

Open Source Java EE Application Server im Vergleich

# Eine echte Alternative

VON SIEGFRIED KLAR UND FRANK PIENKA



Die Zahl der zertifizierten kommerziellen Java-Application-Server nimmt von einer auf die folgende Java EE-Spezifikation ab. Gleichzeitig hat die Zahl der quelloffenen bzw. als Open Source verfügbaren Java EE Application Server zugenommen. Dieser Trend zeigt sich nicht nur in Zahlen, sondern auch in einem wachsenden Marktanteil der Open Source Application Server. Die Reife des Marktes für diese wird auch in der Zunahme von Migrations- und Betriebsthemen sichtbar. Damit werden sie immer weniger zu Spezialprodukten, sondern mehr zu allgemeinen Infrastrukturlösungen. Von daher ist es Zeit für eine Bestandsaufnahme der bestehenden Open-Source-Alternativen Geronimo, Glassfish, JBoss und JOnAS.

Im Einzelnen werden wir auf die verschiedenen Lizenzmodelle sowie die Funktionen der einzelnen Produkte eingehen und Kriterien für eine Beurteilung vorstellen. Bei den untersuchten Produkten haben wir uns auf die am meisten verbreiteten Produkte konzentriert. Da die Mehrzahl der Produkte als Webcontainer entweder Tomcat oder Jetty verwenden, werden wir uns auf die Application-Server-spezifischen Anforderungen beschränken.

Neben den zwei alten Platzhirschen JOnAS und JBoss werden wir auch die bei-

den „Newcomer“ Geronimo und Glassfish vorstellen. Dabei treffen wir mit OpenEJB, Tomcat oder Jetty auch auf alte Bekannte. Außerdem ist auch auf die damit verwandten Produkte WebSphere Community Edition (WAS CE) und Sun Java System Application Server hinzuweisen.

Ein allgemeiner Vergleich kann nur eine grobe Orientierung geben. Um selbst eine Entscheidung für oder gegen einen Java EE Application Server treffen zu können, sollte man eigene Kriterien entwickeln und überprüfen. Das wichtigste Kriterium, das auch die Open-Source-Variante erfüllen sollte, ist die Java EE-Zertifizierung. Außerdem sollten aktuelle Java-Standards und Entwicklungsumge-

bungen unterstützt werden. Diese Kriterien werden von allen hier vorgestellten Servern unterstützt. Unterschiede gibt es bei der Art des Deployments, der Güte der Dokumentation und der Managementfähigkeit in produktiven Umgebungen. Ein wichtiges Unterscheidungskriterium im Gegensatz zu kommerziellen Produkten ist die Art der Lizenz bei Open-Source-Produkten, da diese sich auf die Verwendung auswirkt.

## Lizenzmodelle

Da es für den Open-Source-Bereich nicht nur eine Vielfalt bei den Produkten, sondern auch bei den Lizenzen gibt, wollen wir hier kurz auf die für die untersuchten Ap-



### Fürsprecher

Begleitet wird unser Artikel von vier Statements ausgewiesener oder interessierter Experten des jeweiligen Application Server, für den sie sprechen. Für den Geronimo spricht Geronimo-Buchautor Kristian Köhler, für den JBoss Dirk Weil, dessen Unternehmen diesen Application Server schon lange in Projekten einsetzt. Adam Bien war bei einer Recherche im Projekt angenehm vom Glassfish überrascht und JOnAS ist schon lange bei Lofi Dewanto im Einsatz.

	ASL 2.0	GPL 2.1	ILAN	CDDL 1.0
Beschreibung	Apache Software License	GNU Lesser General Public License	IBM License Agreement for Non-Warranted Programs	Common Development and Distribution License (wird auch für OpenSolaris verwendet)
Link	<a href="http://www.apache.org/licenses/">www.apache.org/licenses/</a>	<a href="http://www.gnu.de/lgpl-ger.html">www.gnu.de/lgpl-ger.html</a>	<a href="http://www-03.ibm.com/software/sla/slabd.nsf/viewbla">www-03.ibm.com/software/sla/slabd.nsf/viewbla</a>	<a href="http://www.sun.com/cddl/">www.sun.com/cddl/</a>
Deutsche Übersetzung	Nein	Ja	Ja	Nein
Copyleft-Effekt	Ohne	Beschränkt	Ohne	Beschränkt

Tabelle 1: Überblick über Lizenzmodelle

plication Server relevanten Lizenzen eingehen, was aber nicht dazu führen soll, eine Diskussion über „die beste Open-Source-Lizenz“ zu entfachen. Diese Entscheidung kann jeder selbst treffen oder sich in den entsprechenden Foren austoben. Bei konkreten Fragen sollte man sich sehr genau mit den verfügbaren Lizenztexten, die zum Teil auch auf Deutsch verfügbar sind, auseinandersetzen und gegebenenfalls sogar eine professionelle Rechtsberatung einholen. Besonders wichtig wird die Lizenz- und Haftungsfrage, wenn es um die Verwendung von Open-Source-Produkten in eigenen Lösungen geht.

- Die *Apache Software License* (ASL) ist die freie Softwarelizenz der Apache Software Foundation (ASF). Die aktuelle Version 2.0 wurde im Januar 2004 veröffentlicht. Sie erlaubt – ähnlich wie BSD – Veränderungen am Quellcode und die ausschließliche Weitergabe in Binärform, sofern die Lizenzbedingungen beigefügt werden oder darauf

hingewiesen wird. Es muss kein Quellcode beigelegt werden. Ebenso dürfen Änderungen am Code vorgenommen werden, ohne dass diese weitergegeben werden müssen. Damit ist diese die freieste der hier vorgestellten Lizenzen. Sie ist kompatibel mit der GPL. Dies ist einer von vielen Gründen, warum gerade Softwarehersteller gerne Apache-Software in ihre Produkte integrieren.

- Die *LGPL* ist die älteste der hier betrachteten Lizenzen und setzt ähnlich wie die *GPL* auf die Idee des Copyleft. Die Software darf für einen beliebigen Zweck genutzt, vervielfältigt und weitergeben oder geändert werden. Veränderte Versionen müssen dabei ebenfalls unter der *LGPL* (oder wahlweise der *GPL*) lizenziert werden. Grundsätzlich darf eine unter der *LGPL* lizenzierte Software nur mit ihrem Quelltext vertrieben werden oder mit der Zusage, den Quelltext auf Anfrage nachzureichen.
- Das *IBM License Agreement for Non-Warranted Programs (ILAN)* erlaubt

die Verwendung und Veränderungen des Programmes. IBM schließt sich selbst von der Haftung aus.

