

# Korrekturen zu "Apache Geronimo. Handbuch für den Java-Applikationsserver"

Frank Pientka, dpunkt.verlag, 2009, ISBN 978-3-89864-517-1

Trotz großer Mühen und umfangreichen Korrekturlesens kommt es leider vor, dass sich Fehler in ein gedrucktes Buch einschleichen.

Da sich der Zeitplan der Nachfolgeversionen von Geronimo (2.2 bzw. 3.0) aus unterschiedlichen Gründen, etwas verzögert hat, ist der meiste Inhalt dieses Buches immer noch aktuell. Die Änderungen oder Neuerungen, die sich durch diese Versionen ergeben, werden im Abschnitt Ergänzungen behandelt.

Gerade diese Ergänzungen enthalten für aktuelle und spezielle Problemstellungen nützliche Hinweise.

Hier sind die uns bis zum **15. März 2010** bekannten Fehler aus unserem Buch wiedergegeben:

## Kapitel 8.6 Frameworks für Weboberflächen

Seite 158, Mitte:

Beim ursprünglichen Simple Object Access Protocol (das *seit Version 1.2 nur noch als Abkürzung SOAP und nicht mehr als Akronym steht*) handelt es sich um ein nachrichtenbasiertes XML-Format.

## Kapitel 9.3.3 Migration von Geronimo zu WebSphere

Seite 180, Tabelle 9-6, dritte Spalte:

Der Inhalt der beiden oberen Zellen - "1.4" - muss ersetzt werden durch "JSE 5.0".

## Kapitel 13.2. Die nächsten Versionen von Geronimo

*Geronimo 2.2 und Java-EE-6.0*

Die aktuelle Geronimo-Version 2.2 wurde im Dezember 2009 veröffentlicht.

Durch die starke Verspätung der Java-EE-6.0-Spezifikation wurden zwar einige Arbeiten an der kommenden Geronimo-Version 3.0 begonnen. Mit einer ersten offiziellen Version ist jedoch erst im dritten Quartal 2010 zu rechnen.

Neben den neuen WebProfilen wird Geronimo auch die OSGi-4.2-Blueprint-Spezifikation unterstützen.

Apache ist der einzige Anbieter, der einen kompletten Java-EE-6.0- und OSGi-4.2-Stack aus dem eigenen Haus anbieten kann, so dass sie auch in vielen anderen Produkten außerhalb von Apache Verwendung finden.

Da die Zukunft von Java Open Source ist und Apache dabei eine immer größere Rolle spielt, wächst damit auch die Bedeutung von Geronimo über Apache hinaus.

Weitere Informationen zur Planung und zum Inhalt der aktuellen Versionen finden Sie unter:

<http://cwiki.apache.org/confluence/display/GMOxPMGT/Apache+Geronimo+Release+Road+maps>

<http://cwiki.apache.org/confluence/display/GMOxPMGT/Geronimo+2.2+Release+Status>

<http://cwiki.apache.org/confluence/display/GMOxPMGT/Java+EE+6+Releases>

<http://cwiki.apache.org/GMOxDEV/road-map-for-jee6-web-profile.html>  
<http://cwiki.apache.org/GMOxPMGT/geronimo-30-release-roadmap.html>  
<http://cwiki.apache.org/GMOxPMGT/geronimo-22-release-status.html>  
<http://www.nabble.com/2.2-progress-td25552905s134.html>  
<http://svn.apache.org/repos/asf/geronimo/sandbox/blueprint>  
<http://wiki.apache.org/incubator/AriesProposal>  
<http://www.nabble.com/OSGI-progress-td25552927s134.html>

## **Kapitel A 4.1. Webservice-Werkzeug**

Seite 266, 4. Zeile statt „`cxf-tools:[sh|.bat]`“ sollte stehen „`cxf-tools.[sh|.bat]`“

# Ergänzung

Da es bei Geronimo selbst und bei einigen von und mit ihm eingesetzten Produkten zu Aktualisierungen gekommen ist, haben wir diese als Ergänzung zu den bestehenden Inhalten aufgenommen. Die meisten Änderungen sind durch die Geronimo-Version 2.2 gekommen, die hier auch der Vollständigkeit aufgeführt sind.

Zur besseren Lesbarkeit haben wir sowohl die Kapitel als auch die Seitenzahlen angegeben:

## Kapitel 1.2 Geschichte

Seite 4

**Geronimo 2.1.4** erschien am 30. März 2009. Neben Fehlerkorrekturen enthält er einige aktualisierte Bibliotheken für Javamail Provider 1.7, OpenJPA 1.2.1, OpenEJB 3.0.1, TXManager 2.1.2, ANT 1.7.1, MyFaces 1.2.6, Derby 10.4.2.0, ASM 3.1, Spring Plugin 2.5.6.

**Geronimo 2.1.5** soll Ende März 2010 erscheinen und enthält wichtige Fehlerkorrekturen und aktualisierte Bibliotheken für Tomcat 6.0.26, xbean 3.6, geronimo-el\_1.0.2\_spec, CXF 2.0.12, myFaces 1.2.8, OpenJPA 1.2.2, Derby 10.5.3.0\_1, OpenEJB 3.0.2, txmanager 2.1.4, ActiveMQ 4.1.2

**Geronimo 2.2** erschien am 30. Dezember 2009 und enthält neben 44 neuen Funktionen, 164 Verbesserungen und 436 Fehlerkorrekturen.

Geronimo 2.2 verbesserte die Erstellung von Assemblies, indem diese nach Funktionalitäten zusammengefasst werden. Neben der Webkonsole können Assemblies auch direkt mit dem Eclipse-Plugin erstellt werden. Außerdem werden von vielen integrierten Produkten aktuelle Versionen unterstützt, wobei hier Jetty7 und ActiveMQ 5.3 (mit Camel 2.0-Integration und Administrationskonsole als Plugin), TranQL Konnektor für PostgreSQL 8.2 und aktuelle Standards (JAX-B 2.1, JAX-WS API 2.1, JPA 2.0, Bean Validation) hervorzuheben sind.

Als neues Assembly gibt es jetzt geronimo-plugin-farm-node, was das verteilte Deployment von Plugins in einer Geronimo-Farm auch ohne Server über RMI und JMX, erlaubt.

Die Webkonsole wurde etwas entschlackt, so dass einige Funktionalitäten, wie z.B. Monitoring, UDDI, als Plugin nachinstalliert werden müssen. Die Webkonsole wurde an einigen Stellen erweitert (Konfiguration des EJB-Containers, Monitoring und die Assembly-Erstellung). Die Konfiguration von Tomcat, kann jetzt z.T., wie in Tomcat gewohnt, in der server.xml-Datei erfolgen und muss nicht mehr nur über die Konfiguration der GBean geschehen.

Die Beschreibung der GBean-Metadaten kann jetzt über Annotationen erfolgen.

Sowohl für Tomcat, als auch für Jetty wird jetzt die Java Authentication Service Provider Interface for Containers-Spezifikation (JASPIC JSR-196) unterstützt, um verschiedene SSO-Provider, wie z.B. openId oder SPNEGO zu unterstützen.

Die Geronimo plugin metadata (META-INF/geronimo-plugin.xml) und Plugin-Katalog-Formate (geronimo-plugins.xml, artifact\_aliases.properties, installed-plugins.properties) sind gegenüber vorherigen Versionen (1.x and 2.0.x) inkompatibel, so dass diese z.B. mit dem Maven-Aufruf

```
mvn org.apache.geronimo.buildsupport:car-maven-plugin:create-pluginlist
```

neu erstellt werden müssen.

Es ist jetzt möglich, das WADI-Cluster für Stateful Session-Beans (SFSB) zu nutzen. Ebenso wurde die Startzeit des Servers und der Webkonsole reduziert, indem die AJAX-Bibliothek

Dojo 1.1.1 abgespeckt wurde und einige Portlet-Plugins (Debug View, Monitoring, Plan Creator) initial nicht mit installiert werden.

Diese Portlets zum Überwachen, Debuggen und Planerstellen sind jetzt optional, wodurch die Webkonsole noch mehr angepasst werden kann. Außerdem werden aktuelle Bibliothekversionen unterstützt und die Werkzeuge Maven, Eclipse und GShell mit neuen Befehlen erweitert.

Eine wichtige Änderung, die ab **Geronimo 2.1.5** oder höher verfügbar ist, betrifft die Umstellung von der commons-logging- auf die slf4j-Bibliothek mit Wrappern für JUL (JavaUtilLogging), JCL (JakartaCommonsLogging) und log4j, wodurch einige Probleme bei unterschiedlichen Logging-Produktversionen behoben sind. Als Standard-Implementierung wird, nach wie vor log4J verwendet.

Erst Geronimo 3.0 wird den Java-EE-6.0-Standard komplett und OSGi intern unterstützen. Statt auf Groovy wird die GShell dann auf Karaf basieren, um auch OSGi-Bundles direkt verwalten zu können.

