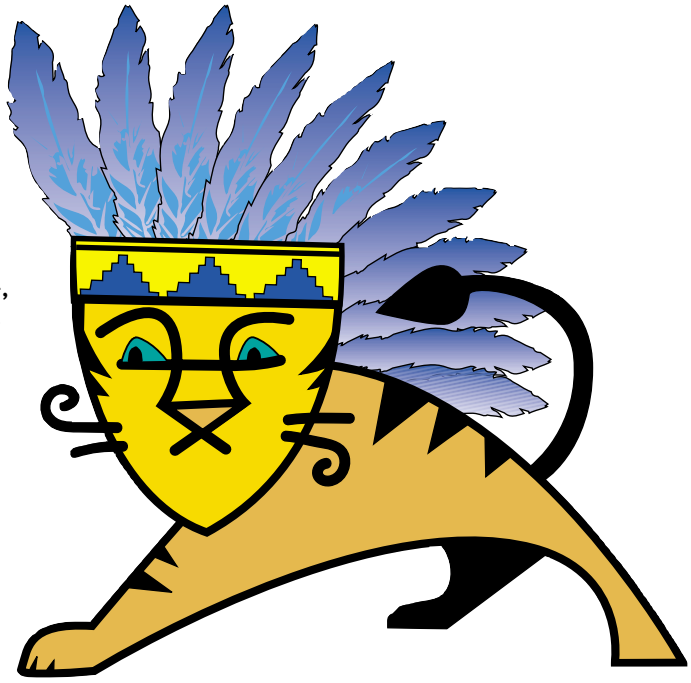
Tomcat mit dem Apache-Webserver nutzen

Vom Mythos einer einfachen Integration

Die Wurzeln des Projekts Tomcat sind eng mit dem Apache-Server httpd verbunden und mit der Zeit ordentlich gewachsen. Die Bande mit dem Apache haben bereits 1997 mit dem legendären Projekt JServ begonnen. Heute ist der Tomcat ein vollständiger und in der Praxis erprobter eigenständiger Web-Container. Trotzdem gibt es jede Menge Szenarien, in denen es erwünscht oder notwendig ist, die Verbindung mit einem Webserver herzustellen. Das neue JK2 2.0.4-Release verbindet den Tomcat mit Apache 2-, Apache 1.3-, Domino Netware-, IIS- und IPlanet/Netscape-Servern. In dieser Kolumne betrachten wir die Integration zwischen Apache und Tomcat etwas genauer.



Der Tomcat mit seiner flexiblen Architektur eignet sich hervorragend als Basis für die unterschiedlichsten Einsatzgebiete: Ob nun als J2EE Web-Container in einem JBoss, JOnAS, als eigenständiger Server oder in Verbindung mit einem Webserver, alle sind als Lösungen seit Jahren erfolgreich im Einsatz. Diese Universalität des Tomcat ist sicherlich einer der wesentlichen Faktoren für die nicht abreißende Erfolgsgeschichte des Jakarta Tomcat-Projektes [1]. Der Apache-Server httpd spielte für die Verbreitung der Java-Servlet-Technologie eine bedeutende Rolle. Seine hohe Akzeptanz verhalf dem Apache-Projekt JServ und damit der Java-Servlet-Technologie zu einem beeindruckend schnellen und anhaltenden Wachstum [2]. Als das Jakarta-Projekt Tomcat 1999 aus der Taufe gehoben wurde, war der Einsatz des Web-Containers Tomcat schon ein Jahr später in vielen Firmen akzeptiert. Aktuell z.B. ergab sich aus einem Quickvote des *Java Magazins*, dass von mehr als 600 Entwicklern über 74 Prozent den Tomcat zur ihrer Web-Container-Plattform erkoren haben [3].

Von Beginn an flossen die Erfahrungen mit dem erfolgreichen Apache Java Protocol (AJP) aus dem Projekt JServ in die weitere Entwicklung des Tomcat ein. Heute steht das AJP/1.3+-Protokoll im „jakarta-tomcat-connector"-Projekt den Tomcat-Versionen 3.3.x, 4.1.x und 5.0.x zur Verfügung [4], [5]. Auf der Seite des Apache-Servers war lange das JK-Modul, der direkte Nachfolger des JServ-Moduls, das Maß aller Dinge in Sachen Webserver-Integration. Das Besondere an der JK-Implementierung ist, dass die Integration auf weitere Webserver ausgedehnt wurde und damit neue Einsatzgebiete für den Tomcat erschlossen wurden.

