Catalina Base: eine praktische Anleitung zur flexiblen Tomcat-Konfiguration

Die Saat liegt im Feld

Die Tomcat-Konfiguration ist mittlerweile so flexibel geworden, dass nur noch alte Hasen in der Lage sind, alle Einstellungsmöglichkeiten auf Anhieb zu überblicken. Dies führt in der Praxis leider auch im produktiven Umfeld häufig – wider besseren Wissens – zur Verwendung der Tomcat-Standardinstallation inklusive der Standard-Projektkonfiguration. Dabei steht bereits seit den frühen Tagen des Tomcat 4-Servers mit der Trennung von der Projektkonfiguration auf der einen und dem Server-Release auf der anderen Seite ein mächtiges Werkzeug zur Verfügung, welches das Hinterlegen mehrerer getrennter Konfigurationen für unterschiedliche Projekte erlaubt. Grund genug für uns, dieses Konzept einmal im Rahmen der TomC@-Kolumne mitsamt einer eigenen, praktischen Erweiterung vorzustellen.

von Peter Roßbach & Lars Röwekamp

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

Wer kennt nicht den Wunsch, mal eben schnell eine Tomcat-Konfiguration zu erstellen, um so die eine oder andere neue Einstellung und deren Auswirkungen für ein Projekt testen zu können. Natürlich sollen diese Änderungen nicht zu Lasten anderer Projekte gehen. In der Praxis sieht die Lösung dann häufig so aus, dass ein neuer Tomcat-Server parallel zu dem bzw. den bisherigen Tomcat-Servern installiert wird und die gewünschten Einstellungen an der im Standardumfang des Tomcat enthaltenen Konfigurationsdatei *$CATALINA.HOME/conf/server.xml* vorgenommen werden. Bei diesem Vorgehen wird in der Regel die Tatsache vernachlässigt, dass es sich bei der Standard-*server.xml* lediglich um ein Beispiel für das Server-Release handelt, nicht aber um eine optimierte Ausgangsbasis für die Konfiguration eines produktiven Systems. Auf vielen derart konfigurierten Servern kann man daher neben den eigentlich angedachten Anwendungen zusätzlich die Tomcat-Dokumentation oder gar die Tomcat-Beispiele erreichen. Ein solches Szenario ist eigentlich immer ein Zeichen dafür, dass es an der Zeit ist, in die Hände zu spucken und über eine bessere Alternative nachzudenken.

Genau solch eine Alternative möchten wir Ihnen im Verlauf dieser Kolumne vorstellen. Dabei werden wir ausgehend von einer normalen Tomcat-Installation zeigen, wie Sie Schritt für Schritt eine Unabhängigkeit von Server-Release und Projektkonfiguration erreichen. Aus dieser Eigenschaft heraus erwachsen starke Konzepte für die Unabhängigkeit und Lebendigkeit ihrer Tomcat-Server. Wäre es nicht wunderbar, mit einem Skript einfach die nächste Konfiguration für ein Experiment oder das nächste Projekt generieren zu können? Diese Kolumne beschreibt einen Weg und natürlich ein Beispiel, Ihre Tomcat-Konfigurationen zu säen und anschließend unabhängig voneinander wachsen zu lassen.

„Wer im Frühjahr nicht reichhaltig sät, kontinuierlich düngt und pflegt, wird keine reichhaltige Ernte einfahren." So oder ähnlich beginnen viele Bauernsprüche und dies erschien auch in uns im Laufe der diversen TomC@-Kolumnen als eine wesentliche Erkenntnis zu reifen. Schon in der Planung der ersten Kolumne im Juni 2003 hatten wir beschlossen, dass jeder Kolumne ein Beispiel hinzugefügt werden sollte [1]. Und wie es in Ihrem und unserem Projektalltag schnell passiert, haben wir das berühmt berüchtigte „Copy & Paste"-Antipattern, wider besseren Wissens, jedes Mal wieder erfolgreich angewendet. Mittlerweile gibt es genug Beispiele, um über eine bessere Strategie nachzudenken. Wir haben uns daher ein Saat-Muster für eine Tomcat-Konfiguration geschaffen. Bevor wir allerdings mit dem Säen und Ernten beginnen, d.h. das Saat-Muster im Detail erläutern, werfen wir zuvor gemeinsam einen Blick auf die Ausgangsbasis eines Tomcat-Release.

