kursieren, sind – neben selbstverständlich weitergehender Modernisierung und Verbesserung von Java als Programmiersprache – die Integration verschiedener Sprachen auf dem JVM und eine bessere Integration mit Native (O/S) Libraries und Bestimmungen. Ganz der NoSQL-Bewegung folgend sind Bestimmungen geplant für „Big Data“, große In-Memory-Daten-Sammlungen, die die jetzige Java-Speichergrenze von zwei GB für Arrays erhöhen soll. Auch soll Java 9 nach jetzigen Plänen eine neue Meta-Objekt-Beschreibung bieten, um einfachen Objekt-Austausch zwischen Sprachen zu ermöglichen. Mehr Unterstützung für Advanced Concurrency auf Multi-Core-Plattformen ist ebenfalls denkbar. Hinzu kommt „reification“, ein Tool, um den Information-Type von Objekten unter „runtime“ zur Verfügung zu stellen und die von vielen gewünschte Korrektur um die ursprüngliche Implementation von Generics in Java SE 5 zu ermöglichen.

Multi-Tenancy steht auch auf der Liste für Java 9, um in Cloud-Umgebungen gleichzeitig mehrere Applikationen innerhalb der gleichen JVM ablaufen zu lassen. Damit verwandt ist die Unterstützung für das Ressourcen-Management bei Cloud-Applikationen. Gleichzeitig ist dieses Thema auch in der nächsten Java-EE-Version geplant.

„Continuations“ ist eine Programmiersprache basierend auf Lambda-Expressionen, die einen fast asynchronen Request von lokalen Methoden unterstützt, wobei



eine Request-Methode keinen Wert wiedergibt, sondern vom „Requestor“ ein Objekt erhält, in dem die Ergebnisse platziert werden können. In Java 9 wird erwogen, Continuations zum Teil von Java zu machen.

Gleichzeitig ist geplant, Java SE und ME wieder zusammenzuführen – mit einem eingeschränkten (core) Footprint, der auf allen Plattformen zur Verfügung stehen wird. Ein weiteres Ziel der näheren Zukunft liegt darin, dass wieder überall (von Card bis zu EE und Cloud) die gleiche Java-Sprache angewendet werden kann.

Die Kooperation mit dem OpenJDK-Projekt sieht auch sehr gut aus. Neben Oracle, RedHat, IBM, SAP und Apple sind auch Azul und Twitter vor Kurzem hinzugekommen. Der Community-Process scheint sich auch hier wieder zu bewegen.

#### Java EE 7

Java EE 6 ist mittlerweile fast zwei Jahre alt. Inzwischen ist Java EE durch sieben Container unterstützt und damit häufig als Mainstream-Technologie in der Praxis anzutreffen. Die meisten Container sind sehr leicht und starten in wenigen Sekunden, was ein sehr großer Entwicklungsschritt gegenüber den traditionellen Applikations-Servern bedeutet. Das ist wiederum ein wichtiger Schritt in Richtung „Cloud“. In der Cloud wird eine Applikation oft nach Nutzen abgerechnet. Das unnötige Einsetzen von Speicher kostet Geld und ein kleiner Memory-Footprint ist damit sehr wichtig geworden. Daneben ist eine schnelle Startzeit essenziell für das richtige Hoch- und Runterskalieren eines Clusters in der Cloud. Obwohl Java EE 6 damit jetzt erst für die meisten Entwickler interessant wird, ist die Entwicklung von Java EE 7 bereits in vollem Gang. JSRs und Expert-Groups werden gebildet und neue APIs veröffentlicht. Manche Spezifikationen wie JAX-RS 2.0 sind bereits größtenteils klar. Das Hinzufügen von APIs ist in einigen Container-Implementierungen bereits integriert, an anderer Stelle sind noch Änderungen erforderlich, wenn beispielsweise JAX-RS-2.0 die Lücken stopfen wird, über die Entwickler in der Praxis gerade stolpern. Für andere Spezifikationen wird noch über neue Ideen nachgedacht. Ein Beispiel dafür ist die Multi-Tenancy-Unterstützung in JPA.