Aber wenn der Tomcat doch eigentlich ein vollständiger HTTP/1.1 Server ist, warum brauchen viele Kundenprojekte dann überhaupt eine Integration in ihren bestehenden Webserver? Na ja, das Web und seine Technologien sind vielfältig. Täglich gibt es neue Kunde von innovativen An-

sätzen. Einige gute Lösungen gab es schon, bevor Java die Welt der Server entdeckt hat. Die Nutzung von Anwendungen, die auf Basis von CGI, Perl, PHP oder ASP geschrieben sind, soll im selben Web möglich sein. Die großen Webserver nutzen klassischerweise die Gegebenheiten der Betriebssysteme ausgezeichnet aus. So werden SSL-Verbindungen meist zügiger geschlossen oder es existieren gute Lösungen für das Caching von statischen oder gar dynamischen Inhalten. Die Wartung so mancher Website ist nur durch das Rewrite-Modul des Apache überhaupt noch möglich. Mancher hat eine organisatorische Trennung zwischen der Bereitstellung seiner Kundenanwendungen und der Webpräsentation mithilfe eines Content Management System (CMS) vollzogen. In meinem Apache/Tomcat-Vortrag auf der JAX 2004-Konferenz hat ein Teilnehmer allerdings ein sehr überzeugendes Argument für die Nutzung des Apache-Servers mit dem Tomcat geliefert: „Mein Chef will das halt einfach so!“ Kurz und gut, in dieser Kolumne stellen wir Ihnen den Nachfolger des JK-Moduls vor und zeigen Ihnen die Integration des Tomcat in den Apache [6], [7], [8], [9].

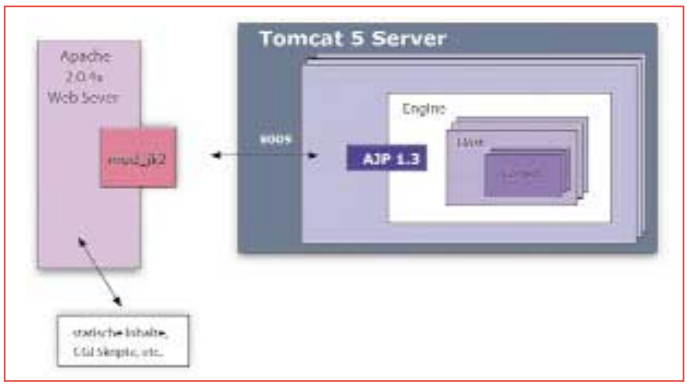


Abb. 1: Verbindung zwischen Apache und Tomcat mit dem JK2

Das JK2-Modul

Der wesentliche Grund für die Entwicklung eines neuen Moduls im Projekt Tomcat zur Integration von Webservern besteht darin, dass man den kommenden Herausforderungen von verbesserten APIs von Webservern und die modernen Eigenschaften, wie der Multithread-Fähigkeit, eine angemessene Unterstützung zukommen lassen möchte. Das JK ist als direkter Nachfolger des Projekts JServ zu verstehen und sehr eng an das Design des Apache 1.3 angelehnt. Hier sind Design und Realisierung des JK2 wesentlich abstrakter geworden und stellen den verschiedenen Webserver-Integrationen eine breitere Basis zur Verfügung. Im Falle des Apache 2 wird seit der 2.0.4-Version des JK2 beispielsweise das Apache Portable Runtime-

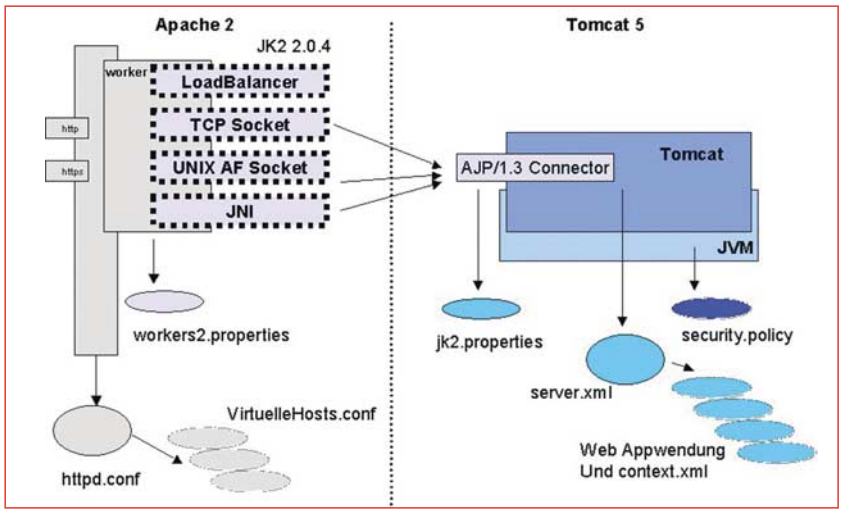


Abb. 2: Konfiguration des Apache mit *mod_jk2*

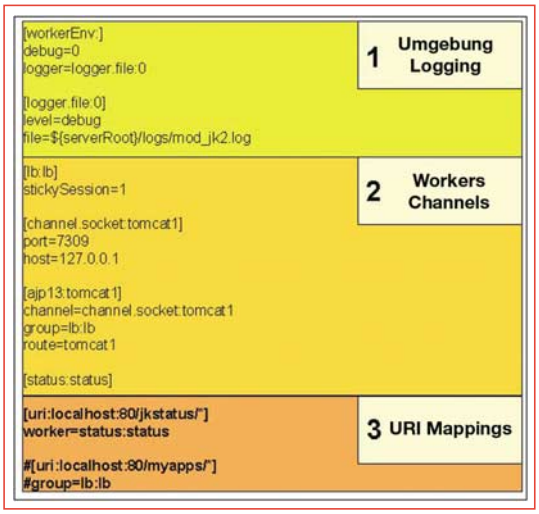


Abb. 3: Die Apache 2-Konfiguration mit *workers2.properties*

API (APR) genutzt und damit ist die Portierbarkeit auf verschiedene Betriebssysteme radikal vereinfacht worden. Natürlich ist die Implementierung durch diesen Schritt erheblich übersichtlicher und einheitlicher geworden. Es ist halt nicht leicht, eine Lösung in der Programmiersprache C für unterschiedliche Betriebssysteme auf Basis etlicher *ifdef*-Schalter im Griff zu behalten. Das Design des JK2 ist nun so aufgebaut, dass man verschiedene Kommunikationsprotokolle bei Bedarf hinzunehmen kann. Wesentlich für uns Anwender ist allerdings, dass nun eine dynamische Konfiguration und eine Statusanzeige Einzug in das Projekt gefunden haben.