Den aktuellen Stand der unterstützten Standard-Specs finden Sie unter <https://svn.apache.org/repos/asf/geronimo/specs/trunk/>

Weitere Informationen zum Stand der Planung neuerer Geronimo-Versionen finden Sie im Geronimo-Wiki unter:

<http://cwiki.apache.org/confluence/display/GMOxPMGT/Geronimo+2.2+Release+Status>

<http://cwiki.apache.org/confluence/display/GMOxPMGT/Apache+Geronimo+Release+Road+maps>

<http://cwiki.apache.org/confluence/display/GMOxPMGT/Java+EE+6+Releases>

<http://cwiki.apache.org/GMOxPMGT/apache-geronimo-board-reports.html>

<http://cwiki.apache.org/GMOxPMGT/apache-geronimo-board-report-2010-01-january.html>

in den Diskussionsforen:

<http://www.nabble.com/Thinking-about-a-2.2-release-td22833029s134.html>

[http://www.nabble.com/-VOTE--Release-Geronimo-Eclipse-Plugin-2.1.4-\(RC1\)-td22920711s134.html](http://www.nabble.com/-VOTE--Release-Geronimo-Eclipse-Plugin-2.1.4-(RC1)-td22920711s134.html)

oder in den Online-Artikeln

<http://www.heise.de/newsticker/meldung/Apache-Geronimo-2-2-Zwischenschritt-mit-Zukunftsperspektive-872040.html>

<http://www.heise.de/developer/artikel/Ausritt-mit-Apache-Geronimo-2-2-942532.html>

## Kapitel 1.4 Produktvarianten

Seite 10:

**WASCE 2.1.1.1** erschien am 15. Dezember 2008

basiert auf Geronimo 2.1.3, Unterstützung für IBM Java SDK 6 Service Release 2 32bit und IBM Java SDK 5 Service Release 8a 64bit-Unterstützung, RAD-7.5- Unterstützung für WAS CE Plugin, neue Plattformen Windows 2008, Windows, Ubuntu 8.10, Fedora 10, neue TranQL XA Konnektoren für Informix Dynamic Server und MS SQL Server 2000/2005

**WASCE 2.1.1.2** erschien am 4. Mai 2009

basiert auf Geronimo 2.1.4, Unterstützung für IBM Java SDK 1.6.0 Service Release 4, IBM Java SDK 1.5.0 Service Release 9, Aktualisierungen für OpenJPA 1.2.1, OpenEJB 3.0.1, MyFaces 1.2.6, Derby 10.4.2.0, Spring 2.5.6, Beispiele, Sicherheitsupdates und aktualisierter Eclipse WTP Server Adapter

**WASCE 2.0.0.4** erschien am 20. Mai 2009 und basiert auf Geronimo 2.0.2  
Unterstützung für IBM 1.5.0 SR9 SDK, Sicherheitsupdates, Aktualisierungen für MyFaces v1.2.6, OpenJPA 1.0.3, genesis 1.3.1, xstream 1.2.2 und aktualisierter Eclipse WTP Server Adapter, Beispiele.

**WASCE 2.1.1.3** erschien am 19. August 2009 basiert auf Geronimo 2.1.4, Tomcat 6.0.20.  
Unterstützung für IBM Java SDK 1.6.0 Service Release 4, IBM Java SDK 1.5.0 Service Release 10, IBM Java SDK 6 SR5, Fedora 11, Ubuntu 9.04 LTS, WASCE Eclipse Plugin (WEP) 2.1.1.3 unterstützt Eclipse 3.5 (Galileo) und Geronimo 1.1, 2.0, 2.1 und braucht nur noch 5 statt 13 MB Platz, DB2 Express-C 9.7

<http://www-01.ibm.com/support/docview.wss?rs=203&cuid=swg27016503>

Alle Änderungen, die in den verschiedenen WAS-CE-Versionen gegenüber Geronimo gemacht wurden, sind unter <http://publib.boulder.ibm.com/wasce/changes/wasce-changes.htm> dokumentiert.

## Kapitel 1.6. Unterstützte Produkte

Seite 5:

Java: Sun, IBM 6.0 (nur WS CE), Java-EE-5.0-zertifiziert  
Plattformen: Windows XP/2003/Vista/2008, Linux (Ix86/PowerPC64) (RedHat, SUSE, Fedora, Ubuntu), AIX (nur WS CE), Solaris SPARC (nur WS CE)

### Kapitel 1.6.2 Unterstützte Plattformen

Seite 16:

Geronimo unterstützt aktuelle folgende Linux-Versionen:

- Asianux Server Version 3
- Red Hat Enterprise Linux, Version 4 Update 8
- Red Hat Enterprise Linux, Version 5 Update 4
- Novell SUSE Linux Enterprise Server, Version 10 Service Pack 3
- Novell SUSE Linux Enterprise Server, Version 11
- Novell SUSE Linux Enterprise Desktop, Version 10 Service Pack 3
- Fedora 11
- Ubuntu 9.04 LTS

Für Windows sind dies:

- Windows Vista SP2
- Windows 2008 Server SP2
- Windows Vista SP1

Als JVM-Ablaufumgebung kann auch Apache Harmony verwendet werden, wenn man die unter <http://cwiki.apache.org/confluence/display/GMOxDOC22/Apache+Harmony> beschriebenen Anmerkungen beachtet.

### Kapitel 2.6.1 Konfiguration über Umgebungsvariablen

Seite 35:

Ab Geronimo-Version 2.2 gibt es noch mehr Möglichkeiten, um Geronimo über Umgebungsvariablen zu konfigurieren.

<http://cwiki.apache.org/GMOxDOC22/command-geronimo-options.html>

Dabei können auch die unter <http://tomcat.apache.org/tomcat-6.0-doc/config/systemprops.html> beschriebenen, Systemvariablen von Tomcat gesetzt werden.

Alternativ können auch die Variablen direkt in den GBeans catalina oder jasper in der Datei config.xml oder in der plan.xml gesetzt werden.

```
<gbean name="JasperSystemProperties"
  class="org.apache.geronimo.system.properties.SystemProperties">
  <attribute name="systemProperties">
    org.apache.jasper.Constants.USE_INSTANCE_MANAGER_FOR_TAGS=true
  </attribute>
</gbean>
<gbean name="CatalinaSystemProperties"
  class="org.apache.geronimo.system.properties.SystemProperties">
  <attribute name="systemProperties">
    org.apache.catalina.STRICT_SERVLET_COMPLIANCE=true
  </attribute>
</gbean>
```

Für Windows: set GERONIMO\_OPTS=-D<OptionClass>=<value>

Für Unix: export GERONIMO\_OPTS=-D<OptionClass>=<value>

```
-Dorg.apache.geronimo.deployment.LenientMFCP=true
-Dorg.apache.geronimo.config.file=path/fileconfig.xml
-Dorg.apache.geronimo.config.substitutions.file=path/fileconfig-substitutions.properties
-Dorg.apache.geronimo.config.substitution.hostName=myHost
-Dorg.apache.geronimo.home.dir=path
-Dorg.apache.geronimo.server.dir=path
-Dorg.apache.geronimo.kernel.config.MPCLSearchOption=optimized
-Dorg.apache.geronimo.deployment.util.DeploymentUtil.jarUrlRewrite=true
-Dorg.apache.geronimo.gbean.NoProxy=false
-Dgeronimo.bootstrap.logging.enabled=false
-Dorg.apache.geronimo.server.name=Instanz
-Dorg.apache.geronimo.repository.boot.path=path
-Dorg.apache.geronimo.saaj.provider=axis2, sun
-Dorg.apache.jasper.Constants.USE_INSTANCE_MANAGER_FOR_TAGS=false
-Dorg.apache.catalina.STRICT_SERVLET_COMPLIANCE to Value=false
-Dcatalina.useNaming=false
-Dorg.apache.catalina.SESSION_COOKIE_NAME= Cookiname, wenn Sie nicht den Standardname
JSESSIONID
-Djava.net.preferIPv4Stack=true
-Djava.security.Provider=SUN
```

Als Beispiel können Sie das Standardverhalten für den EJB-Look-Up ändern:

Für Windows: set GERONIMO\_OPTS=-Dopenejb.localcopy=false

Für Unix: export GERONIMO\_OPTS=-Dopenejb.localcopy=false

Weitere Laufzeitoptionen sind unter

<http://cwiki.apache.org/GMOxDOC22/command-geronimo-options.html>  
beschrieben.

Dabei gibt es mehrere Arten die Geronimo-Skripte zu konfigurieren, wie unter  
<http://cwiki.apache.org/confluence/display/GMOxDOC22/Runtime+issues>  
beschrieben.

Für startup.[bat|sh] oder geronimo.[bat|sh] setzen Sie in der Datei

<Geronimo\_Home>/bin/setjavaenv.[bat|sh] z. B.:

```
if [-z "$JAVA_OPTS"];
then
  JAVA_OPTS="-Xmx256m -XX:MaxPermSize=128m -XX:+HeapDumpOnOutOfMemoryError"
```

fi

Für die GShell können Sie das entweder zu Laufzeit mit dem -J-Schalter

```
gsh> gsh geronimo/start-server -J "-Xmx256m -XX:MaxPermSize=128m -XX:+HeapDumpOnOutOfMemoryError"
```

oder in der Konfigurationsdatei <Geronimo\_Home>/etc/rc.d/start-server,default.groovy tun:

```
// Append some reasonable java flags if none were configured already
if (command.javaFlags.empty) {
    command.javaFlags << '-Xmx256m'
    command.javaFlags << '-XX:MaxPermSize=128m'
    command.javaFlags << '-XX:+HeapDumpOnOutOfMemoryError'}
}
```

## **Kapitel 2.6.2. Konfiguration über die config.xml-Datei**

Bei der config-substitutions.properties-Datei kommen in Geronimo 2.2 folgende Variable dazu

```
AccessTimeout=30 (für EJB 3.1 SingletonContainer)
BMPPoolSize=10 (für BMPContainer)
BulkPassivate=100 (für StatefulContainer)
Capacity=1000 (für StatefulContainer)
FarmName=DEFAULT_FARM (für Cluster-Deployment)
StatefulTimeout=20 (für StatefulContainer)
StatelessPoolSize=10 (für StatelessContainer)
StatelessTimeout=0 (für StatelessContainer)
StrictPooling=true (für StatelessContainer)
MulticastDiscoveryAddress=239.255.3.2 (für EJB-StatefulCluster)
StatelessPoolSize=10 (für StatelessContainer)
MulticastDiscoveryPort=6142 (für EJB-StatefulCluster)
```

und

```
WADIClusterName=DEFAULT_WADI_CLUSTER (Geronimo 2 .1 clusterName)
ClusterNodeName=NODE (Geronimo 2 .1 clusterNodeName)
wurden umbenannt.
```

## **Kapitel 4.1.4 Was ist eine GBean?**

Seite 60:

Die Erstellung einer GBean-Klasse wird in der Geronimo-Version 2.2 durch die Verwendung von Annotationen (@GBean, @Priority, @ParamSpecial, @ParamAttribute, @ParamReference, @Persistent, @Reference) vereinfacht, die im Paket org.apache.geronimo.gbean.annotation definiert sind.