Hier ist noch Input aus der Community erforderlich, um die richtige Form zu finden. Der Ruf dazu erfolgte auch während der JavaOne. Die meisten JSRs sind noch offen und das Feedback ist einfach zu geben, genauso wie das Beteiligen an einer Mailingliste. Ein anderes wichtiges Statement über Java EE 7 besagt, dass es nicht nur um die Cloud geht. Trotz der Cloud-Unterstützung werden bei Java EE 7 sicherlich auch andere Bereiche der Plattform verbessert. Man denke beispielsweise an CDI 1.1, JMS 2.0 und EJB 3.2. Auch eine Java-API für das Manipulieren von JSON steht auf dem Programm. Wenn es um die Cloud geht, sind folgende Themen wichtig für Java EE 7:

- Multi-Tenancy
- Cache/Data-Grid APIs
- NoSQL
- Deployment

#### JavaFX

JavaFX war auf JavaOne prominent positioniert. Dabei wurde vor allem deutlich, dass JavaFX nicht länger eine Spielzeug-Technologie ist, sondern strategisch

wichtig für Oracle. So war es auch keine Überraschung, dass ein paar wichtige Ankündigungen dazu gemacht wurden:

- JavaFX wird im Laufe des nächsten Jahres vollkommen Open Source, ähnlich wie OpenJDK
- Es kommt eine Spezifizierung (JSR) für JavaFX
- Die Mac-Version von JavaFX erscheint noch in diesem Jahr (die Beta-Version steht bereits zur Verfügung)
- Die Linux-Version kommt Anfang 2012

In den Keynotes sind ein paar beeindruckende Demos von JavaFX-Applikationen gezeigt worden. Vor allem die 3D-Demos mit importierten Modellen zeigten, wie stark diese Technik auf dem grafischen Gebiet ist. Es waren auch Demos zu sehen, bei denen JavaFX-Applikationen auf einem Tablet PC abliefen. Die Unterstützung des iOS steht aber noch nicht auf der Roadmap, die Community wird noch zu deren Wichtigkeit befragt. Wie schon letztes Jahr verkündet, ist JavaFX jetzt eine richtige Java-Technologie. JavaFX-Script gibt es nicht

## REST

*In vielen Gesprächen auf der JavaOne war „REST(full)“ aus unterschiedlichen Gründen ein wichtiges Thema. REST ist einfacher als SOAP/WS\*-WebServices und deshalb beliebt. Daneben kann REST in Verbindung mit JSON als Datenformat einfach auf mobilen Devices und in JavaScript-Web-Clients eingesetzt werden. Die JavaOne zeigt einen weiteren Grund für REST – die Cloud. Wenn Java-Applikationen vermehrt in der Cloud-Infrastruktur eingesetzt werden, ist auch die Administration von JVMs in der Cloud mehr und mehr gefragt. Für das Daten-Management dienen neben den bekannten Application-Servern und webbasierten Konsolen auch programmatische Mittel, basierend auf MBeans und JMX, EJBs oder auf „op-proprietary“-Protokollen wie t3 bei WebLogic. Diese Mechanismen funktionieren nicht über HTTP in der Cloud. RESTful APIs sind dann eine naheliegende Alternative.*

