

Tomcat e le servlet

(I lucidi con sfondo celeste non sono essenziali)

Per un tutorial di base (installato Tomcat):
<http://localhost:8080/docs/appdev/>

Tomcat e le servlet

- Tomcat è un progetto Apache
- Tomcat è un *Java application container*, cioè esegue applicazioni Java accessibili via Web: **Java Web Applications** o **Web App**
- Primariamente, il bytecode eseguibile come parte di una Web App è una classe detta **Servlet**
- Il codice eseguibile può essere scritto in altre forme (p.es. JSP), ma, alla fine, sarà tradotto in, ed eseguito come, **servlet**
- Tomcat si può usare in modalità **stand-alone** (è a sua volta un' applicazione Java) oppure come **modulo** del web server **Apache**
- In generale, un *application server* è tale se supporta, oltre alle servlet, l'architettura (di applicazioni per il Web) **Java Enterprise**, di Oracle (un tempo Sun)
- Altri application container: JBoss (Red Hat), GlassFish (Oracle)
 - IIS (Microsoft) non dà più supporto nativo per Java

Tomcat: *CATALINA_HOME*

- Accesso a Tomcat via <http://localhost:8080> (ma vedi *conf.xml*)
- La dir di installazione è detta *CATALINA_HOME*, con le subdir:
 - *bin* – script *startup.sh/shutdown.sh*, di configurazione e alcuni *jar*
 - *lib* – librerie (*jar*) e classi
 - *conf* – file di configurazione (*server.xml* designa il *Connector* port per *http* come 8080, o altro numero, p.es. 28080,)
 - *logs* - log e output files, all'avvio Tomcat cerca o crea *catalina.out*
N.B. *catalina.out* corrisponde al *System.out*, la *standard* output, del codice eseguito da Tomcat
 - *webapps* - web applications eseguite da Tomcat (automaticamente all'avvio per default, v. oltre)
 - *work* – directory di lavoro temporanee per le web applications
 - p.es., ogni pagina JSP viene tradotta in una servlet memorizzata in *work/* e verrà eseguita attraverso questa servlet
 - *temp* - usata dalla JVM che esegue Tomcat per i file temporanei (*java.io.tmpdir*)

26/10/18

Uso di Tomcat / Servlet

3

Dov'è *CATALINA_HOME*?

CATALINA_HOME ha collocazione di default secondo la versione di Tomcat e la piattaforma ospite, p.es.:

- C:\Programmi\Apache Software Foundation\Tomcat 7.0 (Windows)
- /usr/local/share/tomcat che è un link alla cartella /usr/local/apache-tomcat-7.0.47 (Mac OSX, per una possibile, tipica, installazione “a mano”)
- /usr/share/tomcat7 (ArchLinux)
- per Debian/Ubuntu, la situazione è più complessa: *CATALINA_HOME* è /usr/share/tomcat7, ma vedi oltre la discussione su *CATALINA_BASE*

26/10/18

Uso di Tomcat / Servlet

4

Tomcat: cartelle e permessi (approfondimento)

Su distribuzione Archlinux, le scelte di installazione delle directory di Tomcat sono molto tipiche (le vediamo per questo):

```
$ ls -lL /usr/share/tomcat7/                                     # ls -lL dà permessi sui target dei link
drwxr-xr-x 2 root root 4096 Jun  5 21:14 bin
drwxrwxr-x 3 root root 4096 Jun 11 01:26 conf                  # -> /etc/tomcat7
drwxr-xr-x 2 root root 4096 Jun  5 21:14 lib                  # -> /usr/share/java/tomcat7
drwxrwxr-x 2 tomcat log 4096 Jun  5 21:14 logs                # -> /var/log/tomcat7
drwxr-xr-x 2 tomcat tomcat 4096 Jun  5 21:14 temp              # -> /var/tmp/tomcat7/temp
drwxr-xr-x 7 tomcat tomcat 4096 Jun  5 21:14 webapps           # -> /var/lib/tomcat7/webapps
drwxr-xr-x 2 tomcat tomcat 4096 Jun  5 21:14 work              # -> /var/tmp/tomcat7/work
```

- rispetto a un'installazione Tomcat a mano, tutte le dir tranne *bin* sono link, scelta frequente sui server Linux
- come prescrive lo standard Linux (LSB) i file di configurazione vanno in */etc* e quelli di lavoro in */var*
- i permessi restrittivi sulle varie dir sono una scelta “robusta”, rispetto a installazioni “a mano” Win o Mac, con permessi aperti a tutti, ma possono creare problemi a un utente normale non root (v. prossimo lucido)
 - p.es., un problema legato ai permessi si manifesterebbe se si cercasse di “integrare a mano” Netbeans (che gira per l'utente) con un'installazione Tomcat (di sistema)

26/10/18

Uso di Tomcat / Servlet

5

Tomcat: cartelle e permessi / Caveat!

```
$ ls -lL /usr/share/tomcat7/                                     # ls -lL mostra i permessi sulle dir target dei link
drwxr-xr-x 2 root root 4096 Jun  5 21:14 bin
drwxrwxr-x 3 root root 4096 Jun 11 01:26 conf                  # -> /etc/tomcat7
drwxr-xr-x 2 root root 4096 Jun  5 21:14 lib                  # -> /usr/share/java/tomcat7
drwxrwxr-x 2 tomcat log 4096 Jun  5 21:14 logs                # -> /var/log/tomcat7
drwxr-xr-x 2 tomcat tomcat 4096 Jun  5 21:14 temp              # -> /var/tmp/tomcat7/temp
drwxr-xr-x 7 tomcat tomcat 4096 Jun  5 21:14 webapps           # -> /var/lib/tomcat7/webapps
drwxr-xr-x 2 tomcat tomcat 4096 Jun  5 21:14 work              # -> /var/tmp/tomcat7/work
```

- Se un utente comune lancia Tomcat con */usr/share/tomcat7/bin/startup.sh*, Tomcat (come processo) avrà un problema di permesso di scrittura (*w*) sui log in *.../logs* (e anche in *.../temp* e *.../work*)
- Soluzione: Tomcat deve girare (con *sudo*) per l'utente *tomcat* o per *root*:
\$ sudo /usr/share/tomcat7/bin/startup
o meglio essere avviato da *root* come servizio, p.es. via una delle alternative:
 - # */etc/init.d/tomcat6 start* # Debian/Ubuntu (vecchio)
 - # *service tomcat7 restart* # Debian/Ubuntu (più moderno)
 - # *systemctl start tomcat7* # standard Linux recente per la gestione dei servizi
- Meglio ancora: l'utente comune può avviare Tomcat (tipicamente per sviluppo) introducendo un'adatta directory di lavoro *CATALINA_BASE* (v. oltre)

26/10/18

Uso di Tomcat / Servlet

6

Tomcat: *CATALINA_BASE*

Sono possibili installazioni multiple, personalizzate:

- Vi è un'unica *CATALINA_HOME* **condivisa**, con directory *bin* (scripts/jar) e *lib* (classi e lib)
- ogni installazione ha la dir *CATALINA_BASE* **individuale**, con:
 - *conf* - configuration files
 - *logs* - log and output files
 - *webapps* - automatically loaded web applications
 - *work* - temp working dirs for webapps / servlets from JSPs
 - *temp* - used by the JVM for temporary files (java.io.tmpdir)
 - *bin* - configuration scripts in versioni individuali
 - *lib* - librerie specifiche per le app in questa *webapps*
- Per un'installazione singola, di sistema, *CATALINA_HOME* coincide con *CATALINA_BASE* in genere (ma non per Debian)

26/10/18

Uso di Tomcat / Servlet

7

Tomcat: installazione Debian

L'installazione in *CATALINA_HOME* */usr/share/tomcat8* usa una *CATALINA_BASE* di sistema, tipicamente */var/lib/tomcat8*

```
$ ls -l /usr/share/tomcat8      # CATALINA_HOME per Debian/Ubuntu
bin                            # script di startup, shutdown, ...
lib                            # contiene link a file in /usr/share/java
....

$ ls -l /var/lib/tomcat8       # CATALINA_BASE di sistema per Debian/Ubuntu
lib                            # directory vuota all'installazione
policy
webapps
conf -> /etc/tomcat8
logs -> ../../log/tomcat8      # cioè /var/log/tomcat8
work -> ../../cache/tomcat8    # cioè /var/cache/tomcat8

# la directory temporanea è /tmp/tomcat8-tomcat8-tmp/
```

26/10/18

Uso di Tomcat / Servlet

8

Tomcat: installazione personale (Unix)

```
$ mkdir tomcat2 ; cd tomcat2          # tomcat2 sarà CATALINA_BASE
$ mkdir logs temp work webapps        # crea directory individuali in CATALINA_BASE

# Nel seguito, usare CATALINA_HOME appropriata al posto di /usr/share/tomcat

# Nella dir webapps personale conviene replicare le dir ROOT, manager e (se c'è) docs della webapps di sistema
$ cp -R /usr/share/tomcat/webapps/{ROOT,manager,docs} ./webapps/

$ cp -R /usr/share/tomcat/conf .      # replichiamo anche la configurazione di sistema
# NB: per copiare così conf/ servono permessi di lettura, oppure (v. oltre) sudo, poi chown

$ nano conf/server.xml                # con un editor, personalizziamo i port
# Port di default: 8080 (connector port), 8005 (shutdown), 8443 (SSL redirect), 8009 (AJP, per Apache)
# personalizziamoli p.es. in: 28080, 28005, 28443, 28009

# lanciamo tomcat personalizzato
$ CATALINA_BASE=$(pwd) /usr/share/tomcat/bin/startup.sh
$ CATALINA_BASE=$(pwd) /usr/share/tomcat/bin/shutdown.sh

# oppure (modificando le variabili d'ambiente):
$ export CATALINA_BASE=$(pwd) PATH=/usr/share/tomcat/bin:$PATH

$ startup.sh                          # mostrerà le var d'ambiente, tra cui:
Using CATALINA_BASE:  /Users/gp/tomcat2
Using CATALINA_TMPDIR: /Users/gp/tomcat2/temp
# vedi anche logs/catalina...log
$ shutdown.sh
```

26/10/18

Uso di Tomcat / Servlet

9

Tomcat: installazione personale e permessi

Per costruire *CATALINA_BASE/conf*, come detto, conviene copiare e adattare *CATALINA_HOME/conf*

```
$ cp -R /usr/share/tomcat7/conf .      # si replica la configurazione
```

ma non è detto si disponga dei permessi necessari; ad esempio, i permessi non sono adeguati nel caso seguente:

```
$ cd /usr/share/tomcat7
$ ls -lL conf/*.xml
-rw-r----- 1 root tomcat 1394 Apr  1 15:01 conf/context.xml
-rw-r----- 1 root tomcat 6435 May 27 15:41 conf/server.xml
-rw-r----- 1 root tomcat 1636 May 27 15:42 conf/tomcat-users.xml
-rw-r----- 1 root tomcat 152716 Apr  1 15:01 conf/web.xml
```

Qui, se l'utente *gp* è nel gruppo *user*, ma non in *tomcat*, serve:

```
$ sudo cp -R /usr/share/tomcat7/conf .      # si replica la configurazione
$ sudo chown -R gp:users conf/              # cambia ownership per user gp,
                                           gruppo users
```

(NB: *sudo/chown* servirebbero anche se, viceversa, l'utente comune volesse installare le sue app nella *CATALINA_HOME/webapps* di sistema)

26/10/18

Uso di Tomcat / Servlet

10

Tomcat: installazione personale da .zip

Se (vedi slide precedente) copiare la cartella *conf* (o *webapps*) di *CATALINA_HOME* in *CATALINA_BASE* non è possibile o conveniente, si può scaricare la distribuzione .zip o .tar.gz di Tomcat da <http://apache.tomcat.org> e poi estrarne *conf*, p.es. così:

```
$ unzip apache-tomcat-7.0.27.zip */conf/*
```

In generale, ogni utente può semplicemente installare **tutto** Tomcat dallo *zip* (o *tar.gz*), in una sua directory *CATALINA_HOME* completa, in una posizione sotto il proprio controllo:

- p.es. la directory decompressa *tomcat7* (*CATALINA_HOME*) può stare nella home directory dell'utente

N.B.: in questo caso, gli script *bin/startup.sh* o *bin/startup.bat* ... dovrebbero comunque funzionare, ma potrebbero richiedere che si definisca la variabile d'ambiente *JAVA_HOME*

Document root

Tomcat è, a tutti gli effetti, un **Web server** e la sua *document root* è: *CATALINA_BASE/webapps/ROOT*, cioè:

- le pagine *html* e *jsp* in essa (e nelle sue subdir) sono visibili, p.es. il file
 - *CATALINA_BASE/webapps/ROOT/myIndex.html*
è servito alla URL <http://localhost:8080/myIndex.html>
 - *CATALINA_BASE/webapps/ROOT/dslab/index.html*
è servito alla URL <http://localhost:8080/dslab>

Ma Tomcat è prima di tutto un **servlet container**, quindi ci si chiede:

- dove vanno i file .class delle servlet? e
- quali URL daranno accesso ad esse?