**Bei konkreten Lizenzfragen sollte man sich sehr genau mit den verfügbaren Lizenztexten auseinandersetzen und gegebenenfalls eine professionelle Rechtsberatung einholen.**

- Suns *CDDL* wurde von der Open-Source-Initiative (OSI) abgesegnet und bereits für OpenSolaris verwendet. Die Lizenz basiert auf der Mozilla Public License (MPL), sie ist allerdings inkompatibel mit der *LGPL*, da sie mit anderer Software kombiniert werden kann, die nicht unter *CDDL* steht. Es kann sogar der Lizenztext angepasst werden. Deswegen kann auch kommerzi-

Anzeige

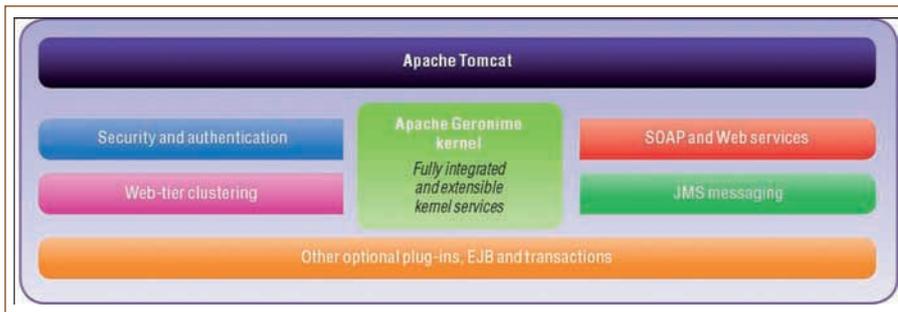


Abb. 1: WAS CE-/Geronimo-Architektur (Quelle: IBM)

elle Software wie der Sun Enterprise Server ohne diese Lizenz veröffentlicht werden.

### Geronimo/WAS CE

Bei Geronimo handelt es sich um den jüngsten zertifizierten Java EE Server. Geronimo wurde 2003 von Gluecode und ehemaligen JBoss Committern als „echter“ Open-Source-Server unter der Apache-Lizenz gegründet. Gluecode selbst wurde 2005 von IBM übernommen und deren Geronimo-Portierung wird heute als WAS CE unter der ILAN-Lizenz geführt.

Die Architektur von Geronimo basiert – ähnlich wie die des JBoss – auf einem Mikrokern, der über Zusatzprodukte leicht und flexibel erweitert werden kann. Er basiert auf den so genannten GBeans (Geronimo Beans), die über das „Inversion of Control“- (IoC-) Muster lose miteinander verknüpft sind.

Sie können sowohl Kernelmodule, Container, Konnektoren oder Module darstellen. Dabei wird ein Deployment-Plan beim Verteilen auf den Server in eine GBean umgewandelt. GBeans unterstützen den Lebenszyklus nach JSR 77

(Java EE-Management-Spezifikation) und können deshalb über JMX verwaltet werden.

Die von Geronimo verwendeten populären Open-Source-Produkte sind hier aufgeführt. Es handelt sich um ausgereifte Produkte (Axis, OpenEJB, CORBA von IONA/Trifork, MX4J, ActiveMQ, Tomcat, Jetty).

OpenEJB ist der Großvater unter den Open-Source-Application-Servern, da er am längsten auf dem Markt ist. Er wurde zwar nie zertifiziert, hat jedoch bisher jede neue Spezifikation implementiert – leider immer auch mit einer Architekturänderung. OpenEJB ist ein leichtgewichtiger, gut zu integrierender Container und Server. Deswegen existieren auch mehrere Integrationen für Tomcat, Geronimo und WebObjects. Da OpenEJB die EJB 1.1-, EJB 2.0- und EJB 2.1-Spezifikation unterstützt, wird er auch gerne für JUnit-Tests von EJBs eingesetzt. Je nach Version wurde die CMP-Persistenz mit unterschiedlichen Produkten (Castor JDO, TranQL) realisiert. Damit wird es



### Apache Geronimo – „Best of Breed“-Ansatz



Kristian Köhler  
(Orientations in Objects),  
Geronimo-Buchautor

Ziel des Geronimo-Projektes war von Anfang an, einen Java EE-konformen Application Server zur Verfügung zu stellen. Basis des Projektes sollte ein „Best of breed“-Ansatz sein. Statt einer Neuimplementierung des kompletten Java EE-Stacks werden bestehende, ausgereifte Java EE-konforme Komponenten als Serverbestandteile integriert. Dieser Integrationscharakter spiegelt sich in der Architektur des Servers wieder. Sie ermöglicht z.B. eine leichte Integration bestehender Software in die Serverlaufzeitumgebung oder die Anpassung des Servers zu einer optimierten Umgebung für die eigene Anwendung. Wird z.B. kein EJB-Container eingesetzt, kann dessen Start leicht verhindert werden oder der EJB Container wird komplett aus dem Server entfernt. Sämtliche am minimalen Kernel des Servers angemeldete Komponenten, seien es Anwendungen oder Serverbestandteile, unterliegen dem Deployment Management API (JSR 88) und können dementsprechend einzeln verwaltet werden. Für ein Deployment kann jede Client

Anwendung eingesetzt werden, die dieses API unterstützt. Als ein Beispiel sei hier Eclipse WTP genannt. Dieser Deployment-Ansatz garantiert Geronimo in Zukunft eine breite Werkzeugunterstützung.

Geronimo arbeitet modulbasiert. Das bedeutet, dass sämtliche Serverbestandteile wie auch Anwendungen in so genannten Modulen gepackt und verwaltet werden. Module können hierbei aus Anwendungen und zugehörigen Serverbestandteilen bestehen. Somit ist es möglich, konfigurierte Serverbestandteile wie z.B. einen Datenbank-Pool mit einer Webanwendung als Modul zu packen. Auf diese Weise kann der Aufwand für die Konfiguration einer Anwendung auf einem weiteren Server minimiert werden. Für die Ablage dieser Module wird ein so genannter ConfigurationStore eingesetzt. Hierbei handelt es sich um eine (zentrale) Ablage, die auch von mehreren Serverinstanzen gemeinsam genutzt werden kann. Einzelne Werte können leicht von den einzelnen Instanzen überschrieben werden. Dies ermöglicht die schnelle Konfiguration von mehreren Geronimo-Instanzen, die z.B. in einem Cluster zusammengeschlossen werden sollen und sich in den Einzelkonfigurationen nicht oder nur geringfügig unterscheiden. Module können ebenfalls über ein zentrales Repository an ent-

fernte Geronimo-Instanzen verteilt werden. Hierzu steht ein Plug-in-Mechanismus zur Verfügung mit dem zur Laufzeit der Funktionsumfang des Application Server erweitert oder auch aktualisiert werden kann. Im Internet stehen bereits Plug-in Repositories zur Verfügung, die Geronimo-Erweiterungen anbieten (z.B. [geronimo.apache.org/plugins/](http://geronimo.apache.org/plugins/)).