*REST an sich ist nicht neu und die JAX-RS-1.1-API ist Teil von Java EE 6. In JSR 339 wird JAX-RS 2.0 als Untermenge von JEE 7 entwickelt. Die betreffende Expert-Group umfasst neben Oracle unter anderem auch Ebay, RedHat und Talend. Mit JAX-RS 2.0 geht das Erwachsenwerden von REST weiter. Eigene Kern-Elemente in der JAX-RS-2.0-Spezifikation sind bessere Client-Libraries für RESTful Services, standardisierte Filter und Handler für Standard Interception und die Bearbeitung von ein- und ausgehenden Berichten, Validation von Parametern und Ressourcen mithilfe von Bean Validation, asynchrone Interaktion auf Basis der asynchronen Bearbeitung von Servlet 3.0 und die Unterstützung für Links auf andere RESTful-Ressourcen.*



mehr und die JavaFX-Applikation wird jetzt einfach in Java geschrieben. Dazu stehen APIs zur Verfügung: eine traditionelle API, die ähnlich aussieht wie Swing, ein auf dem Builder-Pattern basierende API und eine XML-basierte Variante, um Markups zu machen. Damit können Entwickler mit unterschiedlichem Background gut mit

APIs arbeiten. Weil keine separate Sprache mehr existiert, können auch alle IDEs wieder direkt für das Entwickeln von JavaFX-Applikationen angewendet werden.

Swing ist für tot erklärt. Die politisch korrekte Aussage dazu ist, dass momentan nur noch Bugs behoben und kaum neue Entwicklungen gemacht werden. Zum

Glück funktioniert die Integration zwischen Swing und JavaFX und klare Migrationswege sind im Kommen. JavaFX ist damit nicht mehr nur als RIA-Konkurrent für Silverlight und Flex positioniert, sondern muss als Standard-Java-GUI-Technologie der Zukunft gesehen werden. Die Art, wie GUI angeboten wird (Desktop oder Web), ist bestimmt durch den darunterliegenden „renderkit“. Für das Web müssen wir weiterhin Java Web Start und Applets nutzen. Für die Zukunft steht Rendering zu HTML 5 auf dem Programm.

## JMS 2.0

*Letztes Jahr gab es auf der JavaOne 2010 ein Community-Treffen, auf dem abends um zehn Uhr in einer Birds-of-a-Feather-Session (BoF) etwa 40 Personen über die Zukunft von JMS philosophierten. Zu dieser Zeit war Java EE 7 noch nicht erschienen und es gab immer noch keine Klarheit über eine JMS-Version 2.0. In dieser BoF-Session wurden Ideen ausgetauscht, etwa darüber, wie JMS verbessert werden könnte und wo die Schwachstellen sind. Jetzt, ein Jahr später, ist Java EE 7 in vollem Gange und es gibt in der Tat eine JMS-2.0-Spezifikation (JSR 343), erstellt von einer Expertengruppe, in der unter anderem Tibco, RedHat, VMware, Caucho, IBM und Oracle vertreten waren. Während der JavaOne 2011 wurde wieder in einer neuen BoF-Session diskutiert.*

*Ziel dieses JSR ist die Bereinigung der alten JMS-1.1-Schnittstellen aus dem Jahr 2003, die Modernisierung wie bei Java EE 5 und 6 (mit Annotations, Unchecked Exceptions und Injection), die Unterstützung von Funktionen, die in vielen Messaging-Produkten zur Verfügung stehen – wie Minimum Delivery Delay, Bestätigungsnachrichten von Message Delivery, Delivery Count, Batch Delivery –, und das Lösen einiger MDB-Unvollkommenheiten. Eine Folge der vorgeschlagenen Änderungen würde unter anderem bedeuten, dass dieser Code:*

```
public void sendMessage (String payload) {
try {
    Connection conn = null;
    con = cf.createConnection();
    Session sess =conn.createSession(false,Session.AUTO_ACKNOWLEDGE);
    MessageProducer producer = sess.createProducer(dest);
    TextMessage textMessage = sess.createTextMessage(payload);
    messageProducer.send(textMessage);
} catch (JMSEException e) {
    // do something
} finally {
    connection.close();
}
}
```

umgeschrieben werden kann in:

```
@Inject @JMSConnection(lookup="jms/connFactory") @JMSTDestination(lookup="jms/
inboundQueue") MessageProducer producer;
public void sendMessageNew(String payload){
    producer.send(payload);
}
```