Servlet: fonti

- Documentazione su Servlet:
<https://tomcat.apache.org/tomcat-8.0-doc/servletapi/index.html>
- In generale, sulle servlet (ricco): <http://www.coreservlets.com/>,
corredato dal [testo https://tinyurl.com/mhall-unirm](https://tinyurl.com/mhall-unirm),
per approfondimenti: <http://moreservlets.com/>
- Tutorial conciso e ben organizzato sulle servlet:
<http://tutorials.jenkov.com/java-servlets/index.html>
- Di riferimento, da Oracle, sulle servlet:
<http://docs.oracle.com/javaee/7/tutorial/servlets.htm>
- In generale, sulla famiglia di tecnologie Java EE (Enterprise Edition): <http://docs.oracle.com/javaee>
- Breve corso universitario (vedere lezioni 4 e 5):
<http://www.dsi.unive.it/~roncato/informaticaApplicataA/>

Servlet: materiale per studiare

- **Testo consigliato:** Deitel-Deitel, *Java: Tecniche avanzate di programmazione*, <https://tinyurl.com/deitel-java>, cap. 16
- **Lucidi in italiano ispirati al testo consigliato:**
 - Prof. Lo Presti, <http://www.ce.uniroma2.it/~lopresti/didattica2.htm>
 - slide <https://tinyurl.com/lopresti1516-servlet>
 - slide <https://tinyurl.com/lopresti1516-servlet2>
 - Prof. Bartolini, http://twiki.di.uniroma1.it/twiki/view/PW/2013_2014
materiale secondo Deitel-Deitel, ancora più approfondito,
http://twiki.di.uniroma1.it/pub/PW/WebHome/PW_all.pdf,
pagine 69-217
- **Altri buoni lucidi in italiano** (vedere lezioni 4 e 5):
<http://www.dsi.unive.it/~roncato/informaticaApplicataA/>

Tomcat: fonti

- La documentazione online: <http://localhost:8080/docs/> ovvero <https://tomcat.apache.org/tomcat-8.0-doc/>
- Il wiki di Tomcat: <https://wiki.apache.org/tomcat/>
- Per informazioni dettagliate di configurazione: <http://www.coreservlets.com/Apache-Tomcat-Tutorial/detailed-configuration.html>
- Tomcat è un applicazione Java; per usare JRE non di default: https://wiki.archlinux.org/index.php/tomcat#Using_Tomcat_with_a_different_JRE.2FJDK
- Descrizione approfondita dell'architettura di Tomcat: https://www.ntu.edu.sg/home/ehchua/programming/howto/Tomcat_More.html
- Ottimi consigli per security su server di produzione: https://wiki.archlinux.org/index.php/tomcat#Security_configuration

Compilazione e deployment di servlet

- Le servlet API non fa parte della distribuzione Java standard
- Per compilare *MyServlet.java*, serve *\$CATALINA_HOME/lib/servlet-api.jar* impiegato così:

```
$ javac -classpath /usr/share/tomcat7/lib/servlet-api.jar MyServlet.java
```

(si può anche copiare *servlet-api.jar* nella cartella *JAVA_HOME/lib/ext*)
- Il deployment di un file *MyServlet.class* da solo (*stand-alone*) è stato possibile solo fino a **Tomcat 6**.
- Da **Tomcat 7**, per il **deployment** della servlet *MyServlet.class*, occorre che questa sia parte di un *Web application* (vedi oltre) ma spesso ci si affida a un **IDE** (Netbeans/Eclipse...) che assiste nella costruzione della Web App.
 - L'IDE provvede anche a invocare *javac* per compilare le servlet

Servlet stand-alone e *Invoker* (saltare)

- Quali URL danno accesso ai file *.class* delle servlet?
- **Fino a Tomcat 6**, la servlet
.../ROOT/WEB-INF/classes/X.class,
è raggiungibile alla URL <http://localhost:8080/servlet/X>, purché:
 - si attivino la *invoker servlet* e il suo *mapping*, rimuovendo i relativi commenti nel file .../conf/web.xml, e
 - il file .../conf/Context.xml contenga:
<Context privileged="true">
- **Tomcat >=7 non permette** più la **Invoker** servlet (per motivi di sicurezza)
 - quindi la servlet X.class non è più eseguibile “da sola” (*stand-alone*)
 - essa va invece attivata all'interno di una opportuna *Web application* (vedi oltre)

Servlet standalone / 2 (saltare)

Quanto segue è utile per servlet stand-alone con Tomcat 6

- In ...conf/context.xml, che vale per tutte le Web App, conviene, anziché il semplice tag <Context>
 - <Context ... reloadable="true">, per rendere subito visibili i cambiamenti dei class file delle servlet
 - <Context ... allowLinking="true">, per l'esecuzione di class file raggiunti attraverso link simbolici
 - (dir e class file devono avere comunque i permessi “giusti”)

Alcune direttive di configurazione (avanzato)

- In `...conf/context.xml`, conviene: `<Context ... reloadable="true">`, per rendere subito visibili cambiamenti dei file `.class` delle servlet e altre “risorse” sotto osservazione
 - i costi in termini di prestazioni sono da evitare su un Tomcat di “produzione”
- Tomcat cerca tutte le risorse a partire dalla cartella `.../webapps`.
Se in `.../webapps` vi sono dei link simbolici, questi vengono seguiti solo se in `.../conf/context.xml` figurano:
 - `<Context ... allowLinking="true">` per Tomcat 7
 - `<Context> ... <Resources allowLinking="true" ... /> ... </Context>`, per Tomcat 8 e 9
 - N.B.: dir e file raggiunti via link sono poi accessibili solo se hanno i permessi “giusti”
- Se la URL passata a Tomcat termina con slash (/):
 - se non c'è un *welcome-file* (tipo `index.html` o `index.php`) nella directory del server corrispondente alla URL
 - allora Tomcat (6,7,...) mostra il “directory listing”, **purché** `...conf/web.xml` contenga:

```
<servlet>
  <servlet-name>default</servlet-name>
  ...
  <param-name>listings</param-name>
  <param-value>true</param-value>
  ...
</servlet>
```

26/10/18

Uso di Tomcat / Servlet

19

Web App e Tomcat

- Java EE introduce il concetto di *Web Application*, fatta di
 - componenti (`.class` e `.jar`, pagine HTML, JSP,...) e
 - file di configurazionedistribuiti su una gerarchia ben precisa di directory
- Nel seguito vedremo alcune nozioni di base e casi d'uso di **deployment** di webapp su Tomcat
- vedere anche: <http://localhost:8080/docs/appdev/>
- una Web App si può costruire:
 - con un IDE (NetBeans, Eclipse...)
 - o a mano, da riga di comando (complicato, salvo che per casi base come quello illustrato in seguito).

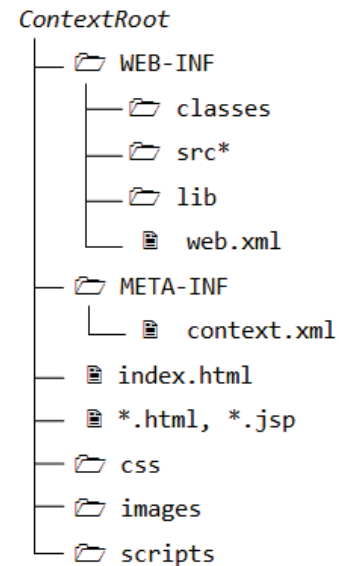
26/10/18

Uso di Tomcat / Servlet

20

Struttura di una Web app

- Radice o *Context root* (vedi oltre): contiene risorse visibili/accessibili ai clienti Web dell'app
- *WEB-INF* contiene risorse non visibili/accessibili ai clienti Web dell'app, riguardanti l'app ma indipendenti dal server
- *META-INF* contiene risorse collegate al server (Tomcat) piuttosto che specifiche della Web App