Hinter dem Geronimo steht neben der Apache Software Foundation ja auch IBM als großer Sponsor. Mit dem Kauf des Geronimo-Supporters Gluecode hat IBM eine klare Aussage in Richtung Geronimo getätigt und Ressourcen für die Weiterentwicklung des Servers zur Verfügung gestellt. Bereits jetzt beteiligen sich weitere Unternehmen an der Entwicklung des Servers, was einer breiten, transparenten und offenen Entwicklungskultur Vorschub leisten sollte. Nicht zuletzt die freizügige Apache-Software-Lizenz (ASL) mit ihrer Möglichkeit, Produktpakete mit Geronimo unter eigenen Lizenzen zu veröffentlichen, macht ein Engagement für kommerzielle Anbieter interessant.

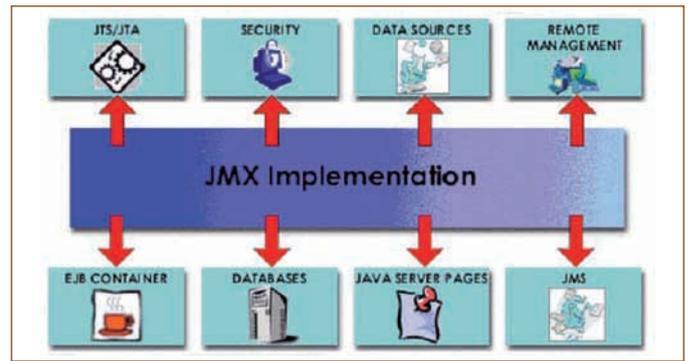
Im Laufe des letzten Jahres hat sich Umfang und Qualität der verfügbaren Informationen in Form von Büchern, Wiki und Online-Dokumentation deutlich verbessert, was ein Ein- und Umstieg auf den Geronimo erleichtern sollte.

für OpenEJB 3.0 einfach sein, mehrere JPA-(Java Persistence API-)Implementierungen zu unterstützen. Geronimo überzeugt durch das automatisierbare Deployment in ein zentrales Repository und seine gute Webkonsole. Außerdem bietet Geronimo für Eclipse und Maven eine gute Integration an.

WAS CE erweitert die Unterstützung von Geronimo um weitere Plattformen, Datenbanken und Directory-Server, sodass von IBM sogar ein offizieller Support dafür erworben werden kann. Bisher liegen die von IBM modifizierten JARs nicht als Sourcen vor, wenn auch vieles davon in Geronimo einfließt.

Anders als die Version 1.0 suggeriert, stellt Geronimo eine – auch für den produktiven Einsatz und die Entwicklung – reife Plattform dar. Schwächen gibt es noch in der Monitoring- und Managementfähigkeit und im EJB Clustering. Die flexible Apache-Lizenz 2.0 ist nicht nur für IBM eine interessante Option

Abb. 2: JBoss-Architektur



für die Integration in die eigenen Produkte. Wenn eine alternative JVM zu der von Sun benötigt wird, wie z.B. für AIX, bietet sich die IBM JVM an. Diese wird zwar nicht von Geronimo, dafür aber von WAS CE unterstützt. Anders als bei Open-Source-Lösungen üblich garantiert IBM zwei Produktversionen pro Jahr und eine Unterstützung der Vorgängerversion von WAS CE für minimal 18 Monate. Das schafft die nötige Planungs- und Investitionssicherheit. In der

sehr guten, mehrsprachigen Dokumentation ist sogar die Migration von und zu verschiedenen Application-Servern beschrieben.

### JBoss

JBoss wurde 1999 als kommerzielle Firma gegründet, um einen Open Source Application Server unter der LGPL zu entwickeln und mit Support und Service Geld zu verdienen. JBoss 4.0 war der erste Open Source Application Server mit

Anzeige

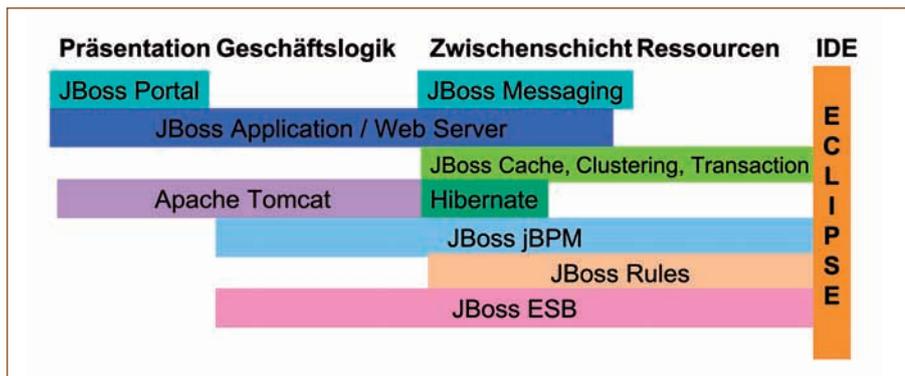


Abb. 3: JBoss-Produktfamilie

der J2EE 1.4-Zertifizierung (EJB 2.1). Durch die lange Historie hat JBoss unter allen Application-Servern den höchsten Marktanteil, wobei dieser jedoch bei Open-Source-Lösungen immer schwierig zu messen ist. Für die Vorgängerversion der aktuellen Version 4.0.4, also die Version 3.2.8, wurde keine eigene J2EE 1.3-Zertifizierung (EJB 2.0) durchgeführt.

JBoss ist bezüglich seiner Konzepte sehr innovativ. So verwendet er eine JMX-Architektur mit einem Mikroker-

nel. Als Webcontainer ist Tomcat bereits integriert. Alternativ kann auch Jetty integriert werden. JBoss bot als Erster einen vollständig in Java implementierten Application Server mit Hot Deployment, Cluster und Failover an. Er kann in sieben Varianten (21 MB bis 73 MB) heruntergeladen und installiert werden (Abb. 2).

