Albrecht Weinert

Tutorial

Tomcat —
mit Windows und Active Directory

Stand: 21.12.2009

Albrecht Weinert
Labor für Medien und verteilte Anwendungen (MEVA-Lab)
Fachbereich Informatik der Hochschule Bochum

Tomcat — mit Windows und Active Directory

V01.01, 23.04.2008 13:57:	neu (ersetzt [11] als Nachfolger)
V01.02, 25.04.2008 08:32:	erste abgeschlossene Version
V01.03, 02.05.2008 13:17:	Rollen mit AD deutlicher; 6.0.x-Besonderh.
V01.04, 14.06.2008 13:17:	"PHP on Tomcat"
V01.05, 20.12.2009 16:43:	Aktualisierungen

Version: V1.05
Zuletzt geändert von A. Weinert am 21.12.2009

Hinweis: Wesentliche Listen, Tabellen, Listings, Bilder etc. sind gemeinsam durchnummeriert.

Hinweis: Die URL dieses Dokuments ist
http://www.a-weinert.de/pub/tomcat-win-ad.pdf .
Die dort zu findende Version könnte neuer sein, als die Vorliegende.

Inhalt

1. Zweck, Voraussetzungen

Das Apache-Projekt Tomcat ist die Referenz-Implementierung von Suns WebService-Standard J2EE *1).

Dieses Handbuch beschreibt die Installation von Tomcat als WWW-Server und als Servlet- und JSP-Container auf einem Server in einer Domain mit Windows Server 2003 enterprise edition und natürlich mit Active Directory (AD). Das Vorliegende ist eine Weiterentwicklung von [11] und beschränkt dessen Betrachtung auf Java 6 und Tomcat 6. Für die AD-Integration wird von vornherein eine geeignete Realm-Klasse vorgesehen *2).

Es geht zum Einen um die ersten Schritte „ab Null“ — gerade eines Tomcat-Neulings. Allerdings wollen auch Fachleute schon in dem Teil wertvolle Anregungen gefunden haben. Motiv für die Veröffentlichung war, dass die bei Tomcat und sonst im Web zu findende — meist sehr gute — Dokumentation gerade die scheinbar einfachen Anfangsfragen nicht oder sogar unzutreffend beantwortet. Für die in einer industrieüblichen Windows-Umgebung meist sinnvolle, wenn nicht gar notwendige Integration in Active Directory gilt dies leider in besonderem Maße.

Der Einsatz von Tomcat als "stand alone Webserver" für auch festen Inhalt (static content) wird kontrovers diskutiert *3). Es ist, wie im hier zugrundeliegenden Anwendungsfall eines Firmen- / Abteilungs-WWW-Servers für Informations- und (Prozess-) Steuerdienste, wesentlich öfter sinnvoll, als mancher glauben mag.

1*): Das Einführen jeder der zahlreichen Abkürzungen beim ersten Auftreten im Text stört den Lesefluss dessen, der sie schon kennt — und nützt andernfalls bei erneutem Auftreten weiter unten nichts. Also bitte ggf. im Abkürzungsverzeichnis (Anhang) schauen.

2*): Wer's dennoch mit JNDIRealm versuchen will / muss findet Hinweise in [11].

3*): Es wird oft stillschweigend vorausgesetzt, dass Tomcat mit einem Apache-Web-Server als "front end" eingesetzt wird. Manche glauben, das müsse so sein. Dem ist nicht so.

1.1 Ziel

Das Vorliegende beschreibt die Inbetriebnahme eines

- Apache-Tomcat als "stand alone" Webserver in einem Windows-Umfeld.

Dieser soll

- statischen Content liefern und
- als J2EE-Container für JSP und Servlets eingesetzt werden.

Der letzte Punkt erlaubt die Unterstützung von Geschäfts- und anderen Prozessen (Datenbankanbindung, Domain-Verwaltung, Prozess-BuB etc.). Dabei kann (und sollte) dann auch AJAX, z.B. mit GWT (siehe Tutorial / Tipps [5]), zum Zuge kommen.

Eine wesentliche Bedingung ist, dass das Ganze in einer

- Windows-Umgebung (Windows Server 2003) und in einer
- Windows-Domäne mit Active Directory (AD-Domain)

integriert laufen soll. Hierin liegt übrigens auch der überwiegende Entwicklungs- und Klärungsaufwand für das Vorliegende. "Integriert" heißt u.A. keine eigene Benutzerverwaltung, sondern für die Authentifizierung von Web-Zugriffen und -Diensten eine

- vollkommene Integration (von Tomcat) in Active Directory.

Dies Alles wurde — so wie hier beschrieben — erfolgreich für kritische Webdienste bis hin zu administrativen Eingriffen in eine W2K3-Domain eingesetzt.

```
  12.04.2007  09:22  54.898.268  jdk-6-doc.zip
  07.08.2009  18:21  77.113.112  jdk-6u17-windows-i586.exe

  16.12.2009  16:55  11.960.931  erg.zip
  16.12.2009  16:49     530.478  frame4j.jar   (in erg.zip)

  25.10.2009  11:49  11.657.986  apache-ant-1.7.1-bin.zip
  29.08.2009  07:55 198.520.136  eclipse-jee-galileo-win32.zip

  20.12.2009  16:22   5.767.535  apache-tomcat-6.0.20.exe
..24.02.2008  14:26   5.401.005  apache-tomcat-6.0.16.exe
  16.12.2009  16:49      25.476  catErgWe.jar
```

Liste 1: Bereitzustellende Installationsdateien; Stand Dezember 2009 (http://tomcat.apache.org/ ,. http://www.frame4j.de/download etc.).

1.2 Voraussetzungen

a.) Sie haben alle Installationsdateien zu (genau) einer Tomcat-Version, ggf. auch der Java-Basis, dem Zubehör etc. aus der Liste 1. Das Vorliegende geht von (mindestens) JDK1.6_10, Tomcat6.0.14 oder .16 aus, sowie ggf. von mindestens Eclipse3, Ant1.6.5, XSLT 2.0 (saxon9.jar) etc. Nehmen Sie nichts Älteres mehr für Neueinstieg sondern die in Liste 1 genannten Versionen oder jüngere.

b.) Sie haben einen geeigneten Server für die Installation von Tomcat und Ihrer Web-Services, vorzugsweise mit Windows Server 2003 (W2K3) und RAID.
Der Server wird in den folgenden Beispielen PD321S genannt.

Eine weitere Installation auf einer W2K3-Workstation — i.A. Ihrem Entwicklungsplatz — ist sinnvoll. Vieles, nicht Alles, lässt sich auch mit "localhost" klären.
Und "meine Lösung / Konfiguration läuft auf genau einem Tomcat-Server" beweist fast nichts.

b.) Dieser Server (PD321S) ist im Netzwerk für alle beabsichtigten (Abteilung, Firma, weltweit) Client-Rechner mit http(s) zugänglich. Er hat genügend Plattenplatz für Alles, was Sie da via HTTP(S) als statischen Inhalt oder als Webservice anbieten wollen. Für die Integration mit AD hat der (Tomcat-) Server Zugriff auf mindestens einem Domain-Controller (besser auf 2). In einem professionellen Umfeld wird der Tomcat-Server selbst (als member computer) der betreffenden Domäne angehören.

c.) Sie haben administrativen Zugriff auf den (Tomcat-) Server; es genügt remote-Zugriff (Einstellungen -> Systemsteuerung -> System -> remote).

d.) Sie haben ein JDK 1.6.0 oder höher auf diesem Server installiert. Ein JRE allein genügt i.A. nicht. Hinweise zur Installation Java-Grundlage finden Sie auch im nächsten Kapitel 1.3.

e.) Diese JDK-Installation ist mit Allem ergänzt, was Sie für Ihre Anwendungen, Servlets etc. benötigen .
Gemeint sind hier Dinge wie das Framework Frame4J, JDBC-Treiber, COMM-Extensions, Java-mail, XSLT2 etc. pp., die sinnvollerweise als "installed extensions" ergänzt wurden; siehe hierzu u.A. auch Kapitel 1.3 und
http://www.frame4j.de
und
http://www.a-weinert.de/pub/java-install.txt [3]).
Das genannte Framework Frame4J (frame4j.jar November 2009 oder jünger) bringt — neben vielen anderen Segnungen — auch die Basis für eine vernünftige, letztlich einzig wirklich lauffähige Intergragtion von Active Directory
Zu solchen sinnvollen Ergänzungen zählt auch XSLT 2.0 (statt nur 1.0), das man beispielsweise durch saxon9.jar nachinstallieren kann (im o.g. erg.zip i.A. drin).

f.) Ihr Suchpfad weist auch auf die Werkzeuge dieses korrekt installierten und nach Ihren Bedürfnissen ergänzten JDK (siehe e. und folgendes Kapitel 1.3). Systemumgebungsvariablen wie ANT_HOME, JAVA_HOME, TMP, TEMP, und ggf. SVNEXE etc. sind korrekt gesetzt.

g.) Für die (optionale) AD-Integration haben Sie administrativen Zugriff auf die Benutzerverwaltung der betreffenden Domäne (in den Beispielen ist die Domain FB3-MEVA.fh-bochum.de) oder Sie können hierzu (geringfügige) Dienste eines freundlichen Domain-Administrators beanspruchen.

h.) Nicht unbedingte Voraussetzung, aber empfehlenswert: Sie verwenden für alle Quellen (.java, .html. .xml, .properties etc. pp.) des betreffenden Webauftritts bzw. der Web-Dienste eine hierfür geeignete Entwicklungsumgebung (z.B. Eclipse 3.5). Sie stellen alle Quell- und Hilfsdateien unter eine Versionsverwaltung (z.B. SVN [16]). Sie haben Erfahrungen mit dem automatischen script-gesteuerten Ausliefern (deploy, bat, cmd, ant) von Java- oder Web-Projekten.
Tipp: Es hat sich als günstig erwiesen, alle (wesentlichen, .xml-) Konfigurations-dateien eines Tomcat-Webservers mit unter Versionsverwaltung zu stellen.

Anmerkung zu a): Die Tomcat-Installationsdateien bestimmen natürlich die installierte Tomcat-Server-Version. Die Wahl ist insofern wichtig, als es zwischen verschiedenen 5er-Versionen und zwischen 5 und 6 durchaus Inkompatibilitäten gibt. Die hier vorausgesetzte Java6- + Tomcat6-Kombination ist derzeit (2008 .. 2010) die sinnvolle Wahl. Tomcat 6.0.14 und 6.0.16 sind bewährt, während bei 6.0.18 und 6.0.20 schon die unveränderte Neuinstallation (unter XP) oft einfach nicht läuft. Hier hilft einfach nur probieren, zumal die erheblichen Probleme oft nur bestimmte Windows-Varianten betreffen. Oft muss man auch auf das unten empfohlene "native" verzichten, damit Tomcat unter Windows läuft.

1.3 Java - Grundinstallation

Eine zeitgemäße und, wenn möglich, gleiche Java-Grundinstallation im Sinne der obigen Punkte d) bis f) auf allen (Tomcat-) Servern und auf allen Entwicklungsplätzen ist sinnvoll. Sie sollte als solide Basis gesetzt und möglichst aufrecht erhalten bzw. gepflegt werden. Siehe die oben angegebenen Verweise.

Da dies (sprich die jeweilige Java-Installation und deren Systemeinbindung) die Basis ist, und Fehler und Inkonsistenz hier schwer zu diagnostizierende Folgefehler haben können, wird über die oben aufgeführten Hinweise hinaus hier ein bewährtes Vorgehen beschrieben.

Die Schritte (auf jedem (künftigen Tomcat-) Server und jedem Test- /Entwicklungsplatz):

0.) Deinstallieren Sie ältere JDKs und JREs
und beseitigen Sie deren nicht mehr benötigte Reste.

1.) Lassen Sie (mit administrativen Rechten)
jdk-6u17-windows-i586.exe
laufen. Nach Zustimmung zu den Lizenzbedingungen ändern Sie den Installationspfad in C:\programme\jdk.
Versionsabhängige Pfadnamen sind nicht empfehlenswert (einfach Mist) und ziehen schon beim vermutlich harmlosen Wechsel zu JDK1.7...einen "Rattenschwanz" von Änderungen nach sich.
Installieren Sie evtl. die Java-Datenbank und ein zusätzliches JRE, das Sie i.A. nur als Browser-plug-in brauchen. (Lesen Sie dazu die Hinweise in http://www.a-weinert.de/pub/java-install.txt [3]) Die JDK-Quellen sollten Sie für Entwicklungsrechner unbedingt mitnehmen; auf Servern sind sie i.A. Nur Platzverbrauch.

2.) Fügen Sie `C:\programme\jdk\bin` Ihrem Suchpfad hinzu.
Ein vernünftiger Suchpfad beginnt so
C:\bat;C:\Programme\util;C:\programme\jdk\bin;C:\WINDOWS\system32;
C:\WINDOWS;C:\WINDOWS\System32\Wbem
bzw. sieht genau so aus.
Vor Allem ist er mit gar Duzenden von Einträgen nicht wesentlich länger.

3.) Setzen Sie auch gleich folgende Umgebungsvariable (System / für Alle)
JAVA_HOME=C:\Programme\jdk
ANT_HOME=C:\Programme\Apache\ant
(letzter natürlich nur, wenn Sie sinvollerweise ant auch installiert haben).

4.) Löschen Sie im Verzeichnis
`C:\WINDOWS\system32\`
die Dateien java.exe und javaw.exe.
Hintergrund: Es sind Kopien von denen die Sie mit C:\programme\jdk\bin eh im Suchpfad haben. Die system32-Kopien sind dann nur schädlich / fehlerträchtig.

5.) Erzeugen Sie (mit md) ein Verzeichnis
`C:\programme\jdk\docs`
und komprimieren es (mit compact). Auf einem reinen Server können Sie auf den Inhalt von diesem docs-Verzeichnis verzichten .... oder ihn als nützlichen Web-Content auch gleich anbieten.

6.) Wechseln Sie (mit cd) in das Verzeichnis
`C:\programme\jdk\`
und dort (genau dort!) packen Sie (mit jar xvf .dat\da\file) die o.a. Dateien
jdk-6-doc.zip und erg.zip
aus.
Als Probe: Von jedem Verzeichnis aus müssen
java ShowPorts
und
java ShowProps
nun was Sinnvolles tun.

Hinweis 1: Beispielhaft genannte Versionen sind ggf. durch jeweils neuere zu ersetzen.

Hinweis 2: Hier verwendete feste Pfadnamen wie C:\Programme\jdk sind Beispiele für sinnvolle und bewährte Vorgehensweisen. Der Kürze und Verständlichkeit des Textes halber werden keine erklärungsbedürftigen Platzhalter, wie %ihrJDKpfad% verwendet oder jedes mal "bzw. Ihr xyz-Pfad" gesagt. Desgleichen werden konsequent Server- und Domainnamen einer dem Vorliegenden zugrundeliegenden erfolgreichen Installation verwendet. Die Anpassung an Ihre Verhältnisse ist jeweils offensichtlich.

Hinweis 3: Für diese und die folgenden Installationsarbeiten genügt (administrativer) remote-Zugriff. Dies gilt auch für Domain-Controller (bei denen eine Tomcat-Installation für administrative Webdienste / Servlets unter gewissen Umständen sinnvoll sein kann). Alle Installationsdateien können auf Netzlaufwerken (Freigaben) liegen.

Kleinigkeit: Eine sowieso fällige Mozilla-Neuinstallation sollte vor der Java-Installation erfolgen.

2. Server - Grundinstallation

Zum Installieren des Tomcat-J2EE-Servers nehmen Sie nur das dankenswerterweise vorbereitete "Windows executable" — hier also beispielsweise apache-tomcat-6.0.20.exe

Hinweis zu "nur": Bei manchen Tomcat-Versionen, wie beispielsweise 5.5.12, gibt es neben dem "Windows executable" auch fertige Windows-Installationen, die man einfach am Zielort auspacken soll. Finger weg! Diese "Fertiginstallationen" funktionieren erfahrungsgemäß nicht. Ebenso wenig funktioniert i.A. das Kopieren einer Tomcat-Installation von einem Rechner zu einem anderen. Mühsam angepasste .xml-Konfigurationsdateien können hingegen oft unverändert weiterkopiert werden. Das Server-Zertifikat (siehe unten) muss natürlich dem Zielrechner angepasst bzw. ausgewechselt werden.

Lassen Sie mit administrativen Rechten

```
20.12.2009  16:22  5.767.535    apache-tomcat-6.0.20.exe
```

auf dem Ziel-Server (hier PD321S) laufen.

Stimmen Sie den Lizenzbedingungen zu, setzen Sie alle Häkchen für eine vollständige Installation — core, service, native (siehe Warnung oben). Ändern Sie das Zielverzeichnis in

```
C:\Programme\Apache\Tomcat
```

Verlangen Sie gleich Port 80, und geben Sie sich ein Administratorpasswort (merken!).

Geben Sie das jre Ihres JDK (siehe oben unter e.) als Laufzeitumgebung an, also

C:\Programme\jdk\jre

Verbieten Sie den sofortigen Start (der richtete zumindest in früheren Versionen ohne fertige Konfiguration, s.u., nur Schaden an) und schließen Sie den Installer. Das Lesen von readme, docs (C:\Programme\Apache\Tomcat\webapps\docs) etc. können Sie auch später machen. Also "finish" ohne Zusatztaten.
Bevor man's vergisst, sollten sie gleich die Datei

```
20.12.2009  16:55   25.476    catErgWe.jar
```

(oder eine neuere) nach

```
C:\Programme\Apache\Tomcat\lib
```

kopieren. Damit ist die Grundinstallation durch.

