

TOMCAT 5.0.x Installation und Konfiguration

Diese Kurzanleitung bezieht sich darauf den Jakarta TOMCAT 5.0.x zu installieren und eine „sicherere“ Konfiguration zu erstellen. Im wesentlichen heißt das, die nicht benötigten Teile zu löschen bzw. zu deaktivieren und unter Umständen die Konfiguration der Java Sandbox ein wenig anzupassen.

1. Entpacken der Binaries [1]:

```
tar -xzf jakarta-tomcat-5.0.<version>.tgz
```

Der TOMCAT kann nach dem Entpacken 'Out of Box' mit Hilfe des Skripts `$(CATALINAHOME)/bin/catalina.sh` gestartet|gestoppt werden.

<i>Argument</i>	<i>Bedeutung</i>
start [-security]	startet den Server im Hintergrund: <code>STDERR/STDOUT</code> wird während des Startens nach <code>\$(CATALINAHOME)/logs/localhost_log.<datum></code> und danach nach <code>\$(CATALINAHOME)/logs/catalina.out</code> umgeleitet. Die zusätzliche Option <code>security</code> startet den TOMCAT im Sandbox Modus (Java Security Manager) mit den Policies aus <code>\$(CATALINAHOME)/conf/catalina.policy</code> .
stop [-force]	stoppt den Server: im wesentlichen wird ein SHUTDOWN (default) an Controlport 8009 (default) geschickt. Die einzelnen Applikationen werden ordentlich beendet. Die Option <code>force</code> erzwingt ein sofortiges herunterfahren des Servers und bricht evtl. laufende Applikationen hart ab.
run [-security]	startet den Server als Vordergrundprozess: <code>STDERR/STDOUT</code> werden nicht umgeleitet. (Option <code>security</code> siehe start)
debug [-security]	startet den TOMCAT im Java Debugger (Option <code>security</code> siehe start)

2. Admin Interface (Administration Applikation)[1,3]

Zur einfacheren Installation bietet der TOMCAT `$(CATALINAHOME)/server/webapps/admin` eine Administrator Oberfläche, die durch eine mitgelieferte Applikation realisiert ist. Nach dem Entpacken des Tomcat's ist diese so konfiguriert, dass man von überall darauf zugreifen und nach Authentifizierung (User/PWD) alle Einstellungen am TOMCAT via WebInterface vornehmen kann. Um dieses zu Realisieren benötigt der TOMCAT Lese UND Schreibrechte auf die zentralen Konfigurationsdateien und das Verzeichnis (die Admin Webapplikation legt in dem `$(CATALINAHOME)/conf` Verzeichnis Sicherheitskopien der zu ändernden Dateien an. Will man das Admin Interface des TOMCAT nutzen, so sollte man sich im klaren sein, dass die Nutzung vor allem mit den Default Einstellungen ein erhebliches Sicherheitsproblem darstellt, die (einfach) ausgenutzt werden können. Da es sich bei dem Admin Interface um eine (normale) Applikation handelt, kann man ziemlich einfach den Zugriff durch Access bzw. Deny Listen einschränken. Hierfür gibt es zwei Möglichkeiten:

Anpassen der `$(CATALINAHOME)/server/webapps/admin/admin.xml`, durch Einfügen eines `<Valve>` Tags unter dem Root Element. Die Zeile

```
<Valve className="org.apache.catalina.valves.RemoteAddrValve"
allow="127.0.0.1" deny=""/>
```

bzw.

```
<Valve className="org.apache.catalina.valves.RemoteHostValve"
allow="localhost" deny=""/>
```

limitiert den Zugriff auf localhost.

Anpassen der globalen Konfiguration `§CATALINAHOME/conf/server.xml` durch Einfügen (bzw. Anpassen) eines AdminContext Tags:

```
<context patch="/admin" ...>
  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
    allow="127.0.0.1" deny=""/>
</context>
```

Wenn man auf das bequemere Editieren mittels der Admin Webapplikation verzichten will, dann kann die Applikation auch gelöscht werden. Die Konfigurationsdateien des TOMCAT unter `§CATALINAHOME/conf` müssen dann für den TOMCAT auch nur noch lesbar aber **NICHT** schreibbar sein!

1. `rm §CATALINAHOME/webapps/admin.xml` – vorhanden wenn der TOMCAT schon mal mit der Admin Applikation gestartet wurde.
2. `rm -r §CATALINAHOME/server/webapps/admin`

3. Manager Interface (Manager Applikation) [1,3]

Das Manager Interface erweitert den TOMCAT um zwei Möglichkeiten.

