Anno Accademico 2007-2008

Laboratorio di Tecnologie Web

Sviluppo di applicazioni web

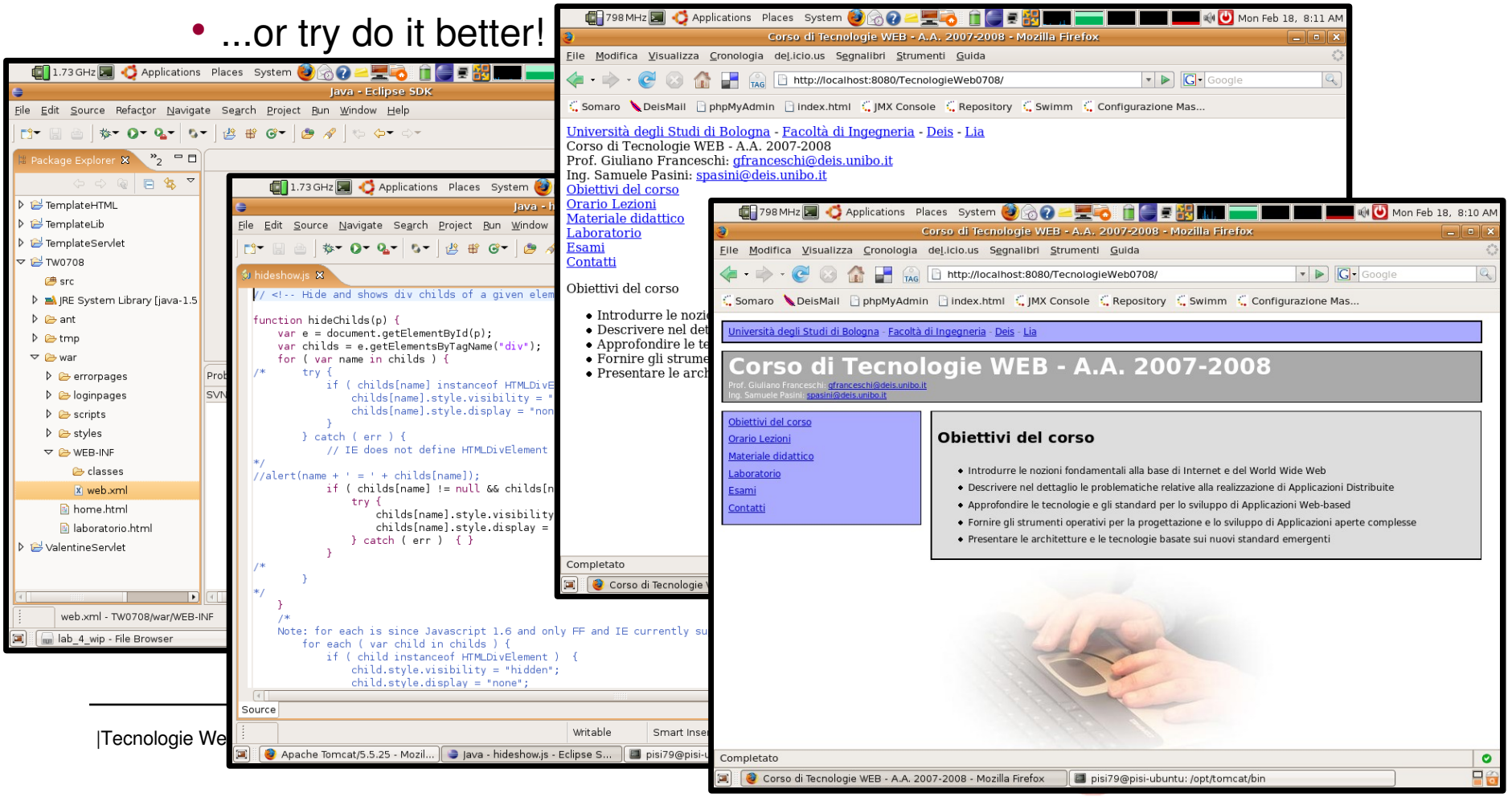
Servlet

http://www-lia.deis.unibo.it/Courses/TecnologieWeb0708/

Before we go on...

> From the course web site you can download an Eclipse project (*TW0708.zip*) resembling the course web site structure:

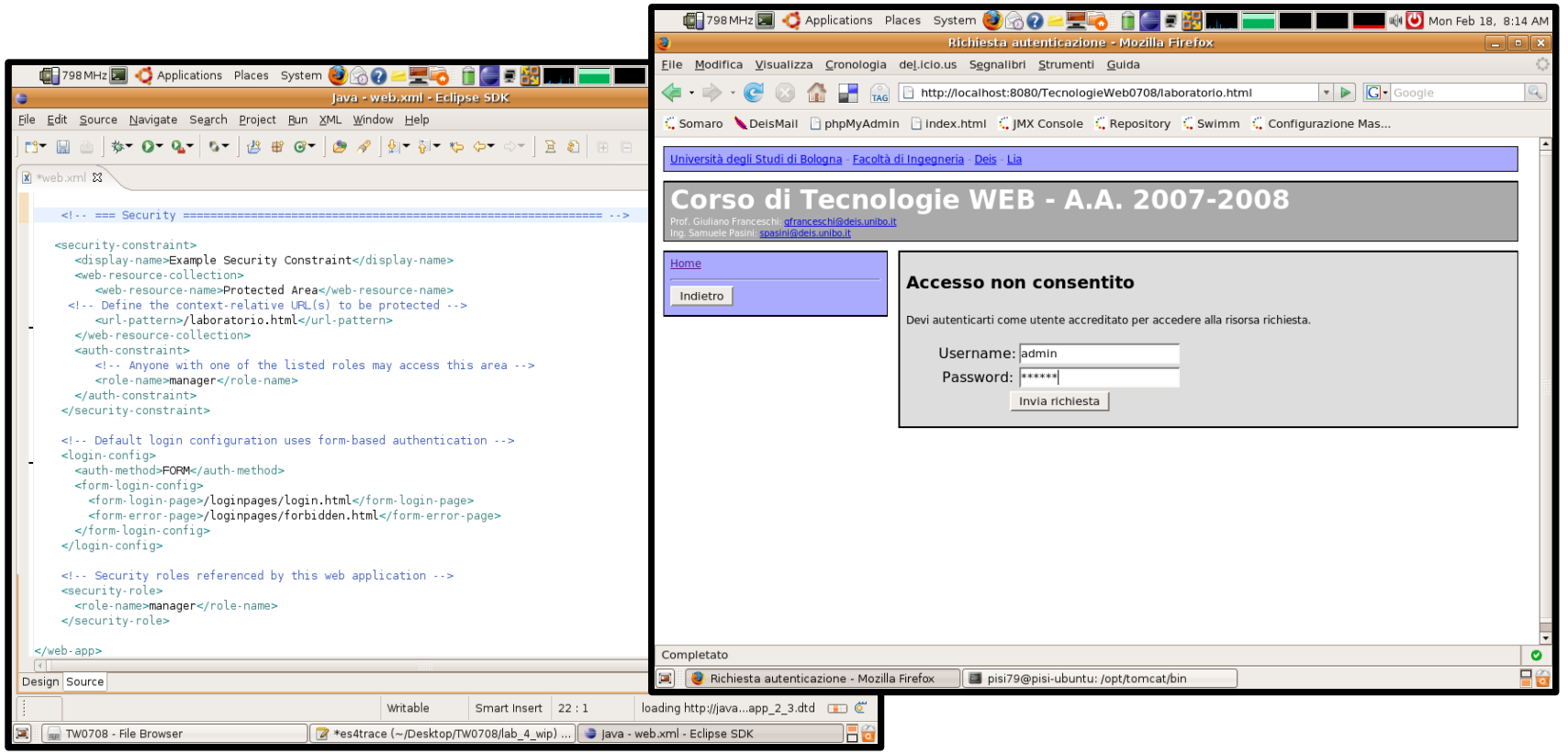
- Download it and import it as usual + set up *environment.properties*
- You can sniff scripts, styles and pages behaviour...
- ...or try do it better!



Inside the course web site replica project: security

> The *web.xml* descriptor shows how to secure part of the site

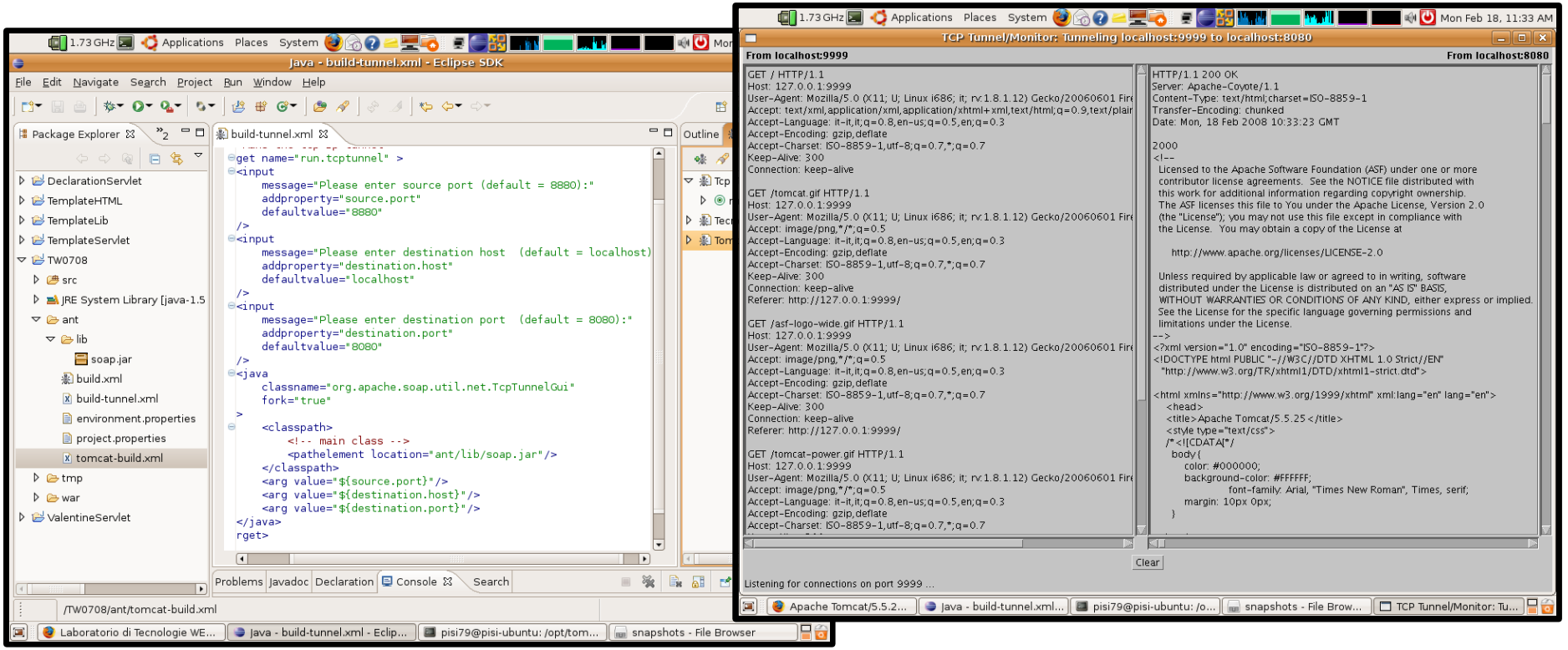
- either via explicit login/error pages or popup forms (commented)
- by leveraging Tomcat users/roles, for the sake of simplicity (by now)