Anmerkung zum Tomcat-Administrator-Passwort — Achtung Falle:
Dieses während der Installation vergebene Passwort ist völlig losgelöst von

Ihren Windows- bzw. Domain-Konten. Dieses Tomcat-Name-Passwort-Paar landet, so wie von Ihnen vergeben, im Klartext in einer .xml-Datei. Diese Sonderbehandlung jeder Anwendung mit Klartextspeicherung von Einzelkonteninformationen für jede Anwendung ist eine Linux-Gewohnheit des Programms. Aus Sicht eines Windows-Administrators ist dies so unfassbar, dass man Kollegen gegenüber schon mal den "körperlichen Beweis" antreten muss. Also hier keinesfalls echte (Domain-, AD-) Administrator- Konteninformationen wiederholen. Mit der AD-Integration — durch obige Kopie nach lib vorbereitet — werden diese "Privateinstellungen" zum Glück eh irrelevant.

Anmerkung zur Wahl von Port 80:
Für das eingangs genannte Ziel eines standalone Web-Servers und J2EE-Containers sind i.A. nur Port 80 (für http) und Port 443 (für https) sinnvoll.
Trotz der o.g. Installationsabfrage und der Antwort 80 bleiben in Konfigurationen (server.xml) und Dokumentationen vielfach die Ports 8080 bzw. 8443 eingetragen. Dies ist hier nicht nur lästig, sondern recht fehlerträchtig.

Wenn Sie der (Neu-) Installation letztlich funktionierende Konfigurationsdateien aus einer Entwicklungsumgebung / einem Versionsverwaltungssystem "unterschieben", ist eh alles wieder in Ordnung. Jedenfalls sollten Sie alle xml-Konfigurationsdateien nach 8080 bzw. 8443 durchsuchen und das Vorkommen ggf. durch 80 bzw. 443 ersetzen.

Hinweis: Wenn Sie nun (prinzipiell sinnvollerweise) mit Windows-Bordmitteln nach Vorkommen von "8080" und "8443" in der Tomcat-Installation suchen, werden Sie wegen eines Explorer-Bugs oft nicht fündig. Im Allgemeinen genügt Folgendes als Reparatur, nach der dann endlich auch Fundstellen in .xml-Dateien gezeigt werden:

```
HKLM\SYSTEM\CurrentControlSet\Control\ContentIndex
    FilterFilesWithUnknownExtensions=1
```

Starten Sie unter

```
Start->Programme->Apache...
```

den Tomcat-Monitor. Falls dies nicht im Startmenu erreichbar ist geben Sie in der Shell

```
C:\Programme\Apache\Tomcat\bin\tomcat6w.exe //MS//Tomcat6
```

ein.

Gehen Sie mit der rechten Maustaste auf das nun erscheinende neue Bodenleistensymbol — ein Kreis mit rotem Punkt und einer Zipfelmütze nach oben links — und befehlen Sie "Start Service".

Bevor Sie irgendwas Anderes tun, schauen Sie auf die (nun entstandenen) Dateien in

```
C:\Programme\Apache\Tomcat\logs
```

Irgendeine Datei > 3 KB deutet auf ein Startproblem (dann in diese reingucken).

Ist dies soweit geglückt, sollten Sie von jedem Rechner in Ihrem Netzwerk aus mit

```
http://pd321s/
```

die vorgefertigte nette Begrüßungsseite — mit einem Bild eines Löwleins — Ihres Tomcat-Servers sehen und von da aus weiternavigieren können. Das gleiche muss — am Rechner selbst (direkt oder remote) eingeloggt — auch mit

```
http://localhost/
```

funktionieren. Insbesondere gelangen Sie so in die wichtige, wenn auch nicht immer korrekte oder vollständige, Tomcat-Dokumentation. Desgleichen müssen auch alle Links im Kasten "Administration" der Tomcat-Begrüßungsseite funktionieren.

Anmerkung zu Log-Dateien:

Eigentlich wäre es sinnvoll, nach der erfolgreichen Installation mit dem o.g. Tomcat-Monitor ins Menu "Configure Tomcat" zu gehen, und dort Temp- und Log-Verzeichnisse außerhalb der Programm-Installation, z.B. auf

```
D:\logFiles\Tomcat\logs
```

zu setzen. Schließlich ist es gute Praxis, mehr statische Programminstallationen (C:\Programme\...) von den sehr dynamischen Cache-und Log-Informationen zu trennen. Wegen der sehr unterschiedlichen Laufwerks-, RAID- und Backup-Anforderungen, sind hierfür sogar unterschiedliche Laufwerke und / oder Partitionen sinnvoll.

Wer nun diese sinnvolle Trennung mit dem genannten Log-Menu-Punkt auch für Tomcat einrichtet, wird enttäuscht. Da aber 20% der erzeugten Log-Files den "Umzug" nicht mitmachen (Bug!), lässt man diesen besser gleich bleiben und hat so alle Logs beieinander in

```
C:\Programme\Apache\Tomcat\logs\ .
```

Bei der Gelegenheit: Falls Sie in einer Log-Dateien Beschwerden der Art

```
"The Apache Tomcat Native library which allows optimal performance
in production environments was not found on the java.library.path"
```

vorfinden, kommentieren dies aus

```
<Listener className="org.apache.catalina.core.AprLifecycleListener" />
```

in der Datei

```
C:\Programme\Apache\Tomcat\conf\server.xml
```

Stopp, Start und die (weltweit verwirrende) Fehlermeldung sollte nicht mehr auftreten.

```
:stopLok
@Echo Lokalen Tomcat stoppen
@echo.
net stop Tomcat6
@if /I %1X==-tomstoppX       goto :ende

:delLokLog
@Echo Lokale Tomcat-Logs löschen
@echo.
java Del "C:\Programme\Apache\Tomcat\logs/+.+"

:startLok
@Echo Lokalen Tomcat starten
@echo.
net start Tomcat6
@goto :ende
```

Listing 2: Script-Auszug, lokalen Tomcat stoppen, Logs weg und starten (u.A. für Tests).

Anmerkung zu Konfigurationsänderungen:

Die meisten Konfigurationsänderungen, wie die eben genannte über Programme oder auch über das direkte Ändern oder Hinzufügen entsprechender .xml-Dateien, erfordern zum wirksam Werden das Stoppen und wieder Starten des Tomcat-Servers. Nicht nur weil der Vorgang Dutzende von Sekunden dauern kann, muss er bei einem produktiv laufenden Server vorsichtig und sparsam eingesetzt werden. Grundlegend Neues, Unverstandenes oder schlecht Dokumentiertes probiert man ja eh besser erst mal an einer jeweils weitgehend gleichen Testinstallation aus.

Es hat sich bei potentiell problembehafteten Änderungen als hilfreich erwiesen, im Stopp-Zustand das oben erwähnte Log-Verzeichnis komplett zu leeren (Listing 2). Dann bekommt man einen besseren Überblick über (nur) die aktuellen Probleme und Ereignisse.

Das Werkzeug "Configure Tomcat" bietet auch die Einstellung der Server-Startart an, also "Manual", "Automatic" oder "Disabled". Für produktive Server ist "Automatic" sinnvoll, für Testinstallationen hingegen i.A. "Manual". Die Veränderung der Startart mit diesem Werkzeug funktioniert nicht immer zuverlässig; am besten geht man hierzu gleich — wie gewohnt und bewährt — in die Windows-Diensteverwaltung.

Hinweis: Wenn Tomcat, so als Dienst gestartet, Probleme bei Dateizugriffen, auch auf seine eigenen .xml-Files, zeigen sollte, könnten Zugriffsrechte für SYSTEM fehlen.

3. Fester Inhalt — static content

Obgleich Tomcat als J2EE-Container zu "Besserem" berufen ist, kann er als Web-Server auch ganz gut festen Inhalt, sprich vorgefertigte .html- und .xml-Seiten, Applets, Stylesheets und Bilder, also so genannten "static content", liefern. Letztlich wird für diese relativ einfache Aufgabe ein vorgefertigtes Servlet bemüht. Für ein paar Tausend solcher (zusätzlicher) statischer Dateien braucht man jedenfalls keinen (zusätzlichen) Apache oder gar den MS-IIS bemühen. Nur wie es mit Tomcat geht, ist nicht so recht dokumentiert.

Beispielhafter Anwendungsfall:

Auf dem Server PD321S liegt auf dessen D:-Platte ab dem Verzeichnis

```
D:\www\fb3
```

"abwärts" ein Abbild eines (statischen) WWW-Auftritts einer FB3 genannten Abteilung. Der Umfang ist mit etwa 8.150 Dateien in 501 Unterverzeichnissen und 12,6 GByte allerdings überschaubar.

Der Server PD321S soll diese Inhalte unter

```
http://pd321s/fb3-etc/
```

und tiefer liefern. "Tiefer" heißt, dass etwas wie

```
http://pd321s/fb3-etc/meva-lab/virt-lab/sorting-demo.html
```

zum Beispiel (mit allen Applets und drumherum) natürlich auch funktioniert.

Dies alles erreicht man mit einer Datei

`C:\Programme\Apache\Tomcat\conf\Catalina\localhost\fb3-etc.xml`

auf dem (Tomcat-) Server mit dem (Mindest-) Inhalt gemäß Listing 3.

```
<Context docBase="D:\www\fb3" />
```

Listing 3: Despriptor in der Tomcat-Konfiguration für statischen Content.

Weitere Bereiche ("Contexte") lassen sich u.A. nach dem gleichen Schema, also mit jeweils einer solchen Mini-.xml-Datei, ergänzen. Das Verzeichnis ...conf\Catalina\localhost müssen Sie fehlenden Falls erzeugen.

Dies gilt auch für Kurz-URLs für den Einstieg in (und nur in) tiefer verschachtelte Unterbereiche mit etwas wie

`docBase="D:\www\fb3-etc\labor\vorhaben\projekt"`

Die Browser-Darstellung solcher Unterbereiche funktioniert nur, falls diese bezüglich relativer Links "self contained" sind, da relative Links nach "weiter oben" (z.B. für Bilder, Stylesheets) nicht gehen. Man erreicht also nur "docBase + abwärts". Dies ist natürlich kein Mangel, sondern eine wesentliche Sicherheitseigenschaft.

3.1 Directory Listings

Zu oft wünschenswertem statischem Inhalt zählt oft auch eine jeweils aktuell generierte Verzeichnisliste, wie sie beispielsweise Bild 4 zeigt.

Bild 4: Directory Listing (mit geändertem DefaultServlet + XSL-Transformator).

Diese Zuordnung zu statisch ist eigentlich nicht ganz richtig, wenn das Listing bei jeder Anfrage aus sich evtl. veränderten Verzeichnisinhalten neu (dynamisch) generiert wird statisch sind hier die Verzeichnisse und die darin liegenden Dateien. Tomcat's für die Lieferung von statischem content zuständiges DefaultServlet erledigt diese Aufgabe auch, falls man es ihm gemäß Listing 5 erlaubt.

Eine solche Aufforderung bzw. Lizenz zum Verzeichnis auflisten (gemäß Listing 5 in ...\conf\web.xml) gilt pauschal für alle Verzeichnisse und über sämtliche à la Listing 3 definierten "Contexte". Gegen ein solches Auflisten von Dateien und Unterverzeichnissen beim Anfordern (http, https) eines Verzeichnisses statt einer Seite, bestehen teilweise berechtigte Sicherheitsbedenken. Im Allgemeinen möchte man ein solches Verhalten nur gezielt für bestimmten Bereiche (Pinwand, lockere Dokumentsammlung, Entwicklung) gezielt haben und sonst nicht. Die Konfiguration des DefaultServlets (Listing 5) gibt solche Feineinstellungen nicht her.
Bei der, wenn das feature irgendwo gewünscht wird, notwendigen Freigabe müssen und können gegebenenfalls diese Sicherheitsfragen auf zwei Weisen geregelt werden:

a) Diese Listenzugriffe Zugriffe auf Verzeichnisse lassen sich mit den in späteren Kapiteln beschriebenen Verfahren feingranular bezüglich geforderter Authentifizierung und gesicherter Datenübertragung absichern. Dies muss für kritische Verzeichnisse dann halt auch (sowieso) gemacht werden.

```
  <servlet>
     <servlet-name>default</servlet-name>
<servlet-class>de.a_weinert.realm.DefaultServlet</servlet-class>
     <init-param> <param-name>debug</param-name>
                 <param-value>90</param-value>  </init-param>
     <init-param>  <param-name>listings</param-name>
                  <param-value>true</param-value> </init-param>
     <init-param>  <param-name>globalXsltFile</param-name>
      <param-value>D:\www\serv-intra\meva-dir-li.xsl</param-value>
     </init-param>
     <init-param> <param-name>readmeFile</param-name>
                  <param-value>readme.htm</param-value>
     </init-param>
     <load-on-startup>1</load-on-startup>
  </servlet>
```

Listing 5: Änderungen in ...\conf\web.xml für directory listings.
(Die Änderung des Servlets setzt `catErgWe.jar` vorraus; s.u..)

b) Verzeichnisse, in denen sich eine der für Tomcat konfigurierten (siehe Listing 6) Verzeichnisstartdateien befindet, werden nicht gelistet. Das DefaultServlet liefert dann die erste (üblicherweise index.html) aus der Liste gefundene Datei oder java server page aus und listet das Verzeichnis trotz ggf. pauschaler Erlaubnis (gemäß Listing 5) nicht auf.

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

Listing 6: Ausschnitt aus ...\conf\web.xml für Verzeichnisstartseiten.

Trivialerweise folgen aus b) die einfachen Regeln:

1.) In Verzeichnissen, die gelistet werden sollen, dürfen Dateien aus Listing 6 nicht enthalten sein. Eine Ergänzungsdatei zur Directory-Listingdarstellung, das readme-File aus Listing 5, muss ggf. anders heißen.

2.) In Verzeichnissen, deren Auflistung unerwünscht ist, muss einfach eine der in Listing 6 aufgeführten Dateien stehen.

Punkt 2 kann per Dienstanweisung oder auch werkzeuggestützt durch Platzieren einer Sperrdatei (des Inhalts "Sie haben sich verlaufen") erreicht werden. Da Verzeichnisse in öffentlichen Webauftritten, in denen keine solche Datei befindet, eh meist sinnlos sind, kommt der Fall ja auch kaum vor. Das "Problem" ist also klein und beherrschbar.

Ein ganz anderer Punkt, ist, dass unter vielen Umständen das vom DefaultServlet selbst gelieferte directory listing nicht passt oder genügt. Möglicherweise störende Mängel sind:

- Die Gestaltung mit Tomcat-Reklame passt nicht zum übrigen Auftritt.
- Sie ist von mäßiger Schönheit.
- Sie bietet zwar Links in Unterverzeichnisse, von dort aber dann kein Link zurück zum dann ja vorhandenen Elternverzeichnis.
- Das DefaultServlet löst keine Probleme mit Umlauten in Dateinamen, die gerade in Aushang- und Notizverzeichnissen unkontrollierbar vorkommen mögen.
- Es generiert solche Probleme eher noch.

Tomcat's DefaultServlet generiert die Verzeichnisinformation als XML und liefert deren html-Darstellung anschließend durch eine XSL-Transformation, (zumindest im Prinzip / ursprünglich mal). Ein Teil der o.a. Probleme / Wünsche kann man durch eine eigene XSLT-Transformation erledigen. Im Listing 5 (Seite 13) ist bereits dargestellt, wie man dem DefaultServlet eine solche eigene Transformation "unterschiebt". XSLT 2 (saxon9.jar) ist schon allein hierfür recht wünschenswert (vgl. "Java-Basis oben).

Eine solche Transformation, die zu einer in Bild 4 gezeigten Darstellung führt und die aufgeführten Probleme angeht, finden Sie als Anregung im Anhang.

Ein Teil der Probleme lässt sich nur durch (vergleichsweise einfache) Änderung des DefaultServlets erreichen. Jede andere Lösung leidet u.A. darunter, dass das vom DefaultServlet gelieferte XML schlecht und undokumentiert ist.

Schlecht ist das XML in dem Sinne, dass es sehr wenig Information liefert und diese dann auch nicht (XML-gerecht) als Daten sondern als amerikanisierte Textdarstellung.

"`4.4 kb` " und "`Fri, 21 Dec` .."

steht so und nur so in der XML-Verzeichnisinformation. Sie sind nicht das möglicherweise als Darstellung gewollte Ergebnis der XSL-Transformation! Das ist nun schlicht XML-widrig.

Und "undokumentiert" meint, dass man das wenige Nützliche, was sonst noch drinsteht durch Lesen der .java-Quelle des DefaultServlets erfährt und nicht aus seiner teilweise sogar irreführenden Dokumentation. Dankenswerterweise sind die Quellen öffentlich.

Diese Mängel sind in Bild 4 nicht mehr erkennbar, da hier ein leicht modifiziertes DefaultServlet eingesetzt wurde:

de.a_weinert.realm.DefaultServlet anstelle von
org.apache.catalina.servlets.DefaultServlet

Die leichten Modifikationen beziehen sich nur auf die Verbesserung bzw. Ergänzung der xml-Verzeichnisliste (und auf das Vermeiden eines Bugs / Crashes beim directory listing). Wenn Sie die weiter oben aufgeführten Basis-Installationen und Ergänzungen haben seht Ihnen dies andere DefaultServlet ohne weiteres (und konfliktfrei, da in einem andern Paket) zur Verfügung.
(in catErgWe.jar zu finden Sie unter http://www.frame4j.de/downloads)

3.2 Statische Startseite

Im Grundzustand liefert der http-Zugriff auf den Tomcat-Server, mit

`http://pd321s`

im Beispiel, die niedliche vorgefertigten Startseite (mit Löwlein oben links). Sie beruht auf

```
C:\Programme\Apache\Tomcat\webapps\ROOT\index.jsp
```

deren einfache textuelle Änderung übrigens nichts zu bewirken scheint.

