



Università degli Studi di Bologna

Facoltà di Ingegneria

Tecnologie Web L-A
A.A. 2009 – 2010

Esercitazione 3

Tomcat

Tutor:

Ing. Pasini Samuele

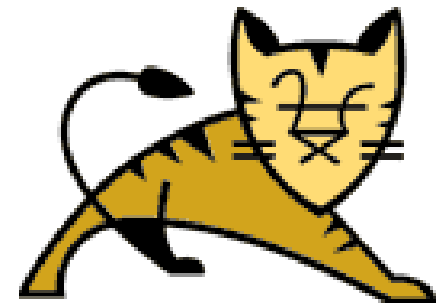
samuele.pasini@unibo.it

Agenda

- Apache Tomcat
 - caratteristiche generali
 - scaricare ed eseguire il Web Server
 - risorse su file system
- Un primo esempio di “*applicazione*” Web
 - struttura del progetto
 - fasi dello sviluppo
 - uso di ANT
 - accesso a risorse statiche (pagine HTML, fogli CSS, ...)

Apache Tomcat

- Un semplice Web Server, interamente scritto in Java
 - permette di pubblicare siti Web
 - fornisce l'ambiente di esecuzioni per applicazioni Web scritte in accordo alle specifiche Java Servlet e JSP
- **Installazione del server**
 - **download** dal sito ufficiale <http://tomcat.apache.org>
oppure
dalla intranet universitaria (sito del corso)
 - **estrazione** del contenuto del file ZIP



Caratteristiche generali

- Struttura su file system
 - **bin**: script e comandi di avvio
 - **common**: librerie Java visibili e condivise da tutte le applicazioni Web in esecuzione sul server
 - **conf**: configurazione di porte, permessi e altre risorse
 - **logs**: file di log (da creare a mano se non esiste!)
 - **server**: codice del server
 - **webapps**: pubblicazione delle applicazioni Web
 - **temp, work**: direttori per le operazioni del server (salvataggio dei dati di sessione, compilazione delle pagine JSP, ...)

Avvio

- E' necessario **impostare la variabile d'ambiente JRE_HOME** affinché “indichi” un'installazione JDK con versione **>= 1.5**
 - `$ export JRE_HOME=...` (linux; per **verificare**: `$ echo $PROVA`)
 - `> SET JRE_HOME=...` (windows; per **verificare**: `> echo %PROVA%`)
- **Lanciare il server** attraverso il comando
 - `$ TOMCAT_HOME/bin/startup.sh` (linux)
 - `> TOMCAT_HOME/bin/startup.bat` (windows)
- **Controllare il corretto avvio**
 - nei log del server stesso
 - `$ tail -f TOMCAT_HOME/logs/catalina.out` (linux)
 - `popup e/o > notepad.exe TOMCAT_HOME/logs/catalina.out` (windows)
 - accedendo a <http://localhost:8080/>

Un semplice progetto di applicazione “web”

- All'interno dell'archivio ZIP dell'esercitazione, nel direttorio *progetti*
 - il file **03_TemplateWebapp.zip** contiene un semplice progetto Java di esempio
 - creato con Eclipse: contiene già tutti i descrittori necessari a essere riconosciuto e configurato correttamente
- Importare il progetto come visto nell'ultima esercitazione, senza esploderne l'archivio su file system (lo farà Eclipse)
 - *File → Import → General → Existing Projects into Workspace → Next → Select archive file*
- Aprire il file di build *ant/build.xml* all'interno della vista ANT di Eclipse
 - *Window → Show view → Other → Ant → Ant*
 - Trascinare il file di build all'interno della nuova vista

Struttura del progetto

- All'interno del direttorio radice
 - **src**: sorgente (file `.java`) dell'applicazione web da sviluppare
 - **test**: sorgente delle routine di test (opzionali) che verificano il corretto funzionamento dell'applicazione
 - **LIBRERIE** (*la visualizzazione può variare da versione a versione di Eclipse*): codice fornito da terze parti necessario allo sviluppo
 - **JRE** le classi base del runtime di Java (es: `java.lang.String`)
 - **API** e loro eventuale **implementazioni**. riferite dall'applicazione a tempo di compilazione, ma non necessarie a tempo di esecuzione (es: `junit.jar` utilizzata solo durante i test, in Eclipse; `servlet-api.jar` e `jsp-api.jar` fornite dal servlet container, durante l'esecuzione)
 - **ant**: strumenti per l'esecuzione automatica di operazioni
 - compilazione, esecuzione dei test, packaging, **deployment**, ...
 - **lib**: direttorio che fisicamente contiene gli archivi `.jar` delle **LIBRERIE**
 - **tmp**: direttorio per scopi temporanei
 - **web**: direttorio radice dell'applicazione web
- Questa struttura caratterizzerà tutti i progetti “web” delle esercitazioni

Struttura dell'applicazione web

- Il direttorio **web** contiene l'esatta struttura dell'applicazione che verrà eseguita all'interno del server
 - risorse “statiche” (dal punto di vista del server): pagine HTML, immagini, fogli di stile CSS, script Javascript, ...
 - metadati dell'applicazione
 - *WEB-INF/web.xml*
(per ora tralasciamo questa parte)
 - bytecode (file *.class*) delle classi Java che costituiscono l'applicazione web
 - *WEB-INF/classes*
(direttorio inizialmente vuoto, usato come destinazione dei sorgenti compilati attraverso il build file di ANT)
 - librerie necessarie a tempo di esecuzione, ma non presenti tra le librerie rese disponibili dal server
 - *WEB-INF/lib*
(direttorio i cui archivi *.jar* sono da aggiungere al build-path di Eclipse, se necessari anche a tempo di compilazione)

Build file di ANT

- Oltre alle normali operazioni, comuni ai progetti di applicazioni “tradizionali”, il file di build che useremo per lo sviluppo di applicazioni web prevede:
 - **packaging** in formato WAR
 - **deployment**
 - copia dell'archivio WAR o dell'equivalente direttorio esploso in un apposito direttorio del server, al fine della attivazione dell'applicazione web
 - **aggiornamento delle sole risorse statiche** dell'applicazione web
 - richiede il deploy in formato “esploso”
 - evita di ricreare da zero l'archivio war in caso di modifiche che non coinvolgono classi Java e descrittori
 - permette quindi di non “spegnere” e “riavviare” l'applicazione sul server (e quindi di non perdere eventuali informazioni di sessioni attive)
 - può richiedere di cancellare la cache del browser (specialmente IE)
 - undeployment
- Inoltre:
 - **avvio del tunnel TCP/IP** per monitorare il traffico HTTP in ingresso e uscita dalle pagine dell'applicazione

Primi passi

- Lanciate il server
- Eseguite il deployment dell'applicazione
- Accedete alla pagina index.html
- Avviate il tunnel TCP/IP
- Eseguite la stessa operazione attraverso il tunnel
- Cancellate il contenuto del tunnel (*clear*)
- Modificate il contenuto della pagina index.html e aggiornate la sua versione sul server per mezzo di ANT
- Eseguite la stessa richiesta (attraverso il tunnel)

- Sondaggio:
 - quanti hanno visto passare nuovo traffico HTTP nel tunnel?
 - quanti usavano Internet Explorer? quanti Firefox?

- Modificate le impostazioni relative all'uso della cache (oppure azzeratela) nel browser e riprovate