### Projekt Avatar: HTML 5 und WebSockets

In der Strategie-Keynote kündigte Oracle das Projekt Avatar an. Grund dafür ist die Entwicklung von HTML 5. Avatar ist auf den ersten Blick ein unklares Projekt für eine sehr eindeutige Entwicklung. HTML 5 wird zunehmend als die Ablösung von Flash für Rich-Visualisierungen in Browsern und auf mobilen Geräten gesehen. Sowohl zwischen den Web-Servern und Web-Clients oder mobilen Clients als auch zwischen den Servern selbst – HTML 5 wird oft mit dem Aufstieg des WebSockets-Protokolls als leichtes, bidirektionales Kommunikationsprotokoll, das echten Push ermöglicht, assoziiert. Mit dem Projekt Avatar möchte Oracle sicherstellen, dass für alle Teile der Java-Plattform (SE, EE, ME und JavaFX) HTML 5 und WebSockets unterstützt werden und dass dieser Support konsistent ist.

Das Projekt Avatar ist „eine komplette Lösung für dynamische Rich-Client-Anwendungen“ und nennt nachdrücklich Java EE (Cloud) als Server und für HTML-5-Browser-Anwendungen, hybride Anwendungen von HTML 5 und Java in einem Browser oder auf mobilen Geräten und (Desktop-) Anwendungen als Client mit Kommunikation basierend auf JSON über WebSockets. Die bidirektionale Kommunikation sollte innerhalb Avatar genutzt werden, um zu Event-basierten Anwendungen zu kommen, unter anderem mit Benachrichtigungen bei Datenänderungen. Avatar umfasst auch eine standardisierte Methode von Offline-Möglichkeiten für Web-Anwendungen. Da es hier speziell um „Slide-Ware“ ging und keine konkreten Beispiele gezeigt wurden, ist noch abzuwarten, wie Avatar irgendwann Einfluss auf die Entwicklung der Java-Plattform



haben wird. Der Gedanke dahinter ist aber sicherlich wichtig, um Java als Plattform zusammen mit dem Web-Programmiermodell zu entwickeln.

### Oracle und das Vermächtnis von Sun

Oracle als Betreuer von Java scheint (in der zweiten Chance) nicht zu enttäuschen. Die Beziehungen innerhalb der Java-Community stabilisieren sich und die Zahlen der Teilnehmer in JCP und OpenJDK erhöhen sich. Die Zusammenarbeit zwischen kommerziellen Konkurrenten wie IBM, Apple, Oracle, SAP und RedHat / JBoss entwickelt sich in Bezug auf die Java-Plattform fruchtbar – abgesehen von der etwas unglücklichen Situation mit Google, der fehlenden Partei bei dieser Konferenz.

Oracle hat das Vermögen der Sun Microsystems jetzt eingehend untersucht. Einige Teile des Nachlasses wurden liebevoll in die Arme geschlossen (wie JavaFX, NetBeans und GlassFish), andere auf das Abstellgleis gesetzt (wie JCAPS, OpenESB, Project Kenai und jMaki). Die Stimmen aus der ehemaligen Sun-Welt sind weitgehend positiv. Nach Jahren von begrenzten Möglichkeiten aufgrund der finanziellen Lage von Sun ist nun für ausgewählte Projekte deutlich mehr Raum für Investitionen, Expansion und Wachstum. Die Stagnation, zu der es seit 2008 in allen Bereichen von Java kam (Plattform und Community), scheint gebrochen. Davon profitieren wird fast jeder.

