IN DIESER KURSEINHEIT

- Einleitung
  - *Installation von Jakarta1*
  - *Was bedeuten all diese Einstellungen und wo kriege ich die SW her?*
- Jakarta
  - *Enable ROOT Context.*
  - *Enable Invoker Servlet*
  - *Port 80*
  - *Servlet Reloading*
  - *`JAVA_HOME` Variable*
  - *DOS Memory Settings*
  - *`CATALINA_HOME` Variable*
  - *Verifizieren der Server-Installation*
  - *Einfache Tests*
- Einrichten Ihrer Entwicklungsumgebung
  - *Allgemeines*
  - *`CLASSPATH`*
- Einfache Test Servlets
  - *Ein Servlet ohne Package*
  - *Ein Servlet mit Packages*
  - *Ein Servlet mit Packages und Hilfsklassen*
- Weitere Tipps
  - *Benutzer 'Manager' einrichten*
  - *Ant als universelles Java Make*
- Das war's

1. *Tomcat installieren Starthilfe*

1.1.Einleitung

Wieder einmal musste ich Servlets für Tests eines XML basierten Java Programms entwickeln und ärgerte mich, keine aktuellen Unterlagen über Installation usw. zur Verfügung zu haben.

Dieses Skript schliesst diese Lücke: ich schreibe einfach auf, was ich alles gemacht habe, bis alles so lief, wie ich es mir so vorstellte.
Damit wir uns verstehen: es geht um die Installation und Konfiguration von Apache Tomcat 4 oder 5 und Servlets Release 2.3 sowie JSP 1.2 (bei Tomcat 4) bzw. Servlet Release 2.4 sowie JSP 2.0 (bei Tomcat 5). Das Einbinden der Servlet Engine in den Standard Web Server von Apache oder andere allgemeine verfügbare Web Server ist komplizierter, und ich brauch's im Moment nicht.

1.2.Installation von Jakarta

Diese Kurzversion hilft Ihnen, falls Sie genügend Erfahrungen gesammelt haben und keine unnötigen Erklärungen lesen möchten:

1. **Installation des J2SDK**
   ich habe die Version 1.4.2 von Java installiert und hoffe, dass einige der grösseren Probleme aus 1.4.1 beseitigt sind (XML war besonders schlimm, aber auch java.net). Dabei habe ich Standardverzeichnisse gewählt:
   `C:\j2sdk1.4.2` und `JAVA_HOME` darauf gesetzt. Bei Eingabe von `java` wird die JVM aus dem JRE gestartet (`Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2-b28)`)

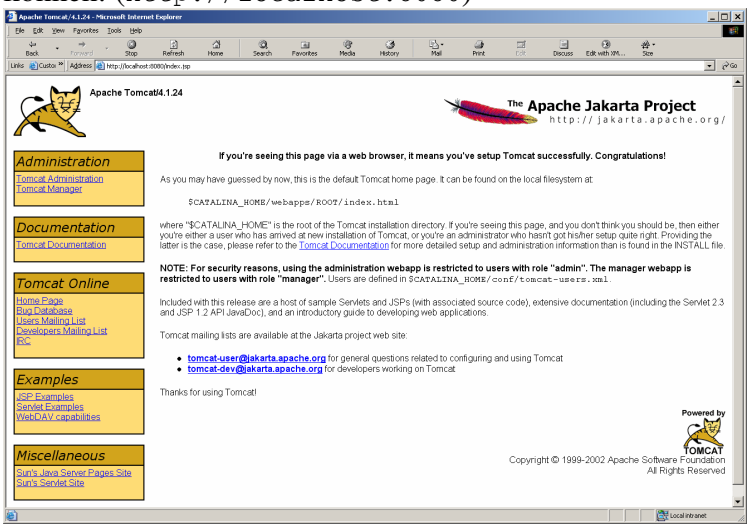
© Josef M. Joller

2. Tomcat

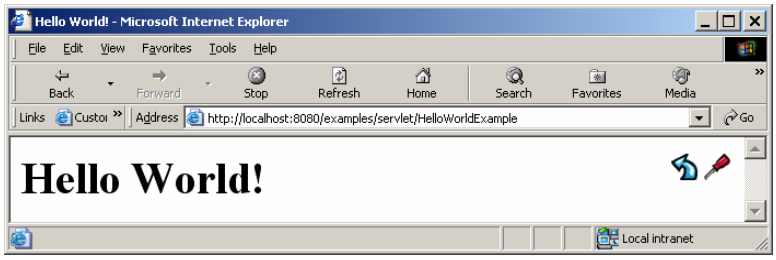
1) **Herunterladen der Software**. Je nach Mirror haben Sie nur Zugriff auf stabile Releases (CNLab-Switch) oder vollen Zugriff, auf auch Beta Releases. Die Version 5 enthält einige Verbesserungen (Performance, Deployment, JMX, JNDI Support).Zur Zeit teste ich Version Jakarta-tomcat-5.0.16.

