

Albrecht Weinert

# Tutorial

Tipps zu Tomcat — für Windows



Stand: 25.04.2008





Albrecht Weinert

Labor für Medien und verteilte Anwendungen (MEVA-Lab)

Fachbereich Informatik der FH Bochum

## Tipps zu Tomcat — für Windows

V01.01, 19.12.2005 10:37: neu  
V01.05, 24.08.2006 16:00: Tomcat Vers. 5.5.17; https; SSL  
V01.09, 30.08.2006 20:28: **Active Directory Integration**  
V01.15, 12.10.2006 09:24: **Ergänzungen aufgrund aktueller Erfahrungen**  
V01.18, 16.10.2006 13:43: dto.,; JNDIRealm  
V01.20, 16.02.2007 11:33: Hinw. auf GWT & AJAX  
V01.22, 20.04.2007 14:34: **kleinere Ergänzungen; keystore, userright, Umleitung**  
V01.24, 28.08.2007 10:35: **Hinweis auf neue Versionen**  
V01.25, 21.11.2007 08:23: **Hinweis FORMS / BASIC**  
V01.26, 10.12.2007 17:15: **JNDIRealm- / AD- Probleme, Klärung**  
V01.28, 25.01.2008 08:40: **XSL und directory listing**  
V01.29, 01.02.2008 16:29: **Erste Hinweise für Tomcat 6.0 (.14)**  
V01.30, 11.02.2008 17:30: **Tomcat 6.0 (.14) verdeutlicht; w. JNDIRealm-Hinw.**  
V01.31, 19.02.2008 05:50: **ADweRealm statt JNDIRealm für Active Directory**  
V01.23, 20.02.2007 11:26: **kleinere redaktionelle Änderungen**  
V01.25, 25.04.2008 08:52: **Hinweis auf [13] als Tomcat6-Nachfolger; Abschluss**

Version: V1.25

Zuletzt geändert von A. Weinert am 25.04.2008

Copyright (c) 2007 Albrecht Weinert. All rights reserved. [a-weinert.de](http://a-weinert.de)

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

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

Hinweis auf Nachfolgeveröffentlichungen: Für Neu-Installationen und Renovierungen mit  
Tomcat 6.0.x aus Java6 greifen Sie zu [13].  
Die URL jenes Dokuments ist: <http://www.a-weinert.de/weinert/pub/tomcat-win-ad.pdf> .

## Inhalt

1. Zweck, Voraussetzungen .....	3
1.1 Ziel .....	3
1.2 Voraussetzungen .....	4
2. Server - Grundinstallation .....	6
3. Fester Inhalt — static content .....	9
4. Statische Startseite.....	13
5. HTTPS — SSL und Nutzerauthentifizierung.....	14
5.1 Vorbemerkungen.....	14
5.2 https und ssl, Hintergrundinformation.....	14
5.3 https und ssl für Tomcat, einfach.....	15
5.4 Erzwingen von https (und von Nutzer-Authentifizierung).....	17
5.5 User-Authentifizierung für Tomcat, einfachst.....	18
6. Ein Servlet.....	19
6.1 Das erste Servlet.....	19
6.2 https und Authentifizierung für ein Servlet.....	21
6.3 Abschließende Anmerkung zum Servlet.....	21
7. Active Directory Integration.....	22
7.1 Ziele und Hintergründe — Konten, Gruppen, Rollen, Rechte .....	22
7.2 Active Directory.....	23
7.2 Lesender JNDI-Zugriff auf AD.....	23
7.3 Test des lesenden LDAP- / JNDI-Zugriffs auf AD.....	24
7.4 AD-Integration in Tomcat.....	26
7.5 AD-Integration — der steinige Weg mit JNDIRealm.....	27
7.6 Authentifizierungs-Abfragen in Servlets.....	30
7.7 Work-around um JNDIRealm-Probleme und die AD-Rollensuche.....	31
7.8 AD-Integration — der Lösungsweg mit ADweRealm.....	35
8. Résumé .....	37
8.1 Erreicht mit Tomcat 5.5 (6.0) .....	37
8.1 Was bleibt zu tun.....	37
8.3 Umstieg von Tomcat 5.5 zu Tomcat 6.0.....	38
Anhang .....	39
A1 Quelle des HelloWorld-Servlets .....	39
A2 Descriptor des HelloWorld-Servlets .....	41
A3 Beispielseiten für FORMS-Authentifizierung .....	43
A4 XSL-Transformator für Verzeichnislisten .....	46
A5 Einige Hinweise zu Tomcat 6.0.14 .....	49
A6 Abkürzungen .....	49
A7 Literatur .....	52

# 1. Zweck, Voraussetzungen

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

Dieses Tutorial 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).

Es geht um die ersten Schritte „ab Null“ — gerade eines Tomcat-Neulings. Allerdings wollen auch Fachleute hier schon 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 \*2). 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\*): 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

Dieses Tutorial beschreibt die Inbetriebnahme eines

- Apache-Tomcat als "stand alone" Webserver.

Dieser soll

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

Letzteres erlaubt die Unterstützung von Geschäfts- und anderen Prozessen (Datenbank-anbindung, 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 für das Vorliegenden 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.

24.08.2006	06:23	5.090.572	apache-tomcat-5.5.17.exe
24.08.2006	06:23	2.340.873	apache-tomcat-5.5.17-admin.zip
24.08.2006	06:23	1.623.862	apache-tomcat-5.5.17-compatible.zip
28.08.2007	09:45	5.074.169	apache-tomcat-5.5.23.exe
28.08.2007	09:46	2.333.977	apache-tomcat-5.5.23-admin.zip
15.12.2007	15:49	5.126.497	apache-tomcat-5.5.25.exe
15.12.2007	15:50	2.374.359	apache-tomcat-5.5.25-admin.zip
31.01.2008	07:58	5.269.478	apache-tomcat-6.0.14.exe
24.02.2008	14:26	5.401.005	apache-tomcat-6.0.16.exe

Liste 1: Tomcat-Installationsdateien in unterschiedlichen Versionen  
(Quelle: <http://tomcat.apache.org/> .... download).

