Java und Open-Source
— Ein Geben und Nehmen —

Bernd Müller
Ostfalia
Hochschule für angewandte Wissenschaften
38302 Wolfenbüttel
bernd.mueller@ostfalia.de

Abstract: Wir zeigen auf der Basis aktueller Java-Spezifikationen und Open-Source-Systeme, dass Java und Open-Source sich gegenseitig befruchten und verschiedene Open-Source-Frameworks im Java-Bereich einen regen gegenseitigen Austausch und eine beachtliche Verwendung anderer Open-Source-Systeme betreiben. Ferner zeigen wir, dass auch Nicht-Open-Source-Systeme im Java-Umfeld von der Verwendungsmöglichkeiten freier Software lebhaften Gebrauch machen.

1 Java und Open-Source

Im Mai 2006 gab Sun bekannt, dass das Java Development Kit (JDK) zukünftig unter einer GPL-Lizenz stehen wird. Im November 2006 wurden zunächst der Compiler und die JVM als Open-Source veröffentlicht. Die Klassenbibliothek wurde schrittweise auf Open-Source umgestellt, da sie proprietären Code enthielt, für den Sun nicht die notwendigen Rechte besaß. Mittlerweile stellt das Projekt OpenJDK [JDK] ein vollständiges Open-Source JDK bereit, so dass auch Linux-Distributionen wie Fedora, die ausschließlich frei Software enthalten (wollen), es verwenden können.

Die Öffnung des Java JDK ist allgemein bekannt. Weniger bekannt ist der Umstand, dass Open-Source-Implementierungen von Java-Frameworks Java-Spezifikationen richtungsweisend und maßgeblich befruchtet haben. Ebenfalls weniger bekannt ist der sehr gut funktionierende Austausch bzw. die Verwendung von Open-Source Systemen sowohl in anderen Open-Source aber auch kommerziellen Systemen. Wir wollen den Bekanntheitsgrad dieser Fakten erhöhen und belegen die

- Beeinflussung von Java-Spezifikationen durch Open-Source
- Verwendung von Open-Source Java-Systeme in kommerziellen Systemen
- Verwendung von Open-Source Java-Systemen in anderen Open-Source Java-Systemen

durch einige ausgewählte Beispiele.

2 Beeinflussung von Java-Spezifikationen durch Open-Source-Implementierungen

Die Java 2 Platform Enterprise Edition Specification 1.4 (J2EE 1.4) besteht aus 21 einzelnen Spezifikationen, die in Tabelle1 aufgezählt sind. Allein die Anzahl der Spezifikationen

Enterprise JavaBeans	Java API for XML Processsing
JavaServer Pages	SOAP with attachements API for Java
Servlets	Java API for XML registries
Java Database Connectivity	Java Enterprise Edition Management Spec.
Java Naming and Directory Interface	Enterprise Edition Deployment Specification
Java Message Service	Java Management Extensions
Java Transaction API	Java Authorization Contract for Containers
Java Transaction Service	Java Authentication and Authorization Service
JavaMail	Web Services for J2EE
JavaBeans Activation Framework	Java API for XML-based RPC
J2EE Connector Architecture	

Tabelle 1: J2EE 1.4 Spezifikationen

macht deutlich, wie komplex die Entwicklung von Unternehmensanwendungen auf Basis von J2EE 1.4 war. Besondere Kritik wurde an der Spezifikation von Enterprise JavaBeans laut, die dem DRY-Prinzip widersprach und mit mehreren Java- und XML-Artefakten pro einzelner EJB als extrem aufwändig und damit praktisch als nicht verwendbar galt.

2.1 Hibernate und JPA

Als Reaktion auf die oben angesprochene Komplexität entstanden z.B. das Spring- und das Hibernate-Framework, wobei letzteres die Persistenz von Java-Objekten in Relationalen Datenbanken realisiert. Wir wollen Hibernate verwenden, um das Umdenken in der Java-Spezifikationsentwicklung zu verdeutlichen. Während früher (J2EE 1.4 und davor) Spezifikationen auf der grünen Wiese entwickelt wurden und zu Spezifikationsungetümen führten, wurden danach existierende Systeme, häufig Open-Source-Systeme, als funktionierendes Beispiel und Blaupause für die Spezifikation verwendet. Die Java Platform Enterprise Edition 5 (Java EE 5) wurde als erste nach diesem Prinzip entwickelt. Die Persistenz wurde in das Java Persistence API (JPA), definiert als Java Specification Request 220 (JSR-220), ausgelagert. Wir wollen diesen JSR im Folgenden als Vergleichssystem für Hibernate heranziehen.

