

Java (== | -> | != | vs) Open Source

Ein Überblick

Clemens Rückle
clemens.rueckle@fh-augsburg.de

29.06.05

Inhaltsverzeichnis:

Einleitung

1. Laufzeitumgebungen

1.1 Sun JRE (!= | vs)

1.2 GNU-Classpath, (==)

1.3 Apache Harmony (==)

2. Java Programmierung

2.1 Sun JDK (-> | vs)

2.2 GCJ (==)

2.3 Andere (==)

2.4 Eclipse (==)

3. Apache Projekte (==)

4. JBoss (==)

Schlussbemerkung

Einleitung

Dieser Aufsatz soll sich damit beschäftigen inwiefern Java beziehungsweise damit zusammenhängende Projekte oder Technologien schon Open Source sind (==), auf Open Source umgestellt werden (->), vermutlich nie Open Source werden (!=) oder einen alternativen Weg gehen (vs). Dazu werden nachfolgend die verschiedenen betrachteten Projekte und Technologien einzeln, sequentiell erörtert.

1. Laufzeitumgebungen

Unter Laufzeitumgebung soll im Folgenden alles, was notwendig ist, um eine Java-Anwendung auf einem Computer mit beliebigem Betriebssystem ablaufen zu lassen, verstanden werden. Dazu gehören Programme wie z.B. Java-Bytecode Interpreter und die Basisbibliotheken in binärer Form, wie sie in verschiedenen von SUN herausgegebenen Referenzwerken beschrieben bzw. im Java Community Process standardisiert sind. Im Folgenden werden die Laufzeitumgebungen von SUN, eine freie Implementierung der Java Basisbibliotheken und ein kürzlich neu entstandenes Projekt zur Entwicklung einer Freien Laufzeitumgebung für Java betrachtet.

1.1 SUN JRE (!= | vs)

Die SUN Java Runtime Environment ist die von SUN herausgegebene Java Laufzeitumgebung. Die JRE wird unter einer nicht-OpenSource-Lizenz, der BCL (Binary Code License) vertrieben. Bisher gab es noch keine Anzeichen seitens SUN, dass diese Lizenzierung in unmittelbarer Zukunft geändert wird. In der OpenSource-Welt wird dies naturgemäß kritisch gesehen, da die Lizenzierung nicht den Kriterien für Freie Software entspricht¹, was bedeutet, dass Software, die selbst Frei ist, also z.B. unter der GPL steht, aber die SUN JRE zur Ausführung benötigt, strenggenommen unbenutzbar, in der von GNU definierten „Freien Welt“, ist.² Ebenfalls oft kritisiert wird, dass die JRE nicht einem Standardisierungsgremium, wie z.B. der Ecma vorgelegt wird oder wurde, wie das z.B. Microsoft mit der Standardisierung der CIL (Common Language Infrastructure) für „.net“ getan hat.³

SUN geht hier allerdings einen eigenen Standardisierungsweg mittels des JCP (Java Community Process). Am JCP sind alle wichtigen Firmen der „Javaindustrie“ beteiligt. Mittlerweile werden sämtliche Änderungen an Java oder der JRE im JCP in Form von JSRs (Java Specification Request) diskutiert und gemeinsam standardisiert.⁴ Sämtliche JSRs sind auch öffentlich einsehbar und ermöglichen so jedem eine standardkonforme Implementierung. Durch den JCP unterliegt die Weiterentwicklung Javas also nicht mehr dem Diktat Suns.

Ob nun die Standardisierung à la Microsoft bei .net oder Sun bei Java besser ist, kann objektiv wohl kaum beantwortet werden.