## 1.2 Voraussetzungen

- a.) Sie haben alle Installationsdateien zu (genau) einer Tomcat-Version aus der Liste 1. Jede der dort aufgeführten Versionen wurde für das Vorliegende erprobt. Für Neuinstallationen empfiehlt sich 6.0.14, .16 (+ Java6) oder jüngere Versionen. Zum Umstieg von 5.5.x auf 6.0.x siehe die Hinweise am Ende und im Anhang A5 (Seite 49).
- 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 Testinstallation auf einer W2K3-Workstation (i.A. Ihrem Entwicklungsplatz) ist sinnvoll. Vieles, nicht Alles, lässt sich auch mit "localhost" klären.
- b.) Dieser Server (PD321S) ist im Netzwerk für alle Ihre Client-Rechner zugänglich und hat genügend Plattenplatz für Alles, was Sie da via HTTP oder HTTPS als statischen Inhalt oder als Webservice anbieten wollen. Für die Integration mit AD hat der (Tomcat-) Server Zugriff auf mindestens einem Domain-Controller. 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\_03 oder höher auf diesem Server installiert. Ein JRE allein genügt i.A. nicht. Wenn Sie bei Java 2 bleiben wollen, bleiben Sie auf Tomcat 5.0.x beschränkt. Ein JDK 1.4.2\_10 (mit 5.5.17 und compat.zip) reicht dann entgegen diverser Installationshinweise, die Java 5 verlangen.

- e.) Diese JDK-Installation ist mit Allem ergänzt, was Sie für Ihre Anwendungen, Servlets etc. benötigen.  
Gemeint sind hier Dinge wie das Framework de.a\_weinert, JDBC-Treiber, COMM-Extensions, Java-mail etc. pp., die sinnvollerweise als "installed extensions" ergänzt wurden; siehe hierzu u.A. auch  
<http://www.a-weinert.de/java/aweinertbib.html>  
und  
<http://www.a-weinert.de/weinert/pub/java-install.txt> [3]).

Das genannte Framework de.a\_weinert (aWeinertBib.jar 21.02.2008 oder jünger) bringt — neben vielen anderen Segnungen — auch die Basis für eine vernünftige, letztlich einzig wirklich lauffähige Intergration von Active Directory  
Zu solchen sinnvollen Ergänzungen zählt auch XSLT 2.0 (statt nur 1.0), das man beispielsweise durch saxon9.jar nachinstallieren kann

- f.) Ihr Suchpfad weist auch auf die Werkzeuge dieses korrekt installierten und nach Ihren Bedürfnissen ergänzten JDK (siehe e.).  
Der Programmsuchpfad beginnt also etwa so:  
Path=C:\bat;C:\Programme\util;C:\programme\jdk\bin;C:\WIND.....

Folgende Systemumgebungsvariablen sind korrekt gesetzt  
ANT\_HOME=C:\Programme\Apache\ant  
JAVA\_HOME=C:\Programme\jdk

- 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 3.3). Sie stellen alle Quell- und Hilfsdateien unter eine Versionsverwaltung (z.B. cvsNT [6]). 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-) Konfigurationsdateien 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 wohl 5.5.09 und 5.5.17 Versionen gibt, bei denen https bzw. SSL einfach nicht gehen. Deinstallieren Sie ggf. solche Versionen und nehmen Sie am besten keine ältere als die in Liste 1. als älteste aufgeführte 5.5.17. Die 6.0.x-Versionen verlangen eine abweichende Konfiguration für HTTPS sowie unbedingt Java 5 oder am besten gleich Java6.

Allgemeine Anmerkung: 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.

## 2. Server - Grundinstallation

Zum Installieren nehmen Sie nur das dankenswerterweise vorbereitete "Windows-executable" — hier also beispielsweise apache-tomcat-5.5.25.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 einfach 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 muss natürlich dem Zielrechner angepasst bzw. ausgewechselt werden.

Hinweis zu Tomcat 6.0.x: Wenn Sie bei Neu-Installationen / Renovierungen Tomcat 6 auf Java6 (und Windows mit Active Directory) einsetzen wollen, greifen Sie statt des Vorliegenden zum [13].

Lassen Sie mit administrativen Rechten

```
24.08.2006 07:23 5.090.572 apache-tomcat-5.5.xy.exe
```

auf dem Ziel-Server (hier PD321S) laufen.

Stimmen Sie den Lizenzbedingungen zu, setzen Sie alle Häkchen für eine vollständige Installation. Ändern Sie das Zielverzeichnis in

```
C:\Programme\Apache\Tomcat 5.5\
```

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
```

Hier können Sie bei 5.5.17 ggf. auch eine JDK4-Installation verwenden, auch wenn von Installer ausdrücklich ein jre1.5 verlangt wird.

Verboten Sie den sofortigen Start und schließen Sie den Installer (sprich obiges

```
apache-tomcat-5.5.xy.exe)
```

mit "finish" ab.

Anmerkung zum Tomcat-Administrator-Passwort — Achtung Falle:

Dieses während der Installation vergebene Passwort ist völlig losgelöst von Ihren Windows- bzw. Domain-Konten. Tomcats Administratorname und -passwort landen, so wie von Ihnen vergeben, im Klartext in einer (.xml-) Konfigurationsdatei. Diese Sonderbehandlung jeder Anwendung mit Klartextspeicherung von Einzelkonteninformationen für jede Anwendung ist eine — aus Sicht eines Windows-Administrators unfassbare — Linux-Gewohnheit des Programms. Also hier keinesfalls echte (Domain-, AD-) Administrator- Konteninformationen wiederholen. Mit der AD-Integration werden diese "Privateinstellungen" glücklicherweise 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 praktisch allen Konfigurationen und Dokumentationen fast durchgängig die Ports 8080 bzw. 8443 eingetragen. Dies ist hier nicht nur lästig, sondern recht fehlerträchtig.



Entsprechende Hinweise weiter unten unbedingt beachten.  
Diese Installationsabfrage ist also eher eine Falle als eine Hilfe..

Weiter in der Installation:

Entpacken Sie nun die Dateien

```
apache-tomcat-5.5.xy-admin.zip
```

und nur im Falle der Version 5.5.17 (xy=17) auch

```
24.08.06 07:23 1.623.862 apache-tomcat-5.5.17-compat.zip
```

in dasselbe temporäre Hilfsverzeichnis mit den Kommandos:

```
cd \temp
jar xfv <downloadVerzeichnis>\apache-tomcat-5.5.xy-admin.zip
jar xfv <downloadVerzeichnis>\apache-tomcat-5.5.17-compat.zip
```

Letztere brauchen Sie nur für xy=17 und nur, falls Sie nicht komplett auf Java 5 oder 6 umgestellt haben.

Das obige Auspacken mit jar.exe erzeugt ein Unterverzeichnis

```
\temp\apache-tomcat-5.5.xy
```

Verschieben Sie (mit XCopy oder mit Explorer-drag&drop) den kompletten Inhalt dieses Verzeichnisses ohne jedes "wenn und aber" nach

```
C:\Programme\Apache\Tomcat 5.5\
```

Damit ist die Grundinstallation des Tomcat-Servers abgeschlossen.

Hinweis zu apache-tomcat-5.5.17-compat.zip: Das Mitinstallieren dieser Datei auch bei JDK1.5.0\_06 scheint allerdings unschädlich zu sein. Wer, wie es inzwischen zu empfehlen ist, konsequent JDK 1.6.0\_03 (oder höher) einsetzt, sollte diese Datei aber weglassen

Nota bene: Man kann auch JDK 1.6 einsetzen, wenn man aus guten Gründen nur 1.4-Spracheigenschaften nutzen will.

Starten Sie unter

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

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

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 weiter navigieren 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 5.5\logs
```

zu setzen. Schließlich ist es gute Praxis, mehr statische Programminstallation (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 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 5.5\logs\ .
```

**Bei der Gelegenheit: Falls Sie in den 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, lassen Sie die Java-Einstellungen in Ruhe! Versuchen Sie es mit dem Auskommentieren von

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

in der Datei

```
C:\Programme\Apache\Tomcat 5.5\conf\server.xml
```

```
:stopLok
@Echo Lokalen Tomcat stoppen
@echo.
net stop Tomcat5
@if /I %1==--tomstopp      goto :ende

:delLokLog
@Echo Lokale Tomcat-Logs löschen
@echo.
java Del "C:\Programme\Apache\Tomcat 5.5\logs/+.+"

:startLok
@Echo Lokalen Tomcat starten
@echo.
net start Tomcat5
@goto :ende
```

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

Stopp, Start und die Fehlermeldung sollte nicht mehr auftreten.

#### 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. Allein schon 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-Dienstverwaltung.

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 (jedenfalls nicht erkennbar) dokumentiert.

Beispielhafter Anwendungsfall:

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

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

"abwärts" ein Abbild (Spiegel) 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 diesen "Spiegel" unter

```
http://pd321s/fb3/
```

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

```
http://pd321s/fb3/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 5.5\conf\Catalina\localhost\fb3.xml
```

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

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

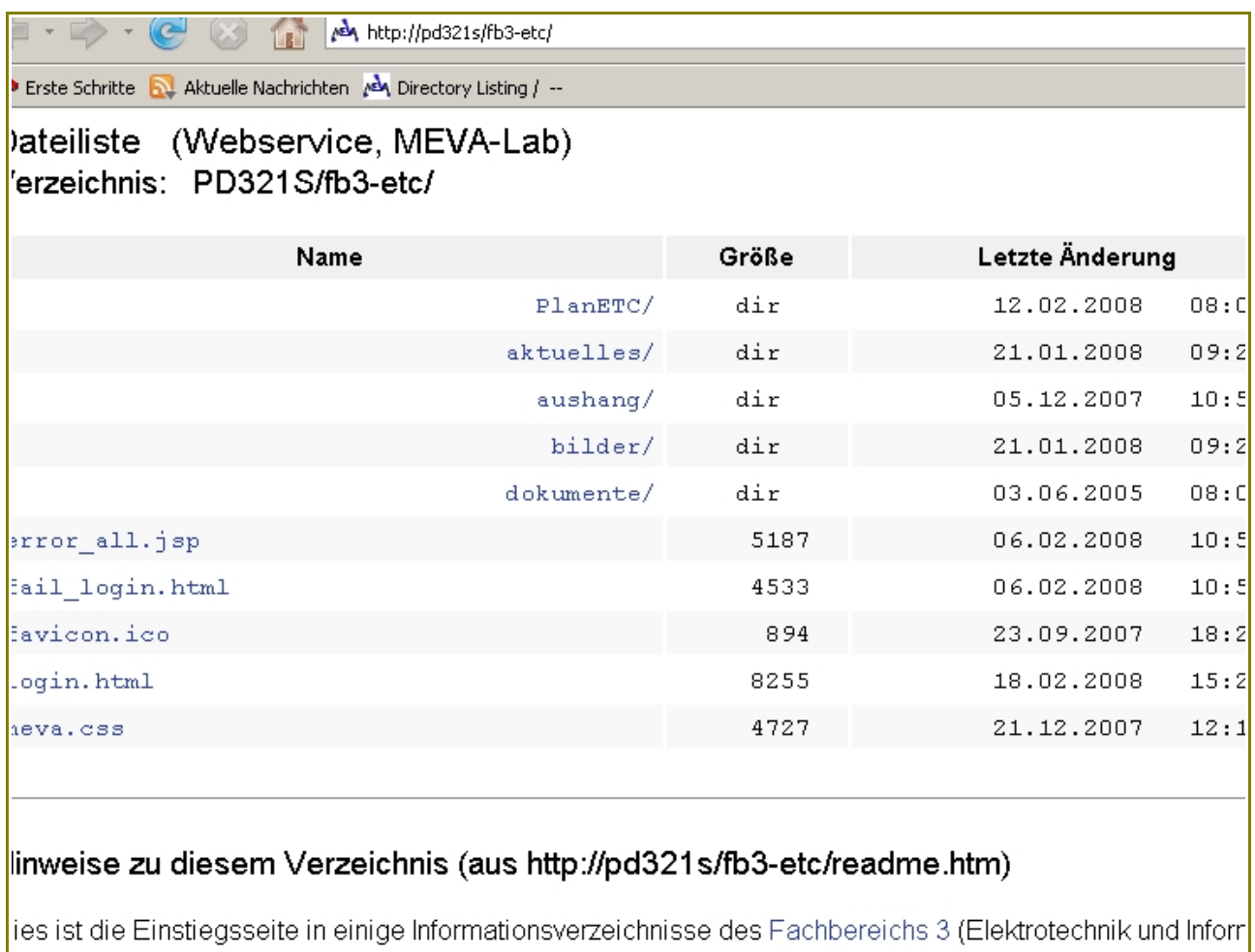
Listing 3: Descriptor in der Tomcat-Konfiguration für statischen Content.

Weitere Bereiche ("Contexte") lassen sich u.A. nach dem gleichen Schema, also mit jeweils einer solchen Mini-.xml-Datei, ergänzen.

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

```
docBase="D:\www\fb3\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.