JBoss hat mit Open Source immer seine eigenen Wege verfolgt. Das wird durch die Übernahme von JBoss durch Red Hat

(2006) voraussichtlich noch verstärkt werden (vom ORG zur COM). Um den ursprünglichen Application Server herum sind mittlerweile eine Vielzahl von Frameworks, Werkzeugen und Servern entstanden. Diese gehen auch in Richtung Integration von Anwendungen und wurden unter der Marke JBoss Enterprise Middleware System (JEMS), einer Open-Source-Plattform für serviceorientierte Architekturen, zusammengefasst. Zu JEMS gehören neben dem Application Server, dem Tomcat und Hibernate unter anderem auch ein Java-Portal, eine Messaging-Plattform, ein Transaktionsmonitor und JBoss Rules.

Ebenso gibt es das Seam-Anwendungs-Framework, das aus den einzelnen Flickern SOA, Ajax, JSF, EJB 3, Portlets, BPM einen perfekt vernähten Patchwork-Teppich anbieten will. Ein zugehöriger Enterprise Service Bus (ESB) und Java Business Integration (JBI) sind in Planung. Eines der neueren Projekte ist auch noch der Collaboration Server mit E-Mail und Kalenderfunktion. Wenn alle Komponenten einen genügend hohen Reifegrad erreicht haben, liegt hier ein System zur kompletten Integration einer IT-Landschaft vor. Damit hat JBoss zukünftig eine große integrierte Middleware-Produktpalette, die sonst im Open-Source-Bereich nur durch Einzelprodukte (der Communitys Apache, ObjectWeb oder Codehaus) abgedeckt wird.

Die Administration erfolgt entweder direkt über das Web Interface zu den MBeans bzw. JMX oder über die Webkonsole, welche zusätzliche Funktionen bietet. Es bleibt jedoch eine rechtliche Unsicherheit, ob die LGPL auf alle Komponenten korrekt angewandt wurde und welche Lizenz RedHat/JBoss in Zukunft verwenden wird.

Ein Wermutstropfen bei JBoss ist auch die nicht immer vorhandene, ausreichend aktuelle Dokumentation, obwohl sich hier im Gegensatz zu früheren Zeiten einiges verbessert hat. Weiterhin sind zahlreiche Bücher und Artikel zum Thema JBoss veröffentlicht worden.

### Glassfish

Bisher ist Glassfish (offizieller Name „Open Source Java EE 5 Application Ser-



## JBoss – die Nr. 1 in Sachen Verbreitung



**Dirk Weil**  
Geschäftsführer Gedoplan & langjähriger JBoss-Anwender

Wir verwenden JBoss in Kundenprojekten nun bereits seit über sechs Jahren. Die anfängliche Unsicherheit, ob dieser Server hinsichtlich Performance und Stabilität mit den kommerziellen Marktführern mithalten kann, ist mittlerweile vollkommen ausgeräumt.

Zunächst profitieren unsere Entwickler von der flexiblen Konfiguration des Servers. Nicht benötigte Services um den zentralen Kernel herum können auf einfache Weise deaktiviert werden. Damit lässt sich eine Entwicklungsumgebung mit moderatem Ressourcenbedarf zusammenstellen, in der auch ein „Server-Jo-Jo“ (Shutdown und Neustart) zeitlich akzeptabel ist. Dabei ist dies in den aktuellen Versionen nur noch selten nötig, da das Hot-Deployment von Anwendungen in aller Regel recht gut funktioniert.

Als Entwicklungsumgebung verwenden wir wie viele andere auch Eclipse. Die Integration von JBoss in Eclipse ist mittels verschiedener Plugins – z.B. JBoss-IDE, MyEclipse, WTP – sehr gut

möglich. Ebenso leicht gelingt dank des Auto-Deployments die Anbindung an das Build-Management mit Ant oder Maven.

Als Verfechter von Standards freuen wir uns über die konsequente Unterstützung der Java EE genauso wie über die hohe Innovationsgeschwindigkeit. JBoss 4 war einer der ersten Open-Source-J2EE 1.4-Server und hatte bereits Ende 2004 eine EJB 3-Preview.

Für unsere Kunden ist wichtig, dass JBoss einen umfangreichen Java-Enterprise-Stack zur Verfügung stellt – vom Persistenz-Provider über EJB- und Web-Container bis hin zu Portal und Process Integration – wengleich bei den letztgenannten Teilen sicher noch ein Teil des Weges vor JBoss liegt. Skalierbarkeit ist ein weiterer Punkt. Hier stellt JBoss eine Cluster-Unterstützung zur Verfügung, die für kleinere Cluster gut funktioniert. Größere Knotenanzahlen im Cluster sollten mit der kommenden Version von JBoss Cache besser unterstützt werden können.

Die Übernahme von JBoss durch Red Hat dürfte für etwas mehr Ruhe auf der rechtlichen Seite sorgen und die Investitionssicherheit erhöhen, nachdem das Vertrauen der Kunden ja im vergangenen Jahr unter der teilweise etwas überzogenen Debatte über Lizenz- und Markenrechte gelitten hatte.

ver“) unter der CDDL veröffentlicht, wobei Teile der Java EE-Bibliotheken unter Suns Binary Code License (BCL) veröffentlicht sind. Deshalb musste das Geronimo-Projekt z.B. auch eine eigene quellfreie Java EE-Implementierung erstellen.

Glassfish ist nicht nur für die derzeitige, sondern auch für die zukünftigen Java EE- und JWS DP-Spezifikationen die bevorzugte Plattform. Glassfish ist im Moment der einzige Open Source Application Server, der offiziell Java EE 5-kompatibel ist.



## Glassfish – nun für den kommerziellen Einsatz frei



**Adam Bien**  
(freier IT Consultant),  
Java EE-Experte und  
neugierig auf Glassfish

Sun hat immer wieder versucht, Application Server zu entwickeln und zu etablieren. Ich selbst bin zunächst auf den Netdynamics Application Server gestoßen. Ich fand diesen nicht schlecht, allerdings hat es damals (ca. Jahr 2000) zehn Wochen gedauert, bis ich eine Demo-CD bekommen hatte – die Software war bereits veraltet, als sie angekommen war ...

Einige Jahre später hatte ich die J2EE 1.3-Referenzimplementierung in Trainings und Workshops verwendet. Es gab mehr Fehler als Features, die Performance war inakzeptabel. Die kommerzielle Version des Application Server basierte auch auf der J2EE 1.3.1 RI und war allein aus diesem Grund für mich völlig uninteressant. Ferner waren die RIs für den kommerziellen Einsatz lizenztechnisch nicht erlaubt (und technisch nicht möglich).

