Tomcat als Unix-Service oder: die kleinen Hürden einer stabilen Tomcat-Unix-Dienstinstallation.

Tomcat als Hürdenläufer

von Peter Roßbach
Gastautor: Thorsten Kamann

➡ www.javamagazin.de/tomcat
➡ tomcat.objektpark.org

Ein Betriebssystem bringt schon von sich aus viele nützliche Funktionen mit, die für einen stabilen unterbrechungsfreien Betrieb der Server-Software sorgen. Unter Unix heißen diese Dienste Dämonprozesse oder Daemons und werden über Skripte an zentraler Stelle im System gesteuert. Sie entsprechen dabei der Funktionsweise des Windows-Dienstmanagers, der im *Java Magazin* 9.2004 beschrieben ist [1]. Diese Skripte stellen Funktionen zum Start, Stopp, Neustart und optional den Status eines Dienstes zur Verfügung. Sie sorgen außerdem beim Starten und Herunterfahren des Betriebssystems für die korrekte Behandlung der einzelnen Dienste. Eine weitere wichtige Eigenschaft ist die Reihenfolge, in der die einzelnen Dienste gestartet werden. Ein Webserver sollte erst gestartet werden, wenn das Netzwerk (TCP/IP) verfügbar ist. Diese Aufgaben muss das Dienstmanagement des Betriebssystems übernehmen. Hier kommen die verschiedenen Systemebenen (Run Level) zur Anwendung und eine feste Reihenfolge der Dienste für Unix zum Einsatz. Für den Administrator ist es sehr wichtig, jederzeit den Status eines Dienstes in Erfahrung bringen zu können. Dies wird bei den meisten Diensten mit dem Status-Befehl im entsprechenden Dienstskript realisiert. Die Anforderungen an eine Unix-Dienstintegration eines Tomcat sind allerdings etwas strenger:

- Integration ins Betriebssystem
- Überwachung der virtuellen Java-Maschine
- konfigurierbares Logging
- Umlenkung der Tomcat-Ausgabe auf eine Konsole
- Neustart auch über eine externe JMX-Konsole

Wenn man eine normale Tomcat-Installation analysiert, dann stellt man leider fest, dass keine der genannten Anforderungen erfüllt wird. Das wollen wir im Folgenden ändern. Der Tomcat liefert den Commons Daemon im Standardrelease mit. Dieser Daemon muss immer erst auf dem Unix-Zielsystem kompiliert werden. Die Konfiguration ist nicht immer intuitiv und der Tomcat nutzt keine freigegebene Version des Projekts [1]. Schon unter Windows hat sich der Java Service Wrapper aus der letzten Kolumne als die bessere Alternative herauskristallisiert [2]. Mit diesem Wrapper lassen sich mit einigen wenigen Schritten alle unsere Anforderungen an eine Systemintegration erfüllen.

Installation von JDK, Tomcat und Java Service Wrapper

Zu Beginn aller Aktivitäten steht die Entscheidung für ein Java 2 SDK. Für den Tomcat ist die Version 1.4.2 empfehlenswert. Für Unix/Linux können Sie zwischen den folgenden SDKs wählen:

- Sun Java 2 [3]
- Blackdown Java 2 (optimiert für Linux) [4]
- IBM Developer Kit [5]
- JRockit [6]

Typischerweise wird das SDK in das Verzeichnis */usr/lib* entpackt. Damit Sie nicht dauernd den oft kryptischen Verzeichnisnamen tippen müssen, erstellt Ihr Systemadministrator einen symbolischen Link auf das Standard-Java:

```
cd /usr/lib
ln -s VERZEICHNIS_DES_SDK java
```

Nach diesem Schritt ist Ihre Java-Installation mit dem Befehl */usr/lib/java/bin/java* erreichbar und auf den meisten Linux-Distributionen als Konvention schon im Systemprofil (*/etc/profile*) eingetragen. Mit dem Befehl *java -version* können Sie kontrollieren, ob alle Nutzer das gewählte Java im Ausführungspfad haben. Im nächsten Schritt wird der Tomcat installiert. Als In-

stallationsverzeichnis bietet sich das Verzeichnis */opt* oder */usr/local* an. Wir empfehlen immer das Verzeichnis */opt*. Laden Sie das Tomcat-Release herunter und entpacken Sie das Archiv in das Zielverzeichnis.