Verzeichnis: PD321S/fb3-etc/

Name	Größe	Letzte Änderung
PlanETC/	dir	12.02.2008 08:00
aktuelles/	dir	21.01.2008 09:20
aushang/	dir	05.12.2007 10:50
bilder/	dir	21.01.2008 09:20
dokumente/	dir	03.06.2005 08:00
error_all.jsp	5187	06.02.2008 10:50
fail_login.html	4533	06.02.2008 10:50
favicon.ico	894	23.09.2007 18:20
login.html	8255	18.02.2008 15:20
meva.css	4727	21.12.2007 12:10

[Linweise zu diesem Verzeichnis \(aus http://pd321s/fb3-etc/readme.htm\)](#)

Dies ist die Einstiegsseite in einige Informationsverzeichnisse des [Fachbereichs 3](#) (Elektrotechnik und Informatik).

Bild 4: Directory Listing (mit default servlet + eigener XSL-Transformator).

Zu oft wünschenswertem statischem Inhalt zählt am auch eine Verzeichnisliste, wie sie beispielsweise Bild 4 zeigt. Diese Zuordnung ist eigentlich nicht ganz richtig, wenn das Listing bei jeder Anfrage aus sich evtl. veränderten Verzeichnisinhalten neu (dynamisch) generiert wird. Tomcats für die Lieferung von statischem content zuständiges default servlet erledigt diese Aufgabe auch, falls man es ihm gemäß Listing 5 erlaubt.

```

<servlet>
  <servlet-name ... Anfabg lassen, wie es war ... /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.

Diese Aufforderung bzw. Lizenz zum Verzeichnis auflisten (gemäß Listing 5) gilt pauschal für alle Verzeichnisse und über sämtliche gemäß Listing 3 definierten sogenannten "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 default servlets (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.
- b) Verzeichnisse, in denen sich eine der für Tomcat konfigurierten (siehe Listing 6) Verzeichnisstartdateien befindet, werden nicht gelistet. Das default servlet 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 zwei 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 default servlet selbst gelieferte directory listing nicht passt oder genügt. Möglicherweise störende Mängel sind:

- Die default-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 zum dann ja vorhandenen Elternverzeichnis.
- Sie löst keine Probleme mit Umlauten in Dateinamen, die gerade in Aushang- und Notizverzeichnissen unkontrollierbar vorkommen mögen.
- Sie generiert solche Probleme eher noch.

Durch Umprogrammieren des default-servlets ließe sich da natürlich Alles machen. Es gibt aber einen anderen Weg. Tomcat's default-servlet generiert die Verzeichnis-liefert die Verzeichnisinformation als XML und liefert deren html-Darstellung anschließend durch eine (eigene) XSL-Transformation. Wie in Listing 5 (Seite 11) bereits dargestellt, lässt sich das default servlet hier eine andere Transformation unterschieben. Die hier gewählte, die zu einer in Bild 4 gezeigten Darstellung führt und die aufgeführten Probleme angeht. Dieses transformierende style sheet finden Sie als Anregung im Anhang.

Eine solche Lösung leidet darunter, dass das vom default-servlet gelieferte XML schlecht und undokumentiert ist.

Schlecht ist das XML in dem Sinne, dass es wenig Information liefert und diese nicht als Daten sondern als amerikanisierte Textdarstellung. "4.4 kb" und "Fri, 21 Dec .." steht so und nur so in der XML-Verzeichnisinformation und ist nicht das gewollte Ergebnis der XSL-Transformation! Das ist in jeder Hinsicht "daneben" und XML-widrig.

Und "undokumentiert" meint, dass man das wenige Nützliche, was sonst noch drinsteht durch Lesen der .java-Quelle des default-servlets 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.real.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).

Eine entsprechend geänderte catalina.jar (auf Basis Tomcat 6.0.14) bzw. eine 6.0.x-versionsunabhängige catErgWe.jar finden Sie unter [www.a-weinert.de/java/index.html](http://www.a-weinert.de/java/index.html).

## 4. 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 5.5\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 10) "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 5.5\webapps\ROOT\index.html
```

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

```
<html><head>
  <title> PD321S - Einfachste Startseite &nbsp; &nbsp; </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.

## 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 einem 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 der Betrieb einer eigenen Zertifizierungsstelle (inhouse CA) oder 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-house-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):

```
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\Apache\Tomcat 5.5\conf\.keystore"
-validity 9999
```

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

```
C:\Programme\Apache\Tomcat 5.5\conf\.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 von dem) das Zertifikat ist, sonst 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 einen eigenen keystore nur für Tomcat an. Dies ist evtl. sinnvoll oder gar notwendig, da nach etwas unklarer Dokumentationslage Tomcat nur ein (1) Zertifikat im benutzten keystore verkräftet und auf der Gleichheit [sic!] von keystore- und Schlüsselwort besteht.

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.

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\Apache\Tomcat 5.5\conf\.keystore"
```

In der Datei

```
C:\Programme\Apache\Tomcat 5.5\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\Apache\Tomcat 5.5\conf\.keystore" />
```

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.

Hinweis: Falls Sie für den betreffenden Server bereits ein Zertifikat haben — beispielsweise für Java, JMX etc. — verwenden Sie dieses natürlich auch für Tomcat. Kopieren Sie dazu den Keystore des Servers (beispielsweise `pd321s-keystore` aus dem entsprechenden JRE-Verzeichnis) ins entsprechende Tomcat-Verzeichnis und ergänzen / ändern Sie den obigen Connector-Eintrag in server.xml mit folgenden zwei Attributen (Beispiel):