2) Die Software ins Verzeichnis `C:\jakarta-tomcat-xyz` **entzippen**
(oder ein Verzeichnis Ihrer Wahl), beispielsweise `C:\jakarta-tomcat-5.0.16`
ODER
2) jakarta-tomcat-xyz.exe starten (in diesem Fall wird das J2SDK gesucht, eine installierte Version als Standard vorgeschlagen und Tomcat ins Programm Verzeichnis installiert).
Im Folgenden benutzen wir einfach die Umgebungsvariable
`CATALINA_HOME` = `C:\Program Files\Apache Group\Tomcat xyz`
beispielsweise `C:\Program Files\Apache Software Foundation\Tomcat` **`5.0`**
bzw. `C:\Program Files\Apache Group\Tomcat` **`4.1`**
oder
`CATALINA_HOME` = `C:\jakarta-tomcat-4.1.29`
bzw. `C:\jakarta-tomcat-5.0.16`

Machen Sie einen **Kurztest**:
Im Verzeichnis `%CATALINA_HOME%\bin` finden Sie eine Batch-Datei zum Starten von Tomcat: `startup.bat`. Dann sollten Sie bereits auf den Web Server zugreifen können: (`http://localhost:8080`)



Wechseln Sie (mithilfe des Menüs) zu den Servlet Beispielen. Auch diese funktionieren, wenigstens das Hello World Beispiel:



© Josef M. Joller

3) **Umgebungsvariable**:
Setzen der Umgebungsvariable `CATALINA_HOME` wie oben erwähnt.

4) **ROOT Context:**
In der Konfigurationsdatei `%CATALINA_HOME%\conf\server.xml` die auskommentierte Einstellung `<Context path="" docBase="ROOT" debug="0"/>` aktivieren. Dieses Element befindet sich ziemlich am Ende von `server.xml`.
In Version 5 ist das Element fehlerhaft (<.../> der Slash fehlt).

5) **Invoker Servlet:**
In der Konfigurationsdatei `%CATALINA_HOME%\conf\web.xml` die auskommentierte Einstellung

```
<!--     <servlet-mapping>
         <servlet-name>invoker</servlet-name>
         <url-pattern>/servlet/*</url-pattern>
     </servlet-mapping>
-->
```

aktivieren (beachten Sie, dass es mehrere Mappings gibt. Sie müssen jene mit * aktivieren).
In Tomcat 5 müssen Sie zudem den in `web.xml` weiter oben stehenden Invoker Servlet Teil aktivieren, sonst wird das Servlet nicht gefunden:

```
  <!-- The "invoker" servlet, which executes anonymous servlet
classes that have not been defined in a web.xml file. Traditionally,
this servlet is mapped to URL pattern "/servlet/*", but you can map
it to other patterns as well.  …The extra path info portion of such a
...-->
<servlet>
         <servlet-name>invoker</servlet-name>
         <servlet-class>
           org.apache.catalina.servlets.InvokerServlet
         </servlet-class>
         <init-param>
             <param-name>debug</param-name>
             <param-value>0</param-value>
         </init-param>
         <load-on-startup>2</load-on-startup>
     </servlet>
```

6) **Standard-Port**:
Falls Sie anstelle des Tomcat Standard-Ports 8080 lieber den Port 80 hätten, können Sie dies in der Datei `%CATALINA_HOME%\conf\server.xml` anpassen:

```
<!-- Define a non-SSL Coyote HTTP/1.1 Connector on port 8080 -->
    <Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
               port="80" minProcessors="5" maxProcessors="75"
               enableLookups="true" redirectPort="8443"
               acceptCount="100" debug="0" connectionTimeout="20000"
               useURIValidationHack="false"
disableUploadTimeout="true" />
```

In Version 5 fehlt die Angabe der Klasse (`className=…`).

© Josef M. Joller

7) **Servlet Reloading:**
Datei `%CATALINA_HOME%\conf\server.xml` anpassen:

```
<!-- Define properties for each web application.  This is only needed
   if you want to set non-default properties, or have web application
   document roots in places other than the virtual host's appBase
   directory.  -->
```

Diese Zeile existiert nicht; Sie müssen diese einfügen!

```
<!-- neu -->  <DefaultContext reloadable="true" />  <!-- neu -->
```

3. **Testen Sie die Installation**, wie oben
(Ihr HTML Root: `%CATALINA_HOME%\webapps/ROOT`).
Ich habe eine mehr oder weniger leere HTML Seite, mit eigenem Titel und einer Überschrift in dieses Verzeichnis kopiert. Da jetzt der Port auf 80 ist, wird diese Seite über `http://localhost` angezeigt:

© Josef M. Joller

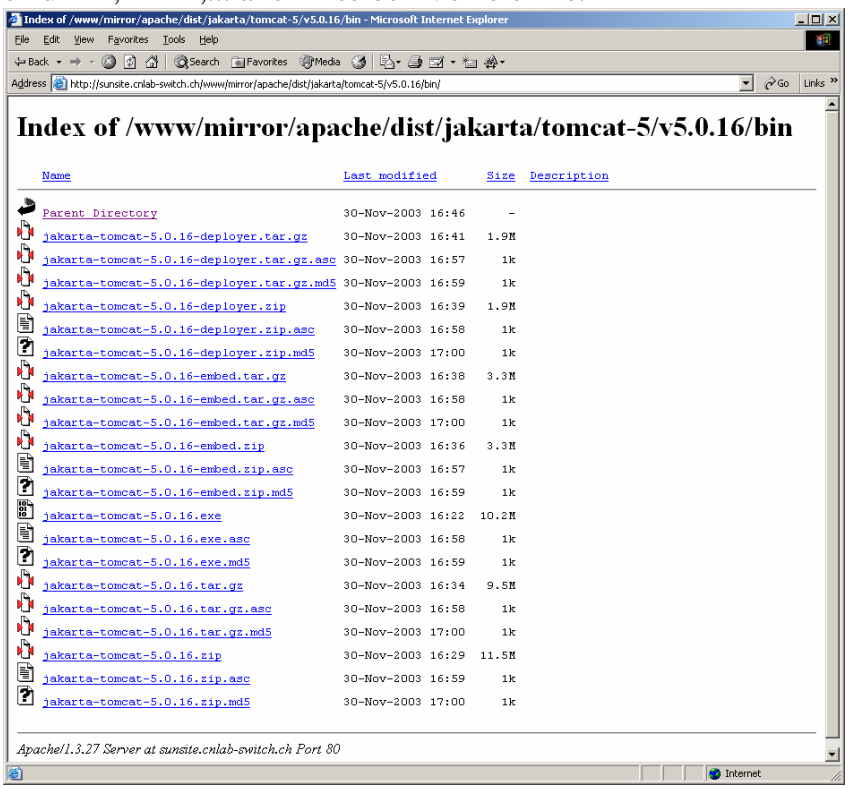
1.2.1. Was bedeuten all diese Einstellungen und wo kriege ich die SW her?

1.2.1.1. Jakarta

Sie können Jakarta in mehreren Versionen herunterladen:

- als ZIP Datei: entweder übersetzt oder im Quellcode
- als EXE Datei mit Installer: entweder mit J2SDK oder ohne

Falls Sie lieber mit dem Installer arbeiten, müssen Sie wie oben beschrieben, ins Verzeichnis `binary` wechseln. Auf der Apache Web Site sieht die Verzeichnisstruktur anders aus. Dort sind ZIP, EXE ,... alle im selben Verzeichnis:



Bezeichnenderweise setzt der Mirror Betreiber auf Apache 1.3.27, nicht auf neuer Versionen.

1.2.2. Enable ROOT Context.

Indem wir die Standard-Web Applikation in Tomcat enablen, können wir Tomcat einfacher nutzen (Servlets und JSP's). Der Eintrag im Konfigurationsfile `server.xml`:

```
<Context path="" docBase="ROOT" debug="0"/>
```

besagt, dass wir den tiefsten Debug Level wollen (höhere Level zeigen mehr Details). Die Ausgabe wird in den Logger geschrieben. 0 ist der Standardwert.

Die entsprechende Zeile ist in Version 5 falsch:

```
<Context ... debug="0" >
```

statt

```
<Context ... debug="0" />).
```


© Josef M. Joller

1.2.3. Enable Invoker Servlet

Das Invoker Servlet vereinfacht ebenfalls das Testen: wenn wir dieses Servlet einschalten, dann können wir die Servlets einfach ins Verzeichnis `WEB-INF/classes` kopieren, ohne irgend eine Änderung an `web.xml` vornehmen zu müssen. Die Servlets stehen dann unter `http://localhost/servlet/ServletName` zur Verfügung. Aus Sicherheitsgründen sollten Sie beim praktischen Einsatz diese Einstellung nicht verwenden!
ACHTUNG: in Tomcat 5 müssen weitere Bereiche unkommentiert werden (siehe oben).
In der Datei `install_dir/conf/`**`web`**`.xml` (nicht `.../`**`server`**`.xml` und auch nicht `web.xml` in den Web Applikations-Verzeichnissen `WEB-INF`):

```
<servlet-mapping>
    <servlet-name>invoker</servlet-name>
    <url-pattern>/servlet/*</url-pattern>
</servlet-mapping>
```

1.2.4. Port 80

Falls Sie noch keinen andern Web Server auf Port 80 haben, vereinfacht diese Einstellung die Schreibweise: statt `http://localhost:8080/`... Schreiben Sie nur noch `http://localhost/`...

```
<Connector
  className="org.apache.coyote.tomcat4.CoyoteConnector"
  port="80" ...
  ... />
```

1.2.5. Servlet Reloading

Diese Option ist sehr hilfreich:

- indem Sie diese einschalten versieht der Server jedes Servlet mit einem Erst-Installationsdatum;
- wenn Sie nun eine neue Version laden, erkennt der Server dies und aktiviert die neuste Version.