Man betrachte zum Beispiel die Sun Hotspot JVM: Oracle hatte durch die BEA-Übernahme mit JRockit bereits eine JVM im Besitz. Diese war im Handel erhältlich für anspruchsvolle Umgebungen, in denen umfangreiche Management- und Feinabstimmungseinrichtungen erforderlich waren. Auch die Garbage Collection sollte vorhersehbar und besser angepasst sein. Oracle hat entschieden JRockit und Hotspot in einer JVM zu verschmelzen – mit den Vorteilen und Stärken von beiden. Diese Konvergenz findet im OpenJDK statt und wird in ein paar Schritten durchgeführt bis zu oder bis nach JDK 8. Zuerst kommen das JCMD Command-Line-Tool, JMX-Agent und MBeans, Java Discovery Protocol (JDP) und Flight Recorder (schon vor JDK 7) an die Reihe. Eine kommerzielle JRockit-Ausgabe mit Enterprise-Unterstüt-

zung bleibt erhalten, aber auch ohne diese Unterstützung ist JRockit jetzt schon unter der gleichen Lizenz verfügbar und damit nutzbar als freie Hotspot-JVM.

GlassFish ist ein weiteres Produkt, das sich mit einem kommerziellen Produkt von Oracle überlappt, mit dem von BEA übernommenen WebLogic Server. GlassFish ist die Referenz-Implementierung der Java EE und damit ein Eckpfeiler der Spezifikation. Sun hat GlassFish sowohl als Open-Source-Applikations-Server als auch mit einer Enterprise-Support-Lizenz angeboten. Die Community war neugierig auf Oracles Pläne.

Auch im Hinblick auf GlassFish gilt, dass Oracle nicht gern zwei vollständige Java EE Application Server unterstützen möchte. Auf der anderen Seite will Oracle mit dem WebLogic Server tatsächlich Geld verdienen, während man für die Java-EE-Spezifikation als Referenz-Implementierung GlassFish verpflichtet ist. Die Lösung ist hier, dass die GlassFish- und die WebLogic-Server-Teams eng zusammenarbeiten und gegenseitig Funktionen und Bibliotheken austauschen und gemeinsam nutzen. Somit ist die JSF-Bibliothek in WebLogic und in GlassFish identisch und wird ebenso rund um den Metro-Web-Service-Stack-Code geteilt. Mit der Zeit wird sich herausstellen, ob WebLogic eine Erweiterung von GlassFish wird – als eine kommerziell angebotene Bereicherung mit erweiterten Funktionen wie Coherence WebCache, Clustering und Verwaltung, basierend auf GlassFish mit Java EE.

Neben einer Referenz-Implementierung ist GlassFish auch ein Schaufenster für neue Ideen in Bezug auf die Änderungen, die notwendig sind, um Java EE 7 Cloud-ready zu machen. Dies bietet Input für die JCP-Expertengruppen, die sich damit beschäftigen werden. GlassFish ist damit ein vielseitiges Produkt und hat den Status „Spielplatz für ein Java-EE-Produkt“ seit Langem verlassen. Dies bedeutet in jedem Fall, dass GlassFish auch unter der Obhut von Oracle – oder vielleicht sogar stärker als je zuvor – als vollständiger, unterstützter und bezüglich Features führender Java-EE-Container eingesetzt werden kann. Die Verbreitung von GlassFish könnte damit stärker steigen.

„Wenn wir uns heute entscheiden müssten, hätten wir nie damit anfangen“, so der

Kommentar einiger Java-Verantwortlicher bei Oracle. „Aber nun, da wir es in den Schoß gelegt bekommen haben, wäre es wirklich schade, sich nicht in vollem Umfang an dem Einstieg zu beteiligen.“ Es geht hier um JavaFX – die von Sun während der JavaOne 2008 präsentierte Technologie für das Benutzer-Interface von morgen. Trotz all des Trubels rundum und des Interesses an JavaFX funktioniert die Technik nicht wirklich berauschend. Kurz vor der Übernahme von Sun bezeichnete Larry Ellison in seiner Keynote auf der JavaOne JavaFX als eines der interessantesten Schmuckstücke in der Sun-Kollektion. In der Tat hat Oracle großartig dort weitergemacht, wo Sun aufgehört hat. JavaFX-Script wurde fallengelassen und durch Java-APIs ersetzt. Dies ist ein erster Schritt in Richtung der Aufnahme von JavaFX in SE, wodurch JavaFX definitiv als Nachfolger und Ersatz von Swing und AWT begrüßt werden kann. Nicht nur auf den Desktop, sondern auch innerhalb von Applets, mit HTML-5-Rendering in üblichen, Browser-basierten Seiten und vielleicht auch wieder auf mobilen Geräten sollte die JavaFX-UI Java-Technologie-of-Choice werden. Ob es mit JavaFX funktioniert, ist sicherlich nicht nur von Oracle abhängig – aber auch dieses Erbstück ist ausgezeichnet.