1.2 GNU-Classpath (==) + freie JVMs

GNU-Classpath ist eine freie Implementierung der Java-Standardbibliotheken und ist, wie auch schon dem Namen entnehmbar, ein GNU-Projekt. Das GNU-Classpath-Projekt entwickelt allerdings keine eigene VM (Virtual Machine) für Java, weshalb zur Ausführung einer Java-Anwendung eine VM aus externer Quelle benötigt wird. Interessanterweise ist es momentan noch nicht möglich die VM aus dem GNU-GCJ-Projekt zu verwenden.⁵

Die GNU-Classpath API ist seit 6.11.04 voll kompatibel zu Version 1.0 des JDKs von Sun. Wenn man sich allerdings die auf den Projektseiten angegebenen Vergleiche zu neueren JDKs ansieht, fällt auf, dass viele wesentliche Bereiche der neuen APIs, wie z.B. „Swing“, noch sehr lückenhaft sind.⁶ Einen Vergleich zur neuesten Version 1.5 bzw 5.0 gibt es noch nicht einmal. Darüberhinaus wird, wie bei GNU üblich, für Windows nur eine Portierung über „Cygwin“ angeboten. Zusätzlich hat die Apache Software Foundation vor kurzem die Entwicklung eines eigenen JDK (Apache Harmony,) unter der sehr liberalen ASL (Apache Software License), angekündigt. All dies wirft die Frage auf, ob dieses Projekt heute wirklich noch notwendig ist bzw. ob es überhaupt noch Sinn macht.

Wenn einem die begrenzte Funktionalität reicht, braucht man wie bereits geschrieben noch eine Virtuelle Maschine für Java. GNU-Classpath ist daher kein All-in-One Paket, weshalb man es meist

1 siehe The Free Software Definition: <http://www.gnu.org/philosophy/free-sw.html>

2 Richard Stallman, Free But Shackled - The Java Trap: <http://www.gnu.org/philosophy/java-trap.html>

3 ECMA-335: <http://www.ecma-international.org/publications/standards/Ecma-335.htm>

4 JCP @ wikipedia: http://en.wikipedia.org/wiki/Java_Community_Process

5 JVMs: <http://www.gnu.org/software/classpath/stories.html#jvm>

6 Vergleich JDK 1.4 GNU-Classpath: <http://www.kaffe.org/~stuart/japi/htmlout/h-jdk14-classpath.html>

nicht direkt, sondern indirekt über eine VM, deren Library auf dem GNU-Classpath basiert, verwendet. Ein Beispiel dafür ist das „Kaffe“-Projekt.⁷

Das „Kaffe“-Projekt ist unter der GPL lizenziert und soll laut eigenen Angaben eine VM besitzen die sogar schneller als die von Sun Herausgegebene ist.

1.3 Apache Harmony (==)

Das Apache Harmony Projekt ist ein sehr neues Projekt, mit dem Ziel einer freien Laufzeitumgebung für Java. Diese soll unter der Apache License lizenziert werden und kompatibel zu Version 1.5 (bzw 5.0) der Java Standard Edition sein. Genauer lässt sich noch nicht sagen, da das Projekt erst vor kurzem gestartet wurde. Der Vorschlag zur Aufnahme in den Apachebrutkasten wurde erst am 6.5.05 gemacht.⁸

Das Apache Harmony Projekt hat wohl bessere Zukunftsaussichten als das GNU-Classpath-Projekt ganz einfach deshalb weil es unter Apache-Schirmherrschaft steht. Dadurch ergeben sich Vorteile, wie „Sourcecode-Spenden“ von Firmen wie z.B. IBM (SWT) sowie möglicherweise kostenlosen Zugriff auf das TCK von Sun. Dieses ist ein Werkzeug um die Kompatibilität einer Java-Laufzeitumgebung zu prüfen. Darüberhinaus, sollen an diesem Projekt einige Leute, die schon an GNU-Classpath beteiligt waren, beteiligt sein.⁹

2. Java Programmierung

Unter dieser Kategorie sind Programmzusammenstellungen zu verstehen, die die Entwicklung von Java-Programmen ermöglichen. Solche Programmzusammenstellungen enthalten Tools wie Compiler, Debugger oder auch graphische Oberfläche zur komfortableren Programmentwicklung und Unterstützung des Programmierers (wie z.B. Vervollständigung von Funktionen und Klassennamen). Im Folgenden werden dazu das JDK von Sun, der GCJ aus der GCC, andere auf der virtuellen Maschine von Java laufende Programmiersprachen und Eclipse betrachtet. Kommerzielle Entwicklungswerkzeuge wie z.B. JBuilder von Borland werden nicht behandelt.