Der Hauptmangel des WWW-Servers im jetzigen Zustand ist, dass man die Links zu weiterem ergänzten Inhalt, wie das "fb3" im obigen Beispiel des Listing 3 (Seite 11) "auswendig" wissen muss. Mit

```
http://pd321s
```

sollte also eine (andere) Startseite geliefert werden, welche direkt oder indirekt Alles anbietet, was man mit diesem Web-Server (öffentlich) zeigen möchte.

Der einfachste Weg hierzu ist das Bereitstellen einer zusätzlichen Datei

```
C:\Programme\Apache\Tomcat\webapps\ROOT\index.html
```

(also direkt neben der oben erwähnten index.jsp) mit etwa folgendem Inhalt:

```
<html><head>
   <title> PD321S – Einfachste Startseite     </title>
</head><body>
   <a href="index.jsp">tomcat-start-page<br /></a>
   <a href="fb3">FB3 - Bereich (auf PD321S)<br /></a>
</body></html>
```

Listing 7: Minimale statische Startseite.

Diese Datei wirkt nun unter

```
http://pd321s
```

als Startseite. Das extrem simple Beispiel bietet nun die Standard-Tomcat-Startseite und den beispielhaften zusätzlichen Content fb3 als Links an. Nun bleibt natürlich die Aufgabe, diese obige (Primitiv-) Startseite durch eine beliebig schön gestaltete zu ersetzen — und mit Links auf jeweils weitere bereitgestellte Inhalte und Dienste zu ergänzen.

Da eine „schöne“ Gestaltung dieser Seite an dieser isolierten Stelle (ohne die einfachen relativen Links zu eigenen CSS, Bildern etc.) schwierig sein kann, bietet sich hier auch eine Umleitungsseite zum „Einstiegsindex“ an.

```
<html><head>
   <meta http-equiv="refresh" content="1;
             URL=wichtel-intra/index.html" />
<meta name="robots" content="noindex,follow">
</head><body> ..... </body></html>
```

Listing 8: Umleitende Startseite.

Hinweis: Kapitel 4 ist absichtlich leer.

5. HTTPS — SSL und Nutzerauthentifizierung

5.1 Vorbemerkungen

Die Authentifizierung von Nutzern, beispielsweise durch Abfrage von Name, Passwort, PIN u.Ä., vor der Lieferung vertraulicher Inhalte einerseits und andererseits die gesicherte, verschlüsselte Übertragung von Informationen mit einen entsprechenden Protokoll (https z.B.) über ein Netzwerk haben sachlich nicht direkt miteinander zu tun.

Man kann also Authentifizierung und gesicherte Übertragung in unterschiedlichster Weise einsetzen — und das einzeln, zusammen oder auch gar nicht. Tomcat unterstützt dies Alles (wenn auch nicht in jeder Version).

Aber einerseits Authentifizierungen verlangen und dann andererseits vertrauliche Inhalte — und / oder Passworte — ohne gesicherte Verbindung transportieren ist zwar technisch möglich aber sachlich meist sinnlos oder zusätzlich gefährlich. In diesem Sinne ist das Eine die Voraussetzung des Anderen.

5.2 https und ssl, Hintergrundinformation

HTTP über SSL erbringt (mindestens) die Verschlüsselung des Datenverkehrs mit dem Schlüsselpaar (private und public key) des Servers. Falls glaubhaft (zertifiziert) ist, dass der so vom Client verwendete public key auch wirklich dem via URL angesprochenen Server gehört, dann "spricht" man auch garantiert mit diesem (und nicht etwa mit einem betrügerischen "Hochstapler"). HTTPS bietet also (mindestens)

- asymmetrische Datenverschlüsselung

und

- Server-Authentifizierung

Und dazu benötigt man

- ein Schlüsselpaar (private/public key pair)

und

- ein Zertifikat über den public key des Servers.

Clients bekommen nur den public key und das Zertifikat, während der Server seinen private key in einem gesicherten sogenannten "keystore" für sich behält. (Beim komplexen Beginn einer ssl-session werden weitere Schlüssel generiert.)

Wichtige solche Zertifikate werden ihrerseits von allgemein bekannten, meist vertrauenswürdigen und i.A. teuren Zertifizierungsstellen (CA) mit letztlich deren (Wurzel-) Zertifikat unterschrieben. Ordentlich gemacht ist dieser Vorgang mit einer notariellen Beglaubigung vergleichbar.

Einfachere und billigere Varianten sind a) der Betrieb einer eigenen Zertifizierungsstelle (inhouse CA) oder b) gleich die jeweilige Verwendung jeweils eines einzelnen selbstunterschriebenen (standalone Wurzel-) Zertifikats.

Das Problem bei solchem einfachen, auch im Folgenden verwendeten und oft legitimen, Vorgehen ist nicht die Glaubhaftmachung der selbst gemachten Wurzelzertifikate gegenüber den Client-Anwendern. Hierzu kann man so genannte Fingerabdrücke auf Visitenkarten und in Katalogen drucken, auf Datenträgern und auf getrennten Web-Seiten ver-

öffentlichen und vieles mehr. Die Glaubhaftmachung ist also für viele externe Nutzergruppen kein Problem und für interne (in house) Nutzer schon gar nicht.

Die Gefahr bei solchem (dem folgendem) Vorgehen liegt vielmehr darin, dass die Nutzer sich angewöhnen, angebotene Serverzertifikate zu akzeptieren — und dies geschieht aus Bequemlichkeit, in Unkenntnis der Zusammenhänge oder bei Zeitdruck dann doch meist ohne nähere Betrachtung und Vergleich mit getrennt bezogenen Fingerabdrücken. Ein Benutzer mit solchem eingeschliffenem Verhalten könnte doch irgendwann "gesichert" mit einem Betrüger kommunizieren. Dem kann man bei in-hose-Anwendungen wiederum dadurch entgegenwirken, indem man administrativ diese Serverzertifikate in die Browser (Firefox, Mozilla etc.) der Anwendungsrechner einträgt.

5.3 https und ssl für Tomcat, einfach

Mit folgendem Kommando (in einer Zeile; siehe auch unten Listing 9 :

```
keytool -genkey -v -keyalg RSA -alias tomcat -keypass changeit
      -storepass changeit  -dname
 "CN=%COMPUTERNAME%, OU=FB3-MEVA, O=fh-bochum.de, L=Bochum, S=NRW, C=DE"
       -keystore "C:\Programme\.keystore"
                -validity 9999
```

erzeugen Sie ein 1.024-Bit RSA Schlüsselpaar und selbst signiertes Zertifikat (MD5 with RSA) für CN=aktuellerRechner, OU=FB3-MEVA, O=fh-bochum.de, etc. pp. und speichern das Ganze im keystore

```
C:\Programme\.keystore
```

"changeit" ist die Anfangseinstellung für Keystore-Passworte bei Tomcat und Java; und, wie der Name sagt, hat man das vielleicht schon geändert.

Bei der -dname folgenden LDAP-Angabe muss CN der Name des WWW-Servers sein, für den (bzw. dann im Betrieb von dem) das Zertifikat ist. Andernfalls gibt es zusätzliche Authentifizierungsbedenken und Abfragen der Clients bzw. der Browser. Die übrigen Angaben zu Organisationseinheit, Firma, Stadt, Bundesland und Staat machen Sie einfach zutreffend.

Hinweis für diejenigen mit hier allererster LDAP-Begegnung:
Ja das sieht Alles so unästhetisch aus, und es ist oft noch viel uneingängiger.

-keystore gibt die Datei für den sogenannten keystore an.

Mit -validy bestimmt man die Gültigkeitsdauer des Zertifikats in Tagen ab Geburt. Der default-Wert von nur 90 Tagen reicht oft nicht einmal für Testinstallationen. (Oft sind Zertifikate zu kurzlebig und werden auch von Firmen trotz Ablauf weiter genutzt.)

Hinweis: Nur falls was schief geht und Sie das Zertifikat durch erneute Generierung ändern wollen, löschen Sie es vorher mit (dem Einzeiler)

```
keytool -delete -v  -alias tomcat -keypass changeit
 -storepass changeit  -keystore
        "C:\Programme\.keystore"
```

In der Datei

```
C:\Programme\Apache\Tomcat\conf\server.xml
```

setzen Sie nun zusätzlich folgenden Connector-Eintrag ein:

```
   <!-- Define a SSL HTTP/1.1 Connector on port 443, 24.08.2006 08:36  -->
   <Connector port="443" maxHttpHeaderSize="8192"
      maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
      enableLookups="false" disableUploadTimeout="true"
      acceptCount="100" scheme="https" secure="true"
      clientAuth="false" sslProtocol="TLS"
      keystoreFile="C:\Programme\.keystore"
      SSLCertificateFile="C:\Programme\localhost.crt "/>
```

Keystore- und Zertifikat-Files hält man sowieso am besten, wie gerade angedeutet, unter neutralem Namen an neutraler Stelle (und nicht in Tomcat-Verzeichnissen, sind ja ja auch für andere Zwecke da). Zum Umbenennen nach "Linux-üblich .PunktVorne) muss man unter Windows die Kommandozeile bemühen

```
ren C:\Programme\pd328s-keystore  .keystore
```

da der Windows-Explorer keine nur aus Extension bestehende Dateinamen akzeptiert.

```
keytool -genkey -v -alias %1 -keyalg RSA
   -keypass changeit  -storepass changeit       -dname
      "CN=%1, OU=FB3-MEVA, O=fh-bochum.de, L=Bochum, S=NRW, C=DE"
    -keystore  %1-keystore -validity 9999

@if not exist %1.cer goto :export
@echo Die Datei %1.cer existiert bereits
@echo Abort
@goto :end

:export
keytool -export -v -alias %1 -storepass changeit
                  -keystore %1-keystore -file %1.cer

keytool -import -v -alias %1 -keypass changeit  -storepass
          changeit  -keystore truststore -file %1.cer -noprompt
@goto :ende
```

Listing 9: Script zur Schlüssel- / Zertifikaterzeugung (Auszug).

Die Erzeugung der "key- und trust-stores" und der Zertifikate erledigt man natürlich besser an einem zentralen Server in der Domain und holt sich die erwähnten Dateien dann mit Kopieren und Umbenennen von dort. Listing 9 zeigt die Erzeugung aller (oben) benögtigen Dateien für einen Server / Rechner.(namens %1). Über das für Tomcat benötigte hinaus wird ein neues Rechnerzertifikat einen gemeinsamen "Trustsore" eingetragen. Mit dessen jeweils aktueller Version vertrauen sich alle so behandelten Rechner (in einer Domain) gegenseitig.

Zurück zur Tomcat-Konfiguration: Ferner ändern Sie bei allen anderen Connectoren in dieser Datei den (schon, vgl. oben, bei Installation unsinnigen) Eintrag

```
redirectPort="8443"
```

bei jedem Auftreten konsequent in

```
redirectPort="443"
```

um. Die Beschreibung für HTTP auf Port 80 in server.xml sieht danach etwa so aus:

```
<!-- Define a non-SSL HTTP/1.1 Connector on port (80)80 -->
<Connector port="80"    maxHttpHeaderSize="8192"
   maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
   enableLookups="false" redirectPort="443" acceptCount="100"
   connectionTimeout="20000" disableUploadTimeout="true" />
```

Nach diesen Ergänzungen und Änderungen in server.xml — und nach Server-Neustart — können Sie Alles, was bisher mit http (Port 80, default) ging, auch mit https (Port 443, default) tun.

Insbesondere muss nun im Erfolgsfalle

```
https://pd321s
```

genau dasselbe liefern wie vorher schon

```
http://pd321s
```

allerdings erst, nachdem Sie den Browser angewiesen haben, das eben erstellte Zertifikat zumindest für diese Sitzung zu akzeptieren.

5.4 Erzwingen von https (und von Nutzer-Authentifizierung)

Nun hat der Nutzer die Wahl, http oder https zu nehmen. I.A. möchte man für bestimmte (kritische) Teile des WWW-Bereichs die Verwendung von https erzwingen, auch wenn der Client eine Seite dieses Teilbereichs mit (nur) http anfordert. Obgleich man es, wie gesagt technisch nicht muss, will man oft auch gleich -- und wie hier der Kürze halber mit dargestellt -- die Nutzer-Authentifizierung für einen solchen Teilbereich.

Beispielhalber sei alles, was es im dem oben vorgestellten WWW-Bereich fb3 in dessen Unterbereich mil-proj und tiefer gibt, mit diesen Bedingungen zu versehen.

Hierzu werden in der Datei

```
C:\Programme\Apache\Tomcat\conf\web.xml
```

folgende zusätzlichen Einträge (am besten ganz hinten vor </web-app>) hinzugefügt:

```
  <security-constraint>
    <web-resource-collection>
      <web-resource-name>Lab Secret Area</web-resource-name>
      <url-pattern>/mil-proj/*</url-pattern>
    </web-resource-collection>
    <user-data-constraint>
      <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>

    <auth-constraint>
      <role-name>role1</role-name>
    </auth-constraint>
  </security-constraint>

  <login-config>
    <auth-method>BASIC</auth-method>
    <realm-name>Eigentlich nicht so geheim, aber Test.</realm-name>
  </login-config>

 <security-role>
    <role-name>role1</role-name>
  </security-role>
```

Zum Ausprobieren muss url-pattern natürlich auf etwas zutreffen, was in Ihrem Web-Angebot (static content oder Servlet) tatsächlich vorhanden ist.

5.5 User-Authentifizierung für Tomcat, einfachst

Mit der beispielhaften Konfiguration für das Erzwingen von https haben wir auch gleich verlangt, dass nur Inhaber der Rolle "role1" den Bereich sehen dürfen und das die einfache "BASIC" (Browser-) Authentifizierung akzeptiert wird.

```
<realm-name>Eigentlich ...
```

ist einfach derjenige Text, den der Benutzer bei der ggf. Aufforderung, sich (BASIC) mit Name und Passwort zu authentifizieren, evtl. zu sehen bekommt.

Falls die Datei

```
C:\Programme\Apache\Tomcat\conf\tomcat-users.xml
```

etwa wie folgt aussieht, kann sich u.A. der Benutzer "Rolle" mit dem Passwort "role1" als Besitzer der oben verlangten Rolle "role1" ausweisen. "Chef" und "Tiger" werden mit ihrem jeweiligen Passwort als Besitzer derselben Rolle "role1" ebenfalls akzeptiert.

```
<!--  tomcat-users.xml -->
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="tomcat" description="Rolle 0"/>
  <role rolename="role1" description="Testrolle"/>
  <role rolename="manager" description="fast Admin"/>
  <role rolename="admin" description="darf alles"/>

  <user username="Katze" password="tomcat" roles="tomcat"/>
  <user username="Tiger" password="tomcat" roles="tomcat,role1"/>
  <user username="Chef"  password="tomcat"  fullName="Oberchef"
                            roles="admin,manager,role1"/>
  <user username="Rolle" password="role1"   fullName="Testnutzer"
                                          roles="role1"/>
</tomcat-users>
```

Die hier einstiegshalber (und default-mäßig) verwendete Vergabe von Benutzernamen und -passworten (sowie die Rollenzuweisung) in der Tomcat-Konfigurationsdatei

`C:\Programme\Apache\Tomcat\conf\tomcat-users.xml`

ist die wohl (als sog. MemoryRealm) die schnellste aber gleichzeitig die wohl denkbar primitivste und schlechteste Lösung. "Linux-like" zieht diese mal wieder eine zusätzliche, jeweils anwendungsbezogene, Nutzerverwaltung auf. Für einen inhouse WWW-Server in einer Windows Server 2003 Domäne möchte man viel lieber — eigentlich ist es ein selbstverständliches "Muss" — die eh vorhandenen (AD-) Nutzerkonten (zur Authentifizierung) und Nutzergruppen (als Rollen, spricht Träger von Rechtezuweisungen) verwenden.

Siehe dazu später das Kapitel 7 "Active Directory Integration".

6. Ein Servlet

Die bekannteren Applets sind Java-Programme, die clientseitig im Browser laufen. Ein Servlet läuft hingegen auf dem (Web-) Server bzw. J2EE-Container (Tomcat). Es kann unter vielem anderen über einen OutputStream oder einen Writer — also jeweils auch dynamisch generiert — html-Inhalt (oder XML für AJAX z.B.) an den Client liefern.

Das notorische Hello-World-Servlet ist natürlich kein sinnvolles Anwendungsbeispiel. Hat man es aber mal zum Laufen gebracht, so hat man auf einem Server ein Programm, das dort bei jedem zugehörigen Seitenzugriff läuft. Mit diesem Ansatz bringt man (als erfahrener Java-Programmierer) nun jeden Web-Service zustande — von Domain-Administration bis hin zur Maschinensteuerung. Mit AJAX mit GWT ([7]) kann man zudem auf elegante Weise Servlets durch Java- statt JavaScript-Programmierung für die Clients ergänzen.

Eine Java-Server-Page (JSP) soll die Programmierung der dynamischen Generierung von html- oder xml-Inhalt vereinfachen. Sie wird von praktisch allen J2EE-Container-Implementierungen, so auch von Tomcat, spätestens beim ersten Zugriff in ein Servlet übersetzt. JSPs sind damit letztlich eine bequeme und oft sinnvolle Möglichkeit bestimmte, nämlich inhalts- und nicht steuerungsorientierte, Servlets (u.A. mit JSTL) zu erstellen.

6.1 Das erste Servlet

Im Verzeichnis (Beispiel)

```
D:\workNoSave\webAppsDev\
```

möchten wir zunächst nur ein Servlet (HelloWorld als "Web-Service") bereitstellen.

6.1.1 Herstellen

Man übersetzt die (Servlet-) Quelle HelloWorld.java aus dem Anhang A2 Seite 47.
Dies geht direkt mit javac oder aus einen einfachen Eclipse-Java-Projekt.

Beides setzt voraus, dass man hat die

```
23.09.05  14:42        97.701   servlet-api.jar
```

aus dem Verzeichnis

```
C:\Programme\Apache\Tomcat\common\lib\
```

auch als "installed extension" in das jeweils benutzte JDK getan hat.