Vorteil: Sie brauchen sich nicht darum zu kümmern, dass jeweils die neuste Version aktiv ist.
Nachteil: ein erhöhter Memory-Bedarf.

Einstellung:
In `conf/server.xml` muss ein neues `DefaultContext` Subelement dem `Service` Element hinzugefügt werden, und dieses muss ein `reloadable` Attribut besitzen, welches auf `true` gesetzt sein muss.

Nach dem Kommentar:

```
<!-- Define properties for each web application.  This is only needed
     if you want to set non-default properties, or have web application
     document roots in places other than the virtual host's appBase
     directory.  -->
```

fügen wir die Zeile:

```
<DefaultContext reloadable="true"/>
```

ein.
ACHTUNG:
Machen Sie jeweils eine Kopie der Konfigurationsdateien bevor Sie irgend etwas ändern!

© Josef M. Joller

1.2.6. JAVA_HOME Variable

Da ich in vielen Anwendungen unterschiedliche Versionen von J2SDK einsetze, verwende ich nie eine Version, die sich im `PATH` befindet.

Meine aktuelle J2SDK Version ist 1.4.2. Deswegen habe ich (Win2000) unter Control Panel, System die Umgebungsvariable JAVA_HOME darauf gesetzt:

```
JAVA_HOME=C:\j2sdk1.4.2_01
```

Bei Win98 und ähnlichen, können Sie die selbe Angabe im `AUTOEXEC.BAT` machen.

Was Sie auch machen könnten wäre in `Catalina.bat` (im `bin` Verzeichnis von Tomcat) eine Abfrage einzubauen:

```
if not "%JAVA_HOME%" == "" goto gotJavaHome
set JAVA_HOME=C:\j2sdk1.4.2_01
:gotJavaHome
```

Auch hier sollten Sie zuerst eine Backup Kopie von `catalina.bat` machen.

1.2.7. DOS Memory Settings

Falls Sie eine Fehlermeldung wegen zuwenig Speicher erhalten, könnten Sie Ihr DOS Memory anpassen:

- Bei der Fehlermeldung `"Out of Environment Space"`können Sie in `install_dir/bin/startup.bat`, "Initial Environment" von Auto auf 2816 setzen
- Vielleicht müssen Sie die selbe Änderung auch in `install_dir/bin/shutdown.bat` vornehmen.

1.2.8. CATALINA_HOME Variable (optional)

Ich persönlich arbeite sehr viel mit Umgebungsvariablen. Deswegen finde ich es sehr praktisch mit `CATALINA_HOME` zu arbeiten.

Je nach Ihrer Installation und Version von Tomcat ist diese natürlich anders:

- Apache Tomcat 4 installiert in der exe Version ins Verzeichnis <Programme>\Apache Group
- Apache Tomcat 4 dagegen installiert in der exe Version ins Verzeichnis C:\Program Files\Apache Software Foundation\Tomcat 5.0
- Falls Sie die ZIP Version entzippen und das Standardverzeichnis (c:\...) wählen, sieht die Umgebungsvariabel wieder anders aus.

© Josef M. Joller

1.2.9. Verifizieren der Server-Installation

Wie oben beschrieben:

- starten Sie den Server mithilfe des Skripts im `bin` Verzeichnis.
- Testen Sie einige der bereits vorhandenen Beispiele.
- Fahren Sie den Server mithilfe des Skripts `shutdown.bat` runter.

Damit Sie effizient arbeiten können, sollten Sie Tomcat in Ihre Entwicklungsumgebung einbinden oder mindestens Shortcuts auf die beiden oben erwähnten Skripts erstellen.

1.2.10. Einfache Tests

Nun sollten Sie die Installation gründlich testen. Ein erster Test ist das Anzeigen einer HTML Seite. Kopieren Sie die Seite `index.html`, `hello.html` und `hello.jsp` aus dem Beispielverzeichnis ins Tomcat Root: `%CATALINA_HOME%\webapps\ROOT`.

Starten Sie nun den Tomcat Server (`startup.bat` im `%CATALINA_HOME%\bin`). Sie sollten eine fast leere Seite (`index.html`) mit zwei Links (`hello.html` und `hello.jsp`) sehen. Beide Links sollten funktionieren.

Falls Sie mit dem Link auf die Java Server Page Probleme haben, stimmt etwas mit `JAVA_HOME` nicht. Falls Sie irgend ein Package Java Programm verwenden, kann dies zu grösseren Suchaktionen führen: Sie müssen immer den Package Namen als Präfix für die Klassen verwenden!

Kreieren Sie einfach noch ein Verzeichnis im Web Root und testen Sie mit den selben Seiten nochmals:
Jetzt müssen Sie statt `http://localhost/Hello.html` und `http://localhost/Hello.jsp`
`http://localhost/directoryName/Hello.html` und
`http://localhost/directoryName/Hello.jsp` aufrufen.

Wenn Sie den Fehler `404` erhalten, dann stehen Ihre Webseiten im falschen Verzeichnis.