Una Web app minimale

Una Web App nella cartella *SimpleApp* con una semplice *Servlet1*:

```
$ ls -lR SimpleApp/
112 01-28-14 08:19 index.html
0 01-28-14 08:21 WEB-INF/
0 01-28-14 08:21 WEB-INF/classes/
650 01-28-14 07:38 WEB-INF/classes/Servlet1.class
239 01-28-14 08:10 WEB-INF/web.xml
```

```
<!-- index.html -->
<html>
<head>
  <title>Test a Simple servlet</title>
</head>
<body>
  <a href="./Simple">Go to servlet</a>
</body>
</html>
```

```
<!-- web.xml -->
<web-app>
  <servlet>
    <servlet-class>Servlet1</servlet-class>
    <servlet-name>Slet1</servlet-name>
  </servlet>
  <servlet-mapping>
    <servlet-name>Slet1</servlet-name>
    <url-pattern>/Simple</url-pattern>
  </servlet-mapping>
</web-app>
```

- *index.html* è solo una comodità, qui porta subito alla URL della servlet
- *web.xml* serve a mappare la servlet sulla URL */Simple*

Mapping e *web.xml*

```
$ ls -lR SimpleApp/
112 01-28-14 08:19 index.html
0 01-28-14 08:21 WEB-INF/
0 01-28-14 08:21 WEB-INF/classes/
650 01-28-14 07:38 WEB-INF/classes/Servlet1.class
239 01-28-14 08:10 WEB-INF/web.xml
```

```
<!-- index.html -->

...<a href="/Simple">
    Go to servlet</a>
...
```

- Il tag `<servlet>` associa la classe ***Servlet1*** al nome *Slet1* (che ha significato solo “interno”)
- Il tag `<servlet-mapping>` mappa il nome *Slet1* sull’URL pattern */Simple* (si noti lo */*)

```
<!-- web.xml -->
<web-app>
  <servlet>
    <servlet-class>Servlet1</servlet-class>
    <servlet-name> Slet1</servlet-name>
  </servlet>
  <servlet-mapping>
    <servlet-name>Slet1</servlet-name>
    <url-pattern>/Simple</url-pattern>
  </servlet-mapping>
</web-app>
```

- ciò rende */Simple* la URL della servlet **relativa al contesto** della Web app, da cui il link nel file *index.html*
- la URL assoluta, sia p.es. `http://localhost:8080/SimpleApp/Simple`, termina con lo URL pattern */Simple*, e, per il resto dipende:
 - da come viene specificato l’host di Tomcat. p.es. `http://localhost:8080`
 - dal nome, p.es. *SimpleApp*, con cui Tomcat serve la Web app (v. oltre)

26/10/18

Uso di Tomcat / Servlet

23

Webapp: nome

Struttura di una Web App nella directory *SimpleApp*:

```
$ ls -ld SimpleApp
drwxr-xr-x 2 gp users 4096 19 jan 12.59 SimpleApp/

$ ls -lR SimpleApp/
112 01-28-14 08:19 index.html
0 01-28-14 08:21 WEB-INF/
...
```

Ma cosa determina il nome di una Web App? Cioè quello con cui Tomcat la serve, p.es. *SimpleApp* nella URL `http://localhost:8080/SimpleApp`

Di norma, è il nome della directory in `$CATALINA_BASE/webapps` in cui risiedono i suoi file,

Webapp: nomi e archivi .war

Struttura di una Web App nella directory *SimpleServletApp*:

```
~ $ ls -ld SimpleServletApp
drwxr-xr-x 2 gp users 4096 19 jan 12.59  SimpleServletApp/

~ $ ls -lR SimpleServletApp/
112 01-28-14 08:19 index.html
  0 01-28-14 08:21 WEB-INF/
...
```

Ma qual è il nome di una Web App? Quello della sua directory *top*?

- si vedrà (più oltre) che il nome con cui Tomcat serve la Web App, p.es. `http://localhost:8080/SimpleServletApp`, è, di norma, il nome della sua directory nella `$CATALINA_BASE/webapps`,

```
~/SimpleServletApp $ jar -cf SimpleServletApp.war . # nomi web app e servlet diversi!
```

Web App e NetBeans

- Un modo semplice di ottenere una Web App è di farla generare a NetBeans a partire da un progetto (comando *Build* e/o *Clean-and-Build* (Shift-F11) o *Run*)
 - NetBeans pone la Web App per il progetto *MyApp* in: `<NetBeansProjects>/MyApp/build/web` all'interno di una gerarchia di directory (v. oltre)
- La Web App per *MyApp* si può anche impacchettare in un archivio .war (WebApp Archive, formato zip)
 - NetBeans (se opportunamente configurato) genera automaticamente il file *Myapp.war* per il progetto *MyApp* al comando *Build*
 - il file *MyApp.war* si troverà nella cartella `<NetBeansProjects>/MyApp/dist`

Esempio di Web App da Netbeans

Una web app *MyServletApp* generata da NetBeans

```
~/NetBeansProjects $ ls -R MyServletApp/build/web
MyServletApp/build/web
MyServletApp/build/web/index.html
MyServletApp/build/web/META-INF
MyServletApp/build/web/META-INF/MANIFEST.MF
MyServletApp/build/web/WEB-INF
MyServletApp/build/web/WEB-INF/glassfish-web.xml
MyServletApp/build/web/WEB-INF/classes
MyServletApp/build/web/WEB-INF/classes/MyServlet.class
MyServletApp/build/web/WEB-INF/classes/.netbeans_automatic_build
MyServletApp/build/web/WEB-INF/classes/.netbeans_update_resources
```

Come detto, Netbeans crea l'archivio *MyServletApp.war* in *MyServletApp/dist*, ma lo si può anche generare “a mano”, dalla dir della Web App *MyServletApp*:

```
~/NetBeansProjects/MyServletApp/build/web $ jar -cf MyServletApp.war .
```

- N.B.: in realtà, il nome dell'archivio si può scegliere a piacimento. P.es.:

```
~/NetBeansProjects/MyServletApp/build/web $ jar -cf MyApp.war .
```

Ma ora, anche se la app generata da Netbeans si chiamava *MyServletApp* (cf. *nome dir*), il deployment (v. oltre) di *MyApp.war* darà comunque luogo a una webapp di nome *MyApp*

Maggiori dettagli: <http://www.adp-gmbh.ch/blog/2004/october/13.html>

26/10/18

Uso di Tomcat / Servlet

27

Deploy Web App dir su Tomcat: a mano

- Tutorial chiaro su deployment di webapp (per webapp Portofino):
http://portofino.manydesigns.com/en/docs/portofino3/3_1_x/installation-guide/deploying-on-tomcat
- In breve: sia *MyApp* la directory di una webapp, strutturata come visto
- essa va posta in *\$CATALINA_BASE/webapps*, cioè, per esempio:
 - *\$HOME/tomcat/webapps*, per un'installazione “personale”, ovvero
 - */usr/share/tomcat/webapps*, se *CATALINA_BASE* coincide con *CATALINA_HOME* o, ancora,
 - */var/lib/tomcat6/webapps* (default di Debian/Ubuntu).
- N.B. se si è sviluppata la webapp *MyApp* con Netbeans, la sua directory sarà *<NetbeansProjects>/MyApp/build/web*, allora:
 - la si copia (in una posizione qualsiasi), rinominandola (es.) *MyApp* e
 - si sposta *MyApp* nella dir di Tomcat *\$CATALINA_BASE/webapps*Analogamente per Eclipse e altri tool.