Die Integration von Apache und Tomcat erfolgt durch die Installation eines Moduls (*mod_jk2*) auf der Seite des Apache. Im Tomcat muss nur ein entsprechender AJP Connector konfiguriert werden und der Austausch kann beginnen (Abb. 1). Nachfolgend werden die einzelnen Schritte und Optionen dieser Integration detaillierter beschrieben. Auf der Heft-CD befindet sich neben den aktuellen Versionen Apache (2.0.49) und Tomcat (5.0.25) eine Beispielkonfiguration für die folgende Apache-Webserver-Integration [10].

Schrittweise Installation und Konfiguration der Integration

Bevor wir einfach loslegen, sollten wir noch ein paar wesentliche Entscheidungen für unser Beispiel überdenken. Das JK2-Modul für den Apache besitzt verschiedene Kommunikationsmechanismen (Abb. 2) [11], [10]. Die meistgenutzte Möglichkeit ist die Integration mit TCP/IP Sockets. Die Realisierung von Sockets ist auf allen Betriebssystemen vorhanden und eine spätere Verteilung auf verschiedene Maschinen ist einfach möglich. Mit Unix AF Socket Pipes steht eine sehr schnelle Lösung für Unix-Maschinen zur Verfügung. Allerdings müssen für die Nutzung der Pipes die Prozesse des Apache und Tomcat auf derselben Maschine ablaufen. Als dritte Option ist die Integration der JVM mit dem Tomcat direkt in den Apache-Prozess via Java Native Interface (JNI) vorgesehen. Weitere Optionen, z.B. die Nutzung des Java NIO Package, sind schon oft auf den Mailinglisten diskutiert worden [12]. Im Beispiel dieser Kolumne kommen die vorbereiteten TCP/IP Sockets zur Anwendung.

Konfiguration des Tomcat

Die Konfiguration der Socket-Lösung auf der Seite des Tomcat ist zunächst recht einfach. Die erfolgreiche Installation des Apache 2 und Tomcat 5 vorausgesetzt [6], [1], wird bloß der AJP/1.3 Connector in der Datei *conf/server.xml* eingetragen (Listing 1). Im Tomcat ist diese Konfiguration schon vorhanden und über den Socketport *8009* erreichbar. Die beste Voraussicht für eine ordentliche Tomcat-Installation bietet immer noch das Erstellen einer eigenen *$catalina.base*-Konfiguration mit dem *webdevPlus* Template, wie dies im Beispiel auf der CD vollzogen ist [13]. Ein guter Startpunkt für eine beherrschbare Konfiguration im Tomcat 5 ist in der Datei *conf/server-minimal.xml* des Release zu finden.

Eine weitergehende Konfiguration des AJP Connector, wie Einstellung des Thread Pools oder Veränderungen der Socket-Eigenschaften, kann mithilfe der bekannten Coyote Connector-Parameter erfolgen [1]. Einen noch darüber hinaus gehenden Einfluss auf die Eigenschaften des AJP Connector können Sie mit der Datei *jk2.properties* nehmen, die später noch detailliert erläutert wird (Abb. 2 und Listing 3) [7]. Der nicht dokumentierte Parameter *jkHome* des AJP Connector sorgt dafür, dass Sie für jeden AJP Connector eine eigenständige Datei *jk2.properties* in einem Verzeichnis *$catalina.base/conf/apache2/conf/* oder *$catalina.base/conf/apache2/etc/* nutzen können. So kann man problemlos für verschiedene Kunden wirklich getrennte Servicekonfigurationen mit entsprechenden eigenständigen Konnektoren verwirklichen.

Der Parameter *jvmRoute* ermöglicht dem LoadBalancer im Apache, auf Basis der markierten Sessions den zugehörigen Tomcat zu finden. Der Listener *ServerLifecycleListener* sorgt für die Integration der JMX-Meta-Informationen und verbessert unsere Apache-Statusintegration, dazu später mehr.

Konfiguration des Apache-Servers

In gewohnter Weise müssen in der zentralen Konfigurationsdatei des Apache *$apache2.home/conf/httpd.conf* entsprechende Anweisungen zur Aktivierung des JK2-

Moduls und Einbindung von dessen Konfiguration eingefügt werden. Im Beispiel steht die Datei *conf/apache2/mod_jk2.conf* dafür zur Verfügung. Deren Inhalt kann komplett übernommen, via Include-Anweisung eingebunden oder beispielsweise unter Suse Linux in das Verzeichnis */etc/apache2/conf.d/* kopiert werden. Die Konfiguration des JK2 erfolgt im Beispiel auf der Basis der Datei *workers2.properties* (Abb. 3).

```
<IfModule !mod_jk2.c>
 LoadModule   jk2_module "D:
              \TomC@-04-08\native\win32\mod_jk2.so"
</IfModule>
<IfModule mod_jk2.c>
 JkSet config.file
   "D:\TomC@-04-08\webdev-server\conf\apache2\
                                  workers2.properties"
</IfModule>>
```

Im Prinzip ist damit die Integration abgeschlossen und nach einem Start bzw. Neustart der beiden Server läuft die Verbindung. Im Beispiel auf der CD wird die Unterschiedlichkeit Ihrer konkreten Serverinstallation durch eine entsprechende Konfiguration der Datei *build.patch* realisiert.