Tomcat Bootstraping

Vor der Nutzung eines Tomcat-Release steht immer das Herunterladen des aktuellen Tomcat-Release. Hier beginnt natürlich schon die Vorentscheidung. Aktuell haben Sie die Wahl zwischen drei verschiedenen Hauptversionen. Die Tomcat 3.3.x- und 4.1.x-Releases sorgen nach wie vor dafür, dass die riesige, vorhandene Nutzerbasis mit aktuellen Versionen für Ihre Produktions- und Kundenservern versorgt wird. Seit etwa einem Jahr ist der aktuelle Entwicklungsstrang für die Unterstützung des Tomcat 5 hinzugekommen. Das modernste und leistungsfähigste Release des Tomcat, das wir jemals gesehen

haben, ist den Kinderschuhen seit diesem Frühjahr entwachsen. Das Redesign der Catalina 4-Architektur und die Erfüllung des Specification Servlet API 2.4 und der JSP 2.0 sind umgesetzt [2], [3], [4]. Die letzten Monate haben im produktiven Einsatz der ersten Sites die Leistungsfähigkeit bewiesen und jede Menge brisanter Probleme sind abgestellt worden. Und auch wir sind mit der TomC@-Kolumne seit Beginn des Jahres auf den Tomcat 5 umgestiegen.

Nun, damit ist es natürlich nicht verwunderlich, dass wir das Tomcat 5.0.19-Release als Basis für unsere Installation gewählt haben [5]. Der Download erfolgt über die Jakarta Site jakarta.apache.org/site/binindex.cgi. Hier wählen Sie den geeigneten Spiegelserver und laden die knapp zehn Megabyte der Binary-Distribution herunter. Voraussetzung für den Einsatz des Tomcat 5 ist mindestens ein JDK 1.3.1, obwohl ein aktuelles JDK 1.4.2 oder 1.5 sicherlich die bessere Wahl ist [6]. Das Archiv *jakarta-tomcat-5.0.xx.[tar.gz|zip]* kann nun an einer beliebigen Stelle im Dateisystem extrahiert werden (Abb. 1).

Im Wesentlichen ist ein Webserver eben auch nur ein normales Java-Programm, d.h., es gibt eine Klasse mit der klassischen *main()*-Methode, die den Start des Servers veranlasst. Im *bin/*-Verzeichnis befinden sich dazu die notwendigen Skripte und Bootstrap-Klassen. Das zentrale *catalina.[sh|bat]*-Skript steuert den Start mit den richtigen Klassen und Optionen für die Java Virtual Machine (JVM) (Tabelle 1 und 2). Die Skripte *startup* und *shutdown* dienen zur Vereinfachung der Steuerung des *catalina*-Skripts. Für die eigene Erweiterung der Umgebung kann das *setenv*-Skript angelegt werden. Der Boot Classpath wird im Skript *setclasspath* zusammengebaut. Alle Skripte stehen in einer Ausprägung für Windows (*.bat*) und Unix, Mac OS X bzw. cygwin (**.sh*) bereit. Neben den Bootstrap-Klassen im *bin/*-Verzeichnis sind im Release im *server/lib/*-Verzeichnis die Bibliotheken und Implementierung des konkreten Tomcat-Servers zu finden. Mit den Webanwendungen teilt sich der Server die Bibliotheken in den *common/lib/*- und *common/endorsed/*-Verzeichnissen. Dort finden zum Beispiel JDBC-Treiber meist zusätzlich ihren Platz. Im Standard-Release befinden sich im *common/lib/*-Verzeichnis die JAR-Archive für Servlets, JSP, Datenbank-Pool DBCP, JNDI und Ant. Im *common/endorsed/*-Verzeichnis ist der aktuelle Apache Java Xerces verfügbar. Eine wichtige Neuerung des Tomcat 5 ist, dass mit der *conf/catalina.properties*-Datei die verschiedenen ClassLoader nun endlich frei definierbar sind. Im *conf/*-Verzeichnis befinden sich die notwendigen Beispielkonfigurationen für einen Test- bzw. Release-Server. Die Anwendungen dieses Tomcat-Testservers befinden sich in den *server/webapps/*- und *webapps/*-Verzeichnissen. Dort befinden sich Anwendungen zur Administration, Management, JSP- und Servlet-Beispiele und die Dokumentation mit Status, Referenz-Guide, kurzem Webanwendungs-Tutorial und Howtos. Die Konfiguration des Servers ist in der *conf/server.xml*-Datei zu finden. Im *conf/Catalina/localhost/*-Verzeichnis befindet sich die Context-Definition für die Webanwendungen Admin und Manager. Für diese Anwendungen müssen weiterhin Nutzer mit den Rollen *admin* und *manager* in *conf/tomcat-users.xml* manuell nachgetragen werden, bevor Sie einen Zugriff darauf bekommen [5], [4]. An dieser Stelle ist im Tomcat 5 auch der Platz für eigene Context-Definitionen, also die spezifischen Konfigurationen einer Webanwendung. Im Tomcat 4 befinden sich diese Dateien noch im *webapps/*-Verzeichnis des jeweiligen virtuellen Hosts.