2.1 SUN JDK (-> | vs)

Das SUN JDK ist das von Sun zum kostenlosen Download angebotene Java Entwicklungs Kit. Das JDK enthält alle benötigten Werkzeuge um in Java geschriebene Programme übersetzen und debuggen zu können. Desweiteren enthält es eine umfangreiche Dokumentation der Basisbibliothek. Das JDK gibt es in verschiedenen Versionen, J2SE für Programme auf einem Desktop-Rechner, J2EE für den Java-Server Bereich und J2ME für die Entwicklung von Software für Mobile Geräte wie z.B. Handys. Die Version für den Desktop Bereich ist dabei unter, den normalen Open Source Lizenzen, ähnlichen Lizenzen im Sourcecode verfügbar. Die Serverversion des JDK wird vermutlich mit dem erscheinen des nächsten Releases auch unter diesen Lizenzen im Sourcecode verfügbar sein. Bei den Lizenzen handelt es sich um die SCCL (Sun Community Source

7 Kaffe: <http://www.kaffe.org/>

8 Proposal: http://mail-archives.apache.org/mod_mbox/incubator-harmony-dev/200505.mbox/%3c3923A844-DEC5-4CC2-ADED-B1F144BB6AF5@apache.org%3e

9 Harmony-FAQ: http://mail-archives.apache.org/mod_mbox/incubator-harmony-dev/200505.mbox/%3c50351021-6408-437D-949A-7AF2AD4DFD0F@apache.org%3e

License)¹⁰ und die JRL (Java Research License)¹¹. In naher Zukunft soll die SCCL allerdings durch zwei neue Lizenzen, die JIUL (Java Internal Use License) und die JDL (Java Distribution License) ersetzt werden. Die JIUL wird dabei weitgehende Änderungen erlauben, allerdings wie der Name schon sagt nur für den Internen Gebrauch. Die JDL wird Änderungen erlauben, solange das veränderte JDK immer noch zu den Standard-Java-Spezifikationen kompatibel ist. Mit diesen neuen Lizenzen bewegt sich SUN also langsam in Richtung Opensource-Welt. Das oben bereits erwähnte, neue Projekt „Apache Harmony“ könnte dabei noch etwas mehr Druck auf SUN aufbauen, die immer noch nicht wirklich Open Source kompatiblen Lizenzen durch solche zu ersetzen.

2.2 GCJ (==)

Das GCJ-Paket ist Bestandteil der GNU Compiler Collection (GCC). Es enthält wie auch das JDK von SUN eine Klassenbibliothek (die Implementierung basiert auf dem oben erläuterten GNU-Classpath-Projekt), eine virtuelle Maschine für Java und einen Compiler um Code für die virtuelle Maschine zu erzeugen. Darüberhinaus enthält das GCJ-Paket noch einen Compiler der Java-Programme direkt in Maschinencode übersetzt. Um diese Programme ablaufen zu lassen benötigt man dann allerdings zusätzlich eine, in eine Bibliothek ausgelagerte, Runtime (libgcj).¹² Diese Bibliothek enthält die Klassenbibliothek und übernimmt Arbeiten wie z.B. den „Garbage Collector“. Durch das Kompilieren verliert das binäre Java-Programm naturgemäß seine Portabilität, dafür ergibt sich der Vorteil eines erschwerten Dekompilierens, da dies bei Maschinencode weitaus schwieriger ist, als bei Bytecode, wie in z.B. Java verwendet. Die Idee Java-Programme zu kompilieren, wenn der Schutz, des in dem Programm enthalten geistigen Eigentums, aus irgendwelchen Gründen notwendig sein sollte, ist sehr interessant, leider lässt sie sich aber aufgrund der nicht vollständigen Klassenbibliothek nur auf sehr eingeschränkte Java-Programme anwenden.

