Die Sonne und das Orakel

Java unter neuer Führung

Ludwig Mittermeier

Oracle kauft Sun Microsystems. Was bedeutet dies für Java und den JCP? Außerdem: das OpenJDK.

Oracle kauft Sun Microsystems

Die weltweite Finanz- und Wirtschaftskrise hat die Börsenkurse drastisch nach unten gedrückt. Eine günstige Gelegenheit also für all diejenigen, die Firmen übernehmen bzw. aufkaufen wollen. Oracle hat sich in dieser Hinsicht in der Vergangenheit schon als sehr aktiv hervorgetan durch die Aufkäufe von Siebel, PeopleSoft, BEA, Virtual Iron und weiteren Firmen.

Nun steht der Kauf von Sun Microsystems durch Oracle an. Das letzte endgültige Wort ist hier allerdings noch nicht gefallen, da es den Kartellbehörden vorbehalten ist. Mal angenommen, die Übernahme läuft wie geplant vonstatten: Was könnte das für Java und insbesondere den JCP bedeuten?

Oracle-Chef Larry Ellison hat auf der diesjährigen JavaOne-Konferenz Anfang Juni in San Francisco angekündigt, Java weiterzuführen und sogar massive Investitionen in diesem Bereich zu tätigen. Darüber hinaus hat Oracle auch zugesagt, das Hardwaregeschäft beizubehalten und Gesamtangebote aus Hardware, Betriebssystem und Software (insbesondere Datenbank) anzubieten. Das klingt durchaus positiv und vielversprechend. Während Sun in den vergangenen Jahren beinahe durchgehend mit wirtschaftlichen Schwierigkeiten zu kämpfen hatte, so könnte es auch sein, dass Java nun unter der Obhut von Oracle eine neue Dynamik erfährt.

Es fehlt indes eine klare Aussage, nach welchem Modell beziehungsweise Prozess Java in Zukunft weiterentwickelt werden wird und welche Rolle der JCP dabei spielen wird.

Der Vorteil von Open Source

Wie auch immer Oracle sich entschließen wird, mit Java und dem JCP fortzufahren, für die Java Community erweist es sich nun als Glücksgriff, dass Sun sich entschlossen hat, die Java-Technologie unter eine Open-Source-Lizenz zu stellen. Dies bedeutet nämlich einen Schutz für die Investitionen all derer, die Java-basierte Produkte herstellen und vertreiben. Selbst wenn im Extremfall die Entscheidung getroffen würde (wonach es im Moment nicht aussieht), die Weiterentwicklung von Java einzustellen, so könnte auf Basis der offengelegten Quellen unter Einhaltung der Open-Source-Lizenz die Weiterentwicklung von Java durch die Community vorgenommen werden. Unter Open-Source-Lizenz (GPL) stehen die Java Virtual Machine (JVM), der Java-Compiler und die Klassenbibliothek.

Besonders die Klassenbibliothek bereitete einigen Aufwand, da sie Teile enthielt, die nicht unter einer Open-Source-Lizenz veröffentlicht werden durften. Diese proprietären Teile mussten erst nach und nach durch Open-Source-Komponenten ersetzt werden. Dazu gehörten beispielsweise Komponenten für die Soundverarbeitung, Kryptographie-Funktionalitäten sowie Systeme zum Farbmanagement und zur Darstellung von Schriftarten. Die Java-Klassenbibliothek befindet sich übrigens bei einer JRE- oder JDK-Installation in der Datei *rt.jar*.

An dieser Stelle möchte ich auch die beiden Projekte GNU Classpath und Apache Harmony erwähnen. GNU Classpath ist ein Projekt unter den Fittichen der Free Software Foundation, welches eine eigene Implementierung der Java-Klassenbibliothek erstellt. Apache Harmony bietet nicht nur die Klassenbibliothek sondern auch eine JVM und einen Compiler. Ihren Ursprung hatten beiden Projekte zu einer Zeit, als Sun noch nicht gewillt war, Java unter eine Open-Source-Lizenz zu stellen. Die Koexistenz von GNU Classpath und Apache Harmony erklärt sich durch Lizenz-Unterschiede zwischen der GPL und der Apache Software License.

Der JCP ist keine Organisation

Die Tatsache, dass die Java-Technologie mittlerweile unter der GPL steht, stellt also, wie gerade erläutert, einen guten Investitionsschutz dar, was auch immer die Zukunft bringen mag. Werfen wir als nächstes einen Blick auf den JCP. Ist der JCP nicht eine eigenständige, von Sun unabhängige Organisation, die also durch die Übernahme von Sun durch Oracle genauso bestehen bleiben wird? Die Antwort lautet ganz klar: Nein!