<http://cwiki.apache.org/GMOxDOC22/gbean-annotations.html>

## **Kapitel 4.1.8 Klassenladerhierarchie**

Seite 73:

Sollte es zu Versionskonflikten mit verschiedenen Spring-Versionen kommen, so setzen Sie folgendes Attribut im Deploymentplan der Anwendung:

```
<hidden-classes>
  <filter>org.springframework</filter>
  <filter>META-INF/spring</filter>
</hidden-classes>
```

## Kapitel 6.2 Monitoring

Seite 108:

Ab Geronimo 2.2 ist die Monitoring-Oberfläche überarbeitet worden und es ergeben sich daraus einige Änderungen. So werden die Einstellungen und Werte der Monitoring-Konsole (Client) in einer eigenen Datenbank `MonitoringClientDB` und einer eigenen Konfigurationsdatei `var\monitoring\snapshot-config.xml` beim Aktivieren der Abfrage gespeichert.

Um ein anderes Datenbankprodukt, z.B. DB2 statt Derby zum Abspeichern der Monitoring-Informationen für die aktiven (in `ActiveMRCDB`) und die historischen Daten (in `ArchiveMRCDB`) zu verwenden, finden Sie Informationen unter:

<http://cwiki.apache.org/confluence/display/GMOxDOC22/Configuring+your+own+Monitoring+Plugin+DataSource>

## Kapitel 7.2. Eclipse verwenden

Seite 104:

Wenn Sie Eclipse 3.5.x (Galileo) zusammen mit Geronimo einsetzen wollen, so können Sie die jeweils aktuellste Version unter <http://people.apache.org/dist/geronimo/eclipse/unstable/> erhalten. Z.B. `geronimo-eclipse-plugin-2.1.5-deployable.zip` oder `geronimo-eclipse-plugin-2.1.5-updatesite.zip`

Das Geronimo Eclipse Plugin (GEP) 2.2.0 enthält einige neue Funktionen, wie Wizards zur Erstellung von einem Datenbankpools, Security-Realms oder zum assemblieren des Servers.

Alternativ können Sie auch das Eclipse-Plugin für WAS CE v2.1.1.3 unter <http://download.boulder.ibm.com/ibmdl/pub/software/websphere/wasce/updates/> verwenden.

Zur Verwendung von OpenEJB 3.1 wurde ein Plugin für Eclipse 3.5 unter <http://apache.org/dist/openejb/eclipse-plugin/update-site/>

zur Verfügung gestellt, das das Arbeiten damit erleichtert.

Um die Geronimo-Deploymentpläne in Eclipse 3.4 gegen ihr Schema zu validieren können Sie wie in

<http://cwiki.apache.org/confluence/display/GMOxDOC22/Using+Eclipse+XML+tools+in+Apache+Geronimo> beschrieben die XSD-Schemas in Eclipse registrieren.

Für Geronimo 2.2 wurde das Eclipse Plugin erheblich erweitert, so dass Sie jetzt auch aus einer bestehenden Anwendung einfacher ein Plugin erstellen oder die Serverkonfiguration anpassen können.

<http://cwiki.apache.org/confluence/display/GMOxDOC22/Converting+applications+into+Geronimo+plugins+using+GEP>

Außerdem können Sie damit auch `openejb-jar.xml`-Dateien editieren.

<http://cwiki.apache.org/confluence/display/GMOxDOC22/Making+deployment+plan+changes+with+Geronimo+Deployment+Plan+Editor>

Um die JSP automatisch, ohne ein erneutes Deployment zu ändern setzen Sie in der `<GERONIMO_HOME>/var/catalina/conf/web.xml`-Datei von Geronimo-Tomcat den Wert für `development` auf `true`.

```
<param-name>development</param-name>
<param-value>true</param-value>
```

## **Kapitel 7.2.5 Die Derby-Datenbank verwenden**

Seite 125:

Zu einer Datenbankquelle können Sie auch zum Deploymentzeitpunkt die nötigen Tabellen mit Inhalt anlegen lassen. Dazu fügen Sie die GBean `org.apache.geronimo.connector.DatabaseInitializationGBean` in die Definition des Ressourcen-Adapters ein:

```
<gbean name="DBInitialization" class="
org.apache.geronimo.connector.DatabaseInitializationGBean">
  <attribute name="testSQL">
    select * from customer
  </attribute>
  <attribute name="path">BankDB.sql</attribute>
  <reference name="DataSource">
    <name>SampleTxDatasource</name>
  </reference>
</gbean>
```

Oder auch alternativ

```
<!-- These GBean will create the tables for the database automatically -->
<gbean name="MyDSGBean"
class="org.apache.geronimo.connector.DatabaseInitializationGBean">
<!-- Execute the createTable.sql SQL statements if the following query does NOT return any
records -->
  <attribute name="testSQL">
SELECT t.tablename FROM SYS.SYSTABLES t WHERE lower(t.tablename)='myTable'
  </attribute>
<!-- Path to the SQL file defined in the module: 'my-sql' where the dependency above is
referencing -->
  <attribute name="path">META-INF/database/derby/createTables.sql
  </attribute>
  <reference name="DataSource">
    <name>jdbc/ DataSource </name>
  </reference>
</gbean>
```

## **Kapitel 7.4.2 Arbeiten mit Maven**

Seite 140:

Das aktuelle Maven-Repository mit Geronimo-Artefakten finden Sie unter:

<http://repo1.maven.org/maven2/org/apache/geronimo/specs/>

Informationen, wie Sie Geronimo-Implementierungen statt der Java-Enterprise-Spezifikationen verwenden können, finden Sie im Anhang B und Tabelle B.1 des freien Maven-Buches, das unter

[http://www.sonatype.com/books/maven-book/reference\\_de/appendix-enterprise.html](http://www.sonatype.com/books/maven-book/reference_de/appendix-enterprise.html) (auf Englisch)

<http://www.sonatype.com/books/maven-book/reference/appendix-enterprise.html> (auf Deutsch) erhältlich ist.

Setzen Sie die `groupId` auf `org.apache.geronimo.specs` und geben als `artifact version` die Version der Spezifikationsimplementierung in ihrer `pom.xml`-Datei an.

Unter

<http://www.mvnrepository.com/artifact/org.apache.geronimo.buildsupport>

finden Sie aktuelle folgende unterstützte Geronimo-Artifakte für Maven

Artifakt	Beschreibung
buildsupport	BUILD-Support für Geronimo
buildsupport-maven-plugin	BUILD-Support für Geronimo Server
car-maven-plugin	The car-maven-plugin handles all aspects of "predeploying" a module into a Geronimo car file and installing car files into a Geronimo server. As part of this it processes plan files to include the module Id and dependencies, and generates geronimo-plugin.xml metadata files and manages a geronimo-plugins.xml catalog in the local maven repository
geronimo-archetype-testsuite	zur Testsuite-Erstellung
geronimo-assembly-archetype	Plugin zur Assembly-Erstellung
geronimo-maven-plugin	Maven 2-Plugin, um Geronimo Server zu installieren, starten, stoppen und Module zu verwalten
geronimo-plugin-archetype	zur Plugin-Erstellung
groovy-build-library	BUILD-Unterstützung mit Groovy-Skripte
testsuite-maven-plugin	zur Geronimo-Testsuite-Erstellung

Eine Abhängigkeit wird dann z.B. für die POM-Abhängigkeit zum geronimo-plugin-archetype in der Version 2.1.4, wie folgt festgelegt:

```
<dependency>
  <groupId>org.apache.geronimo.buildsupport</groupId>
  <artifactId>geronimo-plugin-archetype</artifactId>
  <version>2.1.4</version>
</dependency>
```

Da Geronimo ab Version 1.2 mehr modularisiert wurde, ändern sich auch die groupIds in den Plänen.

**Beispiel:** Abhängigkeit zum Apache-Derby-Modul in Geronimo 1.1.1

```
<dependency>
  <groupId>geronimo</groupId>
  <artifactId>geronimo-derby</artifactId>
  <version>1.1.1</version>
</dependency>
```

**Beispiel:** Abhängigkeit zum Apache-Derby-Modul in Geronimo 2.1.4

```
<dependency>
  <groupId>org.apache.geronimo.modules</groupId>
  <artifactId>geronimo-derby</artifactId>
  <version>2.1.4</version>
</dependency>
```

Alle Module finden Sie unter

<http://www.mvnrepository.com/artifact/org.apache.geronimo.modules>

Die Konfigurationen finden Sie unter

<http://www.mvnrepository.com/artifact/org.apache.geronimo.configs>

Die Anwendungen finden Sie unter

<http://www.mvnrepository.com/artifact/org.apache.geronimo.applications>

Alle Plugins finden Sie unter

<http://www.mvnrepository.com/artifact/org.apache.geronimo.plugins>

Alle Assemblies finden Sie unter

<http://www.mvnrepository.com/artifact/org.apache.geronimo.assemblies>

Um selbst ein Assembly zu erstellen rufen Sie Maven, wie folgt auf:

```
mvn archetype:create \
  -DarchetypeGroupId=org.apache.geronimo.buildsupport \
  -DarchetypeArtifactId=geronimo-assembly-archetype \
  -DarchetypeVersion=2.2-SNAPSHOT \
  -DgroupId=org.apache.geronimo.plugins \
  -DartifactId=geronimo-jetty-liferay \
  -Dversion=1.0-SNAPSHOT
```

Was das Goal

```
<dependency>
  <groupId>org.apache.geronimo.buildsupport</groupId>
  <artifactId>geronimo-assembly-archetype</artifactId>
  <version>2.1.4</version>
</dependency>
```

in der pom.xml-Datei aufruft.