Dies geschieht durch Kopieren nach

```
C:\Programme\jdk\jre\lib\ext\
```

Wenn man die Web-Service-Erweiterungen von Sun, sprich

```
09.01.06  15:56   29.196.014   jwsdp-1_6-windows-i586.exe
```

ausgepackt bzw. installiert hat, findet man dort unter

```
C:\Programme\Sun\jwsdp-1.6\wsi-sampleapp\lib\
```

eine geringfügig andere Datei als die von Tomcat mitgebrachte, die es wohl auch (oder erst recht) tun müsste:

```
14.06.05  22:23   92.625        servlet-api.jar
```

Insbesondere liefert jwsdp-1_6-windows-i586.exe auch die Dokumentation zu den Sun-Servlet-Paketen unter

```
C:\Programme\Sun\jwsdp-1.6\docs\api
```

6.1.2 Bereitstellen (deploy)

Der erste Schritt entspricht dem Bereitstellen (Kapitel 3) von statischem Inhalt:
Mit einer XML-Datei

```
C:\Programme\Apache\Tomcat\conf\Catalina\localhost\webappsdev.xml
```

mit folgendem Inhalt

```
<?xml version="1.0" encoding="UTF-8"?>
<Context docBase="D:/worknosave/webAppsDev" />
```

veröffentlicht man unter

```
http://pd321s/webAppsDev/
```

(im Beispiel) alles, was dort unter

```
D:\worknosave\webAppsDev
```

zu finden ist. Dorthin platziere man eine Datei

```
D:\worknosave\webAppsDev\index.html
```

mit dem (Mindest-) Inhalt

```
<html><head> <title>Servlet Test Range</title> </head>
<body> <h2>Servlets <strike>--</strike> Nur Testbereich</h2>
 <ul> <li><a href="servlet/HelloWorld">HelloWorld (Servlet ausführen)<br /></a>
      </li></ul>
</body></html>
```

Diese (sehr ausbaufähige) Startseite des neuen Servlet- (Demo-) Bereichs bietet so nichts weiter als ein Link auf das oben übersetzte HelloWorld-Servlet (Quelle im Anhang).

Um dem Tomcat nun noch das übersetzte Servlet selbst bekannt zu machen, erstellt man (im Beispiel) unterhalb

```
D:\worknosave\webAppsDev
```

die in Bild 10 gezeigte Datei- und Verzeichnisstruktur.

Dabei ist web.xml die im Anhang aufgeführte einfache Descriptor-Datei (Anhang A2, Seite 47; zunächst erste Fassung ohne Authentifizierung) und HelloWorld.class ist das (mit javac erstellte) Übersetzungsergebnis der ebenfalls im Anhang aufgeführten Servlet-Quelle HelloWorld.java

Für ein einfaches Demo-Servlet ist das (neben dem Neustart von Tomcat) alles.

```
D:\worknosave\webAppsDev\
                         |-   index.html
                         |
                         +---WEB-INF
                             |-   web.xml
                             |
                             +---classes
                                 |-   HelloWorld.class
```

Bild 10: Die Webservice-Verzeichnisstruktur (konkretisiert anhand der Beispiele).

Hinweis: Viele Tomcat-Versionen erfordern, dass die Startklasse eines Servlets im anonymen Paket liegt. Von diesem Servlet benutzte Klassen dürfen in Paketen liegen und kämen dann in entsprechende Unterverzeichnisse von WEB-INF\classes\, falls sie nicht schon über JARs der Installation zugefügt sind.

Diese Einschränkung auf das anonyme Paket ist ein Bug, eine harte Einschränkung bei professioneller Entwicklung und eine fehlerträchtige Falle.

Hinweis 2: Das im beispielhaften Servlet-Descriptor (Anhang A2, Seite 47) aufgeführte Tag `<load-on-startup>` tut, was der Name sagt. Es ist für das HelloWorld-Servlet funktionell nicht nötig und kann entfallen. Es ist aber ein sehr guter Trick, um Servlet- (Klassen-) Ladeprobleme schon beim Tomcat-Start aufzudecken.

"Schon beim Start" heißt insbesondere "nicht Wochen später bei der ersten Anforderung". Insbesondere aber liefert die ggf. Startfehlermeldung eher brauchbare Hinweise auf die Problemursache in den Logs, während diejenigen aufgrund der dann zu Bruch gegangenen allerersten Anfordern via http(s)s meist verschleiernd ist. Subtilere Fehler, wie eine im jdk des Produktivsystems fehlende jar beispielsweise, deckt man mit den Anforderungsfehlermeldungen wohl eher selten auf (wie verzweifelte postings hinreichend zeigen).

6.2 https und Authentifizierung für ein Servlet

Um für dieses beispielhafte Servlet dieselben oder ähnliche Übertragungs- und Authentifizierungsbedingungen zu setzen wie im obigen Beispiel für einen Teil des statischen Content, ergänzen Sie in der Datei

`D:\worknosave\webAppsDev\WEB-INF\web.xml`

am Ende einfach etwa folgenden weiteren "security-constraint":

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Servlet Secret Area</web-resource-name>
    <url-pattern>/servlet/HelloWorld</url-pattern>
  </web-resource-collection>
  <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
  <auth-constraint>
     <role-name>role1</role-name>
  </auth-constraint>
</security-constraint>
```

(Stopp, Start, Testen)

6.3 Abschließende Anmerkung zum Servlet

Das hier geschilderte "handgemachte" Vorgehen für das Einstiegs-Servlet, die Erstellung der Verzeichnisstruktur und der .xml-Dateien lässt sich sinngemäß leicht auf mehrere (und auch wesentlich komplexere) Servlets erweitern. Natürlich bekommt man so etwas (das

Ein-Servlet-Demo und die Erweiterungen) mit entsprechenden Tools (Eclipse, Sun-Studio, Ant, etc.) i.A. einfach "gemacht" oder zumindest weitestgehend unterstützt. Nur:

Die beliebten Vorführungen

"Als Laie mit einem Klick zum Webservice"

machen ja niemanden schlauer. Der Einsatz von Tools, gerade von Eclipse und gerade in Verbindung mit AJAX und GWT, ist natürlich sinnvoll, ja unabdingbar.

Aber dauerhaft wird man nur erfolgreich sein, falls man durchschaut, was diese Werkzeuge jeweils tun.

Es gilt auch und gerade hier das Grundgesetz des (Software-) Engineering:

"With or without a tool — a fool is still a fool."

7. Active Directory Integration

7.1 Ziele und Hintergründe — Konten, Gruppen, Rollen, Rechte

Wie schon im Kapitel 5.5 angemerkt, ist eine jeweils anwendungsbezogene Verwaltung von Nutzerkonten, -passworten und -rechten, wie sie sich bei Tomcat in der Datei

```
C:\Programme\Apache\Tomcat\conf\tomcat-users.xml
```

manifestiert, ein schlechter Lösungsansatz. In einem (Betriebs-) System oder einer Domain mit **n** Stück solchen Anwendungen führt das letztlich zu etwa **n + 1** Benutzer- und Rechteverwaltungen. So etwas ist fehlerträchtig, ein administrativer Albtraum, verwirrend für die Nutzer und letztlich richtig unsicher. Solche wenig zeitgemäßen Ansätze sind aber durch die Verbreitung von Unix und nun Linux immer noch üblich. Dies rührt daher das Unix / Linux per se keine ausdruckstarke ACL-basierte Rechteverwaltung hat.
Die beliebten neun Steinzeit-Bits sind davon ja Jahrzehnte weg.

Moderne Betriebssysteme wie u.A. Windows NT und dessen Nachfolger Server 2003 kennen Konten für Nutzer (im weitesten Sinne) und Gruppen. Nutzer können beliebig vielen Gruppen angehören und für Gruppen gilt dies (rekursiv) auch. Angehörige einer Gruppe erhalten automatisch die der Gruppe direkt oder indirekt zugewiesenen Rechte.

Durch eine gut entworfene Verwendung von Gruppen ist die Verwaltung von Rechten sehr effektiv und rationell machbar. Der sinnvolle Grundansatz ist es, Gruppen einzusetzen für

- organisatorische Einheiten (Abteilungen, Jahrgangsgruppen bei Auszubildenden u.Ä.)

und für

- Rollen (=Pflichten und die dazu notwendige Rechte; z.B. Betreuung der Drucker im dritten Stockwerk u. dergl.).

Versetzung von Mitarbeitern, Zuweisung oder Wegnahme von Pflichten wird im Idealfall nur durch Änderungen von sinnfälligen Gruppenmitgliedschaften bewerkstelligt. Außer für absolut "persönliche Objekte" werden bei einem solchen Ansatz an persönliche Nutzerkonten nie direkt (ACL-) Rechte zugewiesen.

AD- / Domain-Tipp: Das Mittel "Gruppe in Gruppe" ist eine sinnvolle organisatorische Lösung (Mitgliedschaft ist transitiv). In mehr als zwei solchen (sinnvollen) Indirektionsstufen eingesetzt wird es aber kritisch für Überblick (und Performance).

An jedem Objekt (Datei, Verzeichnis, Gerät, Drucker etc.) hängt eine Liste von (genauer eine "Liste von Listen von") Zugriffsrechten und ggf. auch -verboten. Mit diesen ACLs kann sehr granular — "haarklein" — für jeweils beliebig viele Nutzer und Gruppen geregelt werden, welcher Nutzer und welche Gruppe mit diesem Objekt was darf.
Auch die erlaub- bzw. verbietbaren Tätigkeiten gehen über den Unix-Dreiklang "Lesen, Schreiben, Ausführen" weit hinaus.

Mit vernünftigen Vererbungsregeln (bei Windows ab NT) für solche Objekt-Rechte z.B. entlang von Verzeichnisbäumen ist auch deren Verwaltung rationell machbar. Und mit einer vernünftigen Implementierung (leider nicht ab NT sondern erst ab W2K) ist sie auch bei Tausenden von Nutzer- und Gruppenkonten in einer Domain performant.

Praxistipp am Rande zu den oben erwähnten ACL-Verboten: Bleiben lassen!
Um z.B. ererbte Objektrechte für Gruppen und Benutzer zu widerrufen, setze man keine Verbote ein, sondern unterbreche (das geht leicht) die erwähnte Vererbung.

7.2 Active Directory

Die Darstellung der Nutzer- und Gruppenkonten (und sehr vieles mehr) ist seit W2K optional und ab W2K3 fest als "Active Directory" (AD) implementiert. Bei NT ist AD nachrüstbar. AD ist nun nichts anderes als Microsofts Interpretation des weltweit standardisierten Verzeichnisdienstes LDAP. Genauer gesagt: LDAP beschreibt die Art der (sichtbaren) Organisation der Verzeichnisse und das Zugriffsprotokoll.

Die gute Nachricht ist also

- "weltweit standardisiert"

und die andere Nachricht — und das ist bei Standards ja fast immer eine schlechte

- "durch Microsoft interpretiert" — und damit erwartungsgemäß
- "zur Unkenntlichkeit verbogen und mit Nicht-MS-Tools unverwendbar".

Nicht bei "LDAP versus AD": Hier hat Microsoft lediglich die LDAP-Schemata aufgrund der (Windows- und Domain-) Systemnotwendigkeiten erweitert und macht ausgiebig von Sicherheitseigenschaften Gebrauch. Beides ist wirklich sinnvoll.

Active Directory lässt sich also über LDAP abfragen. Damit geht dies unter anderem auch plattformübergreifend mit Java, also JNDI. Ein Nichtberücksichtigen der angedeuteten MS-Besonderheiten führt allerdings zu Funktionsmängeln, die bei unsystematischem Vorgehen oder ungenügender Dokumentation der eingesetzten Software (hier konkret Tomcat's für AD eh wenig geeigneter Klasse JNDIRealm) schwer zu klären sind.

Hierzu findet man beliebig viele negative und in flammenden Worten geschriebene "Erlebnisberichte". Sie sind alle in der Sache letztlich nicht hilfreich.

7.2 Lesender JNDI-Zugriff auf AD

Dies ist die Grundvoraussetzung für das Nutzen von AD über LDAP für Java- (und andere Nicht-MS-) Anwendungen. Und vielfach — z.B. für Authentifizierungen — reicht Lesen ja.

Loggen Sie sich als Domain-Administrator (auch remote) an irgendeinem Domain-Controller ein und gehen Sie nach

- "Benutzer und Computer in Active Directory verwalten"

Hinweis 1: Wenn Sie "nur" J2EE- / Tomcat-Chef, aber nicht Domain-Admin sind, müssen Sie halt einen freundlichen Domain-Admin überreden, das unmittelbar Folgende (das ist nicht viel) für Sie zu tun. Verwenden Sie, falls überhaupt nötig, das bei Windows-Domain-Admins i.A. wirksame "Linux-Horror"-Argument:
Jedes Vermeiden zusätzlicher Nutzerverwaltungen mindert die Gefahr, dass "seine" Anwender ihr Domain-Passwort auch für so was verwenden und dieses dann — oft im Klartext — in irgendwelche **.**user-Dateien gerät.

Hinweis 2: Weisen Sie den Domain-Admin darauf hin, dass er für die folgende AD-Rechte-Vergabe im o.g. AD-Verwaltungstool

- "Ansicht -> Erweiterte Funktionen" (oder so ähnlich)

einschalten muss. Nicht jeder Admin weiß dies, und sucht dann verzweifelt nach der "Sicherheitskarte" für AD-Objekte.

Erzeugen Sie einen neuen Benutzer "ldap leser", Anmeldename "ldr", Passwort "mausilein" (z.B.). Das Passwort muss Ihnen (und später auch Tomcat und Ihren

entsprechenden anderen Java-Anwendungen) bekannt sein. Erzeugen Sie eine Gruppe "LDAPreader", der "ldr" als Mitglied zugefügt wird.

Geben Sie diesem Nutzer und der Gruppe nur minimale Rechte, insbesondere keine administrativen oder gar solche zum "remote login". Am Besten entziehen Sie diesem Nutzer sogar das Recht, sich an irgendeiner Workstation der Domain anzumelden.

Geben Sie der Gruppe "LDAPreader" und damit (indirekt!) dem Nutzer "ldap leser" die Leserechte auf die AD-Wurzel der Domäne — wie gesagt, plus Nichts!. Dies sind die drei "Lesehäkchen" unter den vielen Rechten.

Mit dem Kommandozeilen-Tool dsquery kontrollieren Sie die Existenz der eben geschaffenen Gruppe und des Nutzers im AD. Das Ergebnis sollte etwa so aussehen:

```
E:\temp>dsquery user -name l*
  "CN=ldap leser,CN=Users,DC=FB3-MEVA,DC=fh-bochum,DC=de"
  "CN=le go,OU=mevaStGrp,DC=FB3-MEVA,DC=fh-bochum,DC=de"
  "CN=le garcon,CN=Users,DC=FB3-MEVA,DC=fh-bochum,DC=de"
  "CN=license,OU=SchrottKonten,DC=FB3-MEVA,DC=fh-bochum,DC=de"

E:\temp>dsquery group -name l*
  "CN=Leistungsprotokollbenutzer,CN=Builtin,DC=FB3-MEVA,
        DC=fh-bochum,DC=de"
  "CN=LDAPreader,CN=Users,DC=FB3-MEVA,DC=fh-bochum,DC=de"
```

Die Einschränkung -name l* erspart Ihnen die Anzeige Tausender Konten. Solche dsquery-Anzeigen liefern Ihnen auch die für das Spätere notwendigen Informationen über Microsoft- bzw. Domänen-spezifische strukturelle LDAP- (=AD-) Besonderheiten.

Nun benötigen Sie den administrativen Login nicht mehr: "Danke, lieber Domain-Admin!"

Halt, Stopp, falls der Zugriff auf den freundlichen Domain-Admin doch schwierig war: Evtl. erzeugen Sie gleich noch (für 7.4, siehe unten) zwei Gruppen tcAdmin und tcManager, denen Sie und noch ein nur Ihnen bekannter rechteloser Testnutzer angehören.

7.3 Test des lesenden LDAP- / JNDI-Zugriffs auf AD

Bevor sie einen JNDI-Zugriff auf das Domain-AD für komplexe bzw. möglicherweise komplex zu konfigurierende "Fremdsysteme" (aus Microsoft-Sicht, MAC, Linux, J2EE-Container, sonstige Java-Anwendungen etc.) einsetzen, testen Sie diesen unbedingt mit einfachen "Bordmitteln". Ideal dazu ist ein Java- (JNDI-) LDAP-Browser, z.B. als Datei

```
25.04.2001  17:30           342.395 lbe.jar
```

Dieses Archiv allein genügt. Die Quelle hierfür + Zusatzmaterial ist

http://www-unix.mcs.anl.gov/~gawor/ldap/

oder das Toolsverzeichnis des MEVA-Lab (nur innerhalb der Hochschule Bochum).

Mit diesem LDAP-Browser stellen Sie eine Verbindung zu einem Domain-Controller her. Nehmen Sie sinngemäß (Werte sind Beispiele) die folgenden Einstellungen:

```
Host: 193.175.115.2                    *1)
Port: 389
```

```
Version:  3
Base-DN:  DC=FB3-MEVA,DC=fh-bochum,DC=de  *2)
Anonymous Bind: keines                    *3)
User-DN: CN=ldap leser,CN=Users,DC=FB3-MEVA,DC=fh-bochum,DC=de  *4)
Passwort: mausilein
```

Anm. 1): Besorgen Sie sich die IP-Adresse der Domain-Controllers mit ping.

2): Vollständiger Name Ihrer Domäne als das (unsägliche) LDAP-DC-Gedödel.

3): Geht default-mäßig nicht bei AD; und dies sollte so bleiben.

4): Falle: CN ist immer der Anzeigename, nicht der häufig davon abweichende Login-Name. Letzteres ist die (AD-) LDAP-Eigenschaft sAMAccountName.

Hinweis: Die Verwendung SSL-Verbindungen für LDAP-Anforderungen mit Port 636 erfordert entsprechende Konfigurationen in der Domain. Diese, nämlich Zertifikatserver und Zertifikatverteilung via IIS, sind oft nicht gegeben.

Nach Drücken auf [Connect] sollten Sie mit nun mit dem Java-LDAP-Browser das AD der Domain durchstöbern können. Nutzen Sie dies gleich, um sich die AD-Strukturinformationen zu beschaffen, die Sie (i.A. absolut exakt) zur Konfiguration Ihrer Zielanwendungen (hier also für Tomcat oder später für Ihre AJAX-GWT-Servlets) benötigen.

Wichtig für eine SSO-Anwendung ist u.U., in welchen "Fächern" (i.A. OU= oder CN=) die relevanten Nutzer und Gruppen eingeordnet sind und wie genau die Mitgliedschaft dargestellt wird. Notieren (kopieren) Sie sich hierzu genügend Angaben in der Form:

```
---- Notizblock für später ---
   Alle Nutzer dfb...
 memberOf  CN=FB3Doz,OU=fb3Stud,DC=FB3-MEVA,DC=fh-bochum,DC=de
   Alle Nutzer sfb...
 memberOf  CN=CAX,OU=fb3Stud,DC=FB3-MEVA,DC=fh-bochum,DC=de
    Nutzer dienstGeist
  memberOf CN=MT-Labor,CN=Users,DC=FB3-MEVA,DC=fh-bochum,DC=de
  memberOf CN=Kernteam,CN=Users,DC=FB3-MEVA,DC=fh-bochum,DC=de
  memberOf CN=Replikations-Operator,CN=Builtin,DC=FB3-.....-bochum,DC=de
  memberOf CN=Sicherungs-Operatoren,CN=Builtin,DC=FB3-.....-bochum,DC=de
    Nutzer ldap leser (neu gemacht)
  memberOf CN=LDAPreader,CN=Users,DC=FB3-MEVA,DC=fh-bochum,DC=de
  sAMAccountName ldr
---- Notizblock für später ---
```

Falls dieser Test, also ein lesender LDAP-Zugriff auf AD (mit Java / JNDI), nicht funktionieren sollte, brauchen Sie mit anderen entsprechenden Anwendungen — sprich dem folgenden Abschnitt 7.4 — gar nicht erst weitermachen. Sie müssten dann einen Schritt (Abschnitt) zurückgehen.

7.4 AD-Integration in Tomcat

Nun ist ("nur") noch der Tomcat-Installation beizubringen, das AD der Domäne via LDAP und JNDI anstelle der mit der Datei

```
C:\Programme\Apache\Tomcat\conf\tomcat-users.xml
```

selbstgemachten Nutzer-, Passwort- und Rollenverwaltung zu nutzen.

Hinweis (schon mal für viel später): Wenn man mit diesen Ansatz endgültig "durch ist", sollte man in server.xml unter `<GlobalNamingResources>` die betreffende Ressource löschen oder auskommentieren. `tomcat-users.xml` wird sonst immer wider geöffnet und ohne erkennbare Wirkung geschrieben.

Grundsatz bei der AD-Integration ist diese Abbildung

von Tomcat	nach Active Directory (LDAP)
Nutzer	Domain-Nutzerkonto
Passwort	Passwort oder Ticket dieses Nutzers
Rolle	Domain-Gruppenkonto

Tabelle11: Die Tomcat – AD - Abbildung.

Die Tomcat-Rollennamen "admin" und "manager" werden für die (installierten, s.o.) Tomcat-Admin- und -Management-tools verwendet. Im Hinblick auf die AD-Integration (Abbildungstabelle 1) sind diese Namen unglücklich gewählt. In einer großen Domain sind sie i.A. anderweitig als Nutzer oder Gruppen-Namen vergeben. Jedenfalls würde sie im großen Domain-Kontext kein Mensch würde sie je mit Tomcat in Verbindung bringen.

Als erstes ändert man also diese Tomcat-Rollennamen in (beispielsweise) "tcAdmin" und "tcManager" und erzeugt für Experimente auch zwei so benannte AD-Gruppenkonten.

Bei Tomcat ändern Sie in den Dateien

C:\Programme\Apache\Tomcat\webapps/host-manager/WEB-INF/web.xml

C:\Programme\Apache\Tomcat\webapps\manager\WEB-INF\web.xml

C:\Programme\Apache\Tomcat\conf\tomcat-users.xml

Hinweis: Ort und Anzahl dieser Konfigurations-Verzeichnisse ändern sich von Version zu Version. (Das ist wohl guter Linux-Brauch). Die beste Erforschungs- /Such- und Ersetz-Funktionen bekommt man mit C:\Programme\Apache\Tomcat als Eclipse-Projekt.

bei allen betreffenden Tags <role-name> sowie bei allen Attributen roleName= und roles= die Werte von admin nach tcAdmin bzw. von manager nach tcManager. Diese Änderung auch in tomcat-users.xml erlaubt die Verwendung der neuen Rollennamen auch mit der bisherigen "privaten" (MemoryRealm) Nutzerverwaltung.

Testen Sie diese an sich einfache Änderung der Rollennamen gründlich. So haben Sie eine solide Basis für das Umschalten auf AD — und einen einfachen zeitweisen Rückzug auf das Alte bei evtl. Problemen.

Eigentlich sollte man sich nun auf den Hinweis zu Tomcat-Dokumentation insbesondere zu JNDIRealm beschränken können.

Ja eigentlich, aber dem ist leider nicht so. Vieles funktioniert einfach nicht dokumentations-gemäß, und dabei gibt es auch noch Versionsunterschiede. Es sind relativ zum "Problem" unangemessene experimentelle Mühen aufzuwenden.

7.5 AD-Integration — der steinige Weg mit JNDIRealm

In der Datei

`C:\Programme\Apache\Tomcat\conf\server.xml` ergänzen Sie nun an der hierfür vorgesehenen, aber auskommentierten, Stelle folgendes:

```
<!-- realm added 31.08.2006 15:18, mod.  01.09.2006 11:53, 14:52 -->
<Realm name="ADsso" className="org.apache.catalina.realm.JNDIRealm"
   debug="999"
   connectionURL="ldap://195.37.168.187:389"
   alternateURL="ldap://193.175.113.245:389"

   connectionName="CN=ldap leser,CN=Users,DC=FB3-MEVA,DC=fh-bochum,DC=de"
   connectionPassword="mausilein"

   referrals="follow"

   userBase="DC=FB3-MEVA,DC=fh-bochum,DC=de"
   userSearch="(sAMAccountName={0})"
   userSubtree="true"

   userRoleName="memberOf"

   roleBase="CN=Users,DC=FB3-MEVA,DC=fh-bochum,DC=de"
   roleSubtree="false"
   roleName="cn"
   roleSearch="(member={0})"
 />
```

connectionURL und ggf. alternateURL weisen auf Domain-Controller. Läuft Ihr Tomcat auf einem Domain-Controller, setzen Sie diesen i.A. allein an (erste) Stelle und probieren dabei auch 127.0.0.1 (localhost).

connectionName und connectionPassword stellen das Benutzerkonto dar, das AD lesen darf. Ansonsten sollte es, wie gesagt (s.o.), absolut rechtelos sein. Für alle diese Einstellungen gilt das oben zum LDAP-Browser gesagte; nehmen Sie Ihre dort erprobten Werte.

Die gezeigte Einstellung `referrals="follow"` ist für Active Directory notwendig.

userBase, userSearch und `userSubtree="true"` in der gezeigten Form sucht im gesamten AD nach Nutzerkonten. Wenn Sie alle Nutzerkonten in `CN=Users` haben, können Sie dies in userBase (vorne) ergänzen und `userSubtree="true"` weglassen bzw. false setzen.

userRoleName spezifiziert, wie man Rollen — gleich Gruppenmitgliedschaften — direkt im Directory-Eintrag des Nutzerkontos findet.

Die gezeigten Einstellungen roleBase, roleName und roleSearch suchen Rollen (=Gruppen, die den Nutzer enthalten) nur in CN=Users. Falls dies nicht genügt, setzt man roleBase „ein Stockwerk höher“ und roleSubtree true statt false.

Mit Recht kommt einem das merkwürdig und "doppelt gemoppelt" vor: Zum einen gibt es das an sich sinnvollere direkte Anschauen der Attribute "memberOf" und zum anderen die doppelte Suche nach dem Nutzer und anschließend — aufwändig — noch nach ihn enthaltenden Gruppen.

Das Betrachten einer Active-Directory-Struktur zeigt, dass beide Verfahren zueinander redundant sind — und zwar im Sinne von absolut überflüssig, da AD die Verkettung konsequent bidirektional (member ↔ memberOf) hält. Der Gipfel ist dann, dass die JNDIRealm-Implementierung keine Gruppe-in-Gruppe-Mitgliedschaften ermittelt, sprich den Verkettungen in keiner Richtung folgt.

Hier, sprich im obigen server.xml-Ausschnitt, werden dennoch, wie in Vorlagen weit verbreitet, beide Verfahren parallel eingeschaltet. Letztlich erreicht man mit so mit gewaltigen Zusatzaufwand eine ziemliche Kleinigkeit, und das unzuverlässig.

Hinweis: Ersparen Sie sich zeitaufwändige Experimente zum Ausschalten der Doppelsuche und lassen Sie es so, solange Sie nicht ganz auf den alternativen Ansatz von Kapitel 7.7 übergehen bzw. gleich ADweRealm statt JNDIRealm nehmen.

Eine, wie gesagt, auch so nicht überwindbare Einschränkung ist, dass Nutzer nur Rollen, sprich Gruppen, zugeordnet werden, denen sie direkt angehören. "Gruppen in Gruppen" wird nicht aufgelöst. So was Selbstverständliches funktioniert nicht. So sind in großen Domains dann viele vorhandene Gruppen für Tomcat nicht verwendbar — eine echte Einschränkung und eine fehlerträchtige Falle!

Aus der Sicht eines professionellen Einsatzes in (großen) Domains reichen die schlechten Nachrichten eigentlich. Aber: Hinzu kommt noch, dass die (programmatische) Abfrage

```
request.isUserInRole("tcAdmin") // zum Beispiel
```

unzuverlässig und damit eigentlich überhaupt nicht funktioniert. Eine solche Abfrage liefert bei JNDIRealm zu oft false. Dies gilt gegebenenfalls selbst für die einzige Rolle, ohne die man das betreffende Servlet gar nicht hätte sehen könnte. Dies ist nun alles, gelinde gesagt, ziemlich unbefriedigend.

Dennoch erst mal weiter ...

Nun ergänzt bzw. ändert man noch Folgendes am Ende von

```
C:\Programme\Apache\Tomcat\conf\web.xml  :
```

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

<!--  01.09.2006: Test, Auth. und SSL für .../meva-lab/.... -->
    <security-constraint>
      <web-resource-collection>
        <web-resource-name>Lab Secret Area</web-resource-name>
        <url-pattern>/meva-lab/*</url-pattern>
      </web-resource-collection>

    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>

      <auth-constraint>
         <role-name>tcManager</role-name>
```

```
        <role-name>role1</role-name>
      </auth-constraint>
    </security-constraint>

   <login-config>
      <auth-method>BASIC</auth-method>
      <realm-name>Eigentlich nicht so geheim, aber Test.</realm-name>
   </login-config>

   <security-role>  <role-name>role1</role-name>  </security-role>

   <security-role>  <role-name>tcManager</role-name> </security-role>

   <security-role>  <role-name>CAX</role-name> </security-role>

</web-app>
```

sowie in der Datei

D:\worknosave\webAppsDev\WEB-INF\web.xml

folgendes (siehe auch Anhang, die zweite Fassung):

```
<!--  01.09.2006: Test, Auth. und SSL für Hello-Servlet  -->
    <security-constraint>
      <web-resource-collection>
        <web-resource-name>Servlet Secret Area</web-resource-name>
        <url-pattern>/servlet/HelloWorld</url-pattern>
      </web-resource-collection>
    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>

      <auth-constraint>
         <role-name>FB3Doz</role-name>
         <role-name>tcManager</role-name>
      </auth-constraint>
    </security-constraint>
```

Nun sollten nach Neustart von Tomcat alle Inhalte mit geforderter Authentifizierung mit Domain-Nutzern in entsprechenden Gruppen = Rollen und nicht mehr gemäß den in tomcat-users.xml definierten funktionieren.

Im Erfolgsfall lassen sich nun basiert auf Active Directory und im Rahmen der sowieso stattfindenden Domain-Administration beliebige Tomcat-Rollen definieren, gegen die sich die Domain-Nutzer in ihrer gewohnten Weise authentifizieren.

7.6 Authentifizierungs-Abfragen in Servlets

Die (programmatische) Abfrage

```
request.isUserInRole("tcAdmin") // zum Beispiel
```

erfragt, ob der zum `request` gehörende `session-principal`, sprich der am Browser eingeloggte und erfolgreich authentifizierte Nutzer, die erfragte Rolle hat.

Wie bereits oben erwähnt, funktioniert diese programmatische Abfrage überhaupt nicht. Sie liefert bei JNDIRealm und AD immer nur false. Dies gilt sogar für exakt diejenige Rolle, mit der der Nutzer via

```
<auth-constraint>
   <role-name>tcAdmin</role-name>
</auth-constraint>
```

die betreffende Seite bzw. das Servlet überhaupt nur zu sehen bekam — von weiteren Rollen die er haben, sprich AD-Gruppen, denen er auch angehören kann, ganz zu schweigen. Klar ist, dass mit dieser Situation komplexere Webdienste mit Rollenabfragen für einzelne gewünschte kritische "Taten" schwer darstellbar sind.....

Hinweis: Wenn Sie sich für alle Unerfreulichkeiten von "JNDIRealm mit Active Directory" und diverse "work arounds" interessieren (müssen), lesen Sie dieses Kapitel 7.6 sowie 7.7 in der Vorversion [11] weiter.

Ansonsten nehmen Sie nicht mehr

```
org.apache.catalina.realm.JNDIRealm
```

sondern einfach

```
de.a_weinert.realm.ADweRealm
```

für Active Directory.

Vergessen Sie die JNDIRealm/AD- Probleme und lesen gleich in Kapitel 7.8 weiter.

7.7 Work-around um JNDIRealm-Probleme und die AD-Rollensuche

Siehe den Hinweis gerade: .
Sie finden, wenn Sie den müssen, diese Infos im selben Kapitel von [11].

Wenn Sie den steinigen Weg gegangen sind, haben Sie diese Lage:

- Sie haben die Schritte der Kapitel 7.1 bis 7.4 erfolgreich durchlaufen.
- Sie haben mit den Hinweisen ab Kapitel 7.5 eine Authentifizierung gegen Ihr Active Directory mit JNDIRealm zustande gebracht.
- Sie haben eine hinreichend neue Version des Frameworks de.a_weinert (aWeinertBib.jar oder jünger) installiert.
- Sie haben die damit und durch entsprechend Konfigurationen von <role-name>s etc. auch eine funktionierende Rechteabfrage in Servlets (isUserInRole()) erreicht.

Wenn das Alles so ist, ...
... dann haben Sie es — die Active Directory Integration —ja eigentlich geschafft.

Ein ungutes Gefühl bleibt vor Allem wegen der Auswirkungen auf die Konfiguration der Web_Dienste und tiefgreifender Konsequenzen für die Servlet-Programmierung.

7.8 AD-Integration — der Lösungsweg mit ADweRealm

Wenn ein Authentifizierung gegen ein Active Directory mit JNDIReam, evtl. mit den Hinweisen der hier praktisch entfallenen Kapitel 7.6 unf 7.7 aus dem Vorgänger [11] hinbekommen haben, bleibt ein berechtigtes ungutes Gefühl. Mit einer komplexen, wenig performanten, undurchschaubaren Lösung lebt man auf dünnem Eis. Die sonst nach wie vor lügende isUserInRole-Abfrage lässt sich nur mit Auswirkungen auf die Servlet-Programmierung reparieren.

Und, und, und ... – eigentlich eine untragbare Situation, die in diesem Kapitel an der Wurzel bereinigt wird.

Dies geht nun nur mit einer Realm-Klasse, welche was von Active Directory "versteht". Unter den eben genannten Voraussetzungen können Sie diesen Weg mit der Klasse de.a_weinert.realm.ADweREalm gehen.

Die sehr einfache Verwendung und Konfiguration ist in der (javaDoc-) Dokumentation, www.a-weinert.de/java/docs/frame4j/de/a_weinert/realm/package-summary.html , beschrieben.

Die wesentlichen Vorzüge dieser Lösung sind

- keine Einschränkungen in der AD-Struktur: Group-in-group geht,
- und zwar nun auch für <auth-constraint>s und nicht nur für de.a_weinert.net.LDAPauthRead.`isUserInRole(request, ...)`.
- Problemloses Funktionieren direkt von `request.isUserInRole(s)` in Servlets
- und zwar (nun) ohne jede Auswirkung auf deren Programmierung.

Der letzte Punkt besagt, dass man bei Verwendung von ADweRealm Servlets nicht mehr anders, also nicht mehr mit LDAPauthRead, programmieren muss. ADweRealm selbst verwendet natürlich das auch in Nicht-Tomcat-Client-Server-Anwendungen bewährte .LDAPauthRead. (das man bei den wor-arounds in betreffenden Servlets eingesetzt hätte).

Diese Vorteile . – des das Problem an der Wurzel Packens . – wiegen um so schwerer, als es in einer großen Umgebung ein ja widersinniges Ansinnen ist, vorhandene AD-Strukturen oder bewährte Servlets den Einschränkungen von JNDI-Realm (in Bezug auf Ad) anzupassen.