Bevor wir aber zu einer ernsthaften Prüfung kommen, werfen wir noch mal einen Blick auf die Datei *workers2.properties* (Abb. 3). Die Konfiguration besteht aus drei Teilen: Zuerst legt man eine Umgebung mit allgemeinen Einstellungen fest (Abb. 3 (1)). Danach erfolgt die Definition der konkreten Kommunikationsarten in Form von Workern und Channels (Abb. 3 (2)) und abschließend wird bestimmt, auf welche URIs welcher Worker reagieren soll (Abb. 3 (3)).

Im Beispiel haben wir eine Logger-Datei festgelegt, die alle Ausgaben bis zum *debug*-Level protokolliert. Jedes Element besitzt die Parameter *debug*, *disabled* und *ver* für die Konfigurationsversion. Der LoadBalancer Worker (*lb:lb*) dient dazu, mehrere verschiedene tatsächliche Worker für eine Lastverteilung zusammenzuschalten. Der LoadBalancer im *Sticky-Session*-Modus wählt den jeweiligen Tomcat-Server nur einmal für die erste Anfrage vom Client und sendet danach alle weiteren Anfragen dieser Session immer wieder zum selben Server. Wenn mehrere Server im LB konfiguriert sind, verteilt er die Last und steuert die Anfragen bei einem Ausfall notfalls auf einen anderen Tomcat um. Selbst Hot Standby-Szenarien, Graceful Shutdown einzelner Tomcats oder gewichtete Verteilung lassen sich in der JK2-Version 2.0.4 konfigurieren. Nicht alle Parameter sind zurzeit öffentlich auf der Site dokumentiert und deshalb liegt die Dokumentation der *workers2.properties*-Elemente des CVS Head (2.0.5 dev) auf der Heft-CD bereit.

Der Worker ist eine Abstraktion für den konkreten Kommunikationskanal. Er wird von LB zur Vermittlung von Anfragen herangezogen und sorgt dafür, dass ein entsprechender Channel für die Kommunikation mit dem Tomcat bereitsteht. Dazu bedient er sich im Fall des TCP/IP Channel eines Pools und kann bei störanfälligen Verbindungen vor den eigentlichen Client-Anfragen einen Verbindungstest mit CPING-Anfragen und CPONG-Antworten durchführen. Für jeden AJP Connector auf der Seite des Tomcat ist ein Worker auf der Seite des Apache zuständig. So kann ein TCP/IP Worker in einen Graceful Mode geschaltet werden und damit bewirken, dass auf diesen Tomcat keine neuen Anfragen mehr vom LoadBalan-

Listing 1

Konfiguration des AJP/1.3 Connector in der Datei *conf/server.xml*

```
<Server port="7005"
    shutdown="SHUTDOWN"
    debug="0">
 <Listener
   className="org.apache.catalina.mbeans.
                                ServerLifecycleListener" />
 <Service name="Catalina">
   <Connector
       port="7309"
       protocol="AJP/1.3"
       jkHome="../conf/apache2"/>
   <Engine
       name="Catalina"
       defaultHost="localhost"
       jvmRoute="tomcat1">
   <Host
       name="localhost"
       appBase="webapps"
       unpackWARs="false" />
  </Engine>
 </Service>
</Server>
```