Artefakte zum Genesis-Build-Werkzeug (für Geronimo Konfiguration, Module) finden Sie unter

<http://www.mvnrepository.com/artifact/org.apache.geronimo.genesis>

Eine Zusammenfassung der Maven-Unterstützung finden Sie unter

<http://geronimo.apache.org/maven>

<http://geronimo.apache.org/maven/server/maven-plugins/dependencies.html>

Für die GShell, die ein eigenes Unterprojekt ist finden sich Informationen unter

<http://www.mvnrepository.com/artifact/org.apache.geronimo.gshell>

**Bemerkung:** Maven 2.2.1 oder höher wird benötigt, um die Sourcen oder Beispiele von Geronimo 2.2 zu übersetzen. Für die Beispiele kann mit dem Kommando `mvn clean install -Pit` dann auch ein einfacher Integrationstest durchgeführt werden. Für Geronimo 2.1.5 erfolgt gerade die Umstellung von Maven 2.0.11 auf 2.2.1.

## Kapitel 8.3. Persistenzframeworks

Seite 153:

Wenn Sie statt der internen OpenJPA-Implementierung Hibernate verwenden wollen, müssen Sie eine Klasse `GeronimoTransactionManagerLookup` implementiert

`org.hibernate.transaction.TransactionManagerLookup` erstellen und diese zusammen mit der Hilfsklasse `HibernateUtil` im Deploymentplan der Anwendung deklariert wird. Der Aufruf einer Transaktion erfolgt dann mit:

```
Transaction tr = HibernateUtil.getCurrentSession().getTransaction();;
```

Möchten Sie Hibernate als JPA-Provider verwenden, so deklarieren Sie das in der `persistence.xml`-Datei. Als `transaction-type` können Sie entweder auf `"JTA"` oder `„RESOURCE_LOCAL“` gesetzt werden. JTA wird für verteilte Transaktionen mit dem Zwei-Phasen-XA-Commit und –Protokoll verwendet. `RESOURCE_LOCAL` hingegen wird eher in non-JEE-Umgebungen, wie mit Java SE, verwendet.

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```

xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persisten
<persistence-unit name="PUNIT" transaction-type="JTA">
  <provider>org.hibernate.ejb.HibernatePersistence</provider>
  <jta-data-source>jdbc/myDatabasePool</jta-data-source>
  <exclude-unlisted-classes>>false</exclude-unlisted-classes>
<properties>
  <property name="hibernate.hbm2ddl.auto" value="create-drop"/>
  <property name="hibernate.transaction.manager_lookup_class"
  value=
"org.hibernate.transaction.GeronimoTransactionManagerLookup"/>
</properties>
</persistence-unit>
</persistence>

```

Die fehlende Unterstützung von Hibernate für Geronimo ist schon länger für `net.sf.hibernate.transaction.GeronimoTransactionManagerLookup` unter <http://jira.atlassian.com/browse/CONF-10315>, <http://opensource.atlassian.com/projects/hibernate/browse/HHH-1368> und <http://opensource.atlassian.com/projects/hibernate/browse/HHH-3212> gemeldet, wird leider von JBoss nicht in die aktuellen Versionen ein gepflegt.

Weitere Informationen zu den Klassen und zum Deploymentplan finden Sie unter <http://cwiki.apache.org/GMOxDOC22/jboss-to-geronimo-hibernate-migration.html> <http://qbeukes.blogspot.com/2009/09/geronimo-using-hibernate-as-your-jpa.html> Weitere Beispiele für die Verwendung von JPA mit EJB (Bean Managed Persistence with JPA, Container Managed Persistence with JPA), mit JSF aber ohne EJB (Working with JSF and JPA) oder als Client (Using Java Persistence API in application client) finden Sie mit Source-Code unter <http://cwiki.apache.org/GMOxDOC21/tutorials.html>

## Kapitel 8.4 Messaging-Systeme

Seite 154:

Ab Geronimo 2.2 wird auch ActiveMQ 5.3 unterstützt, das auch Camel 2.0 bereits integriert. Dadurch lassen sich recht einfach die unter <http://activemq.apache.org/enterprise-integration-patterns.html> und <http://camel.apache.org/enterprise-integration-patterns.html> beschriebenen Enterprise Integration Patterns nutzen.

Die Konfiguration von ActiveMQ erfolgt über die Datei

```
$GERONIMO_HOME\var\activemq\conf\activemq.xml
```

Dort können Sie Camel-Elemente, wie Routen oder Komponenten konfigurieren:

```

<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
  <route>
    <from uri="activemq:example.A"/>
    <to uri="activemq:example.B"/>
  </route>
</camelContext>

```

Eine andere Möglichkeit ist die ActiveMQ 3.0 Webconsole, die Sie gesondert unter <http://repo1.maven.org/maven2/org/apache/activemq/activemq-web-console/5.3.0/activemq-web-console-5.3.0.war> erhalten. Diese steht unter <http://repository.apache.org/snapshots/org/apache/geronimo/plugins/activemq-webconsole-tomcat/2.2-SNAPSHOT/activemq-webconsole-tomcat-2.2-20091104.002204-4.car>

auch als Plugin zur Verfügung und kann mit allen abhängigen Komponenten mit  
GSH> deploy/install-plugin activemq-webconsole-tomcat-2.2-20091104.002204-4.car  
installiert und mit <http://localhost:8080/activemq-console/> aufgerufen werden.

Weitere Infos finden Sie unter <http://activemq.apache.org/web-console.html>

Damit Sie diese verwenden können müssen Sie jedoch folgende Umgebungsvariable setzen

```
set GERONIMO_OPTS=-Dwebconsole.type  
=properties -Dwebconsole.jms.url=tcp://localhost:61616  
-Dwebconsole.jmx.url=  
service:jmx:rmi:///jndi/rmi://localhost:1099/JMXConnector  
-Dwebconsole.jmx.user=system -Dwebconsole.jmx.password=manager -Xms512m -Xmx512m -  
XX:MaxPermSize=128m
```

Eine andere Möglichkeit, um Camel-Komponenten zu überwachen, ist die Camel 2.0 Web Console, die Sie über

<http://repo1.maven.org/maven2/org/apache/activemq/activemq-web-console/5.3.0/activemq-web-console-5.3.0.war> erhalten. Eine Beschreibung finden Sie unter <http://camel.apache.org/web-console.html>

### ***Kapitel 9.3.1 Migration von Geronimo 1.x, 2.0, 2.1 zu 2.2***

Seite 169:

Wie in Kapitel 2.6.2. Konfiguration über die config.xml-Datei beschrieben, können Sie am einfachsten Versionsnummern in den Deploymentplänen ersetzen, indem Sie dies in den Dateien client\_artifact\_aliases.properties bzw. artifact\_aliases.properties ablegen. Damit vereinfacht sich die Migration von Geronimo 1.x, 2.0, 2.1 zu 2.2.

### ***Kapitel 9.3.2 Migration von Tomcat zu Geronimo***

Seite 173:

Da sich ab Geronimo 2.2 die Konfiguration des integrierten Tomcat-Webservers verändert hat, wird dadurch die Migration von Tomcat zu Geronimo erleichtert, da jetzt wieder die Datei unter <GERONIMO\_HOME>/var/catalina/server.xml angepasst werden kann.

Je nach zu migrierender Anwendung sind unter <http://cwiki.apache.org/GMOxDOC22/best-practices-tomcat-to-geronimo-migration.html> die verschiedenen Schritte pro Variante (mit Apache Derby, Apache ActiveMQ oder Apache AXIS) skizziert.

### ***Kapitel 9.3.3 Migration von Geronimo zu WebSphere***

Seite 176:

Für die Migration von Hibernate mit EJB 2.1 zu OpenJPA nach dem JPA-Standard 1.0 und EJB 3.0 finden Sie weitere Informationen im developerworks-Artikel:

Migrating legacy Hibernate applications to OpenJPA and EJB 3.0, 22 Aug 2007

[http://www.ibm.com/developerworks/websphere/techjournal/0708\\_vines/0708\\_vines.html](http://www.ibm.com/developerworks/websphere/techjournal/0708_vines/0708_vines.html)

### ***Kapitel 11.1.1 Allgemeine Faktoren***

Seite 191:

In Geronimo hat sich die Konfiguration des Clusters in der config.xml-Datei geändert<sup>1</sup>. In der GBean-Deklaration heißt das Attribut j2eeType von

gbeanInfo="org.apache.geronimo.farm.config.BasicNodeInfo" "GBean" statt "NodeInfo".

Weitere Informationen zum Deployment in einem Cluster ab Geronimo 2.2 finden Sie unter:

<http://cwiki.apache.org/GMOxDOC22/farming-using-deployment.html>

Die Verwendung von Produkten, wie Jetty, Open Terracotta oder GCache zum Clustern mit Geronimo wird unter

<http://cwiki.apache.org/confluence/display/GMOxDEV/Clustering>

beschrieben.

**Bemerkung:** die Verwendung von GCache als Cluster ist mit Geronimo z.Z. nicht möglich.

<http://cwiki.apache.org/GMOxDEV/geronimo-clustering-with-gcache.html>

<http://cwiki.apache.org/GMOxDEV/test-drive-geronimo-ejb-clustering.html>

Damit Webanwendung oder Statefull SessionBeans mit WADI in Geronimo 2.2 geclustert werden können, müssen die Deploymentdeskriptoren erweitert werden.

In der web.xml-Datei setzen Sie das Marker Attribut distributable:

```
<web-app>
.....
  <distributable/>
.....
</web-app>
```

In der Geronimo-web.xml-Datei setzen Sie in den Schemas geronimo-clustering-wadi-X.xsd und geronimo-tomcat-clustering-wadi-X.xsd definierten Attributen für Tomcat:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://
geronimo.apache.org/xml/ns/j2ee/web/geronimo-tomcat-2.0.1">

  <environment>
    <moduleId>
      <groupId>yourGroupId</groupId>
      <artifactId>yourArtifactId</artifactId>
      <version>YourVersion</version>
      <type>war</type>
    </moduleId>
  </environment>
  <context-root>/yourPath</context-root>
  <tomcat-clustering-wadi/>
</web-app>
```

Um Cluster für die stateful Session-Beans (SFSB) zu verwenden, müssen Sie auch den Deployment-Deskriptor openejb-jar.xml um das Element openejb-clustering-wadi ergänzen:

```
<openejb-jar xmlns="http://openejb.apache.org/xml/ns/openejb-jar-2.2"
  xmlns:sys="http://geronimo.apache.org/xml/ns/deployment-1.2">
...
  <openejb-clustering-wadi/>
</openejb-jar>
```

Standardmäßig wird für die Beandaten-Replikation das Multicast-Protokoll verwendet.

Voraussetzung ist, dass das Betriebssystem für dieses Protokoll konfiguriert ist und sich alle Knoten im gleichen physikalischen IP-Subnetz befinden.

Möchten Sie statt des Multicast-Protokolls nur das Unicast-Protokoll verwenden, so müssen Sie jeden Knoten einzeln als GBean mit className, UniqueId (zur eindeutigen Identifikation des Knoten), der host (Hostname oder IP-Adresse des Knotens) und nextWadiStaticMember (Namen des nächsten Knotens in der Reihe, wenn nicht der letzte) in der config.xml-Datei

definieren und mit dem Parameter `disableMCastService` Multicast für die GBean `DefaultDispatcherHolder`, wie folgt, ausschalten.

```
<gbean name="DefaultDispatcherHolder">
  <attribute name="disableMCastService">true</attribute>
</gbean>
```

Weitere Informationen finden Sie unter:

<http://cwiki.apache.org/confluence/display/GMOxDOC22/WADI+Clustering>

oder für Tomcat

<http://tomcat.apache.org/faq/cluster.html>

Leider gibt es Probleme mit dem Routen der WADI - JSESSIONID bei `jvmRoute` mit dem `mod_jk`-Module, was erst in der Version 2.1.5 korrigiert ist.

### **Kapitel 11.3.3. Benutzerverwaltung**

Bei der Verwendung der internen Derby-Datenbank (für `ActiveMRCDB`, `ArchiveMRCDB`, `MonitoringClientDB`, `SystemDatabase`, `UddiDatabase`) sollten Sie beachten, dass diese standardmäßig ohne Benutzer und Passwort gestartet wird. Am einfachsten editieren Sie die Pläne der Datenbankpools und tragen dort einen eigenen Benutzer mit Passwort ein. Dieses wird dann verschlüsselt in der Konfigurationsdatei abgelegt.

Ab der in Geronimo 2.2 enthaltenen Derby 10.4.1.3-Version ist es dann auch möglich sich als Derby-Client mit dem Netzwerkserver zu authentifizieren. (DERBY-3585)

## **Kapitel 11.5 Blogserver Roller**

Seite 225:

Bei der Verwendung der `roller-custom.properties`-Datei muss zusätzlich noch dieser Abschnitt

```
<!-- jndi resources
  <resource-ref>
    <res-ref-name>jdbc/rollerdb</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
  </resource-ref>
-->
```

auskommentiert werden.

Für Roller 4.0 gibt es auch unter <https://svn.apache.org/repos/asf/geronimo/plugins/roller> die Quelle-Dateien für das Roller-Plugin mit MySQL oder Derby als Datenbank und unter <https://svn.apache.org/repos/asf/geronimo/plugins/roller/trunk/roller-themes/src/main/resources/themes/glike/glike-preview.png> ein für Geronimo angepasstes Aussehen, wie es auch vom Geronimo-Blog <http://blogs.apache.org/geronimo/> verwendet wird.

## Kapitel 11.6.2 Integration mit LifeRay

Seiten 229 – 231:

Bei den 5er-LifeRay-Portal-Versionen haben sich einige Änderungen ergeben, die sich auch auf die Verwendung mit Geronimo auswirken. Mit der LifeRay Portal-Version 5.1 gibt es jetzt eine Open-Source-Version LifeRay Portal Standard Edition (LPSE) unter der MIT-Lizenz und eine kommerzielle LifeRay Portal Enterprise Edition (LPEE).

Die für die Verwendung von Geronimo mit LifeRay nötigen Anpassungen werden hier beschrieben.

Um LifeRay unter Windows zu starten, müssen Sie die Standardspeicherwerte der JVM ändern.

Bevor Sie LifeRay das erste Mal starten, sollten Sie die Heap-Größe auf `-Xms128m -Xmx1024m` hoch und die `-XX:MaxPermSize=128m` runter setzen.

Am einfachsten geht das in der Datei `setenv.bat`

```
set JAVA_OPTS=%JAVA_OPTS% -Xms128m -Xmx1024m -Dfile.encoding=UTF8 -Duser.timezone=GMT
```

Für eine 64bit JVM wird von LifeRay folgende Einstellung für Unix empfohlen:

```
JAVA_OPTS="$JAVA_OPTS -XX:NewSize=700m -XX:MaxNewSize=700m -Xms2048m -Xmx2048m -XX:MaxPermSize=128m -XX:+UseConcMarkSweepGC -XX:SurvivorRatio=10"
```

Bei der Verwendung von Derby 10.4.1.3 mit LifeRay 5.1 muss bei der Tabelle `Role_` der Spaltentyp `VARCHAR` angepasst werden.

<http://support.liferay.com/browse/LEP-6730>

```
ALTER TABLE Role_ ALTER DESCRIPTION SET DATA TYPE VARCHAR(1024);
```

Ab der LifeRay Version 5.2 wurde das Dateilayout und auch die Konfiguration der Datenbankverbindungen vereinheitlicht. So können Sie die Datenbankverbindung mit dem `Apache-commons-connection-pool` in der `portal(-ext).properties` Datei ändern oder Sie setzen im LifeRay Entwicklungs-Extension-Environment den Namen der Datenquelle in der Spring-Datei `META-INF/ext-spring.xml` definieren. Die Datenquelle selbst legen Sie dann in Ihrem Applikationsserver an und können so auch dessen erweiterte Funktionalitäten nutzen.

Für die Datei `portal-ext.properties` unter `repository\liferay\liferay-portal-tomcat\5.2.0\liferay-portal-tomcat-5.2.0.car\WEB-INF\classes` sehen die Einträge für die Derby-Datenbank wie folgt aus:

```
jdbc.default.driverClassName=
org.apache.derby.jdbc.EmbeddedDriver
jdbc.default.url=jdbc:derby:portal
jdbc.default.username=APP
jdbc.default.password=APP
```

Weitere JDBC-Einstellungen in der `portal.properties` Datei für andere Datenbanken finden Sie unter: <http://issues.liferay.com/browse/LPS-1113>

Folgende Links geben weitere aktuelle Informationen zur Verwendung von LifeRay und Geronimo:

Datenbankkonfiguration: <http://www.liferay.com/web/guest/community/wiki/-/wiki/Main/Database+Configuration>

Ein aktuelles Problem mit LifeRay und Geronimo finden Sie unter:

<http://issues.liferay.com/browse/LEP-5577>

<http://issues.liferay.com/browse/LEP-6680> Server assembly -- LifeRay as a Geronimo Plugin

Die Deinstallation der Demo-Anwendung 7Cogs ist hier beschrieben:

<http://www.liferay.com/web/guest/community/wiki/-/wiki/Main/7Cogs%20sample%20data>

In der aktuellen Liferay Portal Standard Edition (LPSE) 5.2.3 wird Geronimo 2.1.4 unterstützt.

Für die kommende LPSE 6.0.1 ist eine Unterstützung für Geronimo 2.2 geplant.

## Kapitel 11.8 Prozessserver (NEU)

Eine gute Möglichkeit, BPEL-Prozesse mit Geronimo auszuführen, bietet die Intalio|BPP Community Edition, die mit einem fertig konfigurierten Apache Geronimo Server 2.0.1 für Apache Derby oder MySQL in der aktuellen Version 6.0 ausgeliefert wird. Dabei wird die Ausführung von BPMN 1.1, WS-BPEL 1.0 und 2.0-Prozessen mit der Apache ODE-Engine unterstützt.

Für den Entwurf der Prozesse wird der ebenso freie BPMS Designer, der in die Entwicklungsumgebung Eclipse integriert ist, angeboten.

## Kapitel 12.1 Fehlerbehebung

Damit Sie das http-Zugriffsprotokoll von Tomcat mit Programmen wie AWStats oder Webalizer, auswerten können, sollten Sie das Common Log Format (CLF) mit dem Muster „%h %l %u %t \"%r\" %s %b" oder kürzer pattern="common" verwenden. Alternativ kann für das Combined Log Format "%h %l %u %t %r %s %b%{Referer}i \ %{User-agent}i" (pattern="combined") oder das Extended Log Format (ELF) verwendet werden.