26/10/18

Uso di Tomcat / Servlet

28

Deploy Web App a mano: accesso

- Come detto, la Web App dir va nella dir `$CATALINA_BASE/webapps`; se questa è una cartella di sistema,
 - p.es. `/usr/share/tomcat/webapps` o
 - `/var/lib/tomcat6/webapps` (Debian/Ubuntu)
- allora: **per la copia può servire accesso come user *root* o *tomcat***
- **e le ownership dei file della Web app devono essere quelle previste**

Esempio: deploy a mano su piattaforma Linux/Ubuntu della web app *Arit* (per altre distro, si avranno altri owner/group):

```
~ $ sudo mv Arit/ /var/lib/tomcat6/webapps/  
~ $ sudo ls -ld /var/lib/tomcat6/webapps/Arit/  
drwxr-xr 4 sysadm sysadm 2011-06-16 07:38 /var/lib/tomcat6/webapps/Arit  
~ $ sudo chown -R tomcat6:tomcat6 /var/lib/tomcat6/webapps/Arit/
```

- NB: come “si accorge” della nuova app Tomcat? Ciò dipende dagli attributi nel file *server.xml*. Cf. quanto si dirà per i *.war*

Deploy WAR su Tomcat a mano

- Il deployment si può effettuare, comodamente, da un archivio *WAR* *MyApp.war*, copiandolo a mano in `$CATALINA_BASE/webapps`
- Valgono le osservazioni già fatte, per la Web app non compressa, riguardo a: diritti di chi effettua la copia e ownership del file
- Perché Tomcat “si accorga” del nuovo file *.war* occorre:
 - che il file di configurazione *conf/server.xml* contenga:
`<Host name="localhost" ... autoDeploy="true">`
 - oppure riavviare Tomcat

(in realtà il Deploy al riavvio dipende da un attributo di Host in *conf/server.xml*, cioè **deployOnStartup**; esso non figura nel *server.xml* standard dell'installazione, ma il suo valore di default è *true*)
- Tomcat è in grado di eseguire senza decomprimerlo l'archivio `$CATALINA_BASE/webapps/MyApp.war`,
inoltre, lo decomprimerà anche in `$CATALINA_BASE/webapps/MyApp`,
se nel file di configurazione *conf/server.xml* figura la direttiva (è il default):
`<Host name="localhost" ... unpackWARs="true">`

Contesto di una Web app

Come detto, una Web App che risiede nella directory `$CATALINA_BASE/webapps/MyApp` è accessibile attraverso la URL `http://localhost:8080/MyApp/`

`/Myapp` è detto **contesto (context)** o **context path** della Web App e, in un certo senso, è la webapp, come detto in

http://localhost:8080/docs/deployer-howto.html#A_word_on_Contexts

... a Context is what Tomcat calls a web application.

Si parla di **contesto** perché i riferimenti **relativi** (URL, nomi di file) utilizzati in una Web App vanno riferiti appunto al contesto.

Per **portabilità**, conviene i riferimenti siano **relativi** (al contesto, che viene quindi omesso) piuttosto che **assoluti** (e contenenti il contesto).

P.es., conviene usare URL della forma `./...`

```
<!-- index.html per MyApp portabile -->
<a href="./Simple">Go to servlet</a>
<a href=".">Go to this web app</a>
```

```
<!-- index.html per MyApp non portabile -->
<a href="http://localhost:8080/MyApp/Simple">
Go to servlet</a>
<a href="http://localhost:8080/MyApp">Go to
this web app</a>
```

26/10/18

Uso di Tomcat / Servlet

31

Context descriptor di una Web app (saltare)

Come detto, per la Web App accessibile a `http://localhost:8080/MyApp/`, `/Myapp` si dice **context** o **context path**

La doc http://localhost:8080/docs/deployer-howto.html#A_word_on_Contexts specifica:

*To configure a **Context** within Tomcat, a **Context Descriptor** is required. ... [This] is simply an XML file that contains Tomcat related configuration for a Context, e.g naming resources or session manager configuration...*

*The **locations** for Context Descriptors are:*

- `$CATALINA_BASE/webapps/[webappname]/META-INF/context.xml`
- `$CATALINA_BASE/conf/[enginename]/[hostname]/[webappname].xml`

P.es. per la WebApp *manager* (installata con Tomcat), `context.xml` ha:

```
<Valve className="org.apache.catalina.valves.RemoteAddrValve"
allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" /> <!-- limit access to localhost -->
```

Tra gli attributi del tag `Context` in `context.xml`: `allowLinking`, `reloadable` (v. prima) e:

```
<Context path="/MiaApp"/>
```

per **cambiare** (nome al) **contesto**:

26/10/18

Uso di Tomcat / Servlet

32

Contesto/Path/URL di una Web app

Info su Servlet

Una *Web App* con una semplice servlet `InfoServlet`:

```
// InfoServlet.java - servlet che fornisce info su se stessa
...
public class InfoServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        response.setContentType( "text/ascii" );
        PrintWriter out = response.getWriter(); // get writer

        out.println("servlet class: " + getClass());

        // info from this servlet's request parameter
        out.println("servlet URL: " + request.getRequestURL());
        out.println("context (as path) from present HttpServletRequest: "
            + request.getContextPath());
        out.println("servlet path: " + request.getServletPath());
        out.println("path info: " + request.getPathInfo() );

        // Il contesto e` essenzialmente la dir in cui risiede la webapp
        // * da non confondere con il Servlet Context *

        // info from servlet's ServletContext object
        ServletContext thisServletContext = getServletContext();
        out.println("Container: " + thisServletContext.getServerInfo());
        out.println("context (as path) from servlet's ServletContext: "
            + thisServletContext.getContextPath());
        ...
    }
}
```

Wildcard in URL di Web app

Una Web App con una semplice servlet InfoServlet (eseguibile **senza** invoker servlet)

```
$ ls -lR wildCardServletApp/
112 01-28-14 08:19 index.html
0 01-28-14 08:21 WEB-INF/
0 01-28-14 08:21 WEB-INF/classes/
650 01-28-14 07:38 WEB-INF/classes/Yyy.class
239 01-28-14 08:10 WEB-INF/web.xml
```

```
<!-- index.html -->
<html>
<head>
<title>
Test Yyy servlet
</title>
</head>
<body>
<a href="./Yyy">Go to servlet</a>
</body>
</html>
```

```
<!-- web.xml -->
<web-app>
<servlet>
<servlet-class>Yyy</servlet-class>
<servlet-name>Yyy</servlet-name>
</servlet>
<servlet-mapping>
<servlet-name>Yyy</servlet-name>
<url-pattern>/Yyy.*</url-pattern>
</servlet-mapping>
</web-app>
```

