Università degli Studi di Bologna
Facoltà di Ingegneria

Tecnologie Web L-A
A.A. 2009 – 2010

Esercitazione 4
Servlet

Tutor:
Ing. Pasini Samuele *samuele.pasini@unibo.it*

Agenda

- Importazione e modifica di un progetto di esempio
  - class-path a tempo di compilazione ed esecuzione
  - deployment ed esecuzione
  - descrittore *web.xml*
  - interazione con l'applicazione
  - servlet e mantenimento dello stato
- Funzionalità avanzate
  - esecuzione in modalità debug
- Per approfondire
  - ulteriori esempi

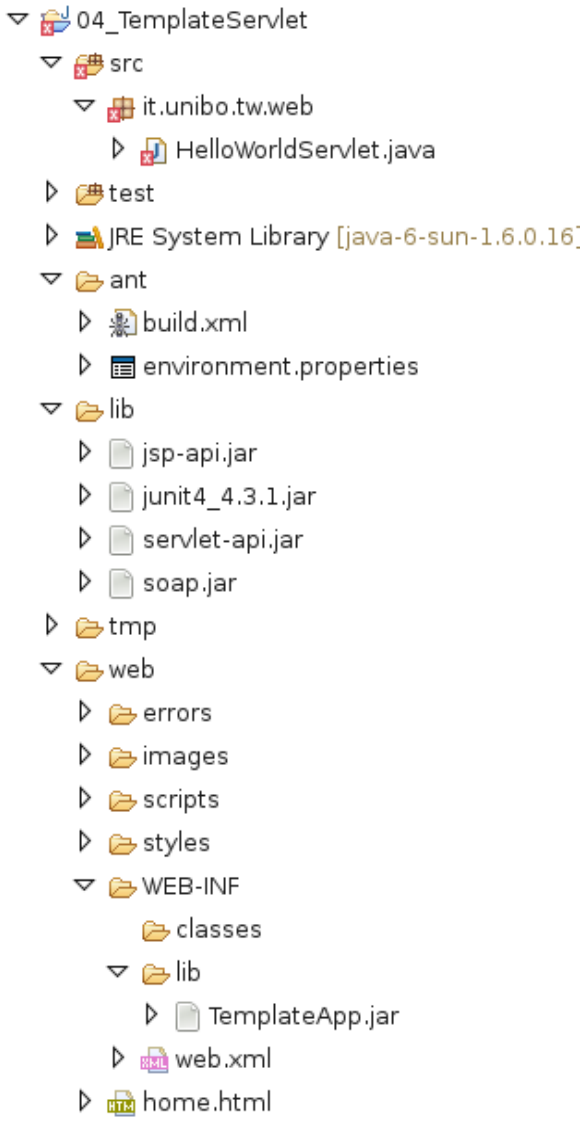
Per cominciare

- All'interno dell'archivio ZIP dell'esercitazione, nel direttorio ***progetti***
  - il file ***04_TemplateServlet.zip*** contiene lo scheletro di un semplice progetto di esempio basato sull'uso di Servlet
  - creato con Eclipse, contiene già tutti i descrittori necessari a essere riconosciuto e configurato correttamente
  - una volta sistemate le “imperfezioni” create ad arte per la presente esercitazione, può essere riutilizzato come base per altri progetti di applicazioni Web

- Importare il progetto come visto nelle precedenti esercitazioni, senza esploderne l'archivio su file system (lo farà Eclipse)
  - *File → Import → General → Existing Projects into Workspace → Next → Select archive file*

Build-path

- Problemi di compilazione (almeno dal punto di vista di Eclipse)
  - aggiungere al *build-path* le librerie necessarie a *compile-time*, ma fornite dal container a *run-time*
    - *lib/servlet-api.jar*
    - *...*
  - aggiungere al *build-path* le librerie necessarie a *compile-time* e da fornire al container a *run-time*
    - *web/WEB-INF/lib/TemplateApp.jar*
      (l'applicazione della 2° esercitazione)
- Il file di build *ant/build.xml* è invece in grado di funzionare perfettamente
  - il *classpath* usato da ANT è indipendente da quello dell'IDE e viene definito dallo stesso file di build
  - gli script di ANT possono perciò eseguire in maniera autonoma, anche in assenza di un IDE...
  - ...a patto che le proprietà relative all'ambiente di esecuzione siano impostate correttamente
    - *ant/environment.properties*



Deployment

- Il progetto contiene
  - una semplice pagina HTML iniziale che “intrattiene” l'utente intanto che le classi dell'applicazione vengono caricate in memoria, al primo accesso
    - *web/home.html*
  - una classe che estende *HttpServlet* e riutilizza il materiale dell'esercitazione #2 per produrre il più classico degli “hello world”
  - fogli di stile, immagini, script, pagine di errore, ...
  - un descrittore XML che “insegna” al web server cosa fare con tutto ciò

- Per il momento
  - avviare Tomcat
    - $ $TOMCAT_HOME/bin/startup.sh
    - > $TOMCAT_HOME\bin\startup.bat
  - controllare i file di log
    - $ tail -f $TOMCAT_HOME/logs/catalina.out
    - > notepad.exe $TOMCAT_HOME/logs/catalina.out
  - compilare, pacchettizzare e pubblicare l'applicazione Web
    - *$/> ant -f $PROJECT_ROOT/ant/build.xml 09a.deploy.war*
  - accedere al contesto Web dell'applicazione
    - $ firefox http://localhost:8080/TemplateServlet
    - > firefox.exe http://localhost:8080/TemplateServlet

- Eppure.....