Ab Geronimo 2.1 können Sie das Zugriffsprotokoll in der

<GERONIMO\_HOME>/var/config/config.xml-Datei in der GBean-Konfiguration des Tomcats abschalten mit

```
<module name="org.apache.geronimo.configs/tomcat6/2.1/car">
```

```
...
```

```
  <gbean name="TomcatEngine">
    <attribute name="initParams">
      name=Geronimo
    </attribute>
    <reference name="TomcatValveChain"/>
  </gbean>
  <gbean name="AccessLogValve" load="false"></gbean>
```

oder das Logmuster ändern mit

```
<module name="org.apache.geronimo.configs/tomcat6/2.1/car">
```

```
<gbean name="AccessLogValve">
  <attribute name="initParams">prefix=access_log.
  suffix=.txt
  pattern=common
  fileDateFormat=yyyy-MM</attribute>
</gbean>
```

Damit überschreiben Sie das initiale Verhalten im Deploymentplan

```
<GERONIMO_HOME>/repository/org/apache/geronimo/configs/tomcat6/2.2/tomcat6-2.2.car/META-INF/plan.xml
```

Informationen zu den Formaten und Auswertungssoftware dafür finden Sie unter:

[http://en.wikipedia.org/wiki/Common\\_Log\\_Format](http://en.wikipedia.org/wiki/Common_Log_Format)

<http://www.w3.org/TR/WD-logfile.html>

[http://en.wikipedia.org/wiki/List\\_of\\_web\\_analytics\\_software](http://en.wikipedia.org/wiki/List_of_web_analytics_software)

<http://tomcat.apache.org/tomcat-6.0-doc/config/valve.html>

<http://cwiki.apache.org/GMOxDOC22/managing-valve.html>

## Kapitel 12.1 Arbeiten hinter einem Proxy

Seite 233:

Da sich ab der Geronimo-Version 2.1.4 die Java-Optionen nicht mehr über JAVA\_OPTS, sondern über GERONIMO\_OPTS gesetzt werden, sollte dies mit

```
GERONIMO_OPTS= -Dhttp.proxyHost=IP-Adresse
```

```
-D http.proxyPort=8080
```

gesetzt werden.

## Kapitel 12.3 Performance und Tuning

Seite 230:

### *JPA-Tuning*

OpenJPA-Caching zur Leistungsverbesserung konfigurieren

Die OpenJPA-Implementierung bietet die Möglichkeit, häufig verwendete Daten zu speichern, um die Leistung zu verbessern. OpenJPA stellt dazu second-level-Datencaches und Abfragecaches bereit, die über die Property `openjpa.DataCache` bzw. `openjpa.QueryCache` in der Datei `persistence.xml` konfiguriert werden.

```
<persistence-unit name="JAVA-EE">
  <jta-data-source>jdbc/orderds</jta-data-source>
  <properties>
    <property name="openjpa.jdbc.DBDictionary" value="derby"/>
    <property name="openjpa.jdbc.Schema" value="APP"/>
    <property name="openjpa.DataCache" value="true"/>
    <property name="openjpa.RemoteCommitProvider" value="sjvm"/>
  </properties>
</persistence-unit>
```

Danach können Sie für die zu cachenden Entities mit der Annotation `@DataCache` und der Zeitangabe in Millisekunden den Zeitraum für das Caching angeben.

Wenn Sie den Datencache in einer verteilten Umgebung aktivieren möchten, müssen Sie die Property `<property name="openjpa.RemoteCommitProvider" value="sjvm"/>` konfigurieren.

Sie können festlegen, dass ein Cache zu bestimmten Zeiten bereinigt wird. Das Merkmal "EvictionSchedule" der OpenJPA-Cacheimplementierung akzeptiert einen Zeitplan für die Bereinigung im Cron-Format. Das Cron-Format gibt die Minuten, die Stunde, den Tag des Monats, den Monat und den Wochentag an, beginnend mit 1 für Sonntag. Das Symbol \* (Stern) bedeutet immer Übereinstimmung. Wenn Sie z. B. festlegen möchten, dass ein Cache jeden Sonntag in jedem Monat um 15 Uhr und 45 Minuten bereinigt werden soll, fügen Sie die folgenden Merkmale der Datei `persistence.xml` und einer zu cachenden Entity-Klasse hinzu:

```
<property name="openjpa.DataCache" value="true(CacheSize=5000 SoftReferenceSize=0
EvictionSchedule='15,45 * * 1')"/>
```

```
import org.apache.openjpa.persistence.DataCache;
@Entity
@NamedQuery(name="product.all",query="select p from Product p")
@DataCache(timeout=6000000)
public class Product implements Serializable {
    @Id
    @Column(name="PRODUCT_ID")
    protected int productId;
```

Standardmäßig kann der OpenJPA- Abfrage-Cache 1000 Elemente speichern, die Anzahl der Objekte können sie jedoch, wie folgt festlegen.

```
<property name="openjpa.QueryCache" value="CacheSize=1000, SoftReferenceSize=100"/>
```

Weitere Informationen zum Optimieren von OpenJPA finden Sie in den Kapiteln 5.2., 6.1, 10. Caching und 14. Optimization Guidelines im Apache OpenJPA User's Guide. Alternativ können Sie auch das unter <http://www.ibmpressbooks.com/articles/article.asp?p=1192350> verfügbare Online Verfügbare Kapitel 8 Apache OpenJPA des Buches „Persistence in the Enterprise: A Guide to Persistence Technologies“ von Roland Barcia, Geoffrey Hambrick, Kyle Brown, Robert Peterson, Kulvir Singh Bhogal, IBM Press, 2008 lesen.

Eine weitere Optimierungsmöglichkeit für OpenJPA betrifft den Zeitpunkt und die Art des Ladens von Objektgeflechten.

OpenJPA unterstützt fetch groups, die auf FetchType.LAZY gekennzeichneten Attributen angewandt werden können, um das Laden zu beschleunigen. So werden alle Objekte von Customer und ListItem geladen, wenn ein Order-Objekt geladen wird.

```
@Entity
@FetchGroups({
    @FetchGroup(name="detail", attributes={
        @FetchAttribute(name="lineItems"),
        @FetchAttribute(name="customers")
    })
})
class Order
```

Werden Objekte häufig geändert oder gelöscht, so ist es sinnvoll, den Cache mit folgender Option auszuschalten.

```
<property name="openjpa.DataCache" value="false"/>
<property name="openjpa.LargeTransaction" value="true"/>
```

Standardmäßig werden 1000 Abfrageausführungen und 100 SoftReferenzen im LRU(Least Recently Used) -Cache zwischengespeichert.

```
<property name="openjpa.QueryCache" value="CacheSize=1000, SoftReferenceSize=100"/>
```

Das Deaktivieren des Abfrage-Caches geht wie folgt:

```
<property name="openjpa.QueryCache" value="false"/>
```

Wenn Sie die Property auf false setzen, können Sie den Abfrage-Cache wieder deaktivieren.

```
<property name="openjpa.QueryCache" value="false"/>
```

Weitere Infos zur Konfiguration des Cachings mit OpenJPA finden Sie unter

[http://openjpa.apache.org/builds/1.0.2/apache-openjpa-1.0.2/docs/manual/ref\\_guide\\_caching.html](http://openjpa.apache.org/builds/1.0.2/apache-openjpa-1.0.2/docs/manual/ref_guide_caching.html)

[http://openjpa.apache.org/builds/latest/docs/manual/manual.html#ref\\_guide\\_caching](http://openjpa.apache.org/builds/latest/docs/manual/manual.html#ref_guide_caching)

Inzwischen können Sie auch EHCache mit OpenJPA als Cache-Provider verwenden.

Unter

<http://sourceforge.net/projects/ehcache/files/ehcache-openjpa>

finden Sie auch eine aktuelle Version des Caching Plugin von EHCache für OpenJPA und unter

[http://ehcache.sourceforge.net/documentation/openjpa\\_provider.html](http://ehcache.sourceforge.net/documentation/openjpa_provider.html)

die passende Beschreibung dazu.

```
<property name="openjpa.DataCacheManager"
value=" ehcache (CacheSize=1000)"/>
<property name="openjpa.DataCache"
value=" ehcache"/>
```