© Josef M. Joller

1.3.1. Einrichten Ihrer Entwicklungsumgebung

1.3.1. Allgemeines

Je nach Entwicklungsumgebung können Sie Tomcat direkt in diese einbinden bzw. diese wird bereits mit Tomcat geliefert (JBuilder). Praktisch sind einige Hilfseinrichtungen:

1) Shortcuts auf Start und Shutdown
   Im Explorer.exe : rechte Maustaste "Create Shortcut"
   Verschieben Sie diese in Ihr Entwicklungsverzeichnis (welches sicher nicht ein Unterverzeichnis von Tomcat sein sollte)!
2) Shortcuts auf die Dokumentation
   Wechseln Sie in das Verzeichnis
   %CATALINA_HOME%\webapps\tomcat-docs und kreieren Sie wie oben einen Shortcut auf index,html. Ich verwende "SendMail" und errichte den Shortcut auf dem Desktop.
3) Wenn Sie gut sind: erstellen Sie ein ANT Skript (build.xml) mit dem Sie übersetzen, testen und deployen können.

1.3.2. `CLASSPATH`

Hier die Libraries, die Sie beim Übersetzen im `CLASSPATH` haben müssen:

```
%CATALINA_HOME%/common/lib/servlet.jar.
```

Zudem muss Ihr Entwicklunsverzeichnis im `CLASSPATH` stehen. Eventuell nimmt Ihnen Ihre Entwicklungsumgebung diese Arbeit ab.

Was ich NIE mache: CLASSPATH als Umgebungsvariable fix definieren. Warum nicht: im schlimmsten Fall testen Sie dann eine Applikation bei sich, sie läuft bestens, aber nur bei Ihnen!

© Josef M. Joller

1.4.Einfache Test Servlets

Im Folgenden bauen und testen wir drei Servlets mit jeweils steigender Komplexität:

1) ein simples Servlet ohne Package
2) ein simples Servlet mit Package
3) ein simples Servlet mit Package und Hilfsklassen

1.4.1. Ein Servlet ohne Package

Das erste Servlet schreibt einfach HTML Code in die Servlet Ausgabe. Damit kann man einfachste Servlet Aufgaben lösen. Alle Dateien, die benötigt werden, finden Sie im Beispiele ZIP Archiv.

1) Name des Servlets: `HelloServlet.java`
2) Übersetzen des Servlets: mit `uebersetzen.bat` oder `ant`!
3) Verschieben zu Tomcat: mit `2_kopiere_zu_tomcat.bat`
4) Starten von Tomcat: mit `startup.bat`
5) Testen des Servlets mit `http://localhost:80/servlet/HelloServlet1`
6) Das war's, ausser Sie wollen jetzt das Ganze mit `ant` nochmals machen:
   Sie finden ein build.xml bei den Beispielen.
   Mit `ant -projecthelp` erhalten Sie eine Liste der verfügbaren Tasks.
   Einige wie etwa `install, list, remove` funktionieren nur, nachdem Sie die Bibliothek `catalina-ant.jar` ins Verzeichnis `%ANT_HOME%\lib` kopiert haben.
   `ANT_HOME` ist bei mir `C:\Ant`. Das Archiv finden Sie bei `%CATALINA_HOME%\server/lib/catalina-ant.jar`.
   Testen Sie zuerst einfache Tasks und schauen Sie sich `build.xml` genau an.

1.4.2. Ein Servlet mit Packages

Als nächstes verwenden wir ein Servlet `HelloServlet2`. In diesem Fall muss das Class-File ins Verzeichnis *%CATALINA_HOME%/webapps/ROOT/WEB-INF/classes/meineservlets*

Der Aufruf erfolgt mittels *http://localhost/servlet/meineservlets.HelloServlet2*.

1.4.3. Ein Servlet mit Packages und Hilfsklassen

Im dritten Servlet wird eine Hilfsklasse aus dem selben Package aufgerufen. In diesem Fall wird die Hilfsklasse ins selbe Verzeichnis (Verzeichnis = Packagename) kopiert.

Der Aufruf geschieht mittels *http://localhost/servlet/meineservlets.HelloServlet3*.

© Josef M. Joller

1.5. Weitere Tipps

Vielleicht möchten Sie den Manager Account einrichten, oder etwas genauer wissen, wie ant funktioniert?

1.5.1. Benutzer 'Manager' einrichten

Es macht wenig Sinn, den Benutzer "manager" gleich mit der Installation auf ein Standard-Passwort zu setzen. Deswegen wird Tomcat ohne Manager-Account verteilt. Wenn Sie den Installer verwenden, fragt er Sie nach einem Passwort; bei der ZIP Version natürlich nicht. Sie können leicht eigene Benutzer einrichten:

1) suchen Sie die Datei `%CATALINA_HOME%\conf\tomcat-users.xml`
2) ergänzen Sie diese durch einen von Ihnen frei wählbaren Benutzer und geben Sie diesem die Rolle eines Managers:
   ```
   <user name="mein_UID" password="mein_PWD"
   roles="standard,manager" />
   ```
   In meinem Fall habe ich einfach "manager/manager" gewählt (lokal).
3) falls Sie nun Tomcat stoppen und erneut starten, steht Ihnen dieser Benutzer zur Verfügung.