```
<Connector port="443"
  ....(wie oben)....
  keystoreFile="C:\Programme\Apache\Tomcat 5.5\conf\pd321s-keystore"
  keyAlias="pd321s" />      <!-- Alias kann i.A. entfallen -->
```

Wenn bei Ihnen Alles (s.o.) passt, können Sie natürlich auch das Kopieren des „Keystores“ lassen und den ursprünglichen Ort der Datei eintragen, also beispielsweise:

```
C:\Programme\jdk\jre\lib\security\pd321s-keystore
```

Wenn Sie Linux-Gewohnheiten beibehalten wollen, müssen Sie ggf. entsprechend zu folgendem Kommando greifen, da der Explorer kein Umbenennen zu ".xyz" erlaubt:

```
ren ".....\pd321S-keystore" .keystore
```

## 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 5.5\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 5.5\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 5.5\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 zunächst die (Servlet-) Quelle HelloWorld.java aus dem Anhang. 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 5.5\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 5.5\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 9 gezeigte Datei- und Verzeichnisstruktur.

Dabei ist `web.xml` die im Anhang aufgeführte einfache Descriptor-Datei (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 9: 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.

## 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 Vorfürungen

"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 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, -passwörtern und -rechten, wie sie sich bei Tomcat in der Datei

```
C:\Programme\Apache\Tomcat 5.5\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 ausdrucksstarke 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, Jahrgangsguppen 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. verbotbaren 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-Rechtevergabe 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", Anmeldenname "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 5.5\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 wieder 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

Tabelle10: Die Tomcat – AD - Abbildung.

Die Tomcat-Rollenamen "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 ("admin" und "manager") oft bereits anderweitig als Nutzer oder Gruppen-Namen vergeben. Auch würden sie im großen Domain-Kontext allgemeiner aufgefasst, und kein Mensch würde sie je mit Tomcat in Verbindung bringen.

Als erstes ändert man also diese Tomcat-Rollenamen 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 5.5\server\webapps\host-manager\WEB-INF\web.xml
```

```
C:\Programme\Apache\Tomcat 5.5\server\webapps\admin\WEB-INF\web.xml
```

```
C:\Programme\Apache\Tomcat 5.5\server\webapps\manager\WEB-INF\web.xml
```

```
C:\Programme\Apache\Tomcat 5.5\conf\tomcat-users.xml
```

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 dokumentationsgemäß, 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 5.5\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 5.5\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.

Eine bewährte Reparatur bietet das Java-Framework `de.a_weinert..` mit der Methode

```
public boolean isUserInRole(
    HttpServletRequest request,
    CharSequence role, InitialLdapContext readCtx)
```

in der einfach zu handhabenden Klasse

```
de.a_weinert.net.LDAPauthRead
```

im Paket `de.a_weinert.net`. Siehe als Ausgang die die Paket-Beschreibung

[http://www.a-weinert.de/java/docs/aWeinertBib/de/a\\_weinert/net/package-summary.html](http://www.a-weinert.de/java/docs/aWeinertBib/de/a_weinert/net/package-summary.html)

Diese Methode (aber nicht die mit `CharSequence user` als erstem Parameter) verwendet zunächst auch einfach das oben erwähnte

```
request.isUserInRole(role) //,
```

gibt sich aber mit einer (wohl sicheren; s.o.) negativen Antwort nicht zufrieden. Dann wird mit dem im `LDAPauthRead`-Objekt hinterlegten Verbindungs- und (Domain-spezifischen) Sucheinstellungen nach der erfragten Rollenzuordnung mit LDAP-Zugriffen gesucht.

Und diese Suche wird auch indirekt im Sinne von "Rolle in Rolle" = "group-in-group" . weitergeführt. Damit sind die oben genannten Einschränkungen der Abbildung einer vernünftigen AD-Gruppen-Handhabung auf Tomcat-Rollenabfragen weg.

Sprich `de.a_weinert.net.LDAPauthRead.isUserInRole(...)` (löst bei richtiger Planung; siehe oben im Kapitel 5.5) alle genannten Probleme.

Dieses Verfahren funktioniert (unter Einhaltung der Tomcat-AD-Hinweise des letzten Kapitels) so gut, dass noch ein schwerwiegender Effekt aufgedeckt wurde:

Die Abfrage nach dem erwähnten `session-principal`, letztlich mit



```
Principal principal = request.getUserPrincipal();
```

lieferte nach schwer durchschaubaren Kriterien des Öfftens `null`.

Nun ja. Wenn man (das Servlet) nicht mal den eingeloggten Nutzer erfährt, ist man die Rollenabfragen und ihre Probleme auch gleich los. Vielleicht war das ja Absicht (Scherz).

Und wohlgemerkt: Der authentifizierte Nutzer war und blieb bei Auftreten dieses Fehlers in der Browser-Session mit all seinen Rechten und credentials eingeloggt. Hiervon konnte man sich zum Einen durch einfache Experimente leicht überzeugen. Und zum Anderen ist es bekannt, dass man diese Sitzungsrechte außer durch Beenden des Browsers praktisch nicht loswird – das ist ja genau das schmerzliche Fehlen einer `sessionReset()`-Funktion.

Die Lösung nun dieses Problems – geboren aus (miss-) interpretierten J2EE-Spezifikationen – ist erschreckend einfach: Man verwende FORMS- statt BASIC-Authentifizierung und lese hierzu die Hinweise im Anhang A3.

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

Die (oben erwähnte) redundante Rollensuche im AD ist, wie gesagt, für das Authentifizieren von Rollen wie

```
<role-name>Kernteam</role-name>
<role-name>tcAdmin</role-name>
```

etc. leider notwendig. Mit dem in diesem Kapitel erläuterten Vorgehen lässt sich das vermeiden. Das Vermeiden zeigt sich symptomatisch an einem verkürzten Eintrag in

```
C:\Programme\Apache\Tomcat 5.5\conf\server.xml :
```

```
<!-- 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"
/>
```

Listing 11: Vereinfachte "Eine Richtung" Konfiguration von JNDIRaelm.

Hinweis noch mal: Diese erfreuliche Kürzung ist nur der Anfang, das Symptom.

Ohne die in diesem Kapitel beschriebenen weiteren Maßnahmen geht so gar nichts.

Zwei positive Effekte seien als Einstiegsmotiv für die so notwendigen folgenden Maßnahmen gleich genannt:

1. Das so konfigurierte Wegfallen der role-Sucherei schlägt sich direkt in einer besseren Leistung beim Einloggen nieder.
2. Außerdem löst man ein — in diesem Falle wohl von Microsoft zu verantwortendes — Problem gleich mit:

Das Rückwärts Auflösen via `roleSearch="(member={0})` der Rollen = Gruppen-Mitgliedschaft funktioniert — soweit mit JNDIRalm überhaupt — nur in sehr kleinen Domains. Sobald eine Gruppe 1500 Mitglieder oder mehr hat, liefert das AD via LDAP anstelle von