Der folgende Code-Ausschnitt zeigt, wie ein Entity mit Hibernate persistent gemacht wird. Der Code ist intuitiv verständlich, so dass wir auf Erläuterungen verzichten.

```
SessionFactory factory =
    new Configuration().configure().buildSessionFactory();
Session session = factory.openSession();
```

```
Transaction tx = session.beginTransaction();
session.save(<ein Entity>);
tx.commit();
...
```

Während die Persistierung von Entities mit J2EE 1.4 nach einem völlig anderen Muster erfolgte, nahm sich die JPA-Expertengruppe Hibernate als Vorbild und definierte eine leichtgewichtige Persistenz. Der folgende Code-Ausschnitt zeigt das obige Beispiel unter Verwendung von JPA.

```
EntityManagerFactory factory =
    Persistence.createEntityManagerFactory("<name>");
EntityManager em = factory.createEntityManager();
EntityTransaction tx = em.getTransaction();
tx.begin();
em.persist(<ein Entity>);
tx.commit();
...
```

Man erkennt deutliche Parallelen, die durch konsistente Umbenennung von *Session* in `EntityManager` besonders hervortreten. Die Inspiration der JPA-Expertengruppe durch Hibernate wird durch dieses einfache Beispiel deutlich und ist auch in vielen anderen Code-Mustern, aber auch auf konzeptioneller Ebene erkennbar. Hibernate ist mittlerweile als JPA-Implementierung zertifiziert. Gavin King, der konzeptionelle Vater von Hibernate, war Mitglied im JPA-Spezifikationsgremiums.

2.2 Hibernate Validator, Seam und Bean Validation

Hibernate Validator [HV] ist ein Projekt, das die Validierung von Objekt-Properties zunächst für Hibernate-Entites, später aber auch für gewöhnliche POJOs realisiert. Die Validierung erfolgt durch die Annotation der Properties z.B. mit `@Min`, `@Max`, `@NotNull`, `@NotEmpty`, `@Future` und `@Past`, deren Namen für sich selbst sprechen. In geschichteten Anwendungen erfolgt häufig eine Validierung auf der GUI- aber auch auf der Persistenz-Ebene, was dem DRY-Prinzip widerspricht. JBoss Seam ist ein Anwendungs-Framework, das für die Oberfläche JSF und für die Persistenz JPA verwendet. Seam zeigte, wie mit Hilfe von Hibernate Validator das DRY-Prinzip respektiert und die auf der Persistenz-Schicht erfolgte Annotation von Properties mit Validierungs-Constraints bis zur Validierung von Eingaben in der Oberfläche hochgehoben werden kann.

Java EE 6 enthält mit *Bean Validation* eine Spezifikation, die Konzepte von Hibernate Validator aufnimmt und in Java EE 6 Containern sowohl für die Oberfläche (JSF) als auch die Persistenz (JPA) verwendet werden kann. Bean Validation ist als JSR-303 spezifiziert und Hibernate Validator ist die Referenzimplementierung. Emmanuel Bernard, ein JBoss-Mitarbeiter und maßgeblicher Kopf von Hibernate Validator, war Leiter des Spezifikationsgremiums.

2.3 Seam und CDI

Das allgemein Martin Fowler zugeschriebene Konzept der *Dependency Injection* (DI) [Fow] wurde in Java EE 5 erstmalig in einer Java-Spezifikation verwendet, um Container-spezifische Objekte und EJBs zu injizieren. Seam erweiterte die dortige Verwendung von DI auf beliebige Objekte, insbesondere die Verwendung derartiger Objekte in der JSF-basierten GUI-Schicht und verallgemeinerte das Prinzip des DI. Durch den Erfolg motiviert, initiierte JBoss den JSR-299, der nach zweimaliger Umbenennung schließlich den Namen *Contexts and Dependency Injection for the Java EE platform* (CDI) [CDI] erhielt. Er führt die durch Seam begonnenen Innovationen fort und erweiterte sie z.B. um Typsicherheit. CDI ist in Java EE 6 enthalten. Gavin King war als maßgeblicher Kopf hinter Seam auch Leiter des Spezifikationsgremiums für CDI. Die Referenzimplementierung von CDI wird unter dem Code-Namen Weld [Wld] als Seam-Teilprojekt von Seam-Entwicklern als Open-Source realisiert.