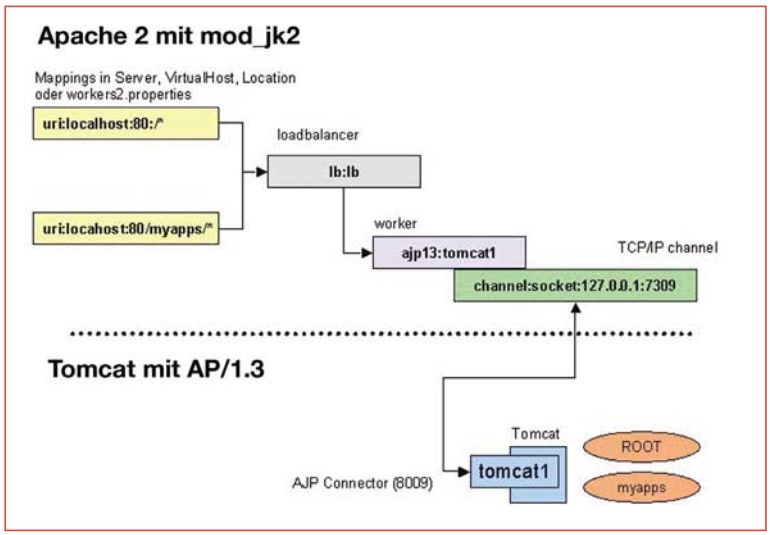


Abb. 4: JK2 TCP/IP-Konfiguration zur Verbindung von Apache und Tomcat

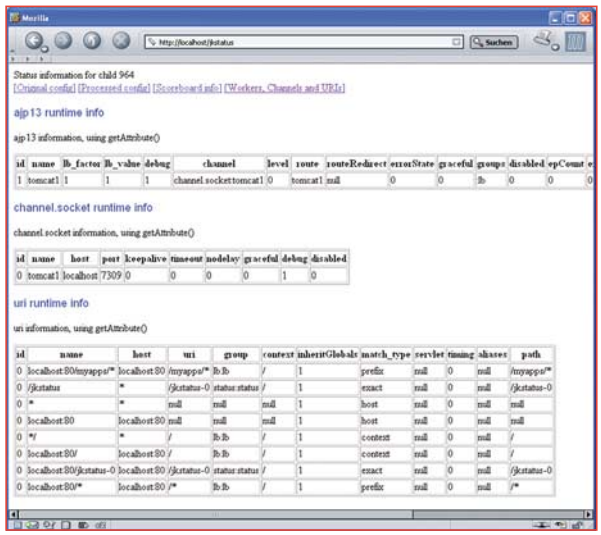


Abb. 5: *jkStatus* Monitoring im Apache 2

cer gelenkt, sondern nur noch bestehende Anfragen einer laufenden Session weitergeleitet werden. Da fühlt sich doch jeder von uns an die netten und ziemlich langen Warteschlangen beim Samstagseinkauf vor drei geschlossenen Kassen erinnert, oder? Ist halt immer ein erhabener Moment, wenn so eine Kasse direkt vor einem, nach ca. 15-minütigem Warten, geschlossen wird.

Eigenschaften des *mod_jk2*

- Vollständige Reimplementierung des beliebten JK-Moduls
- Statusanzeige im Apache und via JMX im Tomcat
- Dynamische Konfiguration direkt im Apache
- Lastverteilung
- Gewichtete Verteilung auf Basis eines Round Robin-Algorithmus
- Fehlerbehandlung und Recovery
- Sticky Session Handling und Stateless Balancing
- Ausfallsicherung
- Cold und Hot Standby
- Graceful Shutdown
- Unterstützung von folgenden Webservern
- IIS
- Apache 1.3.x
- Apache 2.x
- Netware Domino
- Netscape/IPlanet
- Verfügbare Worker
- Status
- LoadBalancer
- TCP/IP Socket
- Unix AF Sockets
- JNI

Der Status Worker sorgt dafür, dass wir endlich die konkrete Konfiguration und den Zustand der Worker im Apache ansehen und verändern können (Abb. 5). Mit der Anfrage *http://localhost/jkstatus* können Sie die gewünschten Informationen erfragen. Nun fehlt uns zum vollständigen Segen noch die Konfiguration des URI Mapping, damit der Apache die Anfragen an unseren Tomcat gezielt weiterleitet. Die *uri*-Elemente sind dafür verantwortlich, diesen Zusammenhang zwischen URI und einem Worker herzustellen (Abb. 3 u. 4). Hierbei sollte schon im einfachsten Fall auf einen LB Worker zurückgegriffen werden, damit man bei einer frequentierten Website einfach weitere Worker und Tomcats hinzunehmen kann. Bei einer umfangreichen Umstellung des URI Mapping unter Zeitdruck sollten Sie mit dieser LB-Zuordnung sicher sein.

Integration der Webanwendung in den Apache-Server

Nun fehlen noch die konkreten Webanwendungen, die wir im Apache auf einem Tomcat-Web-Container bereitstellen möchten. Das Deployment kann einfach im Tomcat mit den üblichen Mitteln der Manageranwendung oder direkt über das Verzeichnis erfolgen [1] (Abb. 4). Die beiden Anwendungen *ROOT* und *myapps* auf der Heft-CD sollen dabei in den Apache Server integriert werden. Dafür tragen wir einfach das entsprechende URI Mapping in die Datei *workers2.properties* ein.

```
[uri:/*]
group=lb:lb

[uri:/myapps/*]
group=lb:lb
```

Mit einem weiteren Aufruf von *http://localhost/jkstatus* wird unter Windows die Konfiguration auch sofort übernommen. Super, aber der Versuch, die neue Anwendung mit *http://localhost/myapps* im Browser zu erreichen, schlägt fehl. In der *ErrorLog*-Datei des Apache findet sich eine Fehlermeldung, dass ein Verzeichnis *…/htdocs/myapps* nicht gefunden werden konnte. Seltsam, versuchen wir es mal mit einem Graceful Restart des Apache. Damit wird die gesamte Konfiguration erneut übernommen. Wow, nun ist alles konsistent und alle Anfragen an die *myapps*- und *ROOT*-Anwendung werden korrekt beantwortet. Leider wird nun eine weitere Anfrage auf *http://localhost/jkstatus* nicht mehr bearbeitet. Das URI Mapping unserer *ROOT*-Anwendung überlagert offensichtlich das Status Mapping. Abhilfe bringt ein direktes Mapping in der *mod_jk2.conf*- bzw. Ihrer *httpd.conf*-Datei und das gleichzeitige Entfernen des *jkstatus* URI-Elements in der Datei *wor-*

kers2.properties (Abb. 3). Nicht vergessen, dass die Änderungen erst nach dem Neustart des Apache aktiv sind.

```
...
<IfModule mod_jk2.c>
  <Location "/jkstatus">
    JkUriSet worker status:status
    Order Deny,Allow
    Deny from all
    Allow from 192.168.1
  </Location>
</IfModule>>
...
```