Die Anwendergruppe GCJs scheint allerdings nicht sonderlich groß zu sein, denn in der Liste „Done with GCJ“¹³ finden sich sehr wenige Projekte, obwohl die erste Version GCJs bereits 1998 erschien. Interessant ist, dass die Liste darüberhinaus teilweise nicht verlinkte oder sogar abgebrochene Softwareprojekte enthält.

2.3 Andere (==)

Neben der von Sun spezifizierten Java-Programmiersprache, gibt es noch eine Reihe weiterer Programmiersprachen für die Compiler geschrieben wurden, welche Bytecode für die Virtuelle Maschine für Java erzeugen. Diese Programmiersprachen stehen oft unter Opensource-Lizenzen wie der GPL- oder BSD-Lizenz und beinhalten innovative Features, von denen manche möglicherweise in zukünftigen Versionen der Java-Programmiersprache Einzug finden werden.

Ein Beispiel ist Jython, dies ist eine Implementierung der Programmiersprache Python. Der Jython-Compiler übersetzt dabei Python-Programme in Java-Bytecode, wodurch diese auf einer virtuellen Maschine für Java ausführbar sind. Die Lizenz dieser Programmiersprache ist die CNRI/Python Lizenz, welche eine der auf der OSI-Website aufgelisteten Lizenzen ist.¹⁴

Ein weiteres Beispiel ist die Programmiersprache Groovy. Groovy ist in der Java-Welt sehr bekannt

10 SCCL: http://java.sun.com/j2se/1.5.0/scsl_5.0-license.txt

11 JRL: http://java.sun.com/j2se/1.5.0/jrl_5.0-license.txt

12 GCJ: <http://gcc.gnu.org/java/>

13 „Done with GCJ“: <http://gcc.gnu.org/java/done.html>

14 OSI-Lizenzen: <http://www.opensource.org/licenses/pythonpl.php>

und wird im Moment sogar vom JCP (Java Community Process) über den JSR 241 (Java Specification Request) spezifiziert. Lizenziert wird Groovy durch eine BSD/Apache ähnlichen Lizenz. Die Sprache ähnelt vom Stil her Sprachen wie Ruby oder Python und ermöglicht so, eine im Vergleich zu Java, schnellere Entwicklung was z.B. für Rapid Prototyping sinnvoll sein kann. Weiterhin bietet Groovy z.B. noch vereinfachten Zugriff auf XML und Datenbanken oder auch z.B. Operator Overloading.¹⁵

Weitere Programmiersprachen findet man auch im Web.¹⁶ Kritisch anmerken muss man hier aber wieder, dass selbst wenn die Programmiersprache „frei“ nach GNU ist, die verwendete JVM dies möglicherweise nicht ist. Dadurch stehen die Resultate der „freien“ Programmiersprachen (Java Bytecode) der „freien“ Welt vielleicht trotzdem nicht zur Verfügung, weil es keine geeignete JVM für den Ablauf des jeweiligen Programms gibt.

2.4 Eclipse (==)

Eclipse ist ein Beispiel für eine freie unter einer Opensource Lizenz (Eclipse Public License¹⁷) stehenden Entwicklungsumgebung. Eclipse basiert auf einem sehr flexiblen Framework (RCP nach OSGi-Standard), wodurch z.B. Plugins oder auch die Verwendung des Frameworks für eigene Anwendungen, die auch nichts mehr mit Eclipse selbst zu tun haben müssen, möglich ist. Mittlerweile ist Eclipse die Entwicklungsumgebung für Java schlechthin geworden und manche Hersteller von kommerziellen Entwicklungsprodukten wie z.B. Profiler oder Optimizer bieten ihre Tool als Plugins für Eclipse an.