2.4 Guice, Spring und Dependency Injection for Java

Kurz vor Fertigstellung des JSR-299 wurde von Google und SpringSource ein weiterer Vorschlag zur Spezifikation von DI in Java eingereicht. Google und SpringSource bieten mit *Google Guice* [GG] und *Spring* [Spr] zwei weit verbreitete DI-Frameworks auf Open-Source-Basis an. Der Vorschlag von *Dependency Injection for Java* als JSR-330 führte zu etlichen Unruhen im Java-Bereich, da eine Konkurrenz zum JSR-299 gesehen wurde. Mittlerweile ist die Spezifikation jedoch verabschiedet und Basis des JSR-299. Bob Lee von Google und Rod Johnson von SpringSource waren die Leiter des Spezifikationsgremiums.

3 Kommerzielle Systeme nutzen Open-Source

Neben der Motivation von Java-Spezifikationen durch Open-Source-Systeme ist die häufige Verwendung von Open-Source-Systemen in kommerzieller Java-Software ebenfalls wenig bekannt. Wir wollen diesen Zustand hier ändern.

Die kommerziellen Java-Entwicklungssysteme (JDK) von Sun [SJK], Oracle [OJK] und IBM [IJK] verwenden eine ganze Reihe von Open-Source-Systemen. Alle genannten Systeme verwenden für die Verarbeitung von XML Systeme der Apache Software Foundation, die unter Open-Source-Lizenzen stehen. Es sind dies Xalan [Xal], ein XSLT-Prozessor, Xerces [Xer], ein XML-Parser und -Generator und XPath [Xpt], die XPath-Implementierung innerhalb Xalans.

Weiterhin verwenden die Entwicklungssysteme von Sun und Oracle BCEL [BCL] (Byte Code Engineering Library), ein System zur Analyse und Manipulation von Java-Byte-Code, während IBM hier eigene, nicht Open-Source-Wege, geht.

Besonders bemerkenswert ist der Umstand, dass Suns JDK die *JavaDB* enthält, ein relationales Datenbankmanagementsystem, das in Java implementiert ist. Das Interessante an JavaDB ist seine Geschichte, die der Geschichte von Firefox, OpenOffice und Eclipse ähnelt, d.h. eine kommerzielle Entwicklungs- und Vermarktungsphase mit anschließender Änderung der Lizenzbedingungen hin zu Open-Source. Die 1996 gegründete Firma Cloudscape entwickelte ein relationales Datenbankmanagementsystem, das ebenfalls den Namen Cloudscape trug. 1999 übernahm Informix Software Inc. die Firma Cloudscape. 2001 übernahm IBM Informix und das Datenbanksystem wurde unter dem Namen *IBM Cloudscape* vermarktet. Im Jahre 2004 übertrug IBM Cloudscape an die Apache Software Foundation unter den neuen Namen *Derby* [DRB], wo es seitdem weiterentwickelt wird. Sun legt Derby seit Version 6 dem JDK bei. Die entsprechenden Dateien im JDK tragen weiterhin den Namen Derby und enthalten die Apache-Lizenz-Datei.

4 Firmenübergreifende Verwendung von OS-Implementierungen

In diesem Abschnitt untersuchen wir die beiden populärsten Open-Source Application-Server. Obwohl Open-Source, werden beide Application-Server zu großen Teilen von Mitarbeitern der jeweiligen Herstellerfirmen implementiert und nur zu geringen Anteilen von anderen Open-Source Entwicklern. Bei den Application-Servern handelt es sich zum einen um den *JBoss Application Server* [JBA], der nach Auskunft von JBoss meistgenutze Java-EE-Application-Server, für den JBoss auch kommerziellen Support anbietet [JEM]. Zum anderen ist dies *GlassFish* [GLF], die Referenzimplementierung von Java-EE 6, für den Sun ebenfalls kommerziellen Support anbietet [GFS]. Da uns keine offiziellen Aussagen der Entwickler über die Verwendung von Fremdsystemen bekannt sind, basieren unsere Ausführungen auf der Analyse der Jar-Dateien der Systeme, die wir zu Dokumentationszwecken angeben. Tabelle 2 zeigt die von GlassFish und JBoss-AS verwendeten OS-Implementierungen.