```
member CN=meier ....  
member CN=müller ....
```

nun plötzlich

```
member;range=0..1499 CN=meier ....  
member;range=0..1499 CN=müller ....
```

Als Konsequenz kann das Hinzufügen von einem einzigen neuen Nutzerkonto zum AD bzw. in eine Gruppe (ausgeführt vom AD-Administrator, der mit dem Tomcat-Chef der Web-Dienste ja nichts in Persona zu tun haben muss) mit einem Schlag die ganze Tomcat-Authentifizierung "knacken". Eine typische "Gestern ging es noch"-Falle, der man mit dem in diesem Kapitel empfohlenen Vorgehen auch aus dem Weg geht.

## Maßnahmen zur Lösung / Vermeidung der Tomcat-JNDIRealm-Probleme

Das Maßnahmenbündel sieht so aus

1. `de.a_weinert.net.LDAPAuthRead.isUserInRole(...)` in Servlets konsequent nehmen. Dies liefert auch indirekte Rollen ("group in group").
2. JNDIRealm nur noch in einer Richtung konfigurieren (siehe oben). Das heißt also: Nur noch in den Nutzern schauen.
3. als Konsequenz von Tomcat's JNDIRealm-Mangel: Nur noch AD-Gruppen für `<auth-constraint>` verwenden, in denen Nutzer direkt Mitglied sind.
4. als Konsequenz von 2: Als `<security-role>s` in Tomcat's und sonstigen `web.xml`-files Rollen mit vollem AD-Namen definieren.

Dabei die für das Tomcat-Management zuständigen unterhalb von

```
C:\Programme\Apache\Tomcat 5.5\server\webapp
```

nicht vergessen und mit "verlängerten" `<role-name>s` `tcAdmin` und `tcManager` versehen.

5. als Konsequenz von 2. und 4: Auch in `<auth-constraint>s` die vollen AD-Rollenamen nehmen.
6. als Konsequenz von 5: Konkrete Rollenangaben in `<auth-constraint><role-name>` nur dann machen, wenn genau eine konkrete direkte Rolle oder maximal zwei das Servlet oder den WWW-Bereich schützen sollen.

7. als Konsequenz von 6: Alle anderen Servlets / Webdienste mit Authentifizierungs-Anforderung mit beliebiger AD-Rolle (und damit beliebigem AD-Nutzer) als Browser-session-Login schützen.

UND (!) die konkrete Tat im Servlet dann über jeweils passende Abfragen mit `de.a_weinert.net.LDAPauthRead.isUserInRole()` schützen.

Zusätzlicher Vorteil: Dann gehen, wie erwähnt, endlich auch "role-in-role" als "group-memberof-group".

Hinweise:

Zu 1.): Das sieht in der Servlet-Quelle an den entsprechenden Stellen sinngemäß so aus:

```
// declarations
import de.a_weinert.net.LDAPauthRead;

String ldap2URL = "ldap://193.175.115.2:389";
LDAPauthRead fb3MEVAad;

// init
fb3MEVAad = new LDAPauthRead("AD von FB3-Meva", ldap2URL,
    "CN=ldap leser,CN=Users,DC=FB3-MEVA,DC=fh-bochum,DC=de",
    "lpassword", true, "FB3-MEVA\\", null);

// MEVA-Lab - AD- LDAP (auch role-Provider für TomCat via JNDIRealm)
fb3MEVAad.setUserRoleCriteria("DC=FB3-MEVA,DC=fh-bochum,DC=de",
    "cn=", true, "memberOf", "cn=");

// use
fb3MEVAad.isUserInRole(request, s, mevaReadCtx)
```

Das ist Alles.

Zu 2.): Siehe Listing 11 / Nimm einfach den oben gezeigten verkürzten Eintrag für

```
<Realm className="org.apache.catalina.realm.JNDIRealm" ...
```

in Tomcat's server.xml.

Zu 4.): Beispielsweise

```
<security-role>
  <role-name>CN=CAX,CN=Users,DC=FB3-MEVA,DC=fh-bochum,DC=de</role-name>
</security-role>
```

zusätzlich zu nur

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

Wann braucht man die "Lang-" und wann die "Kurz-" Fassung der LDAP-/AD-Bezeichnung der Gruppe als Rolle bzw., wie im Beispiel angedeutet, beide?

- Die Langfassung benötigt man, da sie in irgendwelchen `<auth-constraint>`s genau so vorkommt muss, damit nur Nutzer, die der betreffenden Gruppe (direkt) angehören für die Browser-Sitzung zugelassen werden.
- Die Kurzfassung benötigt man (bei Tomcat undokumentiert aber experimentell geklärt) im Falle "`<auth-constraint><role-name>*</role-name><..>`" (siehe unten unter 7). Jeder AD-Nutzer, der sich für eine so (sprich für zunächst beliebige Rollen) geschützte Browser-Sitzung korrekt authentifiziert muss (dennoch) zusätzlich mindestens einer in Kurzfassung genannter Gruppe (direkt!) angehören [sic!].

Hinweis: Die Groß-/Kleinschreibung der Rollennamen muss sich 1 zu 1 nach der Schreibweise im AD / LDAP richten. Im Beispiel mit korrektem "`<role-name>CN=CAX,CN=U...`" würde die Kurzfassung "`<role-name>cax</role-name>`" nicht funktionieren.

Hinweis 2: Hierzu wäre evtl. noch zu klären, ob dies über Aliase, `<role-link>` oder dergleichen auch eleganter geht.

Zu 5.): Voller AD-Name, sprich der "lange" / "distinguished" `<role-name>` ist ggf. auch in `<security-constraint>`s einzusetzen (anstatt oder zusätzlich).

Grund: AD-Auth mit JNDIRealm funktioniert mit einfachem Namen (via `roleName="cn"` z.B.) nur bei der redundanten Rückwärtsauflösung über die Gruppen.

Zu 7.): Authentifizierung der Browser-session mit Authentifizierung eines beliebigen AD-Nutzerkontos erreicht man so (im betreffenden. `web.xml`); Beispiel:

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

Entscheidend ist das einfache