Eclipse war ursprünglich ein Projekt IBMs um eine neue konkurrenzfähige Entwicklungsumgebung zu schaffen, damit man den Anschluss an andere Hersteller wie Borland nicht verliert. Die Ankündigung IBMs Eclipse im Sourcecode freizugeben und der OpenSource-Welt zu Weiterentwicklung zu überlassen kam dann auch ziemlich überraschend, entwickelte sich allerdings zu einer wahren Erfolgsgeschichte.

3. Apache Projekte (==)

Unter dem Namen „Apache“ wird in der IT-Welt meist nur der HTTP-Server „Apache“ verstanden, das Apache Projekt enthält jedoch noch viele weitere Projekte vor allem im Bereich Javas. Beispiele für solche Projekte sind Ant oder Jakarta. All diesen Projekten ist gemein dass sie unter der ASL, der Apache Software License stehen. Diese Lizenz ist, im Gegensatz zur sehr restriktiven GPL, eine sehr offene Lizenz.¹⁸

Ant ist ein Build-Tool und dient als Ersatz der mittlerweile in die Jahre gekommenen Makefiles. Im Gegensatz zu diesen setzt Ant auf eine XML-Struktur und definiert schon eine Reihe vorgefertigter „Tags“ über Java-Klassen. Tags lassen sich auch selbst schreiben, wodurch wiederkehrende Tätigkeiten einfach automatisiert werden können.

Das Apache Jakarta Projekt, ist ein Sammelbecken unterschiedlichster Projekte im Bereich des serverseitigen Javas. Beispiele sind der mittlerweile sehr bekannte Servlet-Container Tomcat¹⁹, der

15 Groovy: <http://groovy.codehaus.org/Home>

16 Sprachen für die JVM: <http://www.robert-tolksdorf.de/vmlanguages.html>

17 Eclipse Public License: <http://www.opensource.org/licenses/eclipse-1.0.php>

18 Apache License: <http://www.opensource.org/licenses/apachepl.php>

19 Tomcat: <http://jakarta.apache.org/tomcat/index.html>

die offizielle Referenzimplementierung der „Java servlet“- und „Java server Pages“-Standards ist. Ein anderes Beispiel ist das Tapestry Projekt²⁰. Dies ist ein Framework um auf möglichst einfache aber doch flexible Weise Webseiten-Projekte Implementieren zu können. Einer der Vorteile dieses Frameworks ist, dass Tapestry-Seiten Standard-HTML-Seiten sind. Dadurch ergibt sich die Möglichkeit, dass die Gestaltung der Webseiten komplett an spezialisierte Fachkräfte weitergeleitet werden kann, die die Seiten mit ihren Werkzeugen editieren können, ohne dass dabei etwas zerstört wird.

Viele Apache Projekte nahmen ihren Anfang in den Entwicklungsabteilungen von Firmen der Javaindustrie wie SUN, IBM oder Oracle. Beispiele dafür sind das bereits erwähnte Tomcat-Projekt welches bei SUN entstand und Pluto welches ein Unterprojekt, des Apache Portals Projekt ist. Pluto ist die offizielle Standardimplementierung der Java Portlet Spezifikation und wurde von IBM entwickelt.²¹ Motivation der Firmen ihre Projekte im Sourcecode freizugeben ist wohl, dass sie daran interessiert sind, offene Industriestandards zu entwickeln. Offene Industriestandards ermöglichen es schließlich einen gemeinsamen Markt aufzubauen, auf dem sich das eigene Produkt, sofern es die Qualitäten besitzt, durchsetzen kann. Wenn es keinen offenen Standard gibt, ergeben sich weniger Chancen für sämtliche Produkte, da viele Softwareentwicklungsfirmen im Bereich der Javaindustrie das Binden an einen einzigen Anbieter scheuen und deshalb auf alternative standardisierte Technologien oder Eigenentwicklungen ausweichen.