Die Übernahme von BEA hat Oracle den Workshop-Plug-in für Eclipse gebracht, von Sun kam sogar eine komplette IDE hinzu, nämlich NetBeans. Wer gedacht hatte, dass Oracle bei drei IDEs (JDeveloper, Eclipse und NetBeans) sehr bald rationalisieren würde, irrte sich gewaltig. Alle drei IDEs sind weiterhin unterstützt. NetBeans ist und bleibt die Plattform für die Popularisierung der Java-Plattform in all ihren Facetten und wird deswegen als Erstes mit der Unterstützung für die neuesten Versionen von Java SE, Java EE und JavaFX ausgestattet. NetBeans wird von Oracle auch als IDE für andere JVM-Sprachen wie JRuby, Groovy und Scala positioniert. Es ist aber Aufgabe der Community, die Nicht-Java-Unterstützung in der NetBeans-Plattform zu definieren.

NetBeans 7.1 wurde während der JavaOne als Beta-Version freigegeben. Diese bietet unter anderem die Unterstützung für JavaFX 2.0. Weitere neue Features sind CSS3-Unterstützung, Tools für visuelles



Debugging von Swing- und JavaFX-Benutzeroberflächen, die in die IDE integrierte Unterstützung für Git sowie neue PHP-Debugging-Funktionen. Neben NetBeans entwickelt Oracle auch Oracle Enterprise Pack for Eclipse (OEPE) weiter – einschließlich der Integration mit dem WebLogic Server (da liegen dann auch die kommerziellen Interessen) – und es gibt eine immer mehr zunehmende Unterstützung für ADF-Entwicklung (vielleicht in Erwartung einer bevorstehenden Community-Edition). Die JDeveloper-IDE ist die dritte in der Reihe. Sie bleibt die strategische Entwicklungsumgebung für die Oracle-Fusion-Middleware-Produkte – von ADF über WebCenter-Suite bis SOA, BPM Studio und Data Integrator.

### Fazit

Nach Jahren mit sehr minimaler Weiterentwicklung oder fast Stillstand, was im Vergleich zu anderen Technologien oder dynamischen Sprachen wie .NET oder iOS/Objective C eigentlich einen Rückschritt bedeutet, ist die Java-Plattform im allen Bereichen wieder in Bewegung. Das zeigt sich im Release von SE 7 (selbst auf Mac OS X), den Plänen für SE 8, den sehr konkreten Entwicklungen und Ambitionen mit EE 7 und dem ernsthaften Weiterentwickeln von JavaFX. Oracle scheint die Verantwortung, die die Rolle als Verwalter von Java mit sich bringt, nun ernst zu nehmen – auch aus klarem Eigeninteresse. Es gab tatsächlich eine Rationalisierung, in deren Rahmen manche Projekte gestrichen wurden, aber gleichzeitig hat Oracle die Investitionen in andere Projekte im Vergleich zu Sun stark anwachsen lassen. Die Versprechungen aus dem Jahre 2010 bezüglich der Weiterführung von Initiativen und konkreten Auslieferungen sind zum größten Teil eingehalten worden. Nach einigem Erkunden und hier und da einem Knistern ist eine breit angelegte und wachsende Community mit etlichen großen Anbietern – vor allem IBM, aber auch Twitter, Azul, SAP und Red Hat / Jboss, VMWare / SpringSource – zusammen mit Oracle (selbst wenn die Kontroversen rund um Google unglücklich sind) entstanden.