Abb. 1: Tomcat-Release-Verzeichnis (CATALINA.HOME)

```
jakarta-tomcat-5.0.xx
├──bin
│   ├──catalina.[sh|bat]
│   ├──startup.[sh|bat]
│   ├──shutdown.[sh|bat]
│   ├──setclasspath.[sh|bat]
│   ├──tomcat.exe
│   ├──jsvc.tar.gz
│   ├──bootstrap.jar
│   ├──commons-logging-api.jar
│   ├──commons-daemon.jar
│   └──jmx.jar (ab 5.0.20)
├──common
│   ├──endorsed
│   └──lib
├──conf
│   ├──Catalina
│   │   └──localhost
│   │       ├──admin.xml
│   │       └──manager.xml
│   ├──catalina.properties
│   ├──catalina.policy
│   ├──server.xml
│   ├──web.xml
│   └──tomcat-users.xml
├──server
│   ├──lib
│   └──webapps
│       ├──admin
│       └──manager
└──webapps
    ├──balancer
    ├──jsp-examples
    ├──ROOT
    ├──servlets-examples
    ├──tomcat-docs
    │   ├──appdev
    │   │   └──sample
    │   │       ├──docs
    │   │       ├──src
    │   │       └──web
    │   ├──architecture
    │   │   ├──requestProcess
    │   │   └──startup
    │   ├──catalina
    │   │   ├──docs
    │   │   │   └──api
    │   │   └──funcspecs
    │   ├──config
    │   ├──jasper
    │   ├──jspapi
    │   ├──servletapi
    │   └──webdav
```

webdev-server Termplate

Für eine kleine Demonstration, Evaluation oder Einarbeitung im Tomcat reicht es völlig, direkt in einem Release zu arbeiten. Einfach starten und mit dem Browser unter der URL http://localhost:8080/ die Dokumentation und Beispiele anschauen. Eigene Anwendungen können einfach in das vorhandene *webapps/*-Verzeichnis kopiert werden und stehen sofort bereit. Für spezielle Konfigurationen wie eigene AccessLogger, Datenbank- und Mail-Ressourcen, Realms oder Connectors sind die entsprechenden Beispiele direkt in der zentralen *conf/server.xml* aus zu kommentieren. Jede Änderung der Konfiguration zieht allerdings einen Neustart des Tomcat-Servers nach sich. Selbst Eingriffe mit der Admin-Webanwendung oder direkt via JMX können diesen Neustart nicht immer verhindern [7].

Nach einer Weile möchten Sie die Konfigurationen mit Ihren Kollegen teilen oder diese in Produktion nehmen. Einmal schnell kopiert und schon ist auch diese Aufgabe bewältigt, oder etwa nicht? Die Wartung bei mehreren Maschinen mit unterschiedlichen Einstellungen und sicherlich notwendigen Release-Wechseln ist mit dieser *Kopiere/Einfüge/Variiere*-Technik schnell nur noch von „wahrhaftigen" Experten beherrschbar. Diese Strategie wird somit schnell zu einem Hemmschuh im Projekt und macht die Bereitstellung eines „Hot Bug Fixes" zu einem unüberschaubaren Risiko.