1. upload/start/stop/restart von Applikationen
2. upload/start/stop/restart via ANT

Damit nicht jeder beliebig Applikationen von überall Installieren/Starten/Stoppen/etc. kann sollte der Manager Zugriff auf bestimmte Hosts/IP-Bereiche durch einen Valve Eintrag in `§CATALINAHOME/webapps` beschränkt werden.

```
<Valve className="org.apache.catalina.valves.RemoteAddrValve"
  allow="127.0.0.1,129.70.128.78" deny=""/>
```

erlaubt beispielsweise den Zugriff auf den Manager vom localhost und einem Rechner mit der IP Adresse 129.70.128.78. **Achtung!** Die Manager Applikation sollte NICHT gelöscht werden, Installieren/Starten/Stoppen /etc. von Applikationen ist dann nur über das Dateisystem möglich und nicht zu empfehlen.

4. Beispiel Applikationen:

Die standardmäßigen Beispiel Applikationen in `§CATALINAHOME/webapps` können ohne weiteres deinstalliert (gelöscht) werden (evtl. alle bis auf ROOT!). Unter Umständen macht es auch Sinn die ROOT Applikation zu löschen bzw. anzupassen (ohne die Beispielapplikationen sind die meisten Links der Standard ROOT Seite „tote“ Links

5. JDBC – Java Data Base Connection[1]

Greifen Applikationen auf eine Datenbank via JDBC zu, dann macht es aus Performancegründen Sinn eine globale Verbindung für alle Applikationen zur Verfügung zu stellen (vgl. [1] – Seite 53). Dieses geschieht durch ein Eintrag im Context bzw. DefaultContext unter dem jeweiligen Host Element in der Datei `$CATALINAHOME/conf/server.xml`.

```
<DefaultContext>
  <Resource name="jdbc/postgres" type="javax.sql.DataSource"/>
  <ResourceParams name="jdbc/postgres">
    <parameter>
      <name>maxWait</name>
      <value>5000</value>
    </parameter>
    <parameter>
      <name>maxActive</name>
      <value>100</value>
    </parameter>
    <parameter>
      <name>password</name>
      <value>XXX</value>
    </parameter>
    <parameter>
      <name>url</name>
      <value>jdbc:postgresql://XXX.XXX.XXX.de:PORT/DATABASE</value>
    </parameter>
    <parameter>
      <name>driverClassName</name>
      <value>org.postgresql.Driver</value>
    </parameter>
    <parameter>
      <name>maxIdle</name>
      <value>2</value>
    </parameter>
    <parameter>
      <name>username</name>
      <value>XXXXXX</value>
    </parameter>
  </ResourceParams>
</DefaultContext>
```

Der wesentliche Unterschied zwischen Context und DefaultContext ist, dass der Default Context für alle Contexte gilt, sofern er nicht explizit durch einen Context überschrieben wird.

6. Homepage [3]

Der TOMCAT bietet von Haus aus eine Homepage (s.o.). Um die Homepage des Tomcats zu ändern, muss in `$CATALINAHOME/webapps/ROOT/WEBINF/web.xml` unterhalb des `<web-app ..>` Tags folgendermaßen die Einstiegsseite eingestellt werden:

```
<welcome-file-list>
  <welcome-file>index.html</welcome-file>
</welcome-file-list>
```

Die `index.html` wird nun in `$CATALINAHOME/webapps/ROOT/` abgelegt.

7. 404/500/... Responsecodes [3]

Der TOMCAT liefert von Haus aus eine sehr ausführliche und leider nicht XHTML konforme Antwort auf Anfragen, die zu Fehlern führen kann, wenn zum Beispiel ein SOAP Toolkit diese mit einem XML Parser parsen möchte. Um stattdessen eine (statische) XHTML Seite zurückzugeben muss in `$CATALINAHOME/conf/web.xml` unterhalb des `<web-app .>` Tags folgendes eingeführt werden:

```
<error-page>
  <error-code>500</error-code>
  <location>/500.html</location>
</error-page>
<error-page>
  <error-code>404</error-code>
  <location>/404.html</location>
</error-page>
```

Die entsprechenden Dateien (`404.html` und `500.html`) wird nun in `$CATALINAHOME/webapps/ROOT/` abgelegt (Hier beispielhaft für die Errorcodes 404 und 500, aber andere sind ähnlich zu behandeln)

8. Konfiguration der Java SandBox[3,4,5]

Java Programme können dynamisch andere Java Klassen von einer Vielzahl von vertrauenswürdigen und NICHT vertrauenswürdigen Stellen nachladen. Der Jakarta TOMCAT zählt zu dieser Klasse von Programmen; eine Webservice Applikation basierend auf Axis wird zur Laufzeit geladen und ausgeführt. Um zu verhindern, dass das Host System unabsichtlich (durch schlampige Programmierung) oder absichtlich (durch böswilligen Code) Schaden nimmt, kann man Java Programme in einer eingeschränkten Umgebung, der so genannten SandBox starten. Durch den in Java 1.2 eingeführten Sicherheitsmanager besteht die Möglichkeit Java Programme sehr fein granuliert mit Zugriffsrechten zu versehen. Im folgenden soll ein Konfiguration anhand einer vereinfachte Darstellung der BiBiServ Webservice Server verdeutlicht werden. Die folgende Tabelle ist eine schematische Darstellung des BiBiServ Webservice Server Umgebung, die im wesentlichen aus dem Standard Tomcat (`/vol/tomcat` im folgenden `$CATALINAHOME`) und einer SunGridEngine(`/vol/sge`) Instanz mit den entsprechenden Unterverzeichnissen besteht.