- *web.xml* è essenziale per mappare servlet → URL *http://localhost:8080/YyyApp/Yyy*

Webapp senza web.xml con annotazioni

Una Web App *AnnotateServletApp* per la servlet *AnnotateServlet* (Tomcat >= 7)

```
$ ls -l AnnotateServletApp/
112 01-28-14 08:19 index.html
0 01-28-14 08:21 WEB-INF/
0 01-28-14 08:21 WEB-INF/classes/
650 01-28-14 07:38 WEB-INF/classes/AnnotateServlet.class
948 01-28-14 07:38 WEB-INF/classes/InfoServlet.class
```

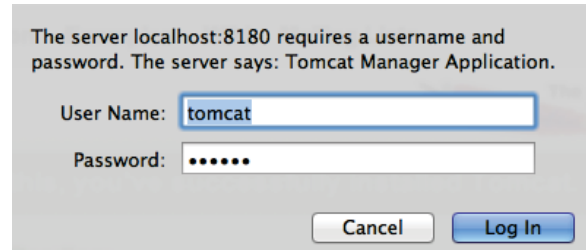
```
// AnnotateServlet.java
import javax.servlet.annotation.WebServlet; // needed for @WebServlet annotation
@WebServlet(urlPatterns = {"/Annotate"}) // needs above import
// The above annotation is processed by the container at deployment time, and the corresponding servlet made
// available at the specified URL patterns. Web app will not need mappings in web.xml. Since: Servlet 3.0

public class AnnotateServlet extends InfoServlet {
    public void doGet( HttpServletRequest req, HttpServletResponse resp )
        throws ServletException, IOException
    { super.doGet(request,response); }
}
```

- la URL per la servlet *AnnotateServlet* è: ...<contesto App>/Annotate, cioè (p.es.): *http://localhost:PORT/AnnotateServletApp/Annotate*
- *web.xml* è assente, ma il mapping *AnnotateServlet.class* → *Annotate* è dato dall'annotazione *@webServlet()*

Tomcat: *manager* app

- Accesso a **Manager** app via <http://localhost:8080/manager/html>
- Occorrono le credenziali di un **utente** con **ruolo** *manager-gui*



The server localhost:8180 requires a username and password. The server says: Tomcat Manager Application.

User Name:

Password:

- il file per username/password/ruoli è **.../conf/tomcat-users.xml**, per i dettagli v. http://localhost:8080/docs/manager-howto.html#Configuring_Manager_Application_Access
- In breve, il file è auto-esplicativo (N.B. viene letto solo al (ri)avvio):

```
<tomcat-users>
<!-- NOTE: By default, no user is included in the "manager-gui" role required to operate the "/manager/html" web
application... to use this app, you must define such a user - username and password are arbitrary.
-->
<!-- definire qui il ruolo manager-gui non serve. La doc dice: "find role names in webapps/manager/WEB-INF/web.xml",
si veda http://tomcat.apache.org/tomcat-7.0-doc/manager-howto.html#Configuring_Manager_Application_Access
-->
  <role rolename="tomcat"/>
  <role rolename="manager-gui"/>
  <user username="tomcat" password="tomcat" roles="tomcat,manager-gui"/>
</tomcat-users>
```

- Per la **host-manager** app (gestione host virtuali), l'utente deve avere il ruolo **admin-gui**

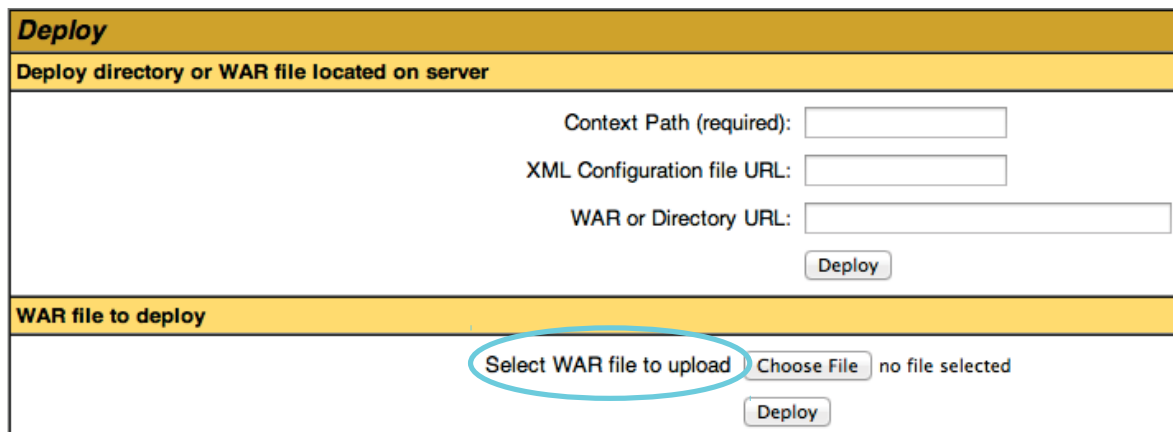
26/10/18

Uso di Tomcat / Servlet

37

Deploy WAR locale via Tomcat manager

- per il deploy, come detto prima, il file **<nomeApp>.war** deve essere posto in **\$CATALINA_BASE/webapps**
- questo compito può essere svolto dalla Manager app di Tomcat: si specifica un **.war** locale (sul PC del browser) e clic su **Deploy**
 - beninteso, Tomcat noterà il file e il Deploy avverrà anche se **AutoDeploy="false"** in **server.xml**
 - l'archivio **.war** verrà decompresso se **unpackWARs="false"** in **server.xml**



Deploy

Deploy directory or WAR file located on server

Context Path (required):

XML Configuration file URL:

WAR or Directory URL:

WAR file to deploy

Select WAR file to upload no file selected

26/10/18

Uso di Tomcat / Servlet

38

Deploy Web app / WAR remoti via Manager App

- Se la Web App dir o l'archivio .WAR sono già sul **server**, la *Manager App* ne consente il *deployment* via GUI
- NB: poichè dir o .WAR sono sul **server**, la *Directory URL* è **remota**, ma, beninteso, se il server è *localhost*, è anche locale!