Neben der *workers2.properties* kann auch die gesamte Konfiguration des JK2 direkt mit entsprechenden Befehlen in den Elementen *Server*, *VirtualHost* und *Location* des Apache definiert werden. So kann man beispielsweise manuelle JK2-Konfigurationen erzeugen, wobei der Apache die statischen Dateien und der Tomcat nur die dynamischen Inhalte beantwortet. Solche Konfigurationen sind allerdings wartungsanfällig und nicht für alle Webanwendungen zu empfehlen. Gerade die Autorisierungen oder das Session-Management einer Anwendung, der Einsatz von Servlet-Filtern, Security Constraints oder einfach schnelle Versionswechsel mit sich ständig verändernden Mappings können zu einer extrem zeitaufwändigen Tätigkeit werden. Fehlerhafte Verweise und die nicht beabsichtigte Gewähr auf geschützte Inhalte sind meist die unerwünschte Folge.

Erste Generatorenansätze sind zwar mit der Klasse *org.apache.jk.config.WebXml2Jk* vorhanden, liefern aber nur eingeschränkt die gewünschten Ergebnisse (Heft-CD-Beispiel im Skript *myapps/genjk2.xml*) und keinerlei automatischen Abgleich mit der Apache-Konfiguration. In dem Centaurus Platform-Projekt arbeiten wir gerade an einem solchen automatischen Abgleich, der auch ein Redeployment und Entfernen der Webanwendungen konsistent mit dem Apache ermöglicht [14].

JK2-Status via JMX im Tomcat

Die Statusinformationen des JK2 im Apache sind sehr detailliert und geben jederzeit exakte Informationen über den Zustand des Moduls (Abb. 5). Es ist sogar möglich, die einzelnen Attribute der Konfiguration direkt zu ändern. Das gesamte JK2-Modul hat sich das JMX-API zum Vorbild genommen und jedes der Konfigurationselemente besitzt eine entsprechende Struktur. Mit dem *jkstatus*-Befehl können alle Attribute der bestehenden Elemente oder die aktuellen Meta-Informationen angezeigt und manipuliert werden. Diese JK2 JMX-Struktur kann sogar

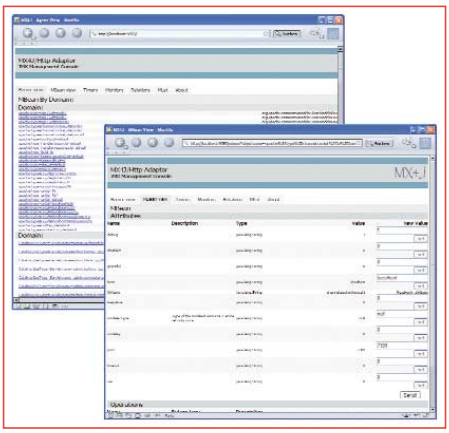


Abb. 6: JK2 Monitoring via JMX im Tomcat

in Ihren Tomcat als JMX MBeans gespiegelt werden. Dazu bedarf es einer speziellen Konfiguration in der Datei *jk2.properties* im Tomcat-Server (Listing 2).

Die Handler *modjk* und *mx* werden über die Handler-Liste aktiviert und für den AJP Connector bereitgestellt. Der *modjk*-Handler bekommt die Parameter des Apache-Servers, mit dem die Spiegelung erfolgen soll. Theoretisch ist es in einem Tomcat möglich, mehrere solche Konfigurationen von verschiedenen Apaches zu laden. Alle fünf Sekunden wird ein Abgleich mit dem Apache durchgeführt. Die MBeans befinden sich in der JMX-Domain *apache* (Abb. 6). Zur Anzeige kann man die MBeans beispielsweise über den JMX HTTP Adaptor des MX4J-Projekts [15] mit dem *mx*-Handler aktivieren. Leider funktionierte der automatische Abgleich zwischen Apache und Tomcat nicht. Natürlich können Sie nun auch mit einem Ant-Skript und dem Manager JMXProxy Servlet Ihre Apache JK2-Konfiguration pflegen, wie es in der TomC@-Kolume im *Java Magazin* 1.2004 beschrieben steht [16]. Allerdings sollten Sie den *jkstatus*-Link und den MX4J Adaptor vielleicht besser durch eine Autorisierung und entsprechende Zugangsbeschränkungen auf interne Server schützen (manuelle *jkstatus*-Konfiguration). Auf einem Produktionsserver sollten Sie das Störungsrisiko gut abwägen. Ein Einsatz des MX4J 2.0.x mit sicherem Remote Client API ist noch nicht direkt vorgesehen. Auf der Centaurus Platform steht ein JMXAdaptor mit konfigurierbarer Zugangsbeschränkung, SSL und flexibler Ausgabesteuerung auf der Basis des neuesten MX4J 2.0.1 zur Verfügung [14], [15].