Wie Stephen Colebourne in seinem Blog [Colebourne] sehr klar analysiert hat, ist der JCP nämlich lediglich ein Prozess unter der Obhut von Sun aber keine eigenständige oder unabhängige Organisation. Das bekannte Java Specification Participation Agreement (JSPA) ist eine rechtliche Vereinbarung, die zwischen einem Individualmitglied oder einer Firma und Sun Microsystems geschlossen wird. Die Situation ist hier deutlich anders als beispielsweise bei Eclipse. Hier gibt es nämlich die sogenannte Eclipse Foundation, eine von IBM unabhängige und rechtlich eigenständige Organisation. Die Mitglieder der Eclipse Foundation schließen rechtliche Vereinbarungen mit dieser Organisation und nicht mit IBM ab.

Oracle wird wohl mit dem Kauf von Sun der Rechtsnachfolger dieser Verträge, die zwischen den JCP-Mitgliedern und Sun geschlossen worden sind. Wie es mit dem JCP weitergeht, ist durch dieses rechtliche Konstrukt aber erst einmal offen und abhängig von Oracles Plänen.

Das OpenJDK

Ein zentraler Bestandteil der Java-Technologie ist das Java Development Kit (JDK) und mit Hinblick auf die Open-Source-Strategie vor allem das OpenJDK. Dieses möchte ich als nächstes erläutern und seine Verbindung zu Java SE 6 und Java SE 7 darstellen. Ursprünglich basierte das OpenJDK-Projekt auf Java SE 7 bzw. dem dazugehörigen JDK 7. Daran wird jedoch schon seit geraumer Zeit entwickelt und es gibt nach wie vor keine finale Version davon. Deshalb wurde der Wunsch, eine Open-Source-Version des JDKs für Java 6 zu haben, immer dringender, sodass es seit Februar 2008 zwei OpenJDK-Pro-

jekte gibt. Zum einen das ursprüngliche OpenJDK-Projekt, basierend auf Java SE 7. Zum anderen das JDK6-Projekt (teilweise auch als OpenJDK6 bezeichnet), welches eine Open-Source-Version des JDK für Java SE 6 bietet.

Interessanterweise gibt es bisher keinen JSR für Java SE 7. Das ist erst einmal überraschend, wird doch so gut wie alles in der Java-Welt in Form eines JSRs spezifiziert. Für Java SE 6 hingegen gab es einen JSR, nämlich JSR 270. Mit dessen TCK können alternative Implementierungen des JDK auf Konformität mit der Java SE 6-Spezifikation überprüft werden. Es sei allerdings nicht unerwähnt, dass es in der Vergangenheit für Open-Source-Projekte oft schwierig war, im Rahmen ihrer finanziellen Möglichkeiten Zugriff auf dieses TCK zu erlangen. Dadurch dass es also keinen JSR für Java SE 7 gibt, fehlt natürlich auch ein entsprechendes TCK. Dies erschwert es alternativen JDK-Implementierungen für Java SE 7, ihre jeweilige Kompatibilität mit der Spezifikation zu überprüfen und zu beweisen. Stephen Colebourne [Colebourne] beklagt sich hierüber in seinem Blog.

Es könnte aber auch sein, dass es noch einen JSR und damit auch ein TCK für Java SE 7 geben wird. Mark Reinhold, Principal Engineer bei Sun, erläutert unter [BlogMR], dass das JDK 7-Projekt in prototypischer Art und Weise ausprobiert, welche Features in Java SE 7 mit aufgenommen werden können. Der dazugehörige JSR wird dann laut seiner Darstellung einfach die bis dahin im JDK 7 realisierten Features festlegen.

Generell ist es aber so, dass es zahlreiche große Open-Source-Projekte gibt, die ein Konzept wie das TCK überhaupt gar nicht kennen. Weder bei Linux noch bei Eclipse gibt es beispielsweise eine Entsprechung zu TCKs und trotzdem funktionieren diese beiden Projekte und noch zahlreiche andere hervorragend. Die Entwicklung von Spezifikation, Referenzimplementierung und TCK in einer Qualität, wie sie bisher üblicherweise vorgenommen wurde, ist aufwändig, teuer und in der Open-Source-Welt nicht unbedingt üblich. Vielleicht muss man sich im Zuge der Open-Source-Strategie von Java mit dem Gedanken anfreunden, dass in Zukunft die Java-Technologie um Bestandteile ergänzt wird, die nicht unter der Obhut eines JSRs stehen und für die es kein TCK gibt.