Als weiterer kleinerer Vorteil kommt hinzu

- ziemliche Vereinfachung der web.xml-Dateien; siehe Listing 12.
- nur noch kurze Rollen reichen und man scheint die <security-role>-Auflistungen (endlich!) ganz weglassen zu können.

Tun Sie also einfach die Datei

`11.04.2008 15:27 58.747 catErgWe.jar` (oder jünger)

nach C:\Programme\Apache\Tomcat\lib\ und ändern / bereinfachen Sie die server.xml gemäß Listing 12.

```
<!--  ADweRealm added 01.02.2008, a Realm really for Active Directory
    Special development logging can be switched on by
     devLog="writable-appendable-file"    (Omit for no extra logging)

  Default attributes (can be omitted if value fits) are:
        userRoleName="memberOf"    userSubtree="true"
        userSearch="(sAMAccountName={0})"
        shortRoles="true"          followRoles="true"
        maxRoles="33"              userPasswordSecret="true"
-->

<Realm name="ADsso" className="de.a_weinert.realm.ADweRealm" debug="999"
     devLog="C:\Programme\Apache\Tomcat\logs\awRe.log"
     allRolesMode="authOnly"

     connectionURL="ldap://193.175.115.2:389"
     alternateURL="ldap://193.175.115.4:389"
     connectionName="CN=ldap reader,CN=Users,DC=FB3-MEVA,DC=fh-bochum,DC=de"
     connectionPassword="***"

     userBase="DC=FB3-MEVA,DC=fh-bochum,DC=de"
     defaultRole="fb3-meva_user"
     shortRoles="short"
   />
```

Listing 12: Die einfache Konfiguration von ADweRealm.

8. Erweiterungen

8.1 PHP

Zur Erinnerung: Dieses Tutorial handelt von J2EE mit ("stand alone") Tomcat auf Windows bis hin zur Integration des Active Directory für die Authentifizierung an Web-Diensten. In solchem Umfeld ist PHP im Allgemeinen kein Thema. Aber manche mögen ja doch ein bisschen mehr. Und eine nicht schweigende Mehrheit glaubt, nun brauche man doch einen Apache. Das stimmt so einfach nicht. (Was man für viele PHP-Anwendungen allerdings braucht, ist ein versionsmäßig passendes mySQL, [14].) Gleichwohl ist, wenn man keine Servlets benötigt, ein (stand-alone) Apache eine prima Sache.

Das folgende beschreibt den bisher einfachsten gefundenen Weg zu "PHP on Tomcat".

Laden Sie sich das freie Resin-Paket herunter, also beispielsweise die Datei

```
10.06.2008   20:16   10.284.736    resin-3.1.6.zip
```

Packen Sie dieses in irgendein (temporäres) Verzeichnis aus, mit:

```
jar xfv  <D:\ToBeDoneViewed\tomcat\undPHP\>resin-3.1.6.zip
```

Aus dem ganzen, großen "Resin" (J2EE) benötigen Sie lediglich von "Quercus" (das ist Resins PHP-Maschine) die folgenden zwei Dateien:

```
04.05.2008   15:26    3.491.668    quercus.jar
04.05.2008   15:26      659.063    resin-util.jar
```

Verschieben oder kopieren Sie diese zwei nach

```
C:\Programme\Apache\Tomcat\lib
```

In der Datei

```
11.06.2008   08:35       56.532    web.xml
```

ergänzen Sie an passender Stelle (bei servlet und bei servlet-mapping) Listing 13.

```
<!-- 11.06.2008 quercus php-Servlet -->
  <servlet>
     <servlet-name>quercus</servlet-name>
     <servlet-class>
        com.caucho.quercus.servlet.QuercusServlet
     </servlet-class>
     <init><php-ini>
           <default_mimetype>text/html</default_mimetype>
     </php-ini></init>
  </servlet>

  <servlet-mapping>
     <servlet-name>quercus</servlet-name>
     <url-pattern>*.php</url-pattern>
  </servlet-mapping>
```

Listing 13: Ergänzung von Tomcats web.xml für PHP mit Quercus.

Wenn Sie nun zusätzlich auch index.php als default-Startseite eines Verzeichnisses haben wollen, sähe die entsprechende Stelle der web.xml etwa so aus:

```
<welcome-file-list>
  <welcome-file>index.html</welcome-file>
  <welcome-file>index.htm</welcome-file>
  <welcome-file>index.jsp</welcome-file>
  <welcome-file>index.php</welcome-file>
</welcome-file-list>
```

Nun stellen Sie noch eine Datei info.php, die etwa dem Vorschlag von Listing 14 entspricht, in ein via http erreichbares Verzeichnis.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type"
          content="text/html; charset=iso-8859-1" />
<meta name="AUTHOR" content="Albrecht Weinert" />
<meta name="GENERATOR" content="Eclipse, CVSkeys.java" />
<meta name="DESCRIPTION" content=
"Labor für Medien und verteilte Anwendungen (MEVA-Lab), Infoseite für php
on Tomcat" />
<meta name="Keywords"
         content="Hochschule Bochum,MEVA,Weinert,Tomcat,Testseite,PHP" />
<title>MEVA-Lab  --  Info, php on Tomcat     </title>
<link rel="stylesheet" type="text/css" href="/serv-intra/meva.css"
      title="Style" />
<link rel="SHORTCUT ICON" href="/serv-intra/favicon.ico" />
</head>   <body class="body">
Serverdatum: <?
   echo Date("d.m.Y H:i:s");
?>
<h3>PHP-Installation im Tomact (J2EE, kein Apache)</h3>
<? phpinfo(); ?>
</body></html>
```

Listing 14: Vorschlag einer einfachen PHP-Testseite "info.php".

Wenn Sie (evtl. erst nach Tomcat-Restart) zur Seite "info.php" gemäß Listing 14 navigieren, sehen Sie etwas wie Bild 15.

Das war schon Alles: PHP steht damit auf dem Tomcat (ohne Apache) auch für serverseitiges Scripting zur Verfügung. Und man kann ja einige nette Dinge damit tun.

Was und wieviel man mit PHP in einem professionellen J2EE-Umfeld tatsächlich tun will und sollte, ist keine so einfache Frage. Das blanke Entsetzen, das die Mehrheit der Java-/J2EE-Entwickler bei ihrer allerersten Begegnung mit PHP befällt wird durch die Instabilität größerer PHP-Pakete gegen kleine Versions- und Umfeldänderungen ja zumindest nicht widerlegt.

Anmerkung: So gibt es riesige PHP-Pakete (Wiki..), deren bereits in PHP als Web-Dienst geschriebenes (integriertes) Installationsprogramm prima (sprich lange und unterhaltsam) läuft. Nach (!) dessen Erfolgsmeldung kommt noch eine Anweisung, eine Datei auf dem Server von Hand zu verschieben ([sic!]). Dies innerhalb eines Web-Dienstes zu erleben, lässt schon nichts Gutes ahnen. Nach braver Ausführung dieser Anweisung und nun dem befohlenen Weiterklick

(im immer noch selben Web-Dienst und mit angezeigter Erfolgs- / Alles OK-meldung) liefert das Ganze — nur noch Nichts (nada, 0 byte, 0 Quelltext, blank page) für sämtliche gelieferten / generierten *.php.

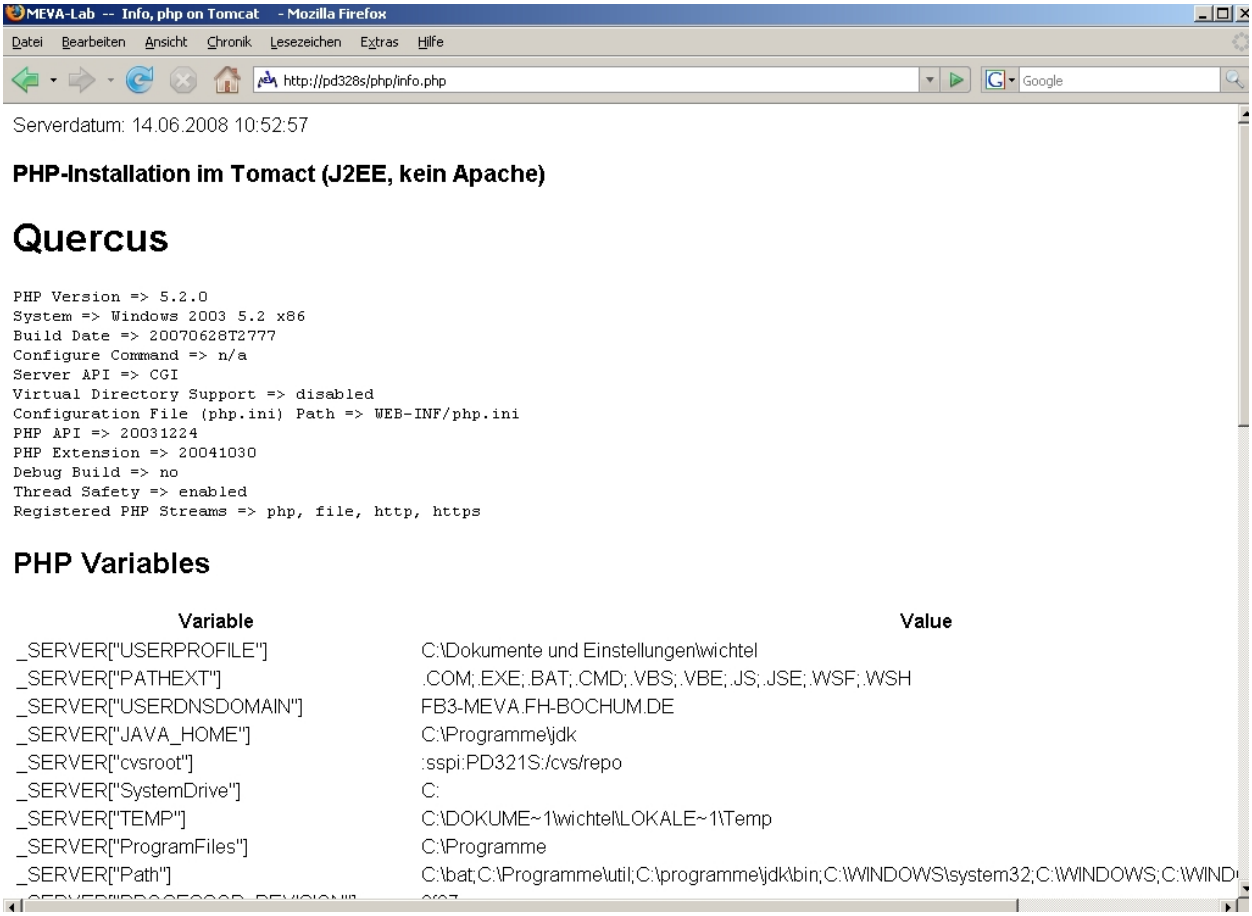

Bild 15: Test der PHP-Grundinstallation mit einer php-info-Seite.

Anmerkung 2 (Ergänzung): Bevor dieses Debakel nun vorschnell der hier vorgestellten Quercus-PHP-Implementierung anlastet, ergoogelt man (MediaWiki blank page), dass dieses Phänomen unerklärt und unbehebbar auch viele andere Plattformen heimgesucht, auch nachdem Wikis bereits lange produktiv liefen, und arme Administratoren schon dem Selbstmord nahegebracht hat.

Das laufen Lassen großer PHP-Pakete auf einem J2EE-Server wird also eher nicht das Motiv für ein "PHP on Tomcat" sein, zumal für viele entsprechende J2EE-/Java-Implementierungen da sind. Das gilt auch für die hier unfairer Weise als Beispiel für PHP-Mängel dargestellte MediaWiki (nächstes Kapitel).

Häufig möchte man den mehr statischen Teil eines Web-Auftrittes (aus einer Versionsverwaltung heraus) auf mehrere Web-Server und Provider verteilen. Ist davon einer extern findet man in den weniger teuren, populären Tarifen fast immer ein PHP aber praktisch nie J2EE vor. Hier wird man ein bisschen serverseitige Dynamik also mit PHP darstellen müssen. Soll nun einfach derselbe Teilbereich auch auf einem internen Tomcat laufen, kann man die *.php dank Quercus einfach so lassen.

Diese interne Wiederverwendbarkeit und die damit verbundenen Vorteile sind das eigentliche Motiv für "PHP on Tomcat".

8.2 Eine Wiki

Auch und gerade auf einem internen J2EE-Server mit Web-Diensten zur Prozesssteuerung, Domain-Verwaltung etc. ist eine (interne) Wiki durchaus sinnvoll. Wie im letzten Kapitel angedeutet muss eine Wiki auf PHP (auf Tomcat) nicht unbedingt Freude machen.

Auf einem J2EE-Server ist auch immer anzustreben etwas von der Größenordnung als richtigen Webdienst zu installieren. Für eine Wiki ist die "very quick Wiki" oder VQWiki eine solche gute Lösung.

Voraussetzungen:

- Sie haben die Installationsdatei und vielleicht auch gleich das Handbuch

```
17.06.2008  09:17         3.595.784 vqwiki-2.8-RC1.war
17.06.2008  08:49           653.173 vqwiki_book.pdf
```

in dieser oder einer anderen passenden Version.

- Auf dem Windows-Server, auf dem ihr Tomcat läuft, haben Sie eine Datenbank, beispielsweise

```
MySQL   5.1.25-rc-community     Community Server (GPL)
```

Das muss nicht unbedingt so, also mySQL und auf demselben Server, sein, aber das ist einfachst handhabe (und im Folgenden vorausgesetzte) Konfiguration.

In dieser Datenbank legen Sie einen neuen Bereich und einen neuen Nutzer mit folgenden Befehlen (als DB-admin) an:

```
create database vqwiki;
grant all on vqwiki.*
          to vqwiki@localhost identified by 'vqwiki';
flush privileges;
exit;
```

Legen Sie einen neuen Dateibereich (als möglichen Arbeitsbereich der Wiki) an mit beispielsweise:

```
E:> md meva-wiki-files
```

Geben Sie vor allem dem Benutzer "system" (unter dem Tomcat normalerweise als Dienst läuft) hierauf Vollzugriff, und anderen Nutzern möglichst wenig Rechte. Danach müsste das sinngemäß so aussehen:

```
D:\temp>cacls E:\meva-wiki-files
     VORDEFINIERT\Administratoren:(OI)(CI)F
                    FB3-MEVA\weinert:(OI)(CI)F
                    FB3-MEVA\Domänen-Admins:(OI)(CI)F
                    NT-AUTORITÄT\SYSTEM:(OI)(CI)F
```

Legen Sie im Web-Dateibereich Ihres Tomcat ein zusätzliches Verzeichnis, wie beispielsweise

```
www\meva-wiki
```

für einen neuen Web-Context an. Kopieren Sie dorthin (aus einem anderen Web-Context) alle Grunddateien, wie Ikonen, Log-In-Form-Seiten, Stylesheet, Fehlerseiten und der-

gleichen. Richten Sie praktisch einen neuen leeren Context gemäß Ihren Grundeinstellungen ein. Danach sieht das dann sinngemäß etwa so aus:

```
Verzeichnis von D:\www\meva-wiki

29.04.2008    14:03         5.188    error_all.jsp
06.02.2008    11:56         4.533    fail_login.html
23.09.2007    19:20           894    favicon.ico
17.06.2008    10:30    <DIR>           images
03.01.2008    17:35         2.257    index_s_f.html
18.02.2008    16:20         8.255    login.html
17.06.2008    09:25    <DIR>           META-INF
21.12.2007    13:19         4.727    meva.css
```

In eine Verzeichnis images können Sie auch schon mal ein gewünschtes Logo (Seitenlängen 90 ... 200 pix) tun; im folgenden Beispiel wäre das

```
/images/meva-logo-aq-kl.png
```

Den neuen leeren Context mit Ihren Basiseinstellungen beschreibenSie mit einer Datei

C:\Programme\Apache\Tomcat\conf\Catalina\localhost\meva-wiki.xml

etwa folgenden Inhalts:

```
<Context docBase="D:\www\meva-wiki">
       <Environment name="logoImageName" type="java.lang.String"
              value="/images/meva-logo-aq-kl.png" override="false"/>
</Context>
```

Wechseln Sie nun in das o.g Context-Verzeichnis (D:\www\meva-wiki) und packen Sie mit

```
jar xfv  D:\ToBeDoneViewed\tomcat\undWiki\vqwiki-2.8-RC1.war
```

die oben genannte Installationsdatei dorthin aus.

Nun können Sie noch (gleich oder später) in der nun vorhandenen Datei

```
D:\www\meva-wiki\WEB-INF\web.xml
```

folgendes ergänzen:

```
<!--  16:58 17.06.2008 Weinert -->
<security-constraint>
    <web-resource-collection>
        <web-resource-name>MEVAwiki</web-resource-name>
        <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>fb3-meva_user</role-name>
    </auth-constraint>
</security-constraint>
```

Mit dieser oder einer ähnlichen sinngemäßen Änderung müssen sich Nutzer Ihrer Wiki für die zugehörige J2EE-/Tomcat-Sitzung (genau so wie Sie es für Ihre übrigen Web-Dienste

ggf. auch möchten) authentifizieren. Ein von VQWiki zusätzlich angebotenes besonderes Log-In-Verfahren sollten Sie nicht nutzen.

Das war's (nach einem evtl. Tomcat-Restart) auch schon. Beim ersten (zwangsweise administrativen) Anmelden an Ihrer neuen Wiki müssen Sie gut geführt, aber dennoch besser mit der entsprechenden Seite des oben erwähnten (pdf) Handbuchs aufgeschlagen, die Grundeinstellungen Ihrer neuen Wiki vornehmen. Dazu gehört auch das Verbinden mit der Datenbank, aber das Alles ist mit den eben bschriebenen Schritten ja gut vorbereitet.

9. Résumée

9.1 Erreicht mit Tomcat 6.0

Dieses Tutorial zeigte von Null an, also auch für Ersteinsteiger, das Aufsetzen eines "stand-alone" Tomcat als J2EE-Container in einer Windows-Domain-Umgebung. Dabei wurden über die Grundinstallation hinaus alle für einen professionellen J2EE-Einsatz notwendigen zusätzlichen Komponenten und Einstellungen behandelt.

Sie haben bzw. können nun also:

- Tomcat als J2EE-Container
- Liefern von static content
- Erstellen und Betreiben von Servlets
- gesicherte Übertragung mit SSL
- Benutzer-Authentifizierung
- Übertragung der Nutzer- und Rollenverwaltung weg von der Anwendung zur Systemumgebung; hier als
- Active Directory – Integration
- Verwendbarkeit von PHP
- eine Wiki

Insbesondere die im Falle vom Vorgehen nach Kapitel 7.8 ist in einem professionellen Windows- / AD-Domain-Umfeld Authentifizierung gegen AD ein "feature" für Firmen-Webdienste. Domain-Nutzer sind Tomcat-Nutzer und Domain-Gruppen sind Tomcat-Rollen – bei Verwendung von ADweRealm gilt / geht das ohne wenn und aber.

9.2 Was bleibt zu tun

1.) Klärung offener Punkte, Erstellen besserer Dokumentation.
Die Klärung vieler Punkte und das Finden mancher Lösung — und die AD-Integration ist nur der herausragenste von vielen — war eher mühseligem, ja "schmerzhaftem" Suchen und nicht vom Hersteller dokumentierten, geordnetem Vorgehen zu verdanken.

Kommentar und Klarstellung am Rande:
Dass Programmier- und Dokumentationsfehler gerade bei umfangreicher und engagierter Arbeit vorkommen, ist nicht der Kritikpunkt. Vielmehr ist es die Art der Behandlung von essentiellen Aspekten für einen professionellen Einsatz von Tomcat in einer AD-Domäne. Hier wird teilweise eine "kümmert uns aus Prinzip nicht"-Haltung à la "Wenn Sie schon das Pech eines Microsoft-Systems haben, .." oder "Take Linux!" offen an den Tag gelegt. Eine solche Haltung, die eigene Mängel mit Hochnäsigkeit kaschieren will, ist ja schon grundsätzlich zu tadeln. Im Zusammenhang mit Tomcat gilt dies aber besonders, wenn man mal voraussetzt, dass ein professioneller und plattformübergreifender Einsatz ein Ziel sein soll. Und für die Existenz eines solchen Ziels spräche ja, dass Tomcat zu SUNs offizieller Referenz-Implementierung für J2EE-Container befördert wurde.

Derselbe "Kommentar am Rande" wäre sinngemäß zu PHP, PEAR, PHPunit etc. zu machen: Für einen professionellen (Windows-) Einsatz macht dies Alles wenig Freude. Festkodierte Linux-Pfade (mal wieder gerne genommen, z.B.) beim Übergang von Version ".16" nach ".17" bringen Alles zum Platzen, und Tests / Unterstützung durch die Autoren ist oft ... (siehe oben).
Zitat eines anderen offenbar auch entsprechend hart Getroffenen (Jonathan Tran):
"It seems that either no one cares about doing this on Windows or only really smart people who find it so simple that they didn't bother to write about it have ever done it."

2.) Finden eines Weges für die "multi-Realm-" Authentifizierung. Bis dato (6.0.14) kann dies keine Tomcat-Realm-Klasse. Auch ADweRealm tut hierfür nur das allermindeste, in dem es nachrangig zur AD-Authentifizierung eine weitere gegen ein beliebiges LDAP erlaubt. Hier hofft man noch, dass Tomcat einmal ein einfach handhabbares "Realm-chain-feature" bringen wird (?).
   Im Zusammenhang mit der AD-Integration (in einer Domain) möchte man unbedingt Folgendes darstellen können:

   Wenn ein Benutzer sich authentifiziert (FORMS) wird zunächst (mit de.a_weinert.realm.ADweRealm) geprüft, ob er ein Domain-Mitglied (Firmenangehöriger) ist und ggf. seine Gruppenmitgliedschaften als Tomcat-Rollen übernommen. Schlägt dies fehl, möchte man sich für ggf. Kunden, Gäste oder ähnliches auf eine weitere (beliebige realm) abstützen. (Wenn diese auch für AD-Mitglieder noch zusätzliche Rollen nachliefern würde, wäre das Ideal erreicht.) Jedenfalls will man in einem Windows-Umfeld i.A. zwar unbedingt die Authentifizierung gegen AD, aber man will AD i.A. nicht (dynamisch) mit durchaus notwendigen Gast-/Kundenkonten aufblähen, die tatsächlich ja keine Domain-Rechte haben.

   de.a_weinert.realm.ADweRealm erfüllt, wie bereits angedeutet, diese MIndest-anforderung durch optionales "dahinter hämger" eines LDAP, in welchem die genannten Gäste bzw. Kunden dann zu finden sein müssen.

3.) Das Anwendungen den eher benutzerobflächlich-geschmacklichen Unterschied von BASIC- und FORMS-Authentifizierung merken, ist ähnlich bearbeitungsbedürftig.

4.) Schöne Fehlerseiten und ihre Probleme (gemacht aber todo: Beschreiben!).

5.) Robuste PHP-Integration (sofern es robustes PHP gibt)

6.) Feedback welcome (a-weinert.de).

Für den sehr empfehlenswerten Einsatz von AJAX mit GWT auf dem gerade etablierten Tomcat siehe [5] im gleichen Verzeichnis.

Wer Alles bisher gesagte erfolgreich nachvollziehen konnte, hat aber eine wirklich gute Basis, auch professionelle Web-Dienste.
Nicht mehr, aber auch — bei aller angeklungenen Kritik — nicht weniger.

Und als plattformunabhängiges Java-Produkt lässt sich Tomcat — ungeachtet der geschilderten Mühen mit der Active-Directory-Integration — um Welten besser in einer industriellen Windows-Umgebung nutzen als manch anderes eher schlecht als recht nach Windows portiertes Linux-C-Produkt.

Anhang

A1 Quelle des HelloWorld-Servlets

```
----  HelloWorld.java  ------------------------------------

import java.io.IOException;
import java.io.PrintWriter;
import java.security.Principal;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import de.a_weinert.Zeit;

/** <b>Einstiegs- und Beispiel-Servlet</b>. <br />
 *  <br />
 *  Ein Objekt dieser Klasse stellt ein kleines {@link HttpServlet}
 *  für einen J2EE-Container, wie beispielsweise Tomcat, dar. Auf einen
 *  HTTP(S)-get-Request hin erzeigt es eine HTML-Ausgabe mit einen Gruß
 *  (&quot;Hello ... &quot;) sowie Datum und Uhrzeit und ein paar weiteren
 *  Informationen.<br />
 *  <br />
 *  Dieses wenn auch nicht minimale so doch sehr einfache Beispiel für ein
 *  Servlet, kann als Einstieg in bzw. Ausgangspunkt für beliebig komplexere
 *  Web-Services dienen.<br />
 *  <br />
 *  Zum Deployment mit Apache-Tomcat siehe die
 *  <a href="http://www.a-weinert.de/docu">MEVA-Lab-Doku</a>
 *  (<a href="http://www.a-weinert.de">A. Weinert</a>).<br />
 *  <br />
 *  <!-- a href="../package-summary.html#co">&copy;</a -->
 *  &copy; Copyright 2006   Albrecht Weinert <br />
 *  <br />
 *  @author   Albrecht Weinert
 *  @version  $Revision: 1.20 $ ($Date: 2007/02/16 13:46:27 $)
 */
 // Bisher    V01.00 (12.01.2006 08:01) : neu
 //           V01.02 (28.08.2006 14:25) : Zus.-Info.
public class HelloWorld extends HttpServlet {

/** Standard-Start einer HTML-Seite einschließlich Title.<br />
 *  <br />
 *  Fest (unparametrierbar) für dieses Beispiel-Servlet.
 */
```

```
   public static final String HTML_START =
        "<html xmlns=\"http://www.w3.org/1999/xhtml\">\n<head>"
      + "<meta name=\"GENERATOR\" content=\"HelloWorld.java (servlet)\" />"
      + "<meta name=\"AUTHOR\" content=\"Albrecht Weinert\" />\n"
      + "<title> Hello World   (Beispiel-Servlet)  
 </title>"
      + "\n</head><body>";


/** Standard-Ende einer HTML-Seite bis Title.<br />
 *  <br />
 *  Fest (unparametrierbar) für dieses Beispiel-Servlet.
 */
   public static final String HTML_END = "<br />\n</body></html>\n";

/** Die Arbeitsmethode des Servlets.<br />
 *  <br />
 */
   public void doGet(HttpServletRequest request,
           HttpServletResponse response) throws ServletException,
                                                IOException {
      StringBuffer bastel = new StringBuffer(666);
      bastel.append(HTML_START);

      Principal principal = request.getUserPrincipal();
      String name = principal == null ?
                                   null : principal.getName();
      String nam = name == null ? "world" : name; // ^ bei ssl
                                              // mit Auth.

      bastel.append("<h3>Hello ").append(nam).append(" !</h3>\n");

      bastel.append("Ihre Adresse: ").append(request.getRemoteHost());
      bastel.append("<br />\n");
      Zeit zt = new Zeit();
      bastel.append("Zeit (Server): ");
      bastel.append(zt.toString("W, der T.m.J, h:m:s"));

      bastel.append("<br />\n");

      String auth = request.getAuthType();
      bastel.append("Authenification: ");
      bastel.append(auth);

      bastel.append("<br />\n");

      String user = request.getRemoteUser();
                    // funktioniert nur mit Tomcats-Linux-Nutzerverwaltung
```

```
      boolean inRole1 = request.isUserInRole("role1");
      boolean inRoleTc = request.isUserInRole("tomcat");
      boolean inRoleAd = request.isUserInRole("admin")  // tcAdmin
          ||  request.isUserInRole("tcAdmin");
      boolean inRoleMa = request.isUserInRole("manager") // tcManager
          || request.isUserInRole("tcManager");
 /// Hinweis: Für professionellen Einsatz besser
 /// de.a_weinert.net.LDAPauthRead.isUserInRole(...) verwenden !

      boolean anyRole = inRoleMa || inRole1  || inRoleTc || inRoleAd;

      bastel.append("User (remote): ");
      bastel.append(user);
      if (anyRole)  bastel.append("  in den Rollen: ");
      if (inRole1) bastel.append(" 1, ");
      if (inRoleTc) bastel.append("tomcat, ");
      if (inRoleMa) bastel.append("manager, ");
      if (inRoleAd) bastel.append("admin");

      bastel.append(HTML_END);
      PrintWriter out = response.getWriter();
      out.print(bastel.toString());
   } // doGet(HttpServletRequest ..)

} // class Helloworld
```

Ein das Framework de.a_weinert nutzendes und somit der Mühen der Auswertung von Initialisierungsdaten, Aufrufparametern und der Internationalisierung weitestgehend enthobenes Hello-Servlet (HelloServ.java) finden Sie unter

http://www.a-weinert.de/java/docs/aWeinertBib/HelloServ.html

Dies demonstriert umfassendere Möglichkeiten und ist so ein wohl besserer Einstieg in die Servlet-Programmierung — wenn der obige "Einsteiger" erst mal läuft. Also weiter damit ...

A2 Descriptor des HelloWorld-Servlets

Für die ersten Servlet-Schritte ohne Authentifizierung genügt folgende Datei im WEB-INF-Verzeichnis des / der Servlets:

```
----  web.xml    (1)   ------------

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
   PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
   "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
```

```
   <servlet>
      <servlet-name>HelloWorld</servlet-name>
      <servlet-class>HelloWorld</servlet-class>
      <load-on-startup>1</load-on-startup>
   </servlet>
   <servlet-mapping>
      <servlet-name>HelloWorld</servlet-name>
      <url-pattern>/servlet/HelloWorld</url-pattern>
   </servlet-mapping>
</web-app>

-------------------------------------
```

Mit zusätzlicher Authentifizierung ergänzt sich diese Datei so

```
----  web.xml    (2)   ------------

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
   PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
   "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
   <servlet>
      <servlet-name>HelloWorld</servlet-name>
      <servlet-class>HelloWorld</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>HelloWorld</servlet-name>
      <url-pattern>/servlet/HelloWorld</url-pattern>
   </servlet-mapping>

<!--  01.09.2006: Test, Auth. und SSL für Hello-Servlet  -->
    <security-constraint>
      <web-resource-collection>
        <web-resource-name>Servlet Secret Area</web-resource-name>
        <url-pattern>/servlet/HelloWorld</url-pattern>
      </web-resource-collection>
    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>

      <auth-constraint>
         <role-name>FB3Doz</role-name>
         <role-name>tcManager</role-name>
```

```
      </auth-constraint>
    </security-constraint>
</web-app>
```

A3 Beispielseiten für FORMS-Authentifizierung

Die Bezüge auf style-sheets, etc. sind Ihren Verhältnissen sinngemäß anzupassen.

Dies ist die Änderung bzw. Ergänzung in
C:\Programme\Apache\Tomcat\conf\ web.xml :

---- web.xml (Erg.) ----------

```
<login-config>
      <auth-method>FORM</auth-method>
      <realm-name>Diese Seiten oder Webdienste
                  erfordern eine Authentifizierung.</realm-name>
      <form-login-config>
      <form-login-page>/login.html</form-login-page>
      <form-error-page>/fail_login.html</form-error-page>
      </form-login-config>
   </login-config>


   <security-constraint>
       <web-resource-collection>
        <web-resource-name> LogInForm </web-resource-name>
        <url-pattern>/login.html</url-pattern>
       </web-resource-collection>
       <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
       </user-data-constraint>
   </security-constraint>
```

Die Formularseite für die Benutzerauthentifizierung:

---- login.html ----------

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!--  Login-Seite  PD321S, PD3022 für FORMS Login
      (nur hochschulintern)
      Version V.$Revision: 1.10 $, $Date: 2007/10/31 12:50:19 $
  -->
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<meta name="AUTHOR" content="Albrecht Weinert" />
<meta name="GENERATOR" content="Eclipse, CVSkeys.java" />
<meta http-equiv="content-language" content="de" />
<meta name="DESCRIPTION"
content="Labor für Medien und verteilte Anwendungen (MEVA-Lab), Login für
Webdienste (intern)" />
<meta name="Keywords"
content="Hochschule Bochum,MEVA,Weinert,AJAX,GWT,webservice,login" />
<title>Das MEVA-Lab  --  Labor für Medien und verteilte
Anwendungen   --  Log-In für Web-Dienste (intern)  
 </title>
<link rel="stylesheet" type="text/css" href="/serv-intra/meva.css"
```

```
title="Style" />
<link rel="SHORTCUT ICON" href="/serv-intra/favicon.ico" />
<meta name="robots" content="noindex,follow" />
</head>
<body class="body"
onload="if(parent.frames.length!=0)top.location=&#39;index.html&#39;;">

<a class="st">Login <strike>--</strike> Authentifizierung für die
Webdienste des MEVA-Lab<br /></a>
<br />
<form method="post" action="j_security_check" name="LogForm" onsubmit="return
chkFormular()">
<table border="0" cellspacing="0" cellpadding="2" id="form" width="544"
summary="Layout-Tabelle top" style="table-layout:fixed">
<tr><th width="170">Kontenname</th><th width="170">Passwort</th>
<th width="90" align="left">Weiter</th><th width="110">Nein Danke</th></tr>
<tr align="center"><td><input type="text" name="j_username" /></td>
<td><input type="password" name="j_password" /></td>
<td colspan="2" align="left"><input type="submit" value="Log in" />  
  <input type="reset" value="Clear"
onclick="document.LogForm.j_username.focus()"/>   <input type="button"
 value="Back" onclick="history.back();" /></td></tr>
</table></form>
<script type="text/javascript">
document.LogForm.j_username.focus();

function chkFormular () {
  if (document.LogForm.j_username.value == "") {
    document.LogForm.j_username.focus();
    return false;
  }
  if (document.LogForm.j_password.value == "") {
    document.LogForm.j_password.focus();
    return false;
  }
  return true;
}
</script>
<br />
<table border="0" cellspacing="0" cellpadding="2" id="lay1" width="544"
summary="Layout-Tabelle top" style="table-layout:fixed">
<tr><td width="11" > </td><td>
Ein Teil der
<a class="hrf" href="http://www.a-weinert.de/service/index.html">Web-Dienste</a>
des <a class="bbi"
href="http://www.a-weinert.de/meva-lab/index.html">MEVA</a>-Lab erfordert
Ihre Authenfizierung mit einem
<a class="hrf" href="http://www.a-weinert.de/service/account.html">Konto</a> der
<a class="hrf" href="http://www.a-weinert.de/service/domain.html">Domain FB3-
MEVA</a>.<br />
<br />
Geben Sie oben Ihren Kontennamen (ohne vorangehendes fb3-meva\) und Ihr
zugehöriges Passwort ein.<br />
<br />
Hinweis: Von der korrekten Eingabe von Konto und Passwort abgesehen können
das Log-In zu dem von Ihnen gewählten Dienst und ggf. dann die Möglichkeiten
damit von Ihrer Zugehörigkeit zu (Active Directory-) Nutzergruppen in der
<a class="hrf" href="http://www.a-weinert.de/service/domain.html">Domain FB3-
MEVA</a>
abhängen.<br />
<br />
```

```
<br />
Zur <a href="/" class="hrf">Server-Startseite.<br /></a>
<br />
Copyright   ©   2006,  
  <a href="http://www.a-weinert.de/weinert/" class="hrf">Albrecht Weinert</
a>,
   Stand: V.$Revision 1.0$ ($Date: 2007/10/24 14:20:21 $; FORMS-auth, J2EE)
</td></tr></table></body></html>
-----------------------------------------------------------------
```