Den *jk2.properties*-Parameter *request.registerRequests* sollten Sie unbedingt auf *false* setzen, ansonsten sammelt der AJP Connector fleißig bei jeder Anfrage ein Statistikobjekt, das nie jemand abholt. Eine sporadische *OutOfMemory* Exception ist leider schon bei mittlerer Last die unausweichliche Folge. In der nächsten Version des JK2 und Tomcat soll diese Schwäche behoben sein.

Fazit

Nehmen Sie sich nun erst mal die Zeit, das CD-Beispiel auf einem Ihrer Rechner auszuprobieren. Dabei sollten Sie sich erst mal mit der beschriebenen Basiskonfiguration anfreunden. In die Details des JK2

Interview mit dem Tomcat Committer: Günter Knauf

Günter Knauf hat wesentlich dazu beigetragen, dass die Entwicklung des JK2-Modul nach 15 Monaten Stillstand neu belebt wurde und eine leistungsfähige Anbindung an den NetWare Apache 2-Server nun existiert.

Java Magazin: Günter, was hat dich dazu angetrieben, dem Tomcat JK2-Modul neuen Schwung zu geben?

Günter Knauf: Ich biete schon seit Jahren Apache Binaries und Module Binaries für NetWare und Win32 auf meiner Site (www.gknw.com/development/apache/) an. Hauptsächlich, weil diese Binaries mit kommerziellen Compilern erstellt werden, die nicht jedem zugänglich sind. Daher ist es auch schwierig, 3rd Party-Module für diese Betriebssysteme zu bekommen, und ich versuche ständig, neue Module für diese Plattformen zu kompilieren. Durch meine Site wurde ich oft nach einem fertigen JK2-Modul gefragt, ebenso in den Novell Developer-Foren, wo ich auch Systemoperator bin. Daher habe ich es einfach mal versucht, das *mod_jk2* auch für NetWare zu kompilieren. Die Voraussetzungen unter NetWare mit Java, Tomcat 4 und Apache 2 waren gut, also fehlte nur „noch" das *mod_jk2*. Einen großen Dank geht an den Australier Norm Wheeler, der unermüdlich getestet, Fehler analysiert und Patches erstellt hat. Ohne seinen Einsatz wäre die erreichte Qualität im Projekt Tomcat Connector in weite Ferne gerückt.

JM: Ist das JK2 Release 2.0.4 für produktive Server freigegeben und worauf muss man doch noch achten?

Knauf: Wir haben *mod_jk2* bereits zwei Monate vor derem endgültigen Release in Produktion auf NetWare, Linux, Win32 eingesetzt. Der Einsatz lief ohne Probleme und mit signifikanter Geschwindigkeitssteigerung auf allen Plattformen. LoadBalancing arbeitet möglicherweise unter Heavy Load-Bedingungen noch nicht ganz fehlerfrei, es scheint in sehr seltenen Fällen vorzukommen, dass sich ein Tomcat-Server einfach aus dem Balancer-Pool ausklinkt.

JM: Was ist der Stand der NetWare Apache 2 JK2-Integration?

Knauf: Das JK2-Modul ist auf allen Betriebssystemen zu 100 Prozent gleich, lediglich das Scoreboard läuft noch nicht, da NetWare offiziell noch kein Shared Memory unterstützt.

JM: Auf welche Impulse und neue Eigenschaften des JK2-Moduls können wir uns in diesem Jahr noch freuen?

Knauf: Schwer zu sagen; ich glaube unsere größte Aufgabe im Projekt Tomcat Connector ist die Dokumentation. Die Dokumentation ist in Teilen schlecht oder schlicht nicht vorhanden. Speziell für die Entwicklung und den Einsatz ist bisher kein API oder keine Designdokumentation in Sicht. Darum hier noch mal ein Aufruf an alle: Jeder kann mithelfen, das Open Source-Projekt Tomcat in diesem Punkt zu verbessern. Wenn ihr also das nächste Mal zwei Stunden im Internet gegoogled habt, bis die Software in eurer Umgebung erstmalig erfolgreich läuft, schreibt es einfach ins Tomcat-Wiki oder in eine der Mailinglisten. ☺

JM: Vielen Dank!

Das Gespräch führte Peter Roßbach.

Listing 2

JMX-Integration einer Apache-Konfiguration mithilfe der *jk2.properties*

```
handler.list=modjk,mx
# Überschreibt den StandardPort
# des Channel-Sockets
channelSocket.port=7309
# JMX-Integration des Apache jkstatus
modjk.webServerHost=localhost
modjk.webServerPort=80
modjk.statusPath=/jkstatus
# Default alle 5 sec
#modjk.updateInterval=5000
#modjk.user=
#modjk.pass=

# JMX mit MX4J 1.1.1 HTTP Adaptor
mx.enabled=true
mx.httpPort=9000
mx.httpHost=localhost
#mx.jrmpPort=1099
#mx.jrmpHost=localhost

# Don't register request at
# JMX statistic (fix memory leak)
# see http://issues.apache.org/bugzilla/
                              show_bug.cgi?id=28321
# (william barker comment)
request.registerRequests=false
```

einzutauchen ist schwierig und eventuell langwierig. Die hier beschriebenen Eigenschaften und Möglichkeiten des JK2 sind hoffentlich ausreichend für eine produktive Standardinstallation. Das JK2 ist ein äußerst leistungsfähiges Modul zur Integration von Webservern und steht für den Apache 2, Apache 1.3, IIS, Domino und Netscape/IPlanet zur Verfügung [7], [11], [17], [18]. Kleine Tücken hat das JK2 Release 2.0.4 noch, aber das Release 2.0.5 ist schon in der Vorbereitung. So sollte beispielsweise immer nach einem Aufruf des Status Worker die aktuelle *workers2.properties* auch aktiv werden. Leider konnte ich die Neukonfiguration immer erst nach einem Graceful Restart des Apache aktivieren.