```
<role-name>*</role-name>
```

welches eine Authentifizierung (mit FORMS, siehe oben) durch ein jedes AD-Nutzerkonto erlaubt, welches eine direkte in Tomcat's web.xml ("lang") hinterlegte direkte AD-Gruppenzugehörigkeit hat.

Nochmals der Hinweis: Diese Gruppen dürfen nun — dank nur noch "Vorwärtssuche"— beliebig groß sein, so dass eine oder wenige Gruppen alle Domain- / AD-Nutzerkonten erfassen können.

Hinweis am Rande: Die Rollenabfrage mit `request.isUserInRole(s)` (die letztlich in `org.apache.catalina.realm.RealmBase / .GenericPrincipal` abgearbeitet wird) funktioniert unter den geschilderten Konfigurationen bestenfalls mit ausführlichen Gruppennamen, und, wie "CN=CVSUsers,CN=Users,DC=FB3-MEVA,DC=fh-bochum,DC=de", und, wie gesagt, auch nur dann, wenn der Nutzer direktes Mitglied ist.

Die im vorigen Kapitel beschriebene AD-Integration läuft einigermaßen stabil. Richtig stabil allerdings erst mit den in diesem Kapitel gerade beschriebenen Maßnahmen, welche am Besten recht früh bei der Planung / der Architektur von Web-Diensten berücksichtigt werden sollten.

## 7.8 AD-Integration — der Lösungsweg mit ADweRealm

- 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` 21.02.2008 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 – und dies Alles auf evtl. Tomcat-versionsabhängiger Basis.

Ja, man sollte das Problem an der Wurzel zu packen. Dies geht aber 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/aWeinertBib/de/a\\_weinert/realm/package-summary.html](http://www.a-weinert.de/java/docs/aWeinertBib/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.

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](#).)

Dies wiegt 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:
- nur noch kurze Rollen reichen und man scheint die `<security-role>`-Auflistungen (endlich!) ganz weglassen zu können.

```
<!-- 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 leser,CN=Users,DC=FB3-MEVA,DC=fh-bochum,DC=de"
connectionPassword="****"

userBase="DC=FB3-MEVA,DC=fh-bochum,DC=de"
defaultRole="fb3-meva_user"