Durch die Aufstellung wird offensichtlich, wie lebhaft die OS-Gemeinschaft agiert. Sun verwendet JBoss-Systeme, JBoss verwendet Sun-Systeme und beide Hersteller bedienen sich bei OS-Implementierungen von Apache, Codehaus, GNU und anderen.

5 Zusammenfassung

Wir haben an Beispielen belegt, dass Open-Source und Java keine getrennten Welten darstellen, sondern die Entwicklung von Java an einem sehr aktiven Open-Source-Ökosystem partizipiert. Dies ist durch die Motivation von Java-Spezifikationen durch erfolgreiche OS-Frameworks zu belegen, von denen wir lediglich beispielhaft JPA, Bean Validation und CDI aufgeführt haben. Weiterhin haben wir Beispiele genannt, in denen Firmen Software unter nicht Open-Source-Lizenzen entwickeln, die Open-Source-Software enthält. Außerdem haben wir gezeigt, dass bei bekannten OS-Application-Servern eine firmenübergreifende Verwendung von OS-Systemen stattfindet. Suns GlassFish verwendet von JBoss

Jar	System	Entwickler(-gruppe)
ant.jar	Ant, Build-Tool [Ant]	Apache
antlr-repackaged.jar	ANTLR, Parser Generator [ATL]	Terence Parr
bean-validator.jar	Hibernate Validator [HV], RI JSR-303	JBoss
asm-all-repackaged.jar	ASM Byte-Code Analyse und Manipulation [ASM]	OW2 Consortium
groovy-1.7.2.jar	Groovy Skriptsprache [GRY]	Codehaus
org.apache.felix.*.jar	Felix OSGI-Platform [FLX]	Apache
weld-osgi-bundle.jar	Weld [Wld], RI JSR-299	JBoss
org.eclipse.persistence.*.jar	EclipseLink [EL], RI JSR-317	Eclipse
web-ajp.jar	AJP-Protokoll-Implementierung [AJP]	Apache
web-core.jar	Apache Tomcat (Catalina) [CAT]	Apache
jettison.jar	Java-API für JSON [JSN]	Codehaus
antlr-repackaged.jar	ANTLR, Parser Generator [ATL]	Terence Parr
bcel.jar	Byte Code Engineering Library [BCL]	Apache
bsf.jar	Bean Scripting Framework	Apache
bsh.jar	BeanShell [BSH]	Pat Niemeyer
commons-collections.jar	Collection-Framework	Apache
commons-logging.jar	Logging-Framework	Apache
dtdparser121.jar	DTD-Parser [DTD]	Wutka Consulting
jsf-impl.jar	Mojarra JSF RI [JSF]	Sun
hsqldb.jar	HSQL-DBMS [HQL]	HSQL Dev-Group
log4j.jar	log4j Logging [L4J]	Apache
mail.jar	Java Mail [JM]	Sun
quartz.jar	Quartz Enterprise Job Scheduler [QTZ]	Terracotta
slf4j.jar	Simple Logging Facade for Java [S4J]	Quality Open Software
dom4j.jar	XML-Bibliothek [D4J]	dom4j
getopt.jar	Zugriff auf Kommandozeilenparameter	GNU
trove.jar	High Performance Collections for Java	GNU
wstx.jar	High-performance XML Processor	Codehaus
concurrent.jar	Concurrent Implementation	SUNY Oswego
jaxb-*.jar	JAXB-Implementierung	Sun
wsdl4j.jar	WSDL-Implementierung	IBM und Sun
xmlsec.jar	Java XML Security	Apache

Tabelle 2: Von GlassFish (erster Teil) und JBoss-AS (zweiter Teil) verwendete OS-Systeme

entwickelte Software, JBoss-AS verwendet von Sun entwickelte Software.