Der Tomcat enthält für eben solche Situationen ein leitungsfähiges Konzept. Die Konfigurationen und Erweiterungen eines Server-Release (*CATALINA.HOME*) kön-

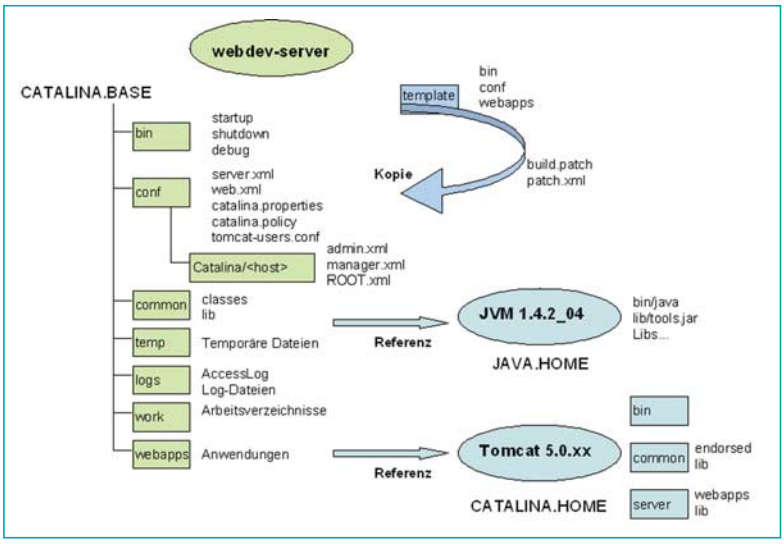

Abb. 2: Zusammenhang zwischen Tomcat-Release und *webdev-server*

nen in einer getrennten Verzeichnisstruktur *CATALINA.BASE* erstellt und hinterlegt werden (Abb. 2 und 3). Somit lassen sich die eigenen Einstellungen vollkommen separat anpassen, während die Klassen und Skripte des Release lediglich referenziert werden. Vorteil eines solchen Vorgehens: Das Tomcat-Release bleibt unverändert bestehen und kann jederzeit ausgetauscht oder aktualisiert werden.

Um Ihnen die Arbeit bei dem Aufsetzen der oben beschriebenen Trennung zu erleichtern, haben wir ein Template namens *webdevPLUS* für eine solche Konfiguration entwickelt, welches sich auf der Heft-CD befindet. Die in *webdevPLUS* enthaltene *CATALINA.BASE*-Struktur hat folgende Eigenschaften:

- Referenzierung auf ein beliebiges Tomcat 5.0.xx-Release (*CATALINA.HOME*)
- Anpassung bestimmter Eigenschaften auf Basis eigener Templates
- Bereitstellung von *startup*-, „*restart*-, *debug*- und *shutdown*-Skripten
- Ausgleich der Differenzen von Betriebsystemen durch Steuerung der Skripte und Patches mit Ant
- eigene Policy-Datei für den Start mit dem Java Security Manager [8]
- eigene *catalina.properties*-Datei für die Zusammenstellung der Klassenpfade
- getestet unter Windows XP, Suse Linux 8.x und 9.0 und Mac OS X
- Das Template von *webdev-server* wendet sich in erster Linie an Entwickler
- Installation von *webdev-server* in einem Saat-Verzeichnis

Befehle	Erklärung
run	Start Catalina im aktuellen Fenster
run –security	Start Catalina im aktuellen Fenster mit einem Security Manager
start	Start Catalina in einem neuen Fenster
start –security	Start Catalina in einem neuen Fenster mit einem Security Manager
stop	Stopp Catalina
jpda start	Start Catalina für einen Remote Debugger via JPDA
debug	Start Catalina in einem Debugger
debug –security	Start Catalina in einem Debugger mit einem Security Manager