1.5.2. Benutzer 'Administrator' einrichten

Analog zum Manager kann man den Administrator einrichten.

1) suchen Sie die Datei `%CATALINA_HOME%\conf\tomcat-users.xml`
2) ergänzen Sie diese durch einen von Ihnen frei wählbaren Benutzer und geben Sie diesem die Rolle eines Administrator:
   ```
   <user name="mein_UID" password="mein_PWD"
   roles="standard,manager,admin" />
   ```
   In meinem Fall habe ich einfach "admin/admin" gewählt (lokal).
3) falls Sie nun Tomcat stoppen und erneut starten, steht Ihnen dieser Benutzer zur Verfügung.

1.5.3. Ant als universelles Java Make

Damit Sie Tomcat Tasks aus Ant benutzen können, müssen Sie zuerst die Bibliothek `catalina-ant.jar` ins Verzeichnis `%ANT_HOME%/lib` kopiert haben. `ANT_HOME` ist bei mir `C:\Ant`. Das Archiv finden Sie bei `%CATALINA_HOME%/server/lib/catalina-ant.jar`. Zudem ist es wichtig, dass Sie die Tasks in Ihrem `build.xml` definieren:

```
<!-- Configure props to access the Manager application -->
<property name="url" value="http://localhost:8080/manager"/>
<property name="username" value="myusername"/>
<property name="password" value="mypassword"/>
<!-- Configure the custom Ant tasks for the Manager application -->
<taskdef name="deploy"    classname="org.apache.catalina.ant.DeployTask"/>
<taskdef name="install"   classname="org.apache.catalina.ant.InstallTask"/>
<taskdef name="list"      classname="org.apache.catalina.ant.ListTask"/>
<taskdef name="reload"    classname="org.apache.catalina.ant.ReloadTask"/>
<taskdef name="remove"    classname="org.apache.catalina.ant.RemoveTask"/>
<taskdef name="resources" classname="org.apache.catalina.ant.ResourcesTask"/>
<taskdef name="roles"     classname="org.apache.catalina.ant.RolesTask"/>
<taskdef name="start"     classname="org.apache.catalina.ant.StartTask"/>
<taskdef name="stop"      classname="org.apache.catalina.ant.StopTask"/>
<taskdef name="undeploy" classname="org.apache.catalina.ant.UndeployTask"/>
```

Für das Servlet 1 habe ich build.xml angepasst; für die andern überlasse ich das Ihnen.

© Josef M. Joller

1.5.4. Servlet Invoker einschalten

Ich habe Eclipse und das Tomcat Plugin von Sysdeco.com installiert und habe zwei Tomcat Installationen getestet:

1) die volle Tomcat/Jakarta Version
2) die Tomcat/Jakarta Version ohne J2SDK

Mit der vollen Installation hatte ich kaum Probleme. Mit der LE Version wurde es harziger. Wichtig ist es, sicherzustellen, dass die Servlets generisch ermöglicht sind:
In `%CATALINA_HOME%\conf\web.xml` muss der folgende Abschnitt, der kommentiert ist, aktiviert werden:

```
<servlet-mapping>
      <servlet-name>invoker</servlet-name>
      <url-pattern>/servlet/*</url-pattern>
</servlet-mapping>
```

Sonst kann der Invoker Ihre Servlets nicht finden und starten. Sie erhalten keine sinnvolle Fehlermeldung, also viel Ärger.

Beachten Sie die Unterschiede in Apache 4 und 5: in 5 müssen Sie zudem das Invoker Servlet weiter oben im web.xml aktivieren (siehe Anleitung ganz oben).

1.6.Deployment von Web Applikation in Tomcat

Nun sollte Tomcat installiert sein und lauffähig. Als nächstes woollen Sie sicher Web Applikationen entwickeln und testen (und eventuell ausliefern). Damit Sie dies tun können, sollten Sie eine grobe Vorstellung von der Verzeichnisstruktur in Tomcat haben. Stellen Sie sicher, dass die Umgebungsvariable TOMCAT_HOME definiert ist, neben `JAVA_HOME` und `CATALINE_HOME`.

Tomcat Verzeichnisstruktur	
/bin	Dieses Verzeichnis enthält die Skripts zum Starten und Beenden von Tomcat
/conf	Dieses Verzeichnis enthält die wichtigsten Konfigurationsdateien: `server.xml` und das globale `web.xml`.
/server	Dieses Verzeichnis enthält die Tomcat Java Archive.
/lib	Diese Verzeichnis enthält Java Archive, die von Tomcat benötigt werden.
/logs	In diesem Verzeichnis befinden sich die Log-Dateien.
/src	In diesem Verzeichnis befindet sich der Quellcode des Tomcat Servers.
/webapps	Diese Verzeichnis enthält alle deployed Web Applikationen (WAR Datei).
/work	Arbeitsverzeichnis für Tomcat (JSP's, Servlets).
/common	Darin finden Sie die Libraries (/lib),Parser (/endorsed), ausgepackte Klassen

Für Ihre Applikation benötigen Sie folgende Verzeichnisse:

Tomcat Applikations-Verzeichnisstruktur	
/	Ihr Applikations-Root
`/WEB-INF/web.xml`	Deployment Descriptor Ihrer Anwendung `web.xml`.
`/WEB-INF/classes/`	Dieses Verzeichnis enthält Ihre Applikation.