Deploy

Deploy directory or WAR file located on server

Context Path (required):

XML Configuration file URL:

WAR or Directory URL:

WAR file to deploy

26/10/18

Uso di Tomcat / Servlet

39

Netbeans per lo sviluppo di Web App

- Netbeans è un potente IDE (Integrated Development Environment) per Java (per cui nasce) e vari altri linguaggi
- E' software libero, controllato da una comunità in cui ha un ruolo preminente **Oracle**
- Supporta lo sviluppo in ambito Java Enterprise, quindi, in particolare di Web App
- Per consentirne l'esecuzione, ha bisogno di un Application server: quello di default è *Glassfish* di Oracle, ma anche *Tomcat* è utilizzabile e si integra bene con Netbeans
- La **soluzione migliore** è scaricare e installare il pacchetto "completo" che contiene *Glassfish* e *Tomcat* in *bundle*, "integrati" e configurati ad-hoc
 - tale *Tomcat* (es. in OSX */Applications/NetBeans/apache-tomcat-7.0.34/*) convive tranquillamente con l'installazione *Tomcat* standard (il *Connector port* sarà 8084 anziché 8080, usato da *Glassfish*)

- I download bundle qui consigliati (includono Tomcat) sono: **Java EE o All**

NetBeans IDE Download Bundles						
Supported technologies *	Java SE	Java EE	HTML5/JavaScript	PHP	C/C++	All
NetBeans Platform SDK	•	•				•
Java SE	•	•				•
Java FX	•	•				•
Java EE		•				•
Java ME						—
HTML5/JavaScript		•	•	•		•
PHP			•	•		•
C/C++					•	•
Groovy						•
Java Card™ 3 Connected						—
Bundled servers						
GlassFish Server Open Source Edition 4.1.1		•				•
Apache Tomcat 8.0.27		•				•
	<input type="button" value="Download"/>	<input type="button" value="Download"/>	<input type="button" value="Download"/>	<input type="button" value="Download"/>	<input type="button" value="Download"/>	<input type="button" value="Download"/>

26/10/18

Uso di Tomcat / Servlet

40

Netbeans: caveat per chi installa – dir target

NetBeans IDE Download Bundles						
Supported technologies *	Java SE	Java EE	HTML5/JavaScript	PHP	C/C++	All
NetBeans Platform SDK	•	•				•
Java SE	•	•				•
Java FX	•	•				•
Java EE		•				•
Java ME						—
HTML5/JavaScript		•	•	•		•
PHP			•	•		•
C/C++					•	•
Groovy						•
Java Card™ 3 Connected						—
Bundled servers						
GlassFish Server Open Source Edition 4.1.1		•				•
Apache Tomcat 8.0.27		•				•
<div>Download Download Download Download Download Download</div>						

- Con Linux, l'eseguibile scaricato installa Netbeans nella dir da cui viene lanciato (utile se l'utente non ha i privilegi per le dir di sistema), p.es.:

```
$ chmod +x netbeans-8.2-linux.sh
$ ./netbeans-8.2-linux.sh
# oppure, con -extract su altra dir, se si hanno i privilegi necessari:
# ./netbeans-8.2-linux.sh -extract /usr/local/
```

26/10/18

Uso di Tomcat / Servlet

41

Netbeans: caveat per chi installa - JDK

NetBeans IDE Download Bundles						
Supported technologies *	Java SE	Java EE	HTML5/JavaScript	PHP	C/C++	All
NetBeans Platform SDK	•	•				•
Java SE	•	•				•
Java FX	•	•				•
Java EE		•				•
Java ME						—
HTML5/JavaScript		•	•	•		•
PHP			•	•		•
C/C++					•	•
Groovy						•
Java Card™ 3 Connected						—
Bundled servers						
GlassFish Server Open Source Edition 4.1.1		•				•
Apache Tomcat 8.0.27		•				•
<div>Download Download Download Download Download Download</div>						
Free, 116 MB Free, 242 MB Free, 142 MB Free, 142 MB Free, 147 MB Free, 277 MB						

* You can add or remove packs later using the IDE's Plugin Manager (Tools | Plugins).

HTML/JS, PHP and C/C++ NetBeans bundles include Java Runtime Environment, and do not require a separate Java installation.

JDK 8 is required for installing and running the Java SE, Java EE and All NetBeans Bundles. NetBeans 8.2 does not run on JDK9! You can download standalone JDK or download the latest JDK with NetBeans IDE Java SE bundle.

Using JDK 9 with NetBeans IDE? Go here.

Important Legal Information:

NetBeans Community Distributions are available under a Dual License consisting of the Common Development and Distribution License (CDDL) v1.0 and GNU General Public License (GPL) v2. Such distributions include additional components under separate licenses identified in the License file. See the Third Party License file for external components included in NetBeans and their associated licenses.

- Netbeans è a sua volta un'applicazione Java ed è **molto pignolo** riguardo al JDK che lo supporta, p.es. NB 8 vuole JDK 8
- Attenzione: se non c'è la versione di JDK richiesta, l'installazione può fallire o, peggio, completarsi e risultare inutilizzabile!

26/10/18

Uso di Tomcat / Servlet

42

Netbeans: caveat per chi installa - JDK

NetBeans IDE Download Bundles

Supported technologies *	Java SE	Java EE	HTML5/JavaScript	PHP	C/C++	All
NetBeans Platform SDK	•	•				•
Java SE	•	•				•
Java FX	•	•				•
Java EE		•				•
Java ME						—
HTML5/JavaScript		•	•	•		•
PHP			•	•		•
C/C++					•	•
Groovy						•
Java Card™ 3 Connected						—
Bundled servers						
GlassFish Server Open Source Edition 4.1.1		•				•
Apache Tomcat 8.0.27		•				•

[Download](#) [Download](#) [Download](#) [Download](#) [Download](#) [Download](#)

Free, 116 MB Free, 242 MB Free, 142 MB Free, 142 MB Free, 147 MB Free, 277 MB

* You can add or remove packs later using the IDE's Plugin Manager (Tools | Plugins).

HTML/JS, PHP and C/C++ NetBeans bundles include Java Runtime Environment and do not require a separate Java installation.

JDK 8 is required for installing and running the Java SE, Java EE and All NetBeans Bundles. NetBeans 8.2 does not run on JDK9! You can download [standalone JDK](#) or download the latest [JDK with NetBeans IDE Java SE bundle](#).

Using JDK 9 with NetBeans IDE? [Go here](#).

Important Legal Information:

NetBeans Community Distributions are available under a Dual License consisting of the [Common Development and Distribution License \(CDDL\) v1.0](#) and [GNU General Public License \(GPL\) v2](#). Such distributions include additional components under separate licenses identified in the [License file](#). See the [Third Party License file](#) for external components included in NetBeans and their associated licenses.

- Se non si può/vuole installare a livello di sistema il JDK richiesto, si può farlo in un'altra directory

26/10/18

Uso di Tomcat / Servlet

43

Netbeans: caveat per chi installa: attivare Tomcat

- (Linux) Il *wizard* di installazione offre la possibilità di personalizzazioni: è **fondamentale** attivare l'opzione di installazione per Tomcat

Customize Installation

Select packs and runtimes to install from the list below.

Component	Status
Base IDE	(Already Installed)
Java SE	(Already Installed)
Java EE	(Already Installed)
Java ME	
HTML5/JavaScript	(Already Installed)
PHP	
C/C++	
Groovy	
Features on Demand	
Runtimes	
GlassFish Server Open Source Edition 4.1.1	(Already Installed)
Apache Tomcat 8.0.27	(Already Installed)

Installation size: 121.4 MB

Next > **Cancel**

26/10/18

Uso di Tomcat / Servlet

44

Tomcat e Netbeans: a mano o bundle?

Integrare “a mano” *Netbeans* con un *Tomcat* “esterno”, installato a parte è complicato, per i detti problemi di utente/ruolo, ma non solo...