Tabelle1: Startbefehle des *catalina.sh*-Skripts

Optionen	Erklärung
JAVA_HOME	Zeigt auf die JVM
CATALINA_HOME	Zeigt auf ein Tomcat-Releaseverzeichnis
CATALINA_BASE	(Optional) Basisverzeichnis für die Konfiguration, falls nicht vorhanden, wird *$CATALINA_HOME* genutzt
CATALINA_OPTS	(Optional) Java-Runtime-Optionen für die Befehle start, stop …
CATALINA_TMPDIR	(Optional) Temporäres Verzeichnis der JVM (*java.io.tmpdir*). Die Voreinstellung ist *$CATALINA_BASE/temp*
JSSE_HOME	(Optional) Zeigt auf die JSSE-Installation (nur JDK 1.3.x)
JAVA_OTPS	(Optional) Java-Runtime-Optionen
JPDA_TRANSPORT	(Optional) JDPA-Transport (*dt_shmem* oder *dt_socket*)
JPDA_ADDRESS	(Optional) JDPA-Adresse für den *dt_socket*-Transport

Tabelle 2: Umgebungvariablen des *catalina.sh*-Skripts

Für die Anwendung des Template muss mindestens ein Apache Ant 1.5.4 installiert und in der jeweiligen Shell erreichbar sein [9]. Also nicht vergessen, nach Download und Installation das *$ANT_HOME/bin*-Verzeichnis in den Systempfad einzutragen. Den detaillierten Inhalt von *webdev-PLUS* hier zu beschreiben wäre etwas zu aufwändig. Wir beschränken uns daher erst einmal auf die Darstellung der wesentlichen Sachverhalte. Eine aktuelle Dokumentation befindet sich im *webdevPLUS*-Release und kann jederzeit aktuell von unser TomC@-Seite unter tomcat.objektpark.org/ abgerufen werden.

Nach dem Auspacken der *TomC@-kolumne-webdevplus-1.1.x*-Archive finden Sie neben der Dokumentation und einigen Skripten das *webdev-server*-Verzeichnis (Abb. 3) In diesem Ordner befindet sich unsere *CATALINA.BASE*-Konfiguration für die Nutzung eines Tomcat 5-Servers. Die Anpassung auf die jeweilige Installation auf Ihrem Rechner wird durch die Pflege der *build.patch*-Datei vorgenommen (Listing 1). Im Wesentlichen müssen dort die Pfade zum JDK (*JAVA_HOME*) und zum Tomcat (*CATALINA_HOME*) eingetragen werden. Mit dem Befehl *ant patch* wird die Anpassung mit den Templates aus dem *webdev-server/template/*-Verzeichnis übertragen. Die wirkliche Be-

schreibung der Anpassungen befindet sich im *webdev-server/patch.xml*-Ant-Skript. Hier werden unter anderem auch die Dateirechte für Unix-Systeme gesetzt und die Ordner *logs*, *work* und *temp* angelegt.

Als Resultat entsteht ein vollständiges *CATALINA.BASE*, das mit den Skripten im *bin/*-Verzeichnis oder dem *build.xml*-Ant-Skript gesteuert werden kann (Abb. 2 und 3). Ein plattformunabhängiger Start erfolgt also mit *ant tomstart* oder *ant tomsecure* mit Unterstützung des Security Manager [8]. Die projektspezifische Konfigurationsdatei *conf/server.xml* ist eine minimale Tomcat-Konfiguration für den Entwickler. Als Erweiterung des Release befindet sich log4j 1.2.8 im *common/*-Verzeichnis. Mit der *common/classes/log4j.properties*-Datei lässt sich die Ausgabe der Serverklassen aussteuern. Der Zugriff auf die Administration und Managementanwendungen sind mit dem Nutzer *manager/tomcat* direkt möglich. Eine eigene *conf/web.xml* und *conf/server.xml* lässt nun zum Beispiel kontrolliert Experimente des Tunings der Japser JSP Engine oder des gesamten Servers zu. Der *webdevPLUS*-Tomcat-Server ist direkt im Browser mit http://localhost:7380 erreichbar und steht damit für Ihre spezifischen Konfigurationen bereit, und zwar ohne dass Sie Ihr eigentliches Tomcat-Release verändern müssen (Abb. 4). Das Template können Sie direkt zum Bestandteil Ihres Projekts machen und so zum Beispiel in die Quellecode-Verwaltung als *webdev-server*-Modul aufnehmen. Die verschiedenen Anpassungen Ihres Projektes oder Ihrer Zielumgebungen können durch verschiedene Patch-Dateien (*buildLINUX.patch*) und das *patch.xml*-Skript gepflegt werden. Das Patchen nutzt die Filtermöglichkeiten des Ant Copy Task.