Besonders überzeugt das JK2 durch die vielfältigen Optionen und Konfigurationsarten, die für die unterschiedlichsten Szenarien einer Webserver-Integration höchst sinnvoll und angemessen sind.

Mit dem JK2 2.0.4 Release ist der Tomcat Community im letzten halben Jahr ein wirklich großer Schritt für eine leistungsfähige Unterstützung von Webservern gelungen.

Ich freue mich auf Kommentare und Anregungen – besuchen Sie also meine TomC@ Site [10] und das TomC@-Forum [19] oder die Tomcat Mailing-Listen [12].

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

News über den Tomcat

Tomcat – jakarta.apache.org/tomcat

- Die Version 5.0.25 ist seit dem 27. Mai 2004 veröffentlicht
  - Verbesserung des Deployment
  - Clustering mit dem *DeltaManager* auf der Basis von JDK 1.4.x
- Die offiziellen Spezifikationstests für Servlet API 2.4 und JSP API 2.0 sind erfolgreich absolviert worden (TCK Passed)
- Erste Diskussion um ein potenzielles Redesign für den Tomcat 5.5 ist angelaufen

Centaurus-Platform – centaurus.sourceforge.net/

- Das 1.0 Beta 5 ist Mitte Juli 2004 veröffentlicht
  - Update auf Tomcat 5.0.2x, Wrapper 3.1.0, MX4J 2.0.1
  - Die neue Management Console:
    - ergonomisches Management eines Tomcat-Servers mit einer Struts-Anwendung umgesetzt
    - gezielte Auswertung des Status einzelner Webanwendungen nach JSR 77
    - direktes Deployment in der Anwendung zusätzlich mit Ant-Tasks
    - Verwendung evtl. vorhandener Confixx-Benutzer ist mit eigenem Realm möglich
    - Einsicht in die eigenen Log-Dateien
    - aktuelle Speicherverbrauchsanzeige
  - MX4J 2.0.1 HTTP Adapter Integration
    - Verwendung eigener XSL Stylesheets
    - Realisierung einer Zugangskontrolle
    - SSL-Unterstützung
- Im nächsten Release geplant:
  - automatische Apache Tomcat-Konfiguration
    - Reflektion des Deployments von Webanwendung
    - Integration mehrerer Centaurus-Profile in einem Apache
    - XML Descriptor
    - Statusanzeige im Tomcat
  - Hypersonic DB-Plugin
  - Erweiterung der Management Console
    - Deutsches Hilfesystem
    - Bessere Statusanzeige
- Weitere Entwickler gesucht und höchst erwünscht

Tomcat Site – tomcat.objektpark.org/

- Angebot individueller Workshops rund um den Tomcat vergrößert
- Neues Catalina Base Template webdevPlus 1.3.1
- Archive der TomC@-Kolumne
- Tipps und Tricks für Ihre Tomcat-Aufgaben

Links & Literatur

[1] jakarta.apache.org/tomcat/

[2] archive.apache.org/dist/java/

[3] javamagazin.de/itr/polls/jm/viewquikvote.php3?func=results&poll_id=16

[4] jakarta.apache.org/tomcat/tomcat-4.1-doc/jk2/common/AJPv13.html, jakarta.apache.org/tomcat/tomcat-3.2-doc/AJPv13.html

[5] jakarta.apache.org/tomcat/tomcat-4.1-doc/jk2/common/AJPv13-extensions-proposal.html

[6] httpd.apache.org/

[7] jakarta.apache.org/tomcat/tomcat-4.1-doc/jk2/

[8] Peter Roßbach, Lars Röwekamp, Michael Kloss: Tomcat 4 Coyote Http Connector – Auf den Hund gekommen, in *Java Magazin* 12/2003

[9] 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

[10] tomcat.objektpark.org/

[11] wiki.apache.org/jakarta-tomcat/Tomcat_2fLinks

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

[13] Peter Roßbach, Lars Röwekamp: Catalina Base: Eine praktische Anleitung zur flexiblen Tomcat-Konfiguration – Die Saat liegt im Feld, in *Java Magazin* 6.2004

[14] centaurus.sourceforge.net/

[15] mx4j.sourceforge.net/, MX4J 1.1.1 ist unter www.ibiblio.org/maven/mx4j/jars/ zu finden.

[16] Peter Roßbach, Lars Röwekamp, Sascha Olliges: eXtreme Administration und Management mit JMX – Kater managen, in *Java Magazin* 1.2004

[17] Aktuelle Apache-Module für Netware und Win32: www.gknw.com/development/apache/

[18] hunter.campbus.com/ (inoffzielle Windows Apache OpenSSL Version)

[19] www.javamagazin.de/tomcat/