Riflessioni e... cose da sistemare

- L'utente...
  - **non può sapere da quale pagina iniziare la navigazione**
  - **non deve ricevere messaggi di errore tecnici (404?)**

- Il Web server, generalmente, ci viene incontro...
  - presentando automaticamente le pagine di benvenuto di default, se presenti, a fronte della richiesta del solo contesto dell'applicazione Web
    - *index.html, index.jsp, ...*
    - ma per complicare le cose, la *"homepage"* di questo progetto si chiama *home.html*
  - creando pagine di errore di default, in caso di problemi
    - almeno è HTML, ma spesso contiene informazioni tecniche che non vorremmo divulgare

- Infine, nonostante l'archivio WAR contenga delle Servlet, il Servlet container non conosce a quali URL devono essere associate e in mancanza di tale informazione, non può renderle disponibili.

- I descrittori XML sono la chiave per risolvere questi problemi ed insegnare al server...
  - come è fatto l'applicazione Web contenuta nel file *.war*
  - come gestire aspetti quali pagine di benvenuto ed errore, criteri di sicurezza, risorse utilizzate, ...
- Per ora, modifichiamo insieme il file
  - *web/WEB-INF/web.xml*

web.xml ...*reloaded*

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>

<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
    version="2.4">

    <display-name>TemplateServlet</display-name>
    <description>
      A servlet-based to project to use as a template for your own ones
    </description>

    <!-- Define servlets that are included in the application -->

    <servlet>
        <servlet-name>HelloWorld</servlet-name>
        <servlet-class>it.unibo.tw.web.HelloWorldServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>HelloWorld</servlet-name>
        <url-pattern>/helloworld</url-pattern>
    </servlet-mapping>

    <!-- Welcome pages -->

    <welcome-file-list>
        <welcome-file>test.html</welcome-file>
        <welcome-file>home.html</welcome-file>
    </welcome-file-list>

    <!-- Handle exceptions and errors -->

    <error-page>
        <error-code>404</error-code>
        <location>/errors/notfound.html</location>
    </error-page>

    <error-page>
        <exception-type>java.lang.Exception</exception-type>
        <location>/errors/exception.html</location>
    </error-page>

</web-app>
```

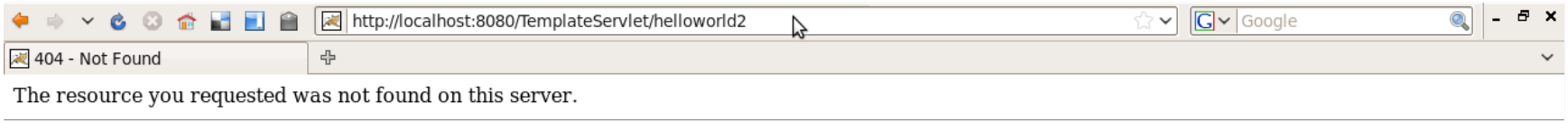
Nuovo deployment

- Accesso al contesto dell'applicazione Web

- Caricamento della servlet *hello world*

- Accesso via HTTP POST... errore (graceful)!

- Accesso a una risorsa che non esiste... errore (graceful)!



Mantenimento dello stato

- Sfruttando le cose apprese a lezione e in laboratorio

  - creare un'applicazione Web (o modificare quella dell'esercitazione) in cui una Servlet è in grado di servire richieste HTTP come segue...
    - HTTP GET:
      - presentazione di una form per l'invio di testo al server mediante HTTP POST
      - valorizzazione del campo di input della form con l'eventuale testo già inviato dall'utente in precedenti interazioni con la stessa Servlet
    - HTTP POST:
      - visualizzazione del testo ricevuto nella pagina HTML di risposta
      - memorizzazione del testo ricevuto

  - per il mantenimento dello stato, scegliere uno tra i seguenti meccanismi
    - salvataggio di attributi in sessione, lato server
    - salvataggio di cookie sul browser, lato client
    - ....*oppure?*

Esecuzione in modalità “debug”

- Per poter eseguire *Tomcat* in modalità “debug” è necessario modificarne lo script di avvio
  - cercare la riga
    - export JAVA_OPTS="$JAVA_OPTS xxx yyy zzz" (*startup.sh)*
    - set JAVA_OPTS=%JAVA_OPTS% xxx yyy zzz (*catalina.bat*)
  - modificarla come segue, facendo attenzione a NON introdurre spazi vuoti dove non indicato!
    - export JAVA_OPTS="$JAVA_OPTS xxx yyy zzz **-Xdebug -Xrunjdwp:transport=dt_socket,address=8787,server=y,suspend=n**"
    - set JAVA_OPTS=%JAVA_OPTS% xxx yyy zzz **-Xdebug -Xrunjdwp:transport=dt_socket,address=8787,server=y,suspend=n**
  - (ri)avviare *Tomcat*

- All'interno dell'IDE occorre poi...
  - creare una apposita configurazione di debug
    - *Run → Debug configurations... → Remote Java application → right-click → New...*
    - ...e indicare la stessa porta di ascolto $PORT
  - avviare tale configurazione

- Infine:
  - specificare opportuni breakpoint nei quali interrompere e monitorare l'esecuzione
    - *...left-click oppure right-click → toggle breakpoint sulla fascia grigia a sinistra del codice, nell'IDE*
  - eseguire il deploy dell'applicazione
  - richiedere via browser la risorsa che determina l'esecuzione del codice contenente i breakpoint
  - osservare l'esecuzione passo-passo (pause/play, step in/out/over)

Per approfondire...

- *Tomcat* fornisce out-of-the-box alcuni esempi relativi all'utilizzo delle Servlet API, utili come riferimento

  - visionabili a partire da:
    - *http://localhost:8080/servlets-examples*

  - funzionamento e estratti del codice sorgente)

  - il codice sorgente completo è comunque disponibile su file system, nella directory di deployment che corrisponde al contesto *"servlet-examples"*