In jedem Projekt gibt es Fehler, die nicht durch einen Test abgedeckt oder einfach aufgezeigt werden können. Mit dem *webdev-server/bin/debug*-Skript kann der Tomcat mit JDPA Socket 8000 gestartet werden. Einer Remote-Kontrolle mit Eclipse, JBuilder oder IDEA steht dann nichts mehr im Wege (Listing 2) [4].

Die Tomcat-webdev-server-Saat

Noch mehr Flexibilität, als im bisherigen Verlauf der Kolumne gezeigt, ist in den meisten Fällen eigentlich nicht mehr unbedingt notwendig. Der wesentliche Erfolg, nämlich die Konfiguration vom eigentlichen Release zu trennen, ist mit dem *webdev-server*-Template erzielt. Aber eine noch nicht genutzte Eigenschaft des Servers zu probieren könnte noch etwas leichter von der Hand gehen. Schnell hat man wie wir fünfzig und mehr Projekte auf seinem Rechner und muss für die neue Aufgabe ein passendes Template erst einmal finden und zusammenstellen. Die Rüstzeit für ein kleines Experiment oder das Beispiel für die nächste TomC@-Kolumne kann so schon einmal ein paar Stunden in Anspruch nehmen. Warum also nicht einfach ein Skript entwickeln, welches auf Basis eines Template der beschriebenen *webdev-server* in einem beliebigen Verzeichnis erstellt? Für verschiedene Betriebssysteme ist dieses Vorhaben zwar eine nicht zu unterschätzende Herausforderung, aber einen Lösungsansatz dafür kann man bereits im Apache XML For-

Abb. 3: *webdev-server*-Verzeichnis (*CATALINA.BASE*)

```
webdev-server
├build.properties
├build.patch
├buildLINUX.patch
├build.xml
├patch.xml
├──bin
├──common (Optional)
│  ├──classes
│  └──lib
├──conf
│  ├──Catalina
│  │  └──localhost
│  │     ├....xml
│  │     ├manager.xml
│  │     └admin.xml
│  ├──server.xml
│  ├──web.xml
│  └──tomcat-userx.xml
├──logs
├──shared (Optional)
│  ├──classes
│  └──lib
├──temp
├──template
│  ├──bin
│  ├──conf
│  └──webapps
├──webapps
│  └──ROOT
│     ├──images
│     ├──tiles
│     └──WEB-INF
│        ├──lib
│        └──tlds
└──work
```

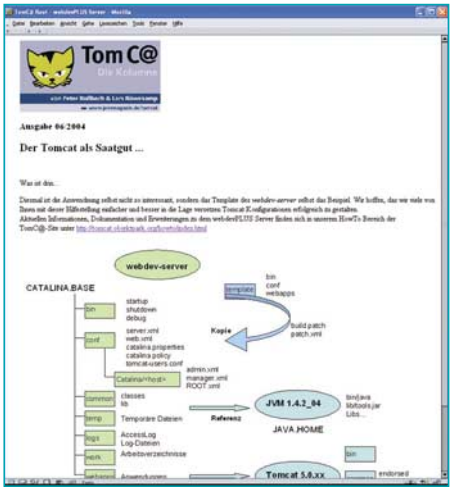

Abb. 4: Root-Anwendung des *webdevPLUS*-Tomcat-Servers

Listing 1

Patch-Werte des *webdev-server* in der Datei *build.patch*

```
java.home=_PATH_TO_JAVA_HOME_
catalina.home=_PATH_TO_CATALINA_HOME_
server.shutdown.port=7305
server.shutdown.key=7305
server.debug=1
connector.http.port=7380
connector.ajp.port=7309
connector.https.port=7443
engine.host=localhost
jpda.address=8000
jvm.minmemory=8
jvm.maxmemory=32
```