Die Zahlen rund um Java bleiben beeindruckend: Die Anzahl der Geräte, auf denen Java in der einen oder anderen

Form aktiv ist, sowie die globale Reihe von Entwicklern und Anbietern ist enorm. Wer darauf besteht, dass Java zum Scheitern verurteilt oder sogar schon gestorben ist, ist ein Nachplapperer von suggestiven Blogs oder schaut nicht auf die wirklichen Fakten.

Oracle sollte dann auch in den kommenden Jahren alles daran setzen, dafür zu sorgen, dass die Community einbezogen wird. Konkrete Entwicklungen schaffen das Vertrauen, dass Java erneut wieder dabei ist, eine moderne Sprache und Plattform zu sein. Um modern weiterzuarbeiten, schicke Sachen zu machen, attraktive Benutzeroberflächen zu entwickeln, um die neueste Technologie zu erschließen und um von den modernen Kenntnissen in Programmiersprachen und Software-Entwicklung zu profitieren, kann man einfach mit der Java-Plattform weitermachen.

*Lucas Jellema  
lucas.jellema@amis.nl*

*Paul Bakker  
paul.bakker@luminis.eu*

*Bert Ertman  
bert.ertman@luminis.eu*

*Ins Deutsche übertragen von  
Michel Keemers und Daniel Dibbets*



Lucas Jellema is Oracle ACE Director für Fusion Middleware. Er arbeitet als CTO, Consultant, Trainer and Autor bei Amis in den Niederlanden.



Paul Bakker ist Senior Developer bei Luminis Technologies in den Niederlanden. Er arbeitet an der Open Source Amdatu PaaS Plattform.



Bert Ertman ist Partner von Luminis in den Niederlanden. Er leitet die niederländische Java User Group Leader (NLJUG) und ist Sun/Oracle Java Champion.

## Weitere Neuvorstellungen im Rahmen der JavaOne und Openworld

Neben JavaFX 2.0 hat Oracle in San Francisco eine Reihe weiterer Neuigkeiten vorgestellt, die auch für Entwickler interessant sind:

- Die Oracle Exalytics Business Intelligence Machine ist das weltweit erste In-Memory-Hard- und Software-System für schnellere Analysen als je zuvor. Die Lösung bietet visuelle Analysen in Echtzeit und enthält neue Typen von Analyse-Anwendungen.
- In Vorschau auf das geplante Release im November 2011 präsentiert Oracle Solaris 11. Das führende Betriebssystem für Unternehmen bietet höchste Verfügbarkeit, Sicherheit und Leistungsfähigkeit sowohl auf SPARC als auch auf x86 Systemen. Entwickelt für Clouds, erlaubt es den sicheren und außergewöhnlich schnellen Einsatz in großen Cloud-Umgebungen.
- Oracle Application Development Framework (ADF) Mobile ist eine Erweiterung des Oracle Application Development Framework und wird die Entwicklung von Java-basierten mobilen Anwendungen weiter vereinfachen. Als Teil von Oracle Fusion Middleware wird es mit Oracle ADF Mobile möglich, native und Web-Funktionen in einer mobilen Anwendung zu vereinen. Basierend auf einem Next-Generation Framework erleichtern die neuen Funktionen die Entwicklung mobiler Anwendungen weiter.
- Mit Verfügbarkeit von Oracle Exastack Optimized erweitert Oracle sein Exastack-Programm. Es erlaubt Independent Software Vendors (ISVs) und anderen Mitgliedern des Oracle PartnerNetwork, ihre Anwendungen auf Oracle Exadata Database Machine und Oracle Exalogic Cloud in punkto Geschwindigkeit und Zuverlässigkeit zu optimieren.