<i>Verzeichnis</i>	<i>Beschreibung</i>
<code>CATALINAHOME/bin</code>	TOMCAT Binaries
<code>\$CATALINAHOME/conf</code>	TOMCAT Konfiguration
<code>\$CATALINAHOME/logs</code>	TOMCAT Logdateien
<code>\$CATALINAHOME/server</code>	TOMCAT Server Libs
<code>\$CATALINAHOME/temp</code>	TOMCAT Spoolverzeichnis
<code>\$CATALINAHOME/common</code>	TOMCAT Libs
<code>\$CATALINAHOME/shared</code>	TOMCAT shared Application Libs
<code>\$CATALINAHOME/webapps</code>	TOMCAT WebApplikation Verzeichnis
<code>/vol/sge/bin</code>	SGE Binaries

Der TOMCAT bzw. die Webservice Applikation brauchen nun unterschiedliche Zugriffrechte um zum Beispiel Programme auszuführen, Log Dateien zu schreiben, Konfigurationsdateien zu lesen usw. Die folgende Tabelle ist eine (unvollständige) Aufstellung der benötigten Zugriffe auf die Systemressourcen.

<i>Codebasis</i>	<i>Zugriff auf</i>	<i>Einschränkung</i>
\$CATALINAHOME /bin/*	alles	keine
\$CATALINAHOME /common/*	alles	keine
\$CATALINAHOME /server/*	alles	keine
\$CATALINAHOME /webapps/*	\$CATALINAHOME/temp/* (Spooldaten)	read,write
	\$CATALINAHOME/webapps/*	read,write
	\$CATALINAHOME/bin/commons-daemon.jar	read
	\$CATALINAHOME/bin/commons-logging.jar	read
	/vol/sge/bin/*	read,execute
	Netzwerk: *:80 (alle Anfragen auf Port 80)	connect, resolve
	Netzwerk: dns.XXX:53 (NameServer)	accept, resolve
\$CATALINAHOME /shared/*	\$CATALINAHOME/temp/* (Spooldaten)	read,write
	\$CATALINAHOME/webapps/*	read,write
	/vol/sge/bin/*	read,execute
	Netzwerk: *:80 (alle Anfragen auf Port 80)	connect, resolve
	Netzwerk: dns.XXX:53 (NameServer)	accept, resolve

Davon ausgehend das der TOMCAT keine „böswilligen“ Absichten hat, dürfen alle direkt zum TOMCAT gehörenden Klassen ohne Einschränkung auf alle Systemressourcen zugreifen (\$CATALINAHOME/(bin|common|server)). Die Webservice Applikationen (\$CATALINAHOME/webapps) hingegen ist nicht vertrauenswürdiger Code der nur auf bestimmte Systemressourcen zugreifen darf. (Spool)Daten können zum Beispiel nur in ein entsprechend ausgewiesenes Spoolverzeichnis (\$CATALINAHOME/temp) schreiben. In der Java Policy Schreibweise sieht das dann exemplarisch für die Webservice Applikation folgendermassen aus:

```
grant codeBase "file:${catalina.home}/webapps/-" {
  permission java.io.FilePermission "$CATALINAHOME/temp/-", "read, write,
delete";
  permission java.net.SocketPermission "*", "connect, resolve";
  permission java.io.FilePermission
"$CATALINAHOME/webapps/-", "read,write,delete";
  permission java.io.FilePermission "$CATALINAHOME/shared/etc/-", "read";
  permission java.io.FilePermission "/vol/sge/bin/-", "execute";
  ... }
```

Zur einfacheren Erstellung solcher Policies gibt das Policytool das mit Sun's SDK

mitgeliefert wird. Dieses GUI des Tool ist allerdings schon ein wenig älter und nicht wirklich intuitiv zu bedienen.

Referenzen:

- [1] TOMCAT „The Definitive Guide“ - J. Brittain und I.F.Darwin – O'REILLY 2003
- [2] BIBIWS – H. Mersch – Diplomarbeit, Universität Bielefeld 2004
- [3] TOMCAT Dokumentation
- [4] Java in a Nutshell 4Aufl– D. Flangan – O'REILLY 2003
- [5] Securing Java - Gary McGraw – Willey 1999 - online vailable for free :
<http://www.securingjava.com/chapter-two/>