Listing 2

Tomcat-Debug-Startskript von *webdevPLUS*

```
@echo off
set _JAVA_HOME=%JAVA_HOME%
set _JSSE_HOME=%JSSE_HOME%

set JAVA_HOME=@java.home@

set CATALINA_BASE=..
set _CATALINA_HOME=%CATALINA_HOME%
set CATALINA_HOME=@catalina.home@
set CATALINA_OPTS=
      -ms@jvm.minmemory@m -mx@jvm.maxmemory@\
        -Dcatalina.config=file:../conf/catalina.properties
set JPDA_TRANSPORT=dt_socket
set JPDA_ADDRESS=@jpda.address@
set CATALINA_OPTS=%CATALINA_OPTS% -Xdebug \
  -Xrunjdwp:transport=%JPDA_TRANSPORT%,\
  address=%JPDA_ADDRESS%,server=y,suspend=n
echo CATALINA_OPTS: %CATALINA_OPTS%
%CATALINA_HOME%\bin\
                catalina run %1 %2 %3 %4 %5 %6 %7 %8 %9

set CATALINA_HOME=%_CATALINA_HOME%
set _CATALINA_HOME=

set JAVA_HOME=%_JAVA_HOME%
set JSSE_HOME=%_JSSE_HOME%
```

rest-Projekt finden [10]. Der notwendige Mechanismus der Saat (seed) einer *webdev-server* ist im *webdevPLUS*-Release schon enthalten. Die vorhandene *webdev-server* wird in einem zentralen Verzeichnis, dem *SEED.HOME,* installiert. Hinzu kommen Skripte für die unterschiedlichen Systeme Windows oder Unix. In der *build.patch*-Datei wird der Eintrag *seed.home* auf Ihr Saatgut-Verzeichnis gesetzt und dann folgendermaßen installiert

```
edit build.patch
cd seed
ant –f seedinstall.xml
export PATH=$seed.home/bin:$PATH
```

Nun ist es möglich, in einem beliebigen Verzeichnis eine *webdev-server* zu erzeugen. Mit dem Befehl *seedtom* geschieht, dass eine Kopie der *webdev-server* erstellt wird und darin direkt das Ant-Skript *patch.xml* abläuft. Somit ist dieser Tomcat-Server sofort einsetzbar. Mit wenigen Änderungen des *seedtom*-Skripts lassen sich im *SEED.HOME*-Verzeichnis eine Reihe Standardkonfigurationen oder weitere Seeds für Webanwendungen, -module oder gesamte Projektstrukturen hinterlegen. Wer gerne eine reichhaltige Ernte haben möchte, braucht halt bei Zeiten gutes Saatgut.

Welche Erweiterungen wünschen Sie noch?

Uns hat der beschriebene *webdevPLUS*-Ansatz schon jede Menge Zeit gespart. Er ist einfach zu bedienen und zu erweitern. Gerade Entwickler, die häufig und gerne neue Releases ausprobieren oder viele Webanwendungen erstellen, können endlich flexibel und schnell eine Serverkonfiguration aufbauen. Ein Wizard ist mit Ant schnell geschaffen. Von unserem aktuellen Stand von *webdevPLUS* können Sie natürlich auch nur *webdev-server* nutzen oder eine eigene *CATALINA.BASE* ableiten. Wir werden in den nächsten Monaten *webdevPLUS* bestimmt erweitern. Jetzt schon enthalten ist ein Linux-Dienst-Skript, mit dem Sie Ihre *webdev-server* dauerhaft in Ihr Suse-Linux-System aufnehmen können. Geplant ist zusätzlich, ein Template für die Erstellung von Webanwendung mit dem Tomcat aufzunehmen.