Nun müssen Sie noch den Java Service Wrapper installieren. Laden Sie sich das für Ihr Betriebssystem geeignete Paket unter der Adresse sourceforge.net/project/showfiles.php?group_id=39428 herunter [7]. Entpacken Sie das Archiv in ein temporäres Verzeichnis und kopieren Sie folgende Dateien in den Verzeichnisbaum Ihrer Tomcat-Version:

- *<wrapper>/bin/testwrapper.sh --> <tomcat>/bin/wrapper.sh*
- *<wrapper>/bin/wrapper --> <tomcat>/bin/wrapper*
- *<wrapper>/lib/libwrapper.so --> <tomcat>/bin/libwrapper.so*
- *<wrapper>/lib/wrapper.jar --> <tomcat>/bin/wrapper.jar*

Jetzt ist fast alles schon fertig, Sie müssen den Wrapper nur noch konfigurieren. Dieses machen Sie mit der Datei *tomcat/conf/wrapper.conf* (Listing 1), [1]. Ein Test der Installation erfolgt durch das Öffnen einer Shell (dem Unix-Gegenstück der Windows-Eingabeaufforderung) und einem Wechsel in das Verzeichnis *bin* der veränderten Tomcat-Installation. Dort führen Sie den Befehl *wrapper.sh console* aus. Falls hier Probleme auftauchen, dann können Sie das Loglevel *wrapper.console.loglevel=DEBUG* in der Datei *wrapper. conf* erhöhen.

In Listing 1 sehen Sie schon eine erweiterte Eigenschaft der Konfigurationsdateien des Wrappers:

```
set.CATALINA_HOME=/opt/tomcat/jakarta-tomcat-5.0.27
set.CATALINA_BASE=/opt/tomcat/jakarta-tomcat-5.0.27
set.JAVA_HOME=/usr/lib/java
```

Sie können beliebig viele Variablen definieren, die dann in der gesamten Konfiguration in der Form *%Variablen_Name%* genutzt werden können. Das Gleiche gilt natürlich für Umgebungsvariablen des Betriebssystems und Variablen, die im aufrufenden Shell-Skript definiert wurden.

Komplexe Konfigurationen können Sie aufteilen und mittels der *Include*-Anweisung in die Hauptkonfiguration (meist die Datei *wrapper.conf*) einbinden: *#include ../conf/wrapper_ext.conf*. Diese Zeile fügt die Datei *wrapper_ext.conf* ein. Relative Pfade haben als Basis das Verzeichnis der Wrapper Executable. Auch die zusätzlichen Konfigurationsdateien können andere Konfigurationen mit der *Include*-Anweisung integrieren. Das können Sie bis zu zehn Ebenen tief durchführen. Vor einer übertriebenen Nutzung dieser Möglichkeit können wir allerdings nur warnen: Übersichtlichkeit und Einfachheit sind oberstes Gebot für sinnvolle Konfigurationen. Der letzte Wert einer Variable oder Wrapper-Definition gilt.

Die vollständige Installation eines zugeschnittenen Tomcat 5.0.27 und Wrapper 3.1.1 finden Sie auf der beiliegenden Heft-CD. Dort brauchen Sie in der Datei *catalina-base-template/makeXXX.properties* nur die entsprechenden absoluten Pfade anpassen. Die von uns gestellten Anforderungen an eine Unix-Systemintegration lassen sich natürlich mit dem Wrapper und ein wenig Shell-Programmierung erfolgreich umsetzen.

Integration ins Betriebssystem

Leider bietet der Wrapper kein Shellskript zur einfachen Integration des Tomcat als Daemon. Das haben wir aber für Sie getan: Auf der Heft-CD finden Sie das Shellskript *tomcat_service.sh*, in dem Sie nur noch die Platzhalter *CATALINA_BASE* und *CATALINA_HOME* durch die entsprechenden absoluten Pfade ersetzen müssen. Dieses Skript bietet die Funktionen *start*, *stop*, *restart*, *status*, *install_service* und *uninstall_service*. Falls Sie eine der Linux-Distributionen SuSE, Red Hat oder Mandrake benutzen, können Sie die Konfiguration des Dienstes mit Abhängig-