Mein Interesse für Glassfish war deshalb anfangs nicht besonders groß – die ersten Java EE 5-Versuche hatte ich mit JBoss 4.0.x RC unternommen, was auf Anhieb sehr gut funktionierte. Auch die Beispielanwendung meines „Enterprise Architekturen“-Buches (Optimierung der Heizungssteuerung) läuft noch auf dem JBoss. Ich wurde nun in einem Java EE-Projekt zu meiner Meinung über Glassfish befragt und war somit „gezwungen“, mir den näher anzuschauen. Nach der einfachen Installation ist mir aufgefallen, dass die Web-basierte Admin-Konsole sehr übersichtlich und mächtig ist. Diese lässt sich durchaus mit kommerziellen Application-Servern vergleichen. Neben dieser gibt es auch eine genauso mächtige Möglichkeit, die Server über Kommandozeile zu administrieren. Die Konfiguration des Servers wird in der Datei `domain.xml` abgelegt. Man kann natürlich auch diese Datei editieren und so den Server (indirekt) administrieren. Dies ist insbesondere in der Entwicklungsphase interessant, wo der Server auf mehreren Entwicklungsplätzen installiert werden soll. Beide Admin-Werkzeuge sind sehr gut dokumentiert.

Was mich interessierte, waren insbesondere die Monitoring- und Management-Fähigkeiten

des Servers – das wird nicht nur in Projekten, sondern auch bei den Herstellern oft vernachlässigt. Alle deployten Komponenten können sowohl von Kommandozeile, Weboberfläche als auch der JConsole überwacht werden. Besonders interessant ist das Feature „Call Flow Monitoring“. Nach der Aktivierung sieht man hier, ähnlich zu einem Sequenzdiagramm, die Aufrufe der Methoden der Session Beans sowie den „Callstack“. Die Performance der einzelnen Methoden und die Gesamtdauer werden auch angezeigt. Die Ergebnisse werden in einer Datenbank (standardmäßig Derby) persistiert, sodass man die Daten auch außerhalb von Glassfish auswerten kann. Mir ist aufgefallen, dass die JPA-Entitäten nicht überwacht werden, und hatte nach diesem Feature in einer Sun-Mailinglist nachgefragt. Ich erhielt bereits nach ca. einer Stunde eine Antwort – es wurde für mich auch ein Feature Request eingerichtet. Die Hilfsbereitschaft der Glassfish-„Gemeinde“ ist (zumindest bisher ☺) überdurchschnittlich. Nach diesen positiven Erfahrungen habe ich mich nun entschieden, Glassfish genauer anzuschauen. Java EE 5-Anwendungen (EAR) können einfach hot- bzw. auto-deployed werden, es reicht die Ablage des Archivs in das „autodeploy“-Verzeichnis. Deployment-Fehler werden mit verständlichen Meldungen protokolliert, was die Suche nach der Ursache deutlich erleichtert. Für das Deployment von Java EE 5-Anwendungen sind keine proprietären Deployment-Deskriptoren oder Features notwendig. Die Dokumentation von Glassfish ist sehr umfangreich, die meisten Features werden zumindest mit einem Blog-Eintrag dokumentiert – meistens existiert auch ein PDF-Dokument. Leider habe ich noch keine Projekterfahrung mit Clustering (ist erst ab der Version 2 verfügbar) und kann nur schwer abschätzen, wie gut Glassfish skaliert bzw. wie hoch die Performance des Servers unter Last ist. Glassfish ist im Gegensatz zu seinen Vorgängern auch für den kommerziellen Einsatz freigegeben und in diesem Fall kostenfrei. Kommerzieller Support wird von Sun angeboten, was in größeren Unternehmen wichtig ist. Glassfish ist aufgrund der ausgeprägten Monitoring- und Management-Möglichkeiten nicht nur für die Entwicklung, sondern auch für die Produktion interessant.

Anzeige

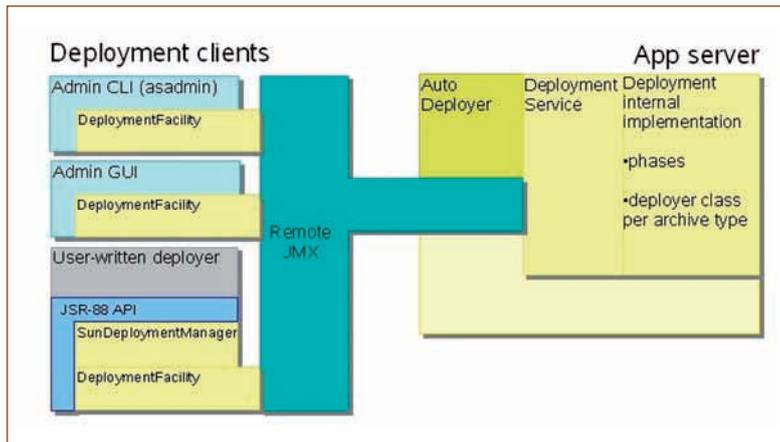


Abb. 4: Glassfish-Architektur für Web-Service-Management (Quelle: Sun)

Glassfish verwendet als Webcontainer Tomcat und als HTTP Connector das Grizzly HTTP Framework, das NIO Sockets unterstützt. Seine Architektur basiert auch auf JMX mit eigenen Erweiterungen (Appserver Management Extensions – AMX). Glassfish erfüllt damit den JSR 77 – Java EE Management Specification.

Die Referenzimplementierung des Java Persistence API im Rahmen des JSR 220 (EJB 3.0) beruht auf TopLink Essentials, welches ebenfalls open source verfügbar ist. Durch Glassfish sind auch einige Java EE-Pakete als Maven Repository verfügbar, was bisher aus rechtlichen Gründen nicht möglich war [12].

Das Deployment kann entweder über die Kommandozeile, ein Verzeichnis oder die Administrationskonsole erfolgen. Letztere macht einen guten Eindruck und ermöglicht die Einstellung vieler Konfigurationen über das Web Interface. Das Self Management Framework vereinfacht das Monitoring und die Administration des Servers.

Da es sich bei Glassfish bzw. den Sun Java System Application Server (SJSAS) um die Java EE-Referenzimplementierung handelt, gibt es sehr viele Beispiele, Handbücher, getestete Frameworks und Tutorials für die Entwicklung oder die Administration, leider jedoch sehr

wenige Informationen über die eigentliche Architektur. Da Glassfish identisch mit dem Sun Java System Application Server, Platform Edition (SJSAS PE 9) ist und auf dem Code der Version 8 des SJSAS beruht, stellt sich bis auf den Support die Frage nach der Existenzberechtigung der kommerziellen Version. Glassfish profitiert jedoch von der sehr guten mehrsprachigen Produktdokumentation und der Verfügbarkeit auf mehreren Plattformen. Auf Solaris mit NetBeans ist Glassfish sicher eine nahe liegende gute Lösung für Web-Service-Anwendungen.