Aus Platzgründen haben wir in dieser Arbeit nur wenige Beispiele anführen können, die die Verbindung von Java mit Open-Source belegen. Die Liste ließe sich deutlich erweitern. Wir hoffen den Leser davon überzeugt zu haben, dass Java und Open-Source in einer Beziehung stehen, die mit „*Ein Geben und Nehmen*“ treffend beschrieben ist.

Literatur

[AJP] Apache Tomcat Connector. Web. http://tomcat.apache.org/connectors-doc/.

[Ant] Ant Java-Build-Tool. Web. http://ant.apache.org/.

[ASM] ASM. Web. http://asm.ow2.org/index.html.

[ATL] ANTLR Parser Generator (ANother Tool for Language Recognition. Web. http://www.antlr.org/.

[BCL] BCEL (Byte Code Engineering Library). Web. http://jakarta.apache.org/bcel/.

[BSH] BeanShell Lightweight Scripting for Java. Web. http://www.beanshell.org/.

[CAT] Apache Tomcat, Catalina Servlet-Container. Web. http://tomcat.apache.org/.

[CDI] JSR 299, Contexts and Dependency Injection for the Java EE platform. Web. http://jcp.org/en/jsr/summary?id=299.

[D4J] dom4j XML-Bibliothek. Web. http://www.dom4j.org/dom4j-1.6.1/.

[DRB] Apache Derby. Web. http://db.apache.org/derby/index.html.

[DTD] Java DTD Parser, Wutka Consulting. Web. http://www.wutka.com/dtdparser.html.

[EL] EclipseLink — Eclipse Persistence Services Project. Web. http://www.eclipse.org/eclipselink/.

[FLX] Apache Felix – OSGi R4 Service Platform. Web. http://felix.apache.org/site/index.html.

[Fow] Fowler Martin. Dependency Injection, Inversion of Control. Web. http://www.martinfowler.com/articles/injection.html.

[GFS] Sun GlassFish Enterprise Server Support. Web. http://developers.sun.com/appserver/support/index.jsp.

[GG] Google Guice. Web. http://code.google.com/p/google-guice/.

[GLF] GlassFish Open Source Application Server. Web. https://glassfish.dev.java.net/.

[GRY] Groovy – An agile dynamic Language for the Java Platform. Web. http://groovy.codehaus.org/.

[HQL] HyperSQL Pure Java Database. Web. http://www.hsqldb.org/.

[HV] Hibernate Validator. Web. http://www.hibernate.org/subprojects/validator.html.

[IJK] JDK von IBM. Web. https://www.ibm.com/developerworks/java/jdk/.

[JBA] JBoss Application Server. Web. http://www.jboss.org/jbossas.

[JDK] OpenJDK. Web. http://openjdk.java.net/.

[JEM] JBoss Enterprise Middleware Subscriptions. Web. http://www.jboss.com/services/subscriptions/.

[JM] JavaMail. Web. http://java.sun.com/products/javamail/.

[JSF] Project Mojarra. Web. https://javaserverfaces.dev.java.net/.

[JSN] Jettison — A JSON StAX Implementation. Web. http://jettison.codehaus.org/.

[L4J] Apache Log4j. Web. http://logging.apache.org/log4j/.

[OJK] JDK von Oracle (JRockit). Web. http://www.oracle.com/technology/products/jrockit/index.html.

[QTZ] Quartz Enterprise Job Scheduler. Web. http://www.quartz-scheduler.org/.

[S4J] Simple Logging Facade for Java (SLF4J). Web. http://www.slf4j.org/index.html.

[SJK] JDK von Sun. Web. http://java.sun.com/javase/downloads/index.jsp.

[Spr] Spring. Web. http://www.springsource.com/products/enterprise.

[Wld] JSR-299 und Weld Overview. Web. http://seamframework.org/Weld.

[Xal] The Apache Xalan Project. Web. http://xalan.apache.org/.

[Xer] The Apache Xerces Project. Web. http://xerces.apache.org/.

[Xpt] Using the JAXP XPath APIs. Web. http://xml.apache.org/xalan-j/xpath_apis.html.