Über ein Plugin unter <http://geronimo.apache.org/plugins/openjpa2/>

ist es ab Geronimo 2.1.3 möglich, die neuere JPA-2.0-Spezifikation zu verwenden. Alle Anpassungen in den Dateien config.xml, client\_artifact\_aliases.properties und

artifact\_aliases.properties erfolgen dann automatisch. Weitere Informationen dazu können Sie der README-Datei unter <https://svn.apache.org/repos/asf/geronimo/plugins/openjpa2/tags/openjpa2-2.1.3-beta/README.txt> entnehmen.

```
<module name=
"org.apache.geronimo.configs/openjpa2/2.2-M3-SNAPSHOT/car" />
  <module name=
"org.apache.geronimo.configs/persistence-jpa20-deployer/2.2-M3-SNAPSHOT/car">
    <gbean name="PersistenceUnitBuilder">
      <attribute name="defaultPersistenceProviderClassName">
        org.apache.openjpa.persistence.PersistenceProviderImpl
      </attribute>
      <attribute name="defaultPersistenceUnitProperties">
        openjpa.Log=commons
        openjpa.jdbc.SynchronizeMappings=
        buildSchema(ForeignKeys=true)
        openjpa.jdbc.UpdateManager=operation-order
        openjpa.Sequence=table(Table=OPENJPASEQ, Increment=100)</attribute>
      <attribute propertyEditor=
"org.apache.geronimo.deployment.service.EnvironmentBuilder"
name="defaultEnvironment">
<environment:environment xmlns:ns2="http://geronimo.apache.org/xml/ns/attributes-1.2"
xmlns="http://geronimo.apache.org/xml/ns/deployment-1.2" xmlns:environment=
"http://geronimo.apache.org/xml/ns/deployment-1.2">
  <dependencies>
    <dependency>
<groupId>org.apache.geronimo.configs</groupId>
  <artifactId>openjpa2</artifactId>
    <type>car</type>
  </dependency>
</dependencies>
</environment:environment>
</attribute>
</gbean>
</module>
```

```
org.apache.openjpa/openjpa/1.0.3/jar=
org.apache.openjpa/openjpa/2.0.0-M3/jar
org.apache.geronimo.configs/persistence-jpa10-deployer/2.2-SNAPSHOT/car=
org.apache.geronimo.configs/persistence-jpa20-deployer/2.2-M3-SNAPSHOT/car
org.apache.geronimo.framework/j2ee-security//car=
org.apache.geronimo.configs/client-security/2.2-SNAPSHOT/car
org.apache.geronimo.configs/j2ee-server/2.2-SNAPSHOT/car=
org.apache.geronimo.configs/client/2.2-SNAPSHOT/car
org.apache.openjpa/openjpa/1.2.1/jar=org.apache.openjpa/openjpa/2.0.0-M3/jar
```

```
org.apache.geronimo.configs/openjpa//car=org.apache.geronimo.configs/openjpa2/2.2-M3-
SNAPSHOT/car
org.apache.geronimo.configs/persistence-jpa10-deployer/2.2-SNAPSHOT/car=
org.apache.geronimo.configs/persistence-jpa20-deployer/2.2-M3-SNAPSHOT/car
org.apache.openjpa/openjpa/1.2.1/jar=org.apache.openjpa/openjpa/2.0.0-M3/jar
org.apache.geronimo.configs/persistence-jpa10-deployer//car=
org.apache.geronimo.configs/persistence-jpa20-deployer/2.2-M3-SNAPSHOT/car
org.apache.openjpa/openjpa/1.0.3/jar=org.apache.openjpa/openjpa/2.0.0-M3/jar
org.apache.geronimo.modules/geronimo-persistence-jpa10//jar=
org.apache.geronimo.modules/geronimo-persistence-jpa20/2.2-M3-SNAPSHOT/jar
```

## Kapitel 12.3.2 Geronimo optimieren

Seite 243:

Gerade bei der Performance der Konnektoren können Sie ab Geronimo 2.2 einige Einstellungen vornehmen. Die Werte hängen jedoch von ihren konkreten Leistungsanforderungen ab, die Sie mit Performancetests validieren sollten, um keine negativen Nebeneffekte zu erzeugen.

Für http/s und NIO sind die Parameter für den Tomcat-Webcontainer unter <http://tomcat.apache.org/tomcat-6.0-doc/config/http.html> und <http://tomcat.apache.org/tomcat-6.0-doc/config/executor.html> beschrieben.

Folgende Parameter mit den Standardwerten sind:

- WebConnectorConTimeout=20000
- maxThreads=150
- minSpareThreads=25
- maxSpareThreads=75
- enableLookups=false
- acceptCount=100
- disableUploadTimeout=false
- tomcatThreadPool
- maxIdleTime=60000
- daemon=true

In folgender Tabelle aus der Tomcat-Dokumentation sind die Möglichkeiten, abhängig vom eingesetzten Konnektor aufgeführt.

	<b>Java Blocking Connector</b>	<b>Java Nio Blocking Connector</b>	<b>APR Connector</b>
Tomcat Version	5.x 6.x	6.x	5.5.x 6.x
Support Polling	Nein	Ja	Ja
Polling Size	N/A	Restricted by memory	Restricted by memory
Read HTTP Request	Blocking	Non Blocking	Blocking
Read HTTP Body	Blocking	Sim Blocking	Blocking
Write HTTP Response	Blocking	Sim Blocking	Blocking
SSL Support	Java SSL	Java SSL	Open SSL
SSL Handshake	Blocking	Non Blocking	Blocking
Max Connections	maxThreads	Polling size	Polling size

Für EJB (BMP, SFSB) und das RMI sind folgende Parameter mit den Standardwerten relevant:

- MinThreadPoolSize=200 (ab Geronimo 2.0)
- MaxThreadPoolSize=500 (ab Geronimo 2.0)
- AccessTimeout=30 (ab OpenEJB 3.1 in Geronimo 2.2)
- Capacity=1000 (ab OpenEJB 3.1 in Geronimo 2.2)
- StatelessPoolSize=10 (ab OpenEJB 3.1 in Geronimo 2.2))
- StatelessTimeout=0 (ab OpenEJB 3.1 in Geronimo 2.2)
- BMPPoolSize=10 (ab OpenEJB 3.1 in Geronimo 2.2)
- StatefulTimeout=20 (ab OpenEJB 3.1 in Geronimo 2.2)
- BulkPassivate=100 (ab OpenEJB 3.1 in Geronimo 2.2)

Um die Stabilität und Performance der HTTP/S-Verarbeitung vom integrierten Tomcat zu verbessern können Sie die Apache Portable Runtime (APR), wie unter <http://tomcat.apache.org/tomcat-6.0-doc/apr.html> beschrieben, verwenden. Das ist besonders sinnvoll, wenn Tomcat auch ohne einen Apache-http-Server verwendet wird.

Je nach Plattform müssen Sie dazu die dazu gehörigen Bibliotheken APR, OpenSSL und die JNI-Wrapper-Header-Dateien (libtcnative) in einen Ordner kopieren, der Teil der java.library.path-Variable ist. Am besten nehmen Sie dazu <GERONIMO\_HOME>/bin. Für Windows können Sie dazu auch das Verzeichnis WINDOWS/system32 nehmen. Da unter Windows oft kein C-Compiler installiert ist, laden Sie sich am besten die neueste Version von tcnative-1.dll und der APR-Bibliothek unter <http://apache.mirrors.esat.net/apr/binaries/win32> oder <http://tomcat.heanet.ie/native/> herunter.

Nach dem Start von Geronimo kann dann über die Webkonsole der Tomcat-APR-HTTP/S-Konnektor konfiguriert werden. Bitte beachten Sie, dass standardmäßig für HTTPS dann APR mit OpenSSL verwendet wird.

Da OpenSSL nicht das JKS (Java KeyStore)-Format unterstützt, müssen Sie in der var/config/config.xml-Datei in der GBean-Konfiguration des Tomcats das Attribut keystoreFile konfigurieren.

```
<gbean name="TomcatWebSSLConnector" class="org.apache.geronimo.tomcat.connector.Https11ConnectorGBean">
  <attribute name="keystoreFile">var/security/keystores/geronimo-default</attribute>
  <attribute name="keystoreType">PKCS12</attribute>
```

Wie Sie solch eine Zertifikatsdatei erstellen ist in der Dokumentation von OpenSSL unter <http://www.madboa.com/geek/openssl/#cert-self> beschrieben.

```
Ein Beispielaufruf lautet openssl req \
  -x509 -nodes -days 365 \
  -newkey rsa:1024 -keyout mycert.pem -out mycert.pem
```

Oder Sie kopieren eine Datei unter dem Verzeichnis openssl\bin\PEM.

```
Um die Warnung (WARN [DTDEntityResolver] Deprecated public id in dwr.xml. Use:
<!DOCTYPE dwr PUBLIC "-//GetAhead Limited//DTD Direct Web Remoting 3.0//EN"
```

```
"http://getahead.org/dwr//dwr30.dtd">) bzgl. DWR in Geronimo 2.2 auszuschalten, ergänzen Sie in der Datei server-log4j.properties folgende Zeilen:
```

```
# Avoid DWR WARN messages during startup and admin console Debug Views
# Can be removed once problem is fixed in DWR
log4j.logger.org.directwebremoting.impl.DTDEntityResolver=ERROR
```

## Kapitel 12.4 Die Installation optimieren

Seite 246:

Geronimo lässt sich ab der Version 2.1.4 einfacher als Dienst in Windows und Linux integrieren.

Für Linux sind unter <https://issues.apache.org/jira/browse/GERONIMO-4622> zwei Startup-Skripte mit Beschreibung hinterlegt, die in zukünftigen Geronimo-Versionen integriert werden sollen.

Für Geronimo 2.1.4 installieren Sie das Plugins „Windows Service Wrapper“ aus dem Repository unter <http://geronimo.apache.org/plugins/geronimo-2.1.4/geronimo-plugins.xml>.

Dann installieren Sie mit <geronimo\_home>\bin\service\_pr.bat install Geronimo als Windows-Dienst unter dem Namen "Apache Geronimo Service - geronimosrv". Mit NET START "Apache Geronimo Service - geronimosrv" können Sie den Dienst starten.

Zum Entfernen des Dienstes rufen Sie folgenden Befehl auf:

```
<geronimo_home>\bin\service_pr.bat remove
```

Weitere Informationen dazu finden Sie unter folgenden Links:

<http://issues.apache.org/jira/browse/GERONIMO-4394>

<http://cwiki.apache.org/GMOxDOC22/running-geronimo-as-a-service.html>

## Kapitel A.1 Installation und Deinstallation

Seite 259:

Für openSUSE stehen neuere Geronimo-2.x-Pakete zum Herunterladen unter

[http://download.opensuse.org/repositories/Java:/packages/openSUSE\\_11.1/noarch/](http://download.opensuse.org/repositories/Java:/packages/openSUSE_11.1/noarch/)

<http://download.opensuse.org/distribution/11.2/repo/oss/suse/noarch/>

bzw. [http://en.opensuse.org/Package Repositories](http://en.opensuse.org/Package_Repositories)

bereit.