Außerdem ist Glassfish das Produkt, mit dem die neuesten Java EE-Standards im Bereich Web Services, Ajax, JSF, Portlet und JBI (JSR 208) entwickelt und getestet werden. Wer die aktuellsten Java EE-Standards benötigt, findet in Glassfish eine gute Lösung. Er ist der einzige Open-Source-Application Server, für den eine SPECJ2004-Benchmark existiert. Auch das zeigt, dass Glassfish/SJSAS auch für große Anwendungen geeignet ist. Außerdem lässt die Integration von Glassfish/SJSAS in die Java Enterprise System Application Platform Suite keine Wünsche offen.

Seine Verbreitung wird sicher zunehmen, da er in SJSAS 9.0, im Java EE 5 SDK und in NetBeans 5.5 bereits integriert ist und das Migrations-Tool mehrere Application Server unterstützt. So nehmen die Anteile von Sun im Application-Server-Markt nach Jahren erstmals wieder leicht zu.



## JOnAS – unter dem Dach eines Konsortiums



**Lofi Dewanto**  
(Deutsche Post),  
JOnAS-Nutzer der ersten  
Stunde

Von großem Vorteil finde ich, dass die Organisationsform „Konsortium“ von ObjectWeb – ähnlich wie bei Apache – es ermöglicht, dass jedes Unternehmen (klein und groß) sowie jeder Entwickler sich an dem JOnAS-Projekt beteiligen können. Die Entwicklungsrichtung des Projektes wird von diesem Konsortium und nicht von einem einzigen Unternehmen gesteuert.

JOnAs ist dank der servicebasierten Architektur sehr schnell zu starten und zu stoppen. Jeder Service kann in einer Property-Datei einfach ein- und ausgeschaltet werden, sodass nur benötigte Services instanziiert werden müssen. Durch die Ähnlichkeit der Verzeichnisstruktur mit Tomcat ist JOnAS für jeden Entwickler, der bereits mit Tomcat gearbeitet hat, sehr einfach

zu verstehen. Somit ist der erste Einstieg in JOnAS für diese Entwicklergruppe sehr einfach. Für Produktionsumgebungen steht das Open-Source-Produkt Enhydra Enterprise zur Verfügung, welches zahlreiche zusätzliche Komponenten für JOnAS direkt mitliefert. Das Installationswerkzeug InstallShield gehört zum Lieferumfang von Enhydra Enterprise, sodass die Installation des Application Server sehr einfach durchzuführen ist. Dieses Werkzeug bietet zudem eine einfache Installation des Load-Balancer (Enhydra Director) und des Application Servers als Dienst unter Windows bzw. als Daemon unter Linux und Unix (Enhydra Tray) an. Für die Entwicklung von Webanwendungen stehen bereits reife Produkte wie XMLC (View-Technik für Webanwendungen – HTML, WML, VoiceXML), EAF (Framework-Technik für Web-Anwendungen) und DODS (objektrelationaler Mapper) zur Verfügung. Diese Produkte werden innerhalb von Enhydra Kelp – Enhydra integrierte Entwicklungsumgebung, welche auf Eclipse basiert – integriert.

## JOnAS

Der von Bull, France Telecom und INRIA 1999 gegründete JOnAS (Java Open Application Server) des ObjectWeb-Konsortiums ist schon ein alter Bekannter unter den Applications-Servern. Manche Dokumentation ist nur in Französisch vorhanden, was JOnAS vor allem in Frankreich populär macht. Für JOnAS wird die LGPL-Lizenz verwendet.

Auch ObjectWeb bietet eine große Anzahl von Open-Source-Middleware-Komponenten an. So nutzt JOnAS auch die Teilprodukte JORAM, JORM, CAROL, SPEEDO, MEDOR, JEREMIE, JOTM und MONOLOG.

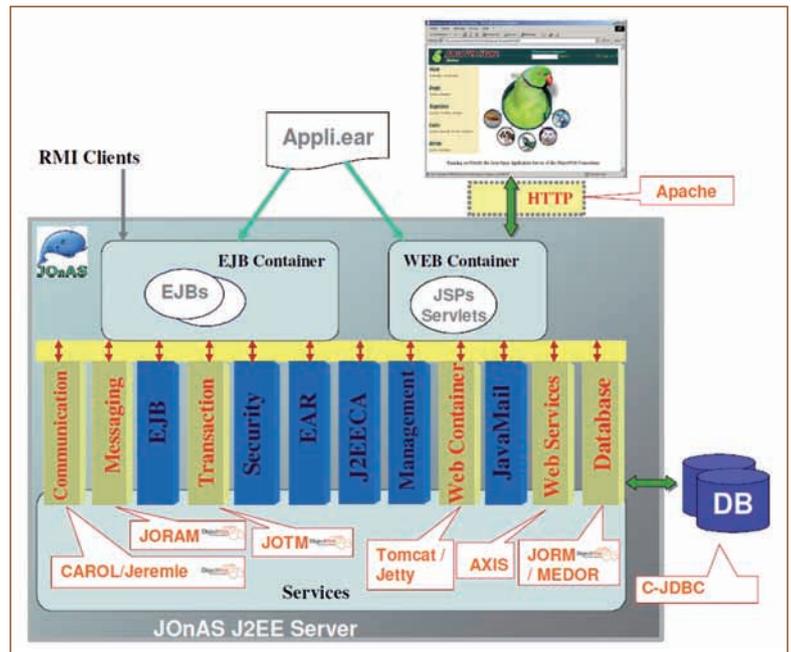
Die aktuelle Version 4.7 des JOnAS ist ein J2EE 1.4-zertifizierter Application

Server. Er hat seit JDK 1.1 und J2EE 1.2 bisher alle aktuellen Java-Versionen unterstützt. Außerdem existiert auch eine recht frühe EJB 3-Implementierung, die den OSGi-Plug-in-Mechanismus nutzt. Er kann sowohl mit Tomcat als auch mit Jetty als Webcontainer betrieben werden.

Für den Betrieb von JOnAS wird lediglich ein JDK der Version 1.4 oder 5 benötigt. JOnAS bietet volle Cluster- und Failover-Funktionalität mit Session-Replizierung. Diese Funktionalitäten stehen sowohl für den EJB-, als auch den Webcontainer zur Verfügung. Für die Entwicklung mit JOnAS bieten Plug-ins für Eclipse und JBuilder Hilfe bei der Erstellung der Deployment-Deskriptoren.