/>
```

Listing 12: Die einfache Konfiguration von ADweRealm.

## 8. Résumé

### 8.1 Erreicht mit Tomcat 5.5 (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

### 8.1 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 herausragende von vielen — war eher mühseligem, ja "schmerzhaftem" Suchen und nicht geordnetem, und nicht Hersteller-dokumentierten 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.

2.) Im Zusammenhang mit der AD-Integration wäre ein strenges Review der Klasse `org.apache.catalina.realm.JNDIRealm` dringend von Nöten. Die in Kapitel 7.7 beschriebenen Work-arounds und die vielen weiteren damit verbundenen Vorteile nehmen dem Punkt aber etwas die Dringlichkeit.

Letztlich sinnvoll anstelle von work-arounds, ist die Lösung an der Wurzel durch Einsatz der Klasse

`de.a_weinert.realm.ADweRealm`

anstelle von `JNDIRealm`. Siehe dazu das Ende von Kapitel 7.7

- 3.) Das Anwendungen den eher benutzerobflächlich-geschmacklichen Unterschied von BASIC- und FORMS-Authentifizierung merken, ist ähnlich bearbeitungsbedürftig.
- 4.) Der eigentliche professionelle Einsatz der J2EE mit allen umfangreichen Möglichkeiten und Aufgaben.  
Dies hier ist kein J2EE-Tutorial sondern eines über  
"Wie bringe ich Tomcat als J2EE-Basis in einer  
industriegängigen Umgebung zum Laufen?"

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 — viel, viel besser in einer industriellen Windows-Umgebung nutzen als manch anderes eher schlecht als recht nach Windows portiertes Linux-C-Produkt.

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

- 5.) JSF-Integration (zusätzliche .jar-Files- in die Tomcat-Installation, ...)
- 6.) Schöne Fehlerseiten und ihre Probleme (todo: Beschreiben!).
- 7.) Feedback welcome (a-weinert.de).

### 8.3 Umstieg von Tomcat 5.5 zu Tomcat 6.0

Erste Erfahrungen mit

31.01.2008 07:58 5.269.478 apache-tomcat-6.0.14.exe

zeigen einige Unterschiede zu 5.5.: Die Verzeichnisstruktur ist etwas anders, ebenso die Handhabung von HTTPS. Hier wird offenbar eine getrennte Zertifikatdatei gewünscht. Das Verzeichnis conf\Catalina\localhost fehlt (kann aber einfach wie für 5.5 angelegt werden).

web.xml und server.xml erfordern auf jeden Fall ein paar Änderungen gegenüber unter 5.5.x lauffähigen Versionen. Am besten passt man sie auf Basis der neu installierten an.

Ein Fehler des neuen DefaultServlet lässt das Auflisten von Verzeichnissen "crashen"!  
(Bug Nr. 44337). Dieser Bug ist mit dem für Directory-Listings eh besseren:

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

gleich mit erledigt.

Auf keinen Fall kann man einer einem laufenden (komplexeren) J2EE-Projekt einfach Tomcat 6.0.x statt 5.5.y "unterschieben". (Ein kleines Experiment mit einer 5.5-Installation kurz vor deren Wegwerfen zeigte, dass es umgekehrt u.U. geht.)

Tomcat 6.0 implementiert die Spezifikationen Servlet 2.5 und JSP 2.1 statt 2.3 bzw. 1.2. Wann die einhergehenden Verbesserungen und Klarstellungen, u.A. bei session-handling und servlet-mapping, den Aufwand eines Umstiegs rechtfertigen, kann nur am Einzelfall entschieden werden.

Hinweise und Erfahrungen dazu finden Sie in [13], das Sie bei Neueinstig mit / Umstieg nach Tomcat 6 auf Java 6 statt des Vorliegenden ([11]) nehmen sollten.



## 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 erzeugt 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 &nbsp;&nbsp;&nbsp; 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 &nbsp; (Beispiel-Servlet) &nbsp;"
&nbsp;</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("Authentication: ");
    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>
</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 5.5\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 &#160;--&#160; Labor für Medien und verteilte
Anwendungen &#160;--&#160; Log-In für Web-Dienste (intern) &#160;
&#160;</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 <del>--</del> 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" /> &nbsp;
&nbsp; <input type="reset" value="Clear"
onclick="document.LogForm.j_username.focus()" /> &nbsp; <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 Authentifizierung 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 &#160; © &#160; 2006, &#160;
&#160; <a href="http://www.a-weinert.de/weinert/" class="hrf">Albrecht Weinert</
a>,
&nbsp; 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 &#160;--&#160; Labor für Medien und verteilte
Anwendungen &#160;--&#160; Log-In für Web-Dienste (intern, failed) &#160;
&#160;</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. &nbsp; </a>
&nbsp; <input type="button" value="Noch mal &nbsp; / &nbsp; 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 Authentifizierung 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="href" 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="href"
href="https://pd321s/serv-intra/java/gwt/remote_meva_acc_c.html">Web-Dienst</a>
können Sie Ihr
<a class="href" href="http://www.a-weinert.de/service/account.html">Konto</a>
ohne Domain-Authentifizierung prüfen.<br />

<br />
Zur <a href="/" class="href">Server-Startseite.<br /></a>
<br />
Copyright &#160; © &#160; 2006, &#160;
&#160; <a href="http://www.a-weinert.de/weinert/" class="href">Albrecht Weinert</
a>,
&nbsp; 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 11) zu Tomcats default-servlet, die in ähnlichen Fällen als Anregung und Problemlöser dienen mag. Die dargestellte Lösung benötigt XSTL 2.0, das man für Tomcat im JDK 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 "&#160;"> ]>
<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"/> &#160;--&#160; </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
&#160;; (Webservice, MEVA-Lab)<br /></a>

<a class="kt" title="Zur Server-Startseite">
<xsl:attribute name="href">http://<xsl:value-of select="$hostName"/>/serv-intra/
</xsl:attribute>   Verzeichnis: &#160;;
<xsl:value-of select="upper-case($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=".."> &#160;; (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="$reLi"/></xsl:attribute>
    <tt><xsl:copy-of select="text()"/></tt>
  </a></td>
<td align="right">
  <xsl:choose>
    <xsl:when test="@type = 'dir'">
      <tt>dir &#160; &#160; </tt>
    </xsl:when>
    <xsl:otherwise>
      <tt><xsl:value-of select="@size"/> &#160; </tt>
    </xsl:otherwise>
  </xsl:choose>
</td>
<td align="right">
<!-- Schaun, ob verbessertes
DefaultServlet, de.a_weinert.realm.DefaultServlet
Albrecht Weinert, 12.02.2008, aktiv war. -->
  <xsl:variable name="dtt" select="@xsdatetime" />
  <xsl:choose>
    <xsl:when test="$dtt != ''">
      <tt>
        <xsl:value-of select="format-dateTime(
adjust-dateTime-to-timezone($dtt), '[D01].[M01].[Y0001] &#160; [H01]:[m01]')"
/>
        &#160; </tt>
      </xsl:when>
      <xsl:otherwise>
        <tt><xsl:value-of select="@date"/> &#160; </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: &#160;Stand
  $Date: 2008/02/27 17:17:42 $ (V.$Revision: 1.9 $), &#160; &#160;
  Copyright &#160; © &#160; 2008, &#160; &#160;
  <a href="http://www.a-weinert.de/weinert/"
  class="hrf">Albrecht Weinert</a><br />
<br />Aktuelle Zeit:
  <xsl:value-of select="format-dateTime($datuhr, '[D]. [MNn] [Y] [h01]:[m01]',
'de', (), ())" />
<br />
</body></html>
</xsl:template></xsl:stylesheet>

```

---

## A5 Einige Hinweise zu Tomcat 6.0.14

Dieser Anhang ist entfernt: Die Veröffentlichung für den Einstieg mit Tomcat 6.0.16 oder jünger auf Java 6 (JDK1.6\_10 oder jünger) ist [13]. In diesem Sinne ist [13] der Nachfolger des Ihnen Vorliegenden [11].

## 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	Bedienen und Beobachten (von Prozessen)
CA	Certification authority
FAQ	Frequently Asked Questions (Hilfetexte in Frage-Antwort-Form)
FB	Fachbereich; insbesondere ...
FB3	Fachbereich Elektrotechnik und Informatik der FH Bochum
FH	Fachhochschule
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

IIOIP Internet Inter-ORB Protocol

IP Internet Protocol

J2EE Java 2 Enterprise 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/weinert/pub/java-install.txt>
- [4] Albrecht Weinert, Java — Tipps und Tricks  
<http://a-weinert.de/weinert/pub/java-tips.txt>
- [5] Albrecht Weinert, AJAX mit GWT — Tipps und Tricks  
<http://a-weinert.de/weinert/pub/gwt-tips.pdf>
- [6] Albrecht Weinert, Tipps zu CVS für Windows — cvsNT  
<http://a-weinert.de/weinert/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/weinert/pub/jmx-ssl-tips.pdf>
- [9] Albrecht Weinert, Windows 2003 Domain Migration von NT4 mit Fremd-DNS  
<http://www.a-weinert.de/weinert/pub/w2k3domain.pdf>
- [10] Albrecht Weinert, Windows Server 2003 — Domain FB3-MEVA Schulungsräume und Infrastruktur — Renovierung 2007  
<http://www.a-weinert.de/weinert/pub/fb3-meva-domain2007.pdf>
- <11> Albrecht Weinert, Tipps zu Tomcat (für Windows) ([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/weinert/pub/fb3-meva-workst2007.pdf>
- [13] Albrecht Weinert, Tomcat — mit Windows und Active Directory (ersetzt [11] als Nachfolger) <http://www.a-weinert.de/weinert/pub/tomcat-win-ad.pdf>

Hinweis: Aus Dateien „.../docu/\*.txt“ könnten inzwischen teilweise „.../weinert/pub/\*.pdf“ geworden sein.

Hinweis: [13] ersetzt in vieler Hinsicht das Vorliegende [11] als Nachfolger.