Nach der derzeitigen Planung wird das JDK 7 Anfang 2010 in Form eines ersten Release Candidate zur Verfügung stehen [JDK7Milestones]. Dann dürfen wir uns über einen verbesserten Garbage Collector (G1) freuen, der kürzere Unterbrechungszeiten und bessere Vorhersagbarkeit als der derzeitige bieten wird. Darüber hinaus wird die JVM eine bessere Unterstützung als bisher für dynamisch typisierte Sprachen bereitstellen (Stichwort *InvokeDynamic*).

Besonders spannend finde ich auch die Ansätze und Diskussion rund um das Thema, das JDK zu modularisieren. Technische Plattformen wie Java und das JDK beziehungsweise das JRE neigen dazu, im Laufe der Zeit immer größer und umfangreicher zu werden. Insbesondere auch für Desktop- und mobile Anwendungen ist es daher sinnvoll, nur diejenigen Teile zu übertragen und zu installieren, die tatsächlich benötigt werden.

Schön wäre es auch, wenn es das Swing Application Framework (JSR 296) in das JDK 7 schaffen würde. Dieses adressiert das Problem, dass es insbesondere für Einsteiger auf dem Gebiet der Java-GUI-Programmierung eine relativ hohe Hürde darstellt, eine Swing-basierte Anwendung zu erstellen. Es lauern viele Stolperfallen auf dem Weg zu einer Applikation mit einer guten Benutzeroberfläche und einem guten Look & Feel. Zudem gibt es einige Aspekte, die nichts mit der eigentlichen Geschäftslogik einer Anwendung zu tun haben, die aber trotzdem immer wieder aufs Neue von jeder Swing-Anwendung behandelt werden müssen. Hier bietet das Swing Application Framework Unterstützung und stellt quasi das Grobgerüst für eine Anwendung zur Verfügung. So muss der Anwendungsentwickler nicht jedes Mal aufs Neue Code schreiben, der das Applikationsfenster erzeugt und anzeigt, die UI-Event-Loop startet und dergleichen mehr. Derzeit trägt der JSR 296 für das Swing Application Framework allerdings den Status *inaktiv* und es ist nicht sicher, wie hier die Entwicklung vorangehen wird.

Neue Geschäftsmöglichkeiten

Im Zuge der diesjährigen JavaOne hat Sun zwei Dinge präsentiert, die neue Geschäftsmöglichkeiten für Java eröffnen könnten, nämlich den Java AppStore und JavaFX 1.2. Der Java AppStore ist natürlich in JavaFX realisiert, befindet sich aber bislang noch in einer geschlossenen Beta-Phase. Als eine zentrale Anlaufstelle soll er es Endanwendern erleichtern, Java-Applikationen zu finden. Das Ganze erinnert einen recht stark an den überaus erfolgreichen iTunes Store von Apple. Vielleicht ist der Java AppStore ja das geeignete Mittel, um die Verbreitung von Java-Desktop-Applikationen weiter voranzutreiben.

Mit der Vorstellung von JavaFX 1.2 und JavaFX TV unternimmt Sun einen weiteren Anlauf, um Java auf neue Geräteklassen zu bringen, nämlich Fernsehgeräte bzw. Set-Top-Boxen.

Fazit

Der Kauf von Sun durch Oracle bietet die Chance, frischen Wind in die Welt von Java zu bringen. Nach bisherigen Aussagen von Oracle, insbesondere von Larry Ellison auf der JavaOne im Juni, will Oracle sowohl das Hard- als auch das Softwaregeschäft von Sun weiterführen und massiv in Java investieren. Allerdings gibt es bisher noch keine konkreten Aussagen darüber, was Oracle mit dem JCP vor hat. Der JCP ist keine eigenständige Organisation sondern vielmehr ein Prozess auf Basis rechtlicher Vereinbarungen, welche die Teilnehmer des JCP mit Sun abgeschlossen haben. Wie es hiermit weitergehen wird, bleibt erst einmal abzuwarten.

Literatur und Links

[BlogMR] JDK 7, `http://blogs.sun.com/mr/entry/jdk7`
[Colebourne] Stephen Colebourne's Weblog, `http://www.jroller.com/scolebourne`
[JavaFX] JavaFX, `http://javafx.com`
[JDK7Milestones] JDK7 Milestones `http://openjdk.java.net/projects/jdk7/milestones`
[OpenJDK] OpenJDK, `http://openjdk.java.net`

Ludwig Mittermeier arbeitet in der Architektur-Abteilung der Siemens Corporate Technology.
Zu seinen bevorzugten Themen gehören Java, OSGi und Eclipse. Ludwig Mittermeier unterstützt Projekte hinsichtlich der Softwarearchitektur und hat bei den JSRs 205 (Wireless Messaging API 2.0) und 229 (Payment API) mitgewirkt.
E-Mail: ludwig.mittermeier@siemens.com.