Die Seite für den bedauerlichen Misserfolg:

---- fail_login.html ----------

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!--  Login-Seite  PD321S, PD3022 für FORMS Login, failure
      (nur hochschulintern)
      Version V.$Revision: 1.10 $, $Date: 2007/10/31 12:50:19 $
  -->
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<meta name="AUTHOR" content="Albrecht Weinert" />
<meta name="GENERATOR" content="Eclipse, CVSkeys.java" />
<meta http-equiv="content-language" content="de" />
<meta name="DESCRIPTION"
content="Labor für Medien und verteilte Anwendungen (MEVA-Lab), Login für
Webdienste (intern)" />
<meta name="Keywords"
content="Hochschule Bochum,MEVA,Weinert" />
<title>Das MEVA-Lab  --  Labor für Medien und verteilte
Anwendungen   --  Log-In für Web-Dienste (intern, failed)  
 </title>
<link rel="stylesheet" type="text/css" href="/serv-intra/meva.css" title="Style"
/>
<link rel="SHORTCUT ICON" href="/serv-intra/favicon.ico" />
<meta name="robots" content="noindex,follow" />
</head>
<body class="body"
onload="if(parent.frames.length!=0)top.location=&#39;index.html&#39;;">

<a class="st">Login <strike>--</strike> Authentifizierung für die Webdienste des
MEVA-Lab<br /></a>
<br />
<table border="0" cellspacing="0" cellpadding="2" id="lay1" width="544"
summary="Layout-Tabelle top" style="table-layout:fixed">
<tr><td width="11" > </td><td>
<form><a class="rg">Log-in-Versuch fehlgeschlagen / log in failed.   </a>
  <input type="button" value="Noch mal   /   try again"
onclick="history.back();" /><br /></form>
<br />
Ein Teil der
<a class="hrf" href="http://www.a-weinert.de/service/index.html">Web-Dienste</a>
des <a class="bbi" href="http://www.a-weinert.de/meva-lab/index.html">MEVA</a>-
Lab erfordert Ihre Authenfizierung mit einem
<a class="hrf" href="http://www.a-weinert.de/service/account.html">Konto</a> der
<a class="hrf" href="http://www.a-weinert.de/service/domain.html">Domain FB3-
MEVA</a>.<br />
<br />
```

```
Ihr Fehlversuch kann folgende Gründe haben<ol>
<li>Das angegebene Paar Kontennamen (ohne vorangehendes fb3-meva\ !) und
Passwort war falsch.</li>
<li>Der von Ihnen gewünschte Web-Dienst erfordert zusätzlich die Zugehörigkeit
des angegebenen Kontos zu bestimmten (Active Directory-) Nutzergruppen in der
<a class="hrf" href="http://www.a-weinert.de/service/domain.html">Domain FB3-
MEVA</a>
 (wie &quot;Kernteam&quot;, &quot;Admins&quot; o.Ä.).</li>
 <li>Sie haben Ihren Browser nicht angewiesen, das Zertifikate des Servers
 (PD321S, PD310S) zu akzeptieren.</ol>

Mit diesem
<a class="hrf"
href="https://pd321s/serv-intra/java/gwt/remote_meva_acc_c.html">Web-Dienst</a>
können Sie Ihr
<a class="hrf" href="http://www.a-weinert.de/service/account.html">Konto</a>
 ohne Domain-Authentifizierung prüfen.<br />

<br />
Zur <a href="/" class="hrf">Server-Startseite.<br /></a>
<br />
Copyright   ©   2006,  
  <a href="http://www.a-weinert.de/weinert/" class="hrf">Albrecht Weinert</
a>,
   Stand: V.$Revision 1.0$ ($Date: 2007/10/24 14:20:21 $; FORMS-auth, J2EE)
</td></tr></table></body></html>
```

A4 XSL-Transformator für Verzeichnislisten

Dies ist eine konfigurierbare Ergänzung (siehe Listing 5 auf Seite 13) zu Tomcat's DefaultServlet, die in ähnlichen Fällen als Anregung und Problemlöser dienen mag. Die dargestellte Lösung benötigt XSTL 2.0, das man für Tomcat im JDK nötigenfalls als saxon9.jar ergänzen kann.

```
----   (pd321s\serv-intra\)     meva-dir-li.xsl    -----------------
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE xsl:stylesheet [ <!ENTITY nbsp " "> ]>
<xsl:stylesheet
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:xs='http://www.w3.org/2001/XMLSchema'
 version="2.0">
<!--  MEVA-Lab Transformator für Tomcat-generierte directory listings
      Version V.$Revision: 1.10 $, ($Date: 2008/02/27 17:17:42 $)
      Copyright (c) 2008 Albrecht Weinert
      Version für den Server PD321S (Hochschule Bochum, Domain FB3-MEVA)
      Bewährt mit XSLT 2.0;
      Dies erfordert saxon9.jar und aWeinertBib.jar in
      C:\Programme\jdk\jre\lib\ext  für das  JDK und damit bei vernünftiger
      Konfigurierung auch gleich für Tomcat
-->
<xsl:output method="xhtml" indent="no" encoding="iso-8859-1" />

<xsl:template match="listing">
  <xsl:variable name="contextPath" select="@contextPath" />
  <xsl:variable name="theDir" select="@directory" />
  <xsl:variable name="hasParent" select="@hasParent" />
<!-- de.a_weinert.net.AdmHelper.getComputername() tut ihr Bestes, um
```

```
     den Namen des Rechners zu ermitteln auf dem dieses XSLT läuft.  -->
  <xsl:variable name="hostName"
      select="admhelper:getComputername()"
      xmlns:admhelper="java:de.a_weinert.net.AdmHelper" />


<html xmlns="http://www.w3.org/1999/xhtml"><head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<meta name="AUTHOR" content="Albrecht Weinert" />
<meta name="GENERATOR" content="Eclipse, Tidy, CVSkeys.java, XMLio.java" />
<meta http-equiv="content-language" content="de" />
<title> Directory Listing
        <xsl:value-of select="$theDir"/>  --  </title>
<link rel="stylesheet" type="text/css" href="/serv-intra/meva.css" title="Style"
/>
<link rel="SHORTCUT ICON" href="/serv-intra/favicon.ico" />
</head>
<body class="body">

<a class="st" id="hdl1" href="http://www.a-weinert.de/meva-lab/">Dateiliste
  (Webservice, MEVA-Lab)<br /></a>



<a class="kt" title="Zur Server-Startseite">
<xsl:attribute name="href">http://<xsl:value-of select="$hostName"/>/serv-intra/
</xsl:attribute>     Verzeichnis:  
<xsl:value-of select="upper-case($hostName)"/>
<xsl:value-of select="$contextPath"/><xsl:value-of select="$theDir"/><br /></a>
<br />
<table class="tw" summary="dir listing" cellpadding="4" width="780" >
<tr><th class="th" width="390">Name</th>
<th class="th" width="100">Größe</th>
<th class="th" width="260">Letzte Änderung</th></tr>

<xsl:if test="$hasParent != 'false'">
<tr class="tw">
  <td align="right"> <a class="hrf" href="../"> . . / </a></td>
  <td> </td><td><a class="hrf" href="../">   (ein Verzeichnis höher) </a></
td>
</tr>
</xsl:if>

<!-- filter:  readme.htm wird als Ergänzung gezeigt und .DS_Store ist
     MAC-generierter Unsinn. Passiert, wenn jemand mit einem MAC
     Dateien an Windows-Server liefert.  -->
<xsl:for-each
    select="entries/entry[(text() != 'readme.htm')
          and (text() != '.DS_Store') and (text() != 'Thumbs.db')]">
    <xsl:sort select="@type"  />
    <xsl:sort select="text()" />

 <tr>
   <xsl:choose>
     <xsl:when test="(position() mod 2 != 0) = ($hasParent != 'false')">
          <xsl:attribute name="class">tp</xsl:attribute>
     </xsl:when>
      <xsl:otherwise>
         <xsl:attribute name="class">tw</xsl:attribute>
      </xsl:otherwise>
   </xsl:choose>
```

```
   <td>
   <xsl:choose>
     <xsl:when test="@type = 'dir'">
           <xsl:attribute name="align">right</xsl:attribute>
     </xsl:when>
      <xsl:otherwise>
         <xsl:attribute name="align">left</xsl:attribute>
      </xsl:otherwise>
   </xsl:choose>
        <xsl:variable name="urlPath" select="@urlPath"/>
        <!-- work (dirty) around XSLT 2.0 attribute coding errors for URLs.
             Even with encoding=iso-8859-1 directives
             umlauts etc. are misscoded (probably Tomcat's UTF8 fault)   -->
        <xsl:variable name="cnt" as="xs:string" select="concat('./', text() )"/>
   <xsl:variable name="reLi" as="xs:string" select="replace($cnt, 'ö', '%F6')"/>
  <xsl:variable name="reLi" as="xs:string" select="replace($reLi, 'ä', '%E4')"/>
  <xsl:variable name="reLi" as="xs:string" select="replace($reLi, 'ß', '%DF')"/>
  <xsl:variable name="reLi" as="xs:string" select="replace($reLi, 'ü', '%FC')"/>
  <xsl:variable name="reLi" as="xs:string" select="replace($reLi, 'Ä', '%C4')"/>
  <xsl:variable name="reLi" as="xs:string" select="replace($reLi, 'Ö', '%D6')"/>
  <xsl:variable name="reLi" as="xs:string" select="replace($reLi, 'Ü', '%DC')"/>

    <a class="hrf"><xsl:attribute name="href"><xsl:value-of
                              select="$reLi"/></xsl:attribute>
          <tt><xsl:copy-of select="text()"/></tt>
        </a></td>
   <td align="right">
      <xsl:choose>
         <xsl:when test="@type = 'dir'">
           <tt>dir     </tt>
         </xsl:when>
         <xsl:otherwise>
        <tt><xsl:value-of select="@size"/>   </tt>
         </xsl:otherwise>
      </xsl:choose>
   </td>
   <td align="right">
   <!--  Schaun, ob verbessertes
         DefaultServlet, de.a_weinert.realm.DefaultServlet
         Albrecht Weinert, 12.02.2008, aktiv war. -->
       <xsl:variable name="dtt" select="@xsdatetime" />
      <xsl:choose>
         <xsl:when test="$dtt != ''">
           <tt>
   <xsl:value-of select="format-dateTime(
  adjust-dateTime-to-timezone($dtt), '[D01].[M01].[Y0001]   [H01]:[m01]')"
/>           </tt>
         </xsl:when>
         <xsl:otherwise>
           <tt><xsl:value-of select="@date"/>   </tt>
         </xsl:otherwise>
      </xsl:choose>
   </td>
  </tr>

</xsl:for-each>
</table><br />

<!-- there is one readme in Tomcats xml-listing or none -->
<xsl:for-each select="readme">
    <xsl:variable name="readFname" select="concat('http://', $hostName ,
```

```
$contextPath, $theDir, 'readme.htm')" />
    <hr size="1" /><br />
<a class="kt" href="#hdl1">Hinweise zu diesem Verzeichnis (aus <xsl:copy-of
select="$readFname" />)<br /></a>
  <br />
  <xsl:value-of disable-output-escaping="yes"  select="text()" />
 </xsl:for-each>

<hr size="1" />

<xsl:variable name="datuhr" select="current-dateTime()" />
<br />
XSL-T.ransformator:  Stand
 $Date: 2008/02/27 17:17:42 $ (V.$Revision: 1.9 $),    
 Copyright   ©   2008,    
 <a href="http://www.a-weinert.de/weinert/"
 class="hrf">Albrecht Weinert</a><br />
<br />Aktuelle Zeit:
 <xsl:value-of select="format-dateTime($datuhr, '[D]. [MNn] [Y] [h01]:[m01]',
'de', (), ())" />
<br />
</body></html>
</xsl:template></xsl:stylesheet>
```

A6 Abkürzungen

ACL	access control list (Liste mit Zugriffsrechten auf ein Objekt)
AD	Active Directory (Microsofts Interpretation von LDAP)
AJAX	Asynchronous JavaScript + XML
API	Application Programme Interface
BuB/S	Bedienen und Beobachten (von Prozessen) Client-Server
CA	Certification authority
FAQ	Frequently Asked Questions (Hilfetexte in Frage-Antwort-Form)
FB	Fachbereich; insbesondere ...
FB3	Fachbereich Elektrotechnik und Informatik der Hochschule Bochum
GSS	Generic Security Service
GUID	Globally Unique Identifier
GWT	Google Webtoolkit, AJAX mit nur Java
HTML	Hypertext Markup Language [RFC 1866]
HTTP	Hypertext Transfer Protokoll. Internet-Protokoll zur Übertragung von Seiten.
HTTPS	HTTP über SSL. Abgesicherte Übertragung.

HW Hardware

IIOP Internet Inter-ORB Protocol

IP Internet Protocol

J2EE Java 2 Enterprise Edition

J2ME Java 2 Micro Edition

J2SE Java 2 Standard Edition

JAAS Java Authentication and Authorization Service

JAF JavaBeans Activation Framework

JAR Java Archive. (.zip + Semantik)

JAXP Java API for XML Parsing

JCA Java Cryptography Architecture (der Java Security API)

JCE Java Cryptography Extensions (zur JCA, Exportrestriktion wegen DAS, DES)

JDBC Java Database Connectivity (Java Datenbankanschluss)

JDC Java Developer Connection (Ein WWW-Service)

JDK Java Development Kit; der Werkzeugsatz für die Entwicklung mit Java

JEB Enterprise JavaBeans (ungleich JavaBeans)

JMX Java Management Extensions

JNDI Java Naming and Directory services Interface

JNI Java Native Interface

JRE Java Runtime Environment; JDK-Subset ohne Entwicklungswerkzeuge.

JSDK Java Servlet Development Kit

JSF Java Server Faces

JSP Java Server Pages

JSSE Java Secure Socket Extension (seit JDK1.4.x integriert)

JSTL JavaServer Pages Standard Tag Library

JVM Java virtual machine; der eigens für Java erfundene Prozessor.
Er wird im Allgemeinen auf dem jeweiligen Zielsystem emuliert.

LAN Local area network; Datennetz für mittlere Entfernungen

LDAP Lightweight Directory Access Protocol

LGPL Lesser GNU Public License

MBean Managed Bean (JMX)

MEVA Labor für Medien und verteilte Anwendungen

MS	Microsoft
NT	Betriebssystem Windows NT (MS)
OMG	Object Management Group
OS	Operating System
PAM	Pluggable Authentication Module
PC	Personal Computer
R&D	Research and Development
RAID	Redundant Array of inexpensive Disks
RDF	Resource Description Framework (W3C)
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SAAJ	SOAP with Attachments API for Java
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SQL	Structured query language, Datenbankbearbeitungssprache
SSL	Secure Socket Layer. Protokollschicht zu Absicherung.
SSO	Single Sign on; Authentifizierung vieler (n) Anwendungen gegen eine (1) "security realm".
TCP	Transmission Control Protocol
TM	Trade Mark (Warenzeichen)
UML	Unified Modelling Language
URI	Uniform Resource Locator
W2K	Betriebssystem Windows 2000 (MS)
W2K3	Betriebssystem Windows Server 2003 (MS)
W3	Amerikanische Kurzform für WWW
W3C	World Wide Web Consortium
WS	Workstation
WSDL	Web Services Description Language
XML	eXtensible Markup Language

A7 Literatur

[1] Ed Ort and Mark Basler, AJAX Design Strategies, SUN 2006
http://java.sun.com/developer/technicalArticles/J2EE/AJAX/.../design-strategies.pdf

[2] Brett McLaughlin, Mastering Ajax, Part 1..4, IBM, 2005
http://www-128.ibm.com/developerworks/web/library/wa-ajaxintro.html

[3] Albrecht Weinert, Zur Installation des JDK (Java Development Kit)
http://a-weinert.de/pub/java-install.txt

[4] Albrecht Weinert, Java — Tipps und Tricks
http://a-weinert.de/pub/java-tips.txt

[5] Albrecht Weinert, AJAX mit GWT — Tipps und Tricks
http://a-weinert.de/pub/gwt-tips.pdf

[6] Albrecht Weinert, Tipps zu CVS für Windows — cvsNT
http://a-weinert.de/pub/cvsnt-tipp.txt

[7] Google, Web-Toolkit, online-Dokumentation (nicht am Stück verfügbar)
http://code.google.com/webtoolkit/documentation/.

[8] Albrecht Weinert, Tipps zu JMX mit SSL
http://a-weinert.de/pub/jmx-ssl-tips.pdf

[9] Albrecht Weinert, Windows 2003 Domain
Migration von NT4 mit Fremd-DNS
http://www.a-weinert.de/pub/w2k3domain.pdf

[10] Albrecht Weinert, Windows Server 2003 — Domain FB3-MEVA
Schulungsräume und Infrastruktur — Renovierung 2007
http://www.a-weinert.de/pub/fb3-meva-domain2007.pdf

[11] Albrecht Weinert, Tipps zu Tomcat (5.x für Windows) ersetzt durch [13]
([13] stattdessen für Tomcat >= 6) http://a-weinert.de/weinert/pub/tomcat-tips.pdf

[12] Albrecht Weinert, Windows Server 2003 — Domain FB3-MEVA
Workstations und Server — Renovierung 2007
http://www.a-weinert.de/pub/fb3-meva-workst2007.pdf

<13> Albrecht Weinert, Tomcat — mit Windows und Active Directory
(ersetzt [11] als Nachfolger) http://www.a-weinert.de/pub/tomcat-win-ad.pdf

[14] Albrecht Weinert, Tipps zu MySQL (mit Java, für Windows)
(2006) http://a-weinert.de/pub/mysqjawi-tipp.txt

[16] Albrecht Weinert, Subversion — mit Windows und Active Directory (2008)
(ersetzt [15]) http://www.a-weinert.de/pub/subversion-win-de.pdf