Listing 1

Eine einfache Wrapper-Konfiguration für den Tomcat 5.0.x

```
set.CATALINA_HOME=/opt/tomcat/jakarta-tomcat-5.0.27
set.CATALINA_BASE=/opt/tomcat/jakarta-tomcat-5.0.27
set.JAVA_HOME=/usr/lib/java
wrapper.java.command=%JAVA_HOME%/bin/java
wrapper.java.mainclass=org.tanukisoftware.wrapper.WrapperStartStopApp
wrapper.java.classpath.1=%CATALINA_HOME%/bin/bootstrap.jar
wrapper.java.classpath.2=%CATALINA_HOME%/bin/wrapper.jar
wrapper.java.classpath.3=%JAVA_HOME%/lib/tools.jar
wrapper.java.library.path.1=%CATALINA_HOME%/bin
wrapper.java.additional.1=-Djava.endorsed.dirs=
    %CATALINA_HOME%/common/endorsed
wrapper.java.additional.2=-Dcatalina.home=%CATALINA_HOME%
wrapper.java.additional.3=-Dcatalina.base=%CATALINA_BASE%
wrapper.java.additional.5=-Dcatalina.config=
    file:%CATALINA_BASE%/conf/catalina.properties
wrapper.java.additional.6=-Djava.security.manager
wrapper.java.additional.7=-Djava.security.policy==%CATALINA_BASE%/
                                        conf/catalina.policy
wrapper.java.initmemory=64
wrapper.java.maxmemory=128
wrapper.app.parameter.1=org.apache.catalina.startup.Bootstrap
wrapper.app.parameter.2=1
wrapper.app.parameter.3=startd
wrapper.app.parameter.4=org.apache.catalina.startup.Bootstrap
wrapper.app.parameter.5=true
wrapper.app.parameter.6=1
wrapper.app.parameter.7=stopd
wrapper.console.format=PM
wrapper.console.loglevel=INFO
wrapper.logfile=%CATALINA_BASE%/logs/catalina.log
wrapper.logfile.format=LPTM
wrapper.logfile.loglevel=INFO
wrapper.logfile.maxsize=10m
wrapper.logfile.maxfiles=3
wrapper.syslog.loglevel=ERROR
wrapper.filter.trigger.1=java.lang.OutOfMemoryError
wrapper.filter.action.1=RESTART
```

keiten von anderen Diensten und dem Run Level direkt im Kopfbereich des Skripts realisieren. Die Debian-Distribution unterstützt diese komfortable Art leider nicht. Dort benutzen wir Standardwerte für die Installation als Systemwerte.

Überwachung der JVM

Die Überwachung der JVM leistet der Java Service Wrapper. Sobald die JVM über einen gewissen Zeitraum nicht auf einen Ping antwortet, wird diese neu gestartet. Dadurch haben Sie eine sehr hohe Ausfallsicherheit. Die Konfiguration dieses Verhaltens ist mit der aktuellen Version 3.1.1 nochmals erweitert worden.

Bei hoch belasteten Systemen kommt es häufiger vor, dass die JVM nicht rechtzeitig auf den Ping des Wrappers antwortet. Passiert das öfters hintereinander, geht der Wrapper davon aus, dass die JVM nicht mehr korrekt arbeitet, und erzwingt ihren Neustart. Um dieses Verhalten zu umgehen, können Sie die Ping-Intervalle und den Timeout erhöhen:

```
wrapper.ping.interval=5
wrapper.ping.timeout=30
```

Mit diesen beiden Einstellungen bestimmen Sie das Intervall und den Timeout. Beide Angaben sind in Sekunden anzugeben. Der Wert für den Timeout muss um mindestens fünf höher als der Wert für das Intervall sein. Wenn Sie den Wert für den Time-out auf Null setzen, ist diese Funktionalität deaktiviert. Zusätzlich zur Überwachung der JVM können Sie auch die CPU überwachen: *wrapper.cpu.timeout=10*. Sobald die CPU nicht innerhalb von zehn Sekunden reagiert, geht der Wrapper von einem Problem aus.

Konfigurierbares Logging

Das Logging ist bereits in unserer Konfiguration *wrapper.conf* enthalten (Listing 1). Details zu den Logging-Eigenschaften des Wrappers finden Sie in der Dokumentation und in unserer letzten Kolumne [8], [1]. Wenn Sie mit einem Service Wrapper arbeiten, sollten Sie sämtliche Logger-Elemente bis auf die der Hosts und Kontexte in der Datei *conf/server.xml* des Tomcat weglassen. Damit erreichen Sie, dass nur eine zentrale Ausgabedatei für die Überwachung des Servers existiert.