© Josef M. Joller

/WEB-INF/lib/	Diese Verzeichnis enthält Java Archive, die von Ihrer Applikation benötigt werden.

1.6.1. Deployen eines bestehenden Web Archives

Als erstes wollen wir ein bestehendes Web Archiv installieren. Dies geschieht in mehreren Schritten:

1. kopieren Sie das WAR Archiv ins Verzeichnis TOMCAT_HOME/webapps
2. ergänzen Sie die Server Konfigurationsdatei : %TOMCAT_HOME%/conf/server.xml

   Beispiel (generiert durch Eclipse):
   ```
   <Context path="/ApachaSOAP" reloadable="true"
   docBase="C:\Java\Eclipse\workspace\ApachaSOAP"
   workDir="C:\Java\Eclipse\workspace\ApachaSOAP\work\org\ap
   ache\jsp" />
   ```

3. Starten Sie Tomcat neu:
   Tomcat entpackt Ihr Web Archiv! Anschliessend können Sie die Datei *.war löschen, da die darin enthaltenen Dateien sich nun in den Subverzeichnissen befinden.

4. Die URL Ihrer Applikation ist in diesem Fall http://localhost:8080/soap/
5. Falls Sie die Applikation mutieren wollen, sollten Sie alle Unterverzeichnisse zuerst löschen, da sonst beim erneuten WAR expandieren alte Dateien nicht gelöscht werden.

1.6.2. Deployen von nicht archivierten Dateien

Falls Sie kein Web Archiv haben, können Sie die Dateien auch direkt ins WebApps Verzeichnis kopieren, beispielsweise mit einem Ant Build Skript. Sie finden ein Beispielskript auf dem Server.

Diese Methode ist sehr einfach und Tomcat generiert aufgrund Ihrer Verzeichnisstruktur beim Neustart den passenden Kontexteintrag in server.xml.

Schauen Sie sich das Skript genau an: die Dateien werden in die passenden Unterverzeichnisse verteilt, genau wie beim WAR.

© Josef M. Joller

1.6.3. Deployen mithilfe des Administrators

Sie können auch Applikationen mithilfe der Tomcat Manager Befehle deployen oder auflisten oder entfernen. Falls Sie den Benutzer „`manage/manage`“ in `tomcat-users.xml` eingetragen haben, beispielsweise als

```
<user username="manage" password="manage" roles="manager"/>
```

können Sie

- in `http://localhost:8080/manager/serverinfo` die Server Informationen abfragen;
- mit `http://localhost:8080/manager/list` die in stallierten und aktiven Tasks auflisten
- ....

Sie finden die vollständige Befehlsliste unter `http://localhost:8080/manager/manager-howto.html`.

1.7.Das war's

Meine Installation funktioniert. Aber das Deployment mit Ant, der Aufbau von war-Archive und vieles mehr steht in der Dokumentation von Tomcat. Ich komme darauf zurück

© Josef M. Joller

1.8.Anhang – Version 5 Spezialitäten

Im folgenden fassen wir einige Spezialitäten zusammen, so wie Sie sie im Readme oder den Release Notes finden. Die meisten Punkte sind für das problemlose Funktionieren einer einfachen Installation nicht wesentlich!

1.8.1. Tomcat 5 und JNI basierte Applikationen

Häufig laden Java Applikationen Bibliotheken aus dem Betriebsystem. Diese Applikationen enthalten typischerweise folgenden Programmcode:

```
static {
      System.loadLibrary("Pfad zur Library");
}
```

Typischerweise darf diese Bibliothek nur einmal geladen werden. Das ist aber nicht so einfach über ein Singleton Design Pattern lösbar, falls Sie Ihre Applikation in ein /WEB-INF/classes oder (WEB-INF/lib) platzieren.

Sie müssen solche Applikationen ausserhalb dieser Verzeichnisse speichern!

1.8.2. Tomcat 5 Standard API's

Standardmässig werden mit Tomcat 5 folgende API's installiert (in „common/lib" oder „shared/lib")

- ant.jar (Apache Ant)
- commons-collections.jar (Commons Collections 2.1)
- commons-dbcp.jar (Commons DBCP 1.1)
- commons-el.jar (Commons Expression Language 1.0)
- commons-logging-api.jar (Commons Logging API 1.0.3)
- commons-pool.jar (Commons Pool 1.1)
- jasper-compiler.jar (Jasper 2 Compiler: JSP)
- jasper-runtime.jar (Jasper 2 Runtime)
- jmx.jar (Sun JMX RI 1.2)
- jsp-api.jar (JSP 2.0 API)
- commons-el.jar (JSP 2.0 Expression Language)
- naming-common.jar (JNDI Context implementation)
- naming-factory.jar (JNDI object factories for J2EE ENC support)
- naming-resources.jar (JNDI DirContext implementations)
- servlet-api.jar (Servlet 2.4 API)

Falls Sie eigene API's allen Web Applikationen verfügbar machen wollen, können Sie diese als Klassen in ein von Ihnen zu kreierendes Verzeichnis „classes" platzieren, oder aber als Archive ins „lib" Verzeichnis.