Zur Administration wird ebenfalls, wie bei den anderen Application-Servern, eine Weboberfläche zur Verfügung gestellt, die sich gut und intuitiv bedienen lässt. Außerdem gibt es eine JSR 160- (Java Management Extensions (JMX) Remote-)kompatible Schnittstelle für die Kommandozeilen, die auch für Skripte [13] genutzt werden kann. Im Punkt Sup-

Abb. 5:  
JOnAS-Architektur [11]



port gibt es, seit Red Hat JBoss gekauft hat und für JOnAS keinen weiteren Support leistet, vor allem kleinere Unternehmen wie beispielsweise die Together Teamlö-

sungen GmbH ([www.together.at](http://www.together.at)). JOnAS ist ein sehr leistungsfähiger Applications Server, nur konnte er sich nicht gegen den JBoss durchsetzen und hat deshalb bisher

Anzeige

den Durchbruch (noch) nicht so richtig geschafft.

### Migration, Wartung und Support

Für die Auswahl oder den Wechsel zu einem Application Server ist auch die Verfügbarkeit von Migrationsmöglichkeiten, Wartung und Support wichtig. Dabei werden diese Themen von den Open-Source-Lösungen oft recht unterschiedlich bis gar nicht behandelt. Sogar der Wechsel innerhalb der Produktfamilie ist nur wenig beschrieben.

Die professionellsten und umfassendsten Angebote in diesem Bereich bieten Sun für SJSAS, Red Hat für JBoss und IBM für WAS CE an.

Die Datenbank-Unterstützung ist bei allen Application-Servern sehr breit. Die meisten haben bereits Derby als Datenbank integriert, sodass beispielsweise eine Migration zu Cloudscape oder DB2 sehr einfach möglich ist.

Dadurch dass die Plattformen Windows, Linux oder Solaris meist unterstützt werden, gibt es mehrere Instal-

lationsoptionen. Die Integration mit bestehenden LDAP-Servern ist bei allen Application-Servern möglich. Die Support-Unterstützung und auch die Release-Pläne sind insgesamt sehr unterschiedlich ausgeprägt.

### Zusammenfassung und Ausblick

Bei der Verwendung von Open-Source-Produkten sollte man auf jeden Fall die Gesamtkosten im Auge haben. Von Fall zu Fall kann es deshalb sinnvoll sein, auch die kostengünstigen Einsteiger-

	JOnAS 4.x	Geronimo 1.1	WAS CE 1.0.1.2	Glassfish 1.0	JBoss 4.x
Lizenz	LGPL 2.1	Apache 2.0	ILAN	CDDL 1.0	LGPL 2.1
Java SE	1.4, 5.0	1.4, (5.0 ohne CORBA)	1.4 (Sun, IBM)	5.0	1.4, 5.0
Java EE	1.4	1.4	1.4	5.0	1.4
Webcontainer	Jetty, Tomcat	Jetty, Tomcat	Tomcat	Tomcat	Jetty, Tomcat
Servlet	2.4	2.4	2.4	2.5	2.4
JSP	2.0	2.0	2.0	2.1	2.0
EJB	2.1, (EJB 3.0 Draft)	2.1	2.1	3.0	2.1 (EJB 3.0 Draft)
JMS	1.1	1.1	1.1	1.1	1.1
WS, ESB (JAXP 1.2, JSR 109, JAX-RPC 1.0, SAAJ 1.2, JAXR 1.0, SOAP 1.1, WS-I Basic Profile 1.0)	JAX-WS 2.0, Celtix	JWSDP 1.5, Celtix	JWSDP 1.5	JWSDP 2.0, JAXB 2.0, StAX 1.0, SOA-Starterkit Open ESB, Tango, FastInfoset, Ajax, JSF	JWSDP 1.5, JBoss Rules, JBoss ESB, EJB 3-Paket, Ajax, JSF
JMX-Support	1.2	1.2	1.2	1.2	1.2
Webkonsole	Ja	Ja	Ja	Ja	Ja, nicht intuitiv
Hot Deploy	war, (alle V5),	Alle	Alle	Alle	Alle
Erweiterte Sicherheit	LDAP, Custom Security	LDAP, Custom Security	LDAP, Custom Security	LDAP, Custom Security	LDAP, Custom Security
Support	-	-	24x7	24x7 (SJSAS 9.0)	24x7
Plattform	Windows, Unix, Linux	Windows, Linux, Solaris (Ix86)	Windows, Linux (Ix86/PowerPC64), AIX	Windows, Linux, Solaris (Ix86, SPARC), Max OS X	Windows, Linux (Ix86) Solaris (Ix86, Sparc)
Installation	ZIP, Installer, 95 bis 66 MB	ZIP, Installer, 64 MB, (Minimal 16,5 MB)	ZIP, Installer, Response-Datei, 41 MB	ZIP, Installer, 250 MB	ZIP, Installer, automatisches Installationskript, 60 MB
IDE	Eclipse, Lombok, Ant, XDoclet, JBuilder	Maven, Eclipse, XDoclet	Maven, Eclipse, RAD, ASTK, XDoclet	Maven, NetBeans, Eclipse, Ant	Maven, NetBeans, Eclipse, Ant, XDoclet
Dokumentation	Dokumentation, Beispiele, Tutorial, Buch	Lückenhafte Dokumentation, mehrere Bücher erhältlich	Mehrsprachige Dokumentation, Beispiele	Dokumentation, Blueprints, Beispiele, Tutorial	Dokumentation, mehrere Bücher
Migrationsplan	Für JBoss	Für Tomcat, BEA, JBoss	WAS 6.x	SJSAS, Tomcat 4.x, WebSphere 5.x, WebLogic 8.x, JBoss 3.x,	JEMS, Seam
Betrieb, Ausfallsicherheit, Skalierbarkeit	HTTP-, EJB-Cluster, load balancing, Replikation	HTTP-Session-Cluster, Replikation	HTTP-Session-Cluster, Replikation	SPECjAppServer 2004	HTTP-, EJB-Cluster, Replikation
Datenbank-Support (direkt)	IBM DB2, Oracle, Cloudscape, HSQL embedded, Microsoft SQL, MySQL, Postgres, Sybase, Mckoi, Progress	Derby embedded	IBM DB2, Cloudscape embedded, Oracle, Microsoft SQL Server, MySQL	Oracle, Microsoft SQL, MySQL, Postgres, Sybase, IBM DB2, Derby embedded	IBM DB2, Oracle, Microsoft SQL Server, MySQL, Hypersonic embedded