Umlenkung der Tomcat-Ausgabe auf eine Konsole

Dies erreichen Sie, indem Sie den Wrapper im Konsolenmodus starten: *bin/wrapper.sh console*. Dafür müssen Sie allerdings den Tomcat neu starten. Besser ist es, die Datei *catalina.log* zu überwachen. Mit dem Unix-Befehl *tail* geht eine manuelle Überwachung recht einfach von der Hand: *tail -f logs/catalina.log*.

Neustart auch über eine externe JMX-Konsole

Seit der Version 3.1.0 ist der Wrapper auch durch ein MBean steuerbar. Auf der Heft-CD haben wir eine Tomcat-*WrapperLifecycleListener*-Klasse für diese Integration realisiert. Mit der folgenden Konfiguration aktivieren Sie den Listener und können dann mit jeder JMX-Konsole den Wrapper kontrollieren (Abb. 1).

```
<Server port="7405" shutdown="SHUTDOWN" debug="0">
  <Listener className="org.apache.catalina.mbeans.Server
                                        LifecycleListener"/>
  <Listener
className="org.objektpark.catalina.wrapper.Wrapper
                                        LifecycleListener"/>
...
</Server>
```

Weitere Fragen zur Systemintegration

Eine zentrale Frage stellt sich beim Unix-Betrieb mit Tomcat immer wieder: Wie vermeidet man, dass der Tomcat unter dem Root-Konto ausgeführt wird? Im Skript *tomcat-service.sh* können Sie eine/n beliebige/n Benutzer und Gruppe angeben, unter dem/r der Tomcat ausgeführt wird. Wir schlagen für diese Aufgabe den Benutzer *nobody* und die Gruppe *nobody* vor, da dieser Benutzer keine weiteren Rechte im System besitzt. In diesem Fall kann es aber zu Problemen mit den Dateirechten im Tomcat-Verzeichnis kommen. Aus diesem Grund führt das Skript *tomcat-service.sh* die entsprechenden Zuordnungen für Rechte der Eigentümer und Gruppen für Sie direkt aus.

Wie kann der Tomcat über Port 80 erreicht werden?

Wenn der Nutzer auf einen nicht privilegierten Port wechselt, kann der (privilegierte) Port 80 auf Unix wegen fehlender Rechte nicht mehr genutzt werden. Für diese Problemstellung gibt es zwei Lösungen:

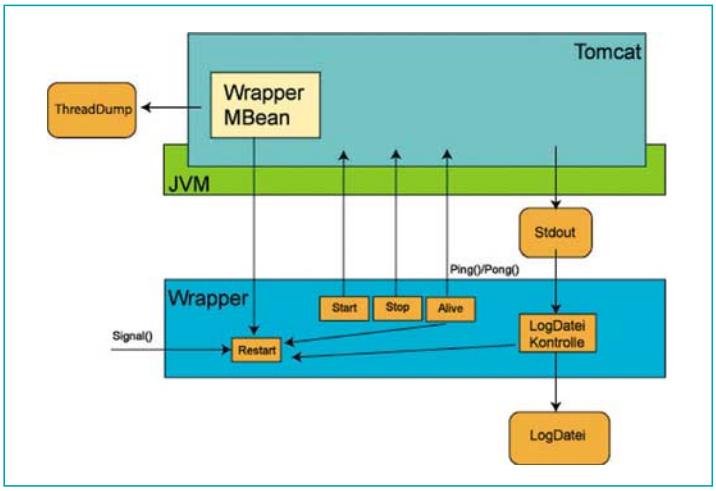

Abb. 1: Wrapper-Architektur

- Ein Apache-Webserver wird vorgeschaltet, der die Anfragen auf Port 80 annimmt und an den Tomcat weiterleitet (*mod_jk*, *mod_jk2* oder *mod_proxy*) – eine Kombination, die häufig genutzt wird [8], [9].
- Mittels eines entsprechenden *iptables*-Eintrags kann der Port auch intern umgeleitet werden. Diese Lösung bietet sich an, wenn kein Webserver genutzt werden soll [10]:

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -i
eth0 -j REDIRECT --to-port 7380
```

Und schließlich: Was passiert bei *java.lang.OutOfMemoryError*-Ausnahmen? Solche Fehler können vom Wrapper gefiltert und ein Neustart des Tomcat eingeleitet werden. Dieses Verhalten ist schon in unserer Datei *wrapper.conf* auf der Heft-CD entsprechend aktiviert.

Fazit

Wie Sie sehen, sind alle Wünsche an eine Systemintegration mit dem Java Service Wrapper recht einfach zu erfüllen. Allerdings ist dies leider auch mit einiger zeitraubender Arbeit verbunden. Auf der Heft-CD finden Sie natürlich ein Beispiel, das Sie mit wenigen Handgriffen testen können.

Wenn man diese Konfiguration des Wrappers beherrscht – die Lernkurve ist erfreulich flach –, kann man schon mit wenig Aufwand eine große Wirkung erzielen. Viele Eigenschaften, die man üblicherweise mit Shellskripten programmiert hätte, sind mit dem Wrapper bereits realisiert. Falls aber doch Probleme oder Fragen zum Wrapper auftauchen, gibt es in der sehr kompetenten Mailing-Liste schnell Hilfe [11].

Wir freuen uns auf Kommentare und Anregungen – besuchen Sie also meine TomC@ Site [13] und das TomC@-Forum [14] oder die Tomcat-Mailing-Listen [15]. ■

Peter Roßbach (pr@objektpark.de) ist als freier J2EE-Systemarchitekt, Entwickler und Trainer tätig. Peter Roßbach ist seit kurzem Tomcat-Committer.

Thorsten Kamann (thorsten.kamann@planetes.de) ist als Softwareentwickler und Administrator für Win32- und Linux-Systeme tätig.

Tomcat-News

- Das Release Tomcat 5.0.27 ist freigegeben
- *mod_jk 1.2.6* ist freigegeben
- Tomcat 5.next
  - Ein vollständig neuer Deployer ist realisiert.
    - Ein *META-INF/context.xml* in Verzeichnissen findet endlich Berücksichtigung
    - Der DefaultContext wird durch eine *conf/context.xml* und *conf/<engine>/<host>/context.xml.default* ersetzt
    - Es existieren nun eine Datei *conf/web.xml* sowie *conf/<engine>/<host>/web.xml.default* als Voreinstellung für den Web Deployment Descriptor
    - Autoreloading der Webanwendung kann abhängig von Veränderungen an beliebigen Dateien durchgeführt werden
  - Die zentrale Tomcat-Konfiguration *server.xml* wird vereinfacht
- Die Neuentwicklung des AJP Adaptor auf Basis des Apache 2-Moduls *Httpd Proxy* wurde begonnen

Links & Literatur

[1] Peter Roßbach, Thorsten Kamann: Der feine Unterschied. Tomcat als Serviceunternehmen: erfolgreiche Dienstleistung unter Windows, in *Java Magazin* 9.2004,

[2] wrapper.sourceforge.net/

[3] java.sun.com/j2se/1.4.2/download.html

[4] blackdown.org/java-linux/java-linux-d1.html

[5] www-106.ibm.com/developerworks/java/jdk/linux140/

[6] dev2dev.bea.com/products/wljrockit142/, edocs.bea.com/wljrockit/docs81/

[7] sourceforge.net/project/showfiles.php?group_id=39428

[8] Peter Roßbach (Hrsg.); Andreas Holubek, Thomas Pöschmann, Lars Röwekamp, Peter Tabatt: Tomcat 4X: Die neue Architektur und moderne Konzepte für Webanwendungen im Detail; Software & Support Verlag, 2002

[9] Peter Roßbach: Vom Mythos einer einfachen Integration. Tomcat mit dem Apache-Server nutzen, in *Java Magazin* 8.2004

[10] www.klawitter.de/tomcat80.html, linux.org.mt/article/tomcat-ports

[11] sourceforge.net/mail/?group_id=39428

[12] Peter Roßbach, Lars Röwekamp, Thorsten Kamann: Mein Tomcat ist eine Festung … Tomcat als Trutzburg, in *Java Magazin* 4.2004

[13] tomcat.objektpark.org/

[14] www.javamagazin.de/tomcat/

[15] www.mail-archive.com/tomcat-user@jakarta.apache. org/, www.mail-archive.com/tomcat-dev@jakarta.apache.org/

[16] jakarta.apache.org/tomcat/

[17] jakarta.apache.org/commons/daemon/

[18] wrapper.tanukisoftware.org/doc/english/properties.html

[19] centaurus.sourceforge.net/

[20] Peter Roßbach, Lars Röwekamp: Die Saat liegt im Feld. Catalina Base: eine praktische Anleitung zur flexieblen Tomcat-Konfiguration, in *Java Magazin* 6.2004