Eine weitere sehr nützliche Initiative ist das deutsche Centaurus-Projekt von Thorsten Kamann und Peter Roßbach [11], [8]. Das Ziel des Projekts ist, eine eigenständige Tomcat 5-Distribution mit HostCreator, Service Wrapper, Memory- und Security Watcher auf Basis einer Plugin-Architektur für den produktiven Einsatz zu schaffen. Die Motivationen des Centaurus-Projekts sind, einen stabilen Betrieb von Tomcat-Servern sicherzustellen und kontrollierte Erweiterungen zu erlauben. Ein Schwerpunkt liegt auf der Unterstützung von Hostern, die auf dem eigenen Windows- oder Linux-Root-Server den Tomcat betreiben. Wir hoffen, einen weiteren Beitrag zur schnellere Verbreitung von Java-Webtechnologie auf der Basis des Tomcat zu liefern.

Wir freuen und schon auf Kommentare und Anregungen – besuchen Sie also unsere TomC@-Seite [1] und das TomC@-Forum [12] oder die Tomcat-Mailing-Listen [13]. ■

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

Lars Röwekamp (lars.roewekamp@openknowledge.de) ist als IT-Berater mit den Schwerpunkten neue Technologien und OO/Enterprise Java für seine Firma OpenKnowledge tätig.

News aus der Tomcat Community

Mal wieder etwas über das TomC@-Projekt und nicht nur Technik.

Neues aus dem Tomcat-Projekt …

- Der Tomcat 5 soll monatlich aktualisiert werden (5.0.19 Ende Februar und 5.0.23 Anfang Mai).
- Die JK2-Konnektoren für die Apache httpd- und IIS-Integration liegen nun in der Version 2.0.4 bereit.
- Die Konfiguration und Implementierung des Clustering ist seit dem 5.0.19-Release umgestellt und erweitert worden. Es werden nicht mehr alle Sessioninformationen repliziert, sondern mithilfe des DeltaManager nur noch geänderte Session-Daten.
- Der Jasper hat jede Menge Bugfixes für den JSP 2.0-Support erhalten.
- Natürlich sind in jedem Release einige sehr unschöne Sachen gefixt.
- Mit dem 5.0.21 Alpha-Release berücksichtigt der Tomcat das neue Apache 2.0-Lizenzmodel.
- Einsatz des aktuellen Xerces Parser 2.6.2.
- Beliebige Zusammenstellung der Klassenpfade via *catalina.properties.*
- Der Tomcat 5 läuft auch schon inoffiziell mit dem neuen MX4J 2.0 und damit können schon erste Erfahrungen mit dem Client JMX API nach JSR 160 gesammelt werden [14], [15].

Neues auf unser Tomc@ Site tomcat.objektpark.org

- Im Bereich „Howto“ beginnen wir Beispiele zu sammeln.
- Aktuelle Termine für Schulungen und Konferenzen.
- Sämtliche Beispiele und ausgewählte Texte befinden sich im Archiv der Kolumne.
- Im Bereich „Professioneller Service“ können Sie sich über unser Beratungsangebot informieren.
- Im Bereich „Tomcat – Das Buch“ befinden sich erste Informationen zu unserem Tomcat-Buchprojekt.

Links & Literatur

[1] tomcat.objektpark.org/

[2] java.sun.com/products/servlet/

[3] java.sun.com/products/jsp/

[4] 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, S. 112 ff.

[5] jakarta.apache.org/tomcat/

[6] java.sun.com/j2se/1.4.2/

[7] Peter Roßbach, Lars Röwekamp, Sascha Olliges: TomC@ – die Kolumne: Kater managen – eXterne Adminstration und Management mit JMX, in *Java Magazin* 1.2004

[8] Peter Roßbach, Lars Röwekamp: TomC@ – die Kolumne: Mein Tomcat ist eine Festung … – Tomcat als Trutzburg: Sicherheit für Leib und Leben, in *Java Magazin* 4.2004

[9] ant.apache.org/

[10] xml.apache.org/forrest/

[11] centaurus.planetes.de/

[12] www.javamagazin.de/tomcat/

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

[14] jcp.org/aboutJava/communityprocess/final/jsr160/

[15] mx4j.sourceforge.net/

[16] www.apache.de/dist/jakarta/tomcat-connectors/jk2/binaries