Tabelle 2: Vergleich der konkurrierenden Java EE Application Server ([www.theserverside.com/tt/articles/article.tss?l=ServerMatrix](http://www.theserverside.com/tt/articles/article.tss?l=ServerMatrix), [en.wikipedia.org/wiki/Matrix\\_of\\_Application\\_Servers](http://en.wikipedia.org/wiki/Matrix_of_Application_Servers))

Versionen der kommerziellen Hersteller zu berücksichtigen. Schwächen haben die meisten der hier vorgestellten Open Source Application Server mit der Ausfallsicherheit – der JBoss-Cluster und JOnAS seien hier ausgenommen – und dem Betrieb (Monitoring, Management). Dort fehlen derzeit noch Funktionen, die jedoch durch Einzelprodukte oder Eigenlösungen ersetzt werden können. Alle Server lassen sich durch eine JMX-Administrationskonsole, wie z.B. MC4J [14], administrieren.

Immer wichtiger für einen produktiven Einsatz wird ein professionelles Supportangebot, sodass die Ressourcen im eigenen Haus gezielter eingesetzt werden können und doch eine gewisse Betriebssicherheit gegeben ist. Die Investitionssicherheit ist bei quelloffenen Lösungen wegen ihrer Nähe zu Standards und Zertifizierung oft sogar sehr hoch.

Alle besprochenen Produkte zeigen einen hohen Reifegrad und ausreichende Funktionalität. Die Unterstützung der Entwicklung für Open-Source-Werkzeuge ist sehr gut, sodass einer Verwendung dort nichts im Wege steht.

Ähnlichkeiten zeigen alle betrachteten Produkte in der Verwendung von Tomcat als gemeinsamem Webcontainer. Im Gegenzug ist hier aber die Tendenz zu beobachten, dass teilweise abgespeckte Versionen des Application Server als Webserver (Little-G, JBossWeb) angeboten werden. Der Mehrwert gegenüber Tomcat bleibt bisher jedoch fraglich.

Unterschiede gibt es jedoch bei der Realisierung der Persistenzschicht (Hibernate, TranQL, JORAM, TopLink), der Portalfunktion und der Web-Service-Dienste (Axis, JAX{B, P, M, WS}). Ganz interessant ist auch die Rolle von BEA, die erst vor kurzem die EJB 3.0-Persistenzimplementierung OpenJPA (basiert auf dem Produkt Kodo der Firma SolarMetric) an die Apache Software Foundation übergeben hat. Außerdem verwendet Geronimo das ursprünglich von BEA stammende XMLBeans-Framework. Hier ist sicherlich auch wichtig, welche Anforderungen durch die Architektur oder die verwendeten Datenbanken an die Auswahl des Application Server gestellt werden.

Für einen eigenen Funktionsvergleich der Produkte bietet sich die Sun-PetStore-Anwendung an, da sie für die meisten Produkte angepasst wurde. Für eine Performanzeinschätzung sollte man eine eigene Benchmark entwickeln.

Da auch in Zukunft die Verwendung von Integrationslösungen wie Web Services oder Portale zunehmen wird, ist es zu begrüßen, dass hier auch im Open-Source-Bereich Komplettlösungen zur Verfügung stehen, die auf bewährten Open-Source-Produkten beruhen.

Spätestens wenn alle Produkte Java EE 5.0/Java 6 unterstützen, sollte man noch einmal die Produkte miteinander vergleichen. Dabei werden auch das Management und das Deployment von Web Services eine größere Rolle spielen.



**Siegfried Klar** ([s.klar@impaq-pluralis.com](mailto:s.klar@impaq-pluralis.com)) und **Frank Pientka** ([frank.pientka@impaq-pluralis.com](mailto:frank.pientka@impaq-pluralis.com)) arbeiten beide als Senior Consultant bei der Impaq Pluralis AG in Dortmund. Beide sind seit mehreren Jahren im Bereich Java und Java EE tätig. Ihre Schwerpunkte sind Application Server (WebSphere, Geronimo, JBoss), Build-Management und Software-Architektur.

Anzeige

### ■ Links & Literatur

- [1] OSI Repository mit mehreren Open-Source-Lizenzen: [www.opensource.org/licenses/](http://www.opensource.org/licenses/)
- [2] Informationen über Open-Source-Lizenzen: [de.wikipedia.org/wiki/Open-Source-Lizenz](http://de.wikipedia.org/wiki/Open-Source-Lizenz), [www.ifross.de/ifross\\_html/lizenzcenter.html](http://www.ifross.de/ifross_html/lizenzcenter.html)
- [3] Tomcat: [tomcat.apache.org](http://tomcat.apache.org)
- [4] WAS CE-Dokumentation: [publib.boulder.ibm.com/wasce/Front\\_en.html](http://publib.boulder.ibm.com/wasce/Front_en.html) (auch Offline-Dokumentation)
- [5] Geronimo-Dokumentation: [wiki.apache.org/geronimo](http://wiki.apache.org/geronimo), [chariotsolutions.com/geronimo/geronimo-html-one-page.html](http://chariotsolutions.com/geronimo/geronimo-html-one-page.html)
- [6] OpenEJB: [openejb.codehaus.org](http://openejb.codehaus.org)
- [7] JBoss AS-Dokumentation: [labs.jboss.com/JBossas/docs](http://labs.jboss.com/JBossas/docs)
- [8] Glassfish: [glassfish.dev.java.net](http://glassfish.dev.java.net), [en.wikipedia.org/wiki/GlassFish](http://en.wikipedia.org/wiki/GlassFish)
- [9] Java EE-Zertifizierung: [java.sun.com/javaee/overview/compatibility.jsp](http://java.sun.com/javaee/overview/compatibility.jsp)
- [10] TopLink Essentials: [www.oracle.com/technology/products/ias/toplink/jpa/](http://www.oracle.com/technology/products/ias/toplink/jpa/)
- [11] JOnAS: [jonas.objectweb.org](http://jonas.objectweb.org)
- [12] Maven2 Repository: [maven-repository.dev.java.net/nonav/repository/](http://maven-repository.dev.java.net/nonav/repository/)
- [13] Java Management Extensions (JMX) Remote API 1.0: [www.jcp.org/en/jsr/detail?id=160](http://www.jcp.org/en/jsr/detail?id=160)
- [14] MC4J Management Console: [mc4j.org](http://mc4j.org)
- [15] Kristian Köhler, Christan Dedek: Geronimo schnell + kompakt, entwickler.press 2006