Wie Sie oben sehen, steht Ihnen das Commons Logging API (aus Xerces 2) zur Verfügung.

© Josef M. Joller

1.8.3. Tomcat 5 und XML Parser

Tomcat benutzt intern einen XML Parser (beispielsweise um server.xml oder web.xml zu analysieren). Falls Sie den Standard XML Parser (Xerces 2) durch einen andern ersetzen wollen, dann müssen Sie einfach `xercesImpl.jar` im Verzeichnis „common/endorsed“ durch einen andern Parser ersetzen. Dieser muss aber JAXP 1.2 API kompatibel sein.

1.8.4. JAVAC Memory Leaks

Der (alte) Java Compiler führte zu vielen Memory Leaks. Das führt bei JSP's in Umgebungen, welche diese oft und wiederholt laden, zu grossen Problemen, die Sie nur durch wiederholtes (periodisch) Starten der JVM beheben können.

Dieses Problem wurde in JDK1.4 behoben.

1.8.5. Linux

Probleme gab/gibt es auch mit Linux und bestimmten Kombinationen des Kernels mit glibc und dem Sun Hotspot 1.2 und 1.3. Die IBM JVM Version ist in diesem Umfeld die bessere Wahl.

Das Problem kann beseitigt werde, indem Sie die default Stack-Grösse passend wählen:
bash shell:"ulimit -s 2048";
tcsh: "limit stacksize 2048".

GLIBC 2.2 / Linux 2.4 Benutzer sollten folgende Umgebungsvariable definieren:
export LD_ASSUME_KERNEL=2.2.5.

Das Gleiche gilt für Redhat Linux 9.0, falls Sie Stabilitätsprobleme haben

1.8.6. CGI und SSI Unterstützung

Falls CGI und SSI eingeschaltet werden, könnten bei gleichzeitigem eigenen Security Manager Sicherheitsprobleme entstehen. Deswegen ist in Tomcat 5 standardmässig das Laden eines eigenen Security Managers disabled.

CGI können Sie folgendermassen einschalten:

1) benennen Sie $CATALINA_HOME/server/lib/servlets-cgi.renametojar um in $CATALINA_HOME/server/lib/servlets-cgi.jar
2) in $CATALINA_HOME/conf/web.xml, entfernen Sie die Kommentare in den Deklarationen:

```
<servlet>
        <servlet-name>cgi</servlet-name>
        ...
</servlet>

und im Servlet Mapping:
<servlet-mapping>
        <servlet-name>cgi</servlet-name>
        <url-pattern>/cgi-bin/*</url-pattern>
</servlet-mapping>
```


© Josef M. Joller

Alternativ können Sie diese Servlet Deklarationen und Mappings zu Ihrem Web Applikation Deployment Descriptor hinzufügen.

SSI können Sie auf folgende Art und Weise aktivieren:

1) benennen Sie das Archiv $CATALINA_HOME/server/lib/servlets-ssi.renametojar um, in $CATALINA_HOME/server/lib/servlets-ssi.jar
2) in $CATALINA_HOME/conf/web.xml, entfernen Sie die Kommentierung in den zwei Bereichen Servlet Deklaration und Servlet Mapping:

```
<servlet>
        <servlet-name>ssi</servlet-name>
        ...
</servlet>

bzw. das Servlet Mapping:
<servlet-mapping>
        <servlet-name>ssi</servlet-name>
        <url-pattern>*.shtml</url-pattern>
</servlet-mapping>
```

Auch in diesem Fall könnten Sie diese Angaben genauso im Deployment Descriptor Ihre Web Applikation machen.

1.8.7. Security manager URLs

In Tomcat 4.0 war die codeBase URLs für JARs
jar:file:${catalina.home}/webapps/examples/WEB-INF/lib/driver.jar!/-

In Tomcat 4.1 und 5.0:
file:${catalina.home}/webapps/examples/WEB-INF/lib/driver.jar

1.8.8. Enabling Invoker Servlet

Ab Tomcat 4.1.12 wurde das Invoker Servlet disabled. Wie oben beschrieben, können Sie es wieder enablen, indem Sie die folgende Zeile im server.xml freischalten:

$CATALINA_HOME/conf/web.xml : entfernen Sie die Kommentierung der "/servlet/*" Servlet-Mapping Definition.

Apache empfiehlt, das Invoker Servlet in produktiven Installationen zu disablen.

© Josef M. Joller

© Josef M. Joller