- a volte *Tomcat* “non risponde più su *localhost:8080*”, dopo l'integrazione con *Netbeans*
 - nel caso, verificare con *telnet localhost 8080* se *Tomcat* è attivo, quindi se la cartella *ROOT* in *.../webapps* è integra... nei casi peggiori, re-installare *Tomcat* ...

La soluzione migliore è scaricare e installare il pacchetto *JavaEE* o *All* che contengono *Tomcat* in *bundle*, già “integrato” e configurato ad-hoc

- tale *Tomcat* (p.es in OSX */Applications/NetBeans/apache-tomcat-7.0.34/*) convive tranquillamente con l'installazione *Tomcat* standard (il *Connector port* sarà 8084 anziché 8080, utilizzato da *Glassfish*)

NetBeans IDE Download Bundles

Supported technologies *	Java SE	Java EE	HTML5/JavaScript	PHP	C/C++	All
NetBeans Platform SDK	•	•				•
Java SE	•	•				•
Java FX	•	•				•
Java EE		•				•
Java ME						—
HTML5/JavaScript		•	•	•		•
PHP			•	•		•
C/C++					•	•
Groovy						•
Java Card™ 3 Connected						—
Bundled servers						
GlassFish Server Open Source Edition 4.1.1		•				•
Apache Tomcat 8.0.27		•				•

Download Download Download Download Download Download

26/10/18

Uso di Tomcat / Servlet

45

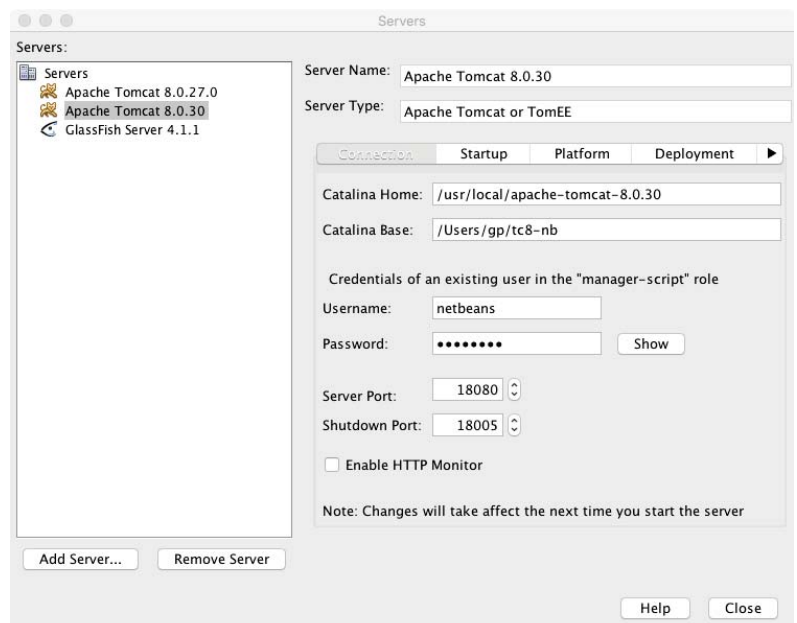
Tomcat e Netbeans: a mano...

Per chi però proprio volesse “**provarci**” (a integrare NB con TC esterno):
<http://wiki.netbeans.org/AddExternalTomcat>

https://wiki.archlinux.org/index.php/Netbeans#Integrate_with_tomcat

dà un'ottima spiegazione su:

- come integrare via Wizard (menu *Tools-Servers*) di Netbeans
- Wizard assai migliorato in NB8
- **notare**, a fianco:
 - **Catalina Home** di sistema
 - **Catalina Base** “privata”
 - **Username** viene creato se non esiste, con i ruoli corretti (v. *tomcat-users.xml*)
 - scelta **Port** non-standard (resa permanente in *server.xml*)



Ancora, riguardo al deploy via NetBeans su Tomcat esterno (non lo raccomando!):
<http://wiki.netbeans.org/DeploymentOnExternalTomcatforNB6>

26/10/18

Uso di Tomcat / Servlet

46

Netbeans: dietro le quinte

Build per un'app *WSAppTC* invoca il command line tool *ant*:

```
$ ~ ant -f ~/NetBeansProjects/WSAppTC/build.xml -DforceRedeploy=false dist
```

build.xml contiene direttive per *ant* e ne richiama altre per la costruzione della web app, dalla directory *WSAppTC/nbproject*; *ant* ha diversi compiti:

- compila con *javac* i file sorgente (ricompila selettivamente, solo i file cambiati e le loro dipendenze, come *make*)
- crea e riempie le directory *build* e *build/web*, questa con la *web app* da pubblicare (“deploy”)
- crea la dir *dist*, in cui confeziona l'archivio *dist/WSAppTC.war*, contenente la web app; anche il *.war* si può usare per il deploy

Clean cancella, sempre con *ant*, le directory *build* e *dist*:

```
ant -f ~/NetBeansProjects/WSAppTC/build.xml -DforceRedeploy=false clean
```

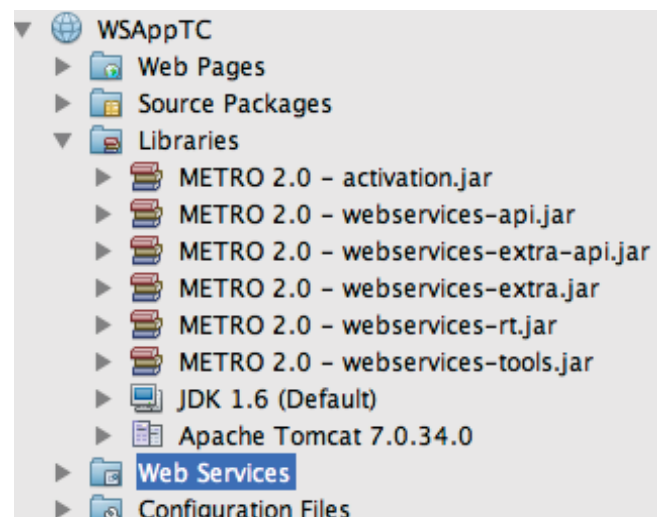
Netbeans dietro le quinte / librerie

In figura, le librerie usate da una Web app che implementa un Web Service da usare con Tomcat:

- le *Metro* vanno incluse nella Web App (salvo che si installino una volta per tutte, per Tomcat o JDK, in *ext/*)
- librerie JDK e Tomcat non vengono incluse (sono già nell'ambiente)

Al **build**, Netbeans invoca *ant*, che:

- copia alcuni (template) file di configurazione da *web/* (e altre) a *build/web/*
- copia le librerie *Metro* (stack che implementa Web services) a corredo di Netbeans, nella Web app generata
- invoca *javac* rispetto alle librerie di Metro e Tomcat



Implementare WS: GlassFish vs. Tomcat

Si può usare NB con GlassFish e deploy dell'app su Tomcat?

- in linea di massima sì, ma non sempre
- sì, se le librerie richieste per Tomcat e Glassfish coincidono
- no, altrimenti: è il caso dei Web Services