4. JBoss (==)

JBoss.inc ist eine Firma mit genau einem Produkt dem JBoss Enterprise Middleware System (JEMS)²². JEMS ist eine Sammlung verschiedenster Softwarekomponenten die notwendig sind um Enterprise-Software zu schreiben. Enthalten sind vor allem verschiedene Programme die auf Standards der Java Enterprise Edition beruhen. Sämtliche Bestandteile JEMSSs sind Open Source. Die Meisten werden dabei von der JBoss.inc selbst in Zusammenarbeit mit Leuten aus der Java-OpenSource-Szene entwickelt und unter der LGPL (Lesser General Public License)²³ lizenziert. Der Download der Software sowie die dazugehörige Dokumentation sind kostenlos, Geld wird nur durch Support, Training und Zertifizierungen im Bereich von JEMS verdient. Dieses Geschäftsmodell wird von JBoss.inc als Professional Open Source bezeichnet.²⁴

Ein Beispiel aus der JEMS-Suite ist der JBoss Java-Application-Server. Wie oben bereits erwähnt, ist dieser Server kostenlos herunterladbar und aufgrund seiner Lizenz (LGPL) auf beliebig vielen Systemen einsetzbar. Diese Eigenschaften haben den JBoss-Server mittlerweile zu einem der beliebtesten Java-Application-Server in der Entwicklung gemacht. Umfragen sprechen von rund 50 % Marktanteil. In Produktivumgebungen ist der Marktanteil allerdings etwas geringer laut offiziellen Angaben liegt er bei rund 33%²⁵. Damit ist der JBoss-Sever allerdings immer noch Marktführer gemeinsam mit dem kommerziellen Produkt „WebSphere“ von IBM. Der Marktanteil ist wohl daher etwas geringer, da im Produktiveinsatz, die Vorteile der kommerziellen Java-Application-Server wie WebSphere von IBM oder WebLogic von Bea stärker zum Tragen kommen. Vorteile haben die kommerziellen Application-Server vor allem in der Effizienz bezüglich Speicherverbrauch und der Performance.

20 Tapestry: <http://jakarta.apache.org/tapestry/index.html>

21 Pluto: <http://portals.apache.org/pluto/>

22 JBOSS: <http://www.jboss.org/products/index>

23 LGPL: <http://www.opensource.org/licenses/lgpl-license.php>

24 Professional Open Source: <http://www.jboss.com/company/pos>

25 JBOSS Marktanteil: <http://www.javamagazin.de/itr/news/psecom,id,19434,nodeid,10.html>

Schlussbemerkung

Wie man sieht ist im Umfeld von Java bereits jetzt schon einiges Opensource. Interessanterweise sind viele dieser Anwendungen z.B. aus dem Apache Jakarta Projekt oder Eclipse, ehemalige Projekte verschiedenster Firmen der Java Industrie, die jetzt durch die Opensource-Szene weitergepflegt, bzw im Falle von Eclipse schon stark erweitert bzw umgestaltet wurden. Ein Grund für diese Erfolgsgeschichten ist sicherlich, dass einige der Mitarbeiter, aus den Zeiten in denen die jeweiligen Projekte noch in Firmenhand waren, auch jetzt noch an diesen Projekten beteiligt sind.

In der Java Welt kann man Opensource sicherlich als stark Innovationen Förderndes Mittel ansehen. Schließlich müssen Kommerzielle Anbieter in Konkurrenz zu den kostenlosen und oft sehr guten freien Produkten treten. Um sich hier Durchzusetzen verlangt es guten Service und viel wichtiger noch innovative Produkte, die Features besitzen die es in der Freien Welt bisher noch nicht gibt. Da die Entwicklung der freien Produkte aber auch nicht stehen bleibt, sondern im Gegenteil sehr rasant vonstatten geht, müssen die Anbieter ihre Produkte fortlaufend modernisieren und komfortabler gestalten. Im Sinne der Java-Entwickler und der Java-Industrie kann dies nur von Vorteil sein. Es lässt sich also sagen, dass Opensource in der Java-Welt schon jetzt ein sehr wichtiger Bestandteil ist und auch in Zukunft bleiben wird.