Für die WebSphere Application Server Community Edition unter SUSE Linux Enterprise Server 11 (SLES 11)) bietet Novell den Priority Support und auch Patches dafür als RPM an.

Weitere Infos dazu finden Sie unter

<http://www.novell.com/partners/ibm/wasce.html>

[http://www.novell.com/products/linuxpackages/server11/i386/websphere-as\\_ce.html](http://www.novell.com/products/linuxpackages/server11/i386/websphere-as_ce.html)

[http://www.novell.com/products/linuxpackages/server11/i386/java-1\\_6\\_0-ibm.html](http://www.novell.com/products/linuxpackages/server11/i386/java-1_6_0-ibm.html)

<http://download.novell.com/patch/finder/>

<http://www-01.ibm.com/support/docview.wss?uid=swg27015951>

Ab SLES 10 SP1 wurden die RPM-Pakete von Geronimo auf websphere-as\_ce umbenannt.

Die RPM-Pakete für WAS CE bzw. Geronimo können Sie wie folgt installieren:

```
rpm -Fhv websphere-as_ce.rpm websphere-as_ce-doc.rpm
    websphere-as_ce-samples.rpm
rpm -Fvh websphere-as_ce.rpm
rpm -Fvh java-1_6_0-ibm-1.6.0_sr6-1.1.1.i586.rpm java-1_6_0-ibm-jdbc-1.6.0_sr6-
1.1.1.i586.rpm
rpm -ivh geronimo-servlet-2_5-api-1.2-2.3.noarch.rpm
geronimo-ejb-3_0-api-1.2-2.3.noarch.rpm
geronimo-jpa-3_0-api-1.2-2.3.noarch.rpm
geronimo-jsp-2_1-api-1.2-2.3.noarch.rpm
```

Für den Debian Package Manager (dpkg) finden Sie die Pakete von WAS CE 2.1.1.3 für die Ubuntu 8.04 "Hardy" i64-Reihe unter:

<http://archive.canonical.com/ubuntu/pool/partner/w/wasce-server/>

<https://launchpad.net/ubuntu/+source/wasce-server>

<https://launchpad.net/ubuntu/hardy/hppa/wasce-server/2.1.1.3final-4>

Der Aufruf für die Installation lautet dann: dpkg -i dateiname.deb bzw.

```
dpkg -i wasce-server_2.1.1.3final-4_all.deb
```

## Kapitel A 4.1 Webservice-Werkzeug

Seite 266:

Für das CXF-Framework gibt es ab Geronimo 2.2 folgendes Werkzeug, das alternativ zu dem Jaxws-tools-Werkzeug eingesetzt werden kann:

```
cxf-tools.[sh|.bat]
```

```
Usage: cxf-tools <toolName> <tool options>
```

where <toolName> is:

```
java2ws    - generate portable artifacts from class
wsdl2java  - generate portable artifacts from WSDL
```

Eine Beschreibung der erweiterten Werkzeug-Optionen für CXF finden Sie unter:

<http://cwiki.apache.org/GMOxDOC22/cxf-tools.html>

Die Sie auch mit `cxf-tools.[bat|sh] java2ws -h` oder `cxf-tools.[bat|sh] wsdl2java -h` anzeigen lassen können.

Der Aufruf `jaxws-tools.[bat|sh] wsgen [options] <SEI>` wird verwendet, um von einer Service-Endpunktschnittstelle (SEI) die implementierenden Java-Klassen zu erzeugen (nach dem *Code-First-Prinzip*). Der Aufruf `jaxws-tools wsimport [options] <WSDL_URI>` wird verwendet, um aus einer WSDL-Beschreibung (<WSDL\_URI> steht für eine WSDL-Datei oder einer URL zu einer `http/server/context?wsdl`) die implementierenden Java-Klassen zu erzeugen (nach dem *Contract-First-Prinzip*). Sie können mit folgendem Aufruf `jaxws-tools.[bat|sh] wsgen -version`

oder

```
jaxws-tools.[bat|sh] wsimport -version
```

einfach herausfinden welche JAX-WS-Version Sie einsetzen. Für Geronimo 2.1.4 ist das z.B. `2.0_01-b59-fcs` und für Geronimo 2.2 `JAX-WS RI 2.1.4-b01`.

Die aktuelle JAX-WS-Version `JAX-WS RI 2.1.6` die mit `JDK 1.6.0_17` ausgeliefert ist lässt sich mit dem Aufruf `wsimport -version` ausgeben.

## Anhang A.5 GShell

Seite 268:

```
jaxws
wsdl2java      Generate JAX-WS artifacts from WSDL
wsimport       Generate JAX-WS artifacts from WSDL
java2ws        Generate JAX-WS artifacts from class
wsgen          Generate JAX-WS artifacts from class
```

Abb. 106 Liste aller GShell-Befehle für Geronimo 2.1.4 ergänzen

```
Für Geronimo 2.2.0 kommen noch folgende GShell-Befehle hinzu
deploy/assemble Assemble a Geronimo-Server(CSA)
deploy/login Saves username and password for this connection
deploy/new-instance Create a new instance
geronimo/wait-for-server Wait for a Geronimo server to start
remote/rsh Connect to a remote GShell server
remote/rsh-server Start a remote GShell server
cluster/deploy Administer cluster, deploy a Plugin in a farm
cluster/heartbeat Monitor cluster heartbeat
jaxws/java2ws Generate JAX-WS artifacts from class
jaxws/wsgen Generate JAX-WS artifacts from class
jaxws/wsimport Generate JAX-WS artifacts from WSDL
jaxws/wsdl2java Generate JAX-WS artifacts from WSDL
```

Um einen Server auf Kommandozeile als Custom Server Assembly(CSA) zu assemblieren rufen Sie folgenden Befehl in der GShell auf.

```
deploy/assemble --help
--secure          Use secure channel
-a (--artifact) VAL server artifact name
-f (--format) VAL zip or tar.gz
```

-g (--groupId) VAL	server groupId
-h (--help)	Display this help message
-l (--list)	refresh plugin list
-m (--mode) VAL	custom assembly mode
-p (--port) N	Port, default 1099
-s (--hostname, --server) VAL	Hostname, default localhost
-t (--path) VAL	assembly location
-u (--username) VAL	Username
-v (--version) VAL	server version
-w (--password) VAL	Password

Standardmäßig wird der neu assemblierte Server (hier testServer) im Verzeichnis /var/temp/assembly abgelegt, wenn der Benutzer keine weiteren Informationen, wie Versionsnummer, groupId oder ArtifactId angibt.

Der vollständige Befehl lautet:

```
deploy/assemble -g org.apache.geronimo.assemblies -a testServer -v 1.0 -u system -w manager
```

Weitere Informationen finden Sie unter:

<http://cwiki.apache.org/confluence/display/GMOxDOC22/Assembling+a+server+via+comma+nd+line>

Ein weiterer GShell-Befehl kommt in Geronimo 2.2 für die Assembly-Erstellung mit geronimo-plugin-farm-node (<http://cwiki.apache.org/GMOxDOC22/plugin-based-farming.html>) hinzu. Mit cluster/deploy --help

können Sie Plugins auch remote hinzufügen, entfernen oder anzeigen mit folgenden Optionen:

ACTION	Action (add/remove) to perform
--secure	Use secure channel
-a (--pluginartifact) VAL	Plugin Artifact to perform action on
-c (--cluster) VAL	Cluster to perform action on
-f (--farm) farm name	
-h (--help)	Display this help message
-l (--pluginlist) VAL	Plugin List to perform action on
-p (--port) N	Port, default 1099
-s (--hostname, --server) VAL	Hostname,IP address, default localhost
-u (--username) VAL	Username
-w (--password) VAL	Password

**Beispiel :**

```
deploy/list-plugins --username system --password manager
org.apache.geronimo.samples/sample-datasource/2.2-SNAPSHOT/car
deploy/deploy bank-ear-2.2-SNAPSHOT.ear plan.xml
cluster/deploy add -f cluster1 -l pluginList1 -a org.apache.geronimo.samples/bank-
tomcat/2.2-SNAPSHOT/car
cluster/deploy remove -l pluginList1 -a org.apache.geronimo.samples/bank-tomcat/2.2-
SNAPSHOT/car
cluster/deploy remove -f cluster1 -l pluginList1
deploy/list-plugins -rr -r http://geronimo.apache.org/plugi
ns/geronimo-2.2/
```

## Anhang D Beispielanwendungen

Seite 283:

Die Anwendung <http://localhost:8080/Bank> ist auch ein Beispiel, wie zum Deploymentzeitpunkt im Datenbank-Ressource-Adapter die Datenbanktabellen mit Inhalten angelegt werden können.

Für die Geronimo Version 2.2 finden Sie die Plugins unter folgenden URLs, die beim Update der Repository-Liste in der Webadminkonsole auch automatisch hinzugefügt werden.

<http://geronimo.apache.org/plugins/geronimo-2.2/>

<http://geronimo.apache.org/plugins/samples-2.2/>

<http://geronimo.apache.org/plugins/openjpa2/>

## Kapitel H Literatur

Seite 307:

Apache Geronimo 2.1: Quick Reference, Manu T. George, Vamsavardhana Reddy  
Chillakuru, packtpub, November 2009

ActiveMQ in Action, Bruce Snyder, Dejan Bosanac, Rob Davies, Manning Juni 2010

---

<sup>i</sup> <https://issues.apache.org/jira/browse/GERONIMO-4504>