Inside the course web site replica project: ANT & tunnel

> There's a particular ANT build file (*build-tunnel.xml*) that runs a configurable tcp tunnel in a GUI (= monitor) mode.

- You can listen on port, say, 9999 and forward requests to 8080
- Run the ANT target, browse to *http://localhost:9999* instead of *http://localhost:8080* and see what happens



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Server-side code: servlets

> Servlets are J2EE modular components that run on the server-side and process requests to programmatically generate dynamic responses

> How to create a servlet

- Write code and descriptors (for instance within an Eclipse project)
- Add static resources (HTML pages, styles) and client-side scripts if needed
- Package all the stuff into a .war file
- Deploy it to the web server

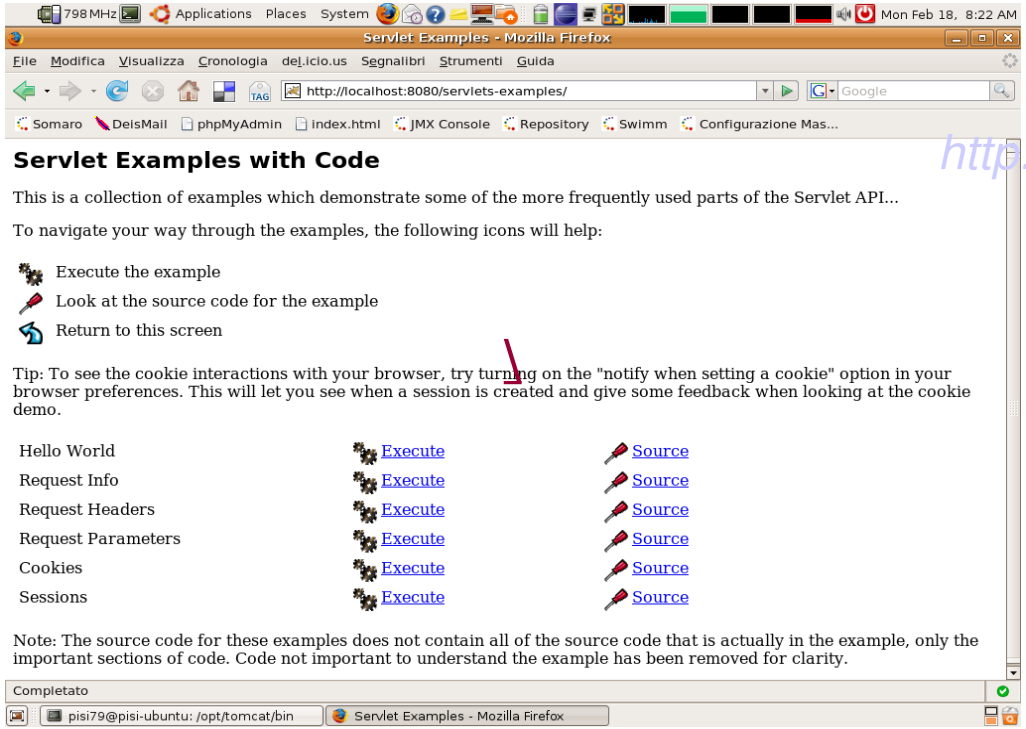
You can use the *TemplateWebapp* project as a starting point

or the *TemplateServlet* one (with just some sample code inside, in addition,

just to get you started immediately)

Servlet examples

> Tomcat provides, out-of-the-box, some servlet (and JSP) examples that can be useful to learn how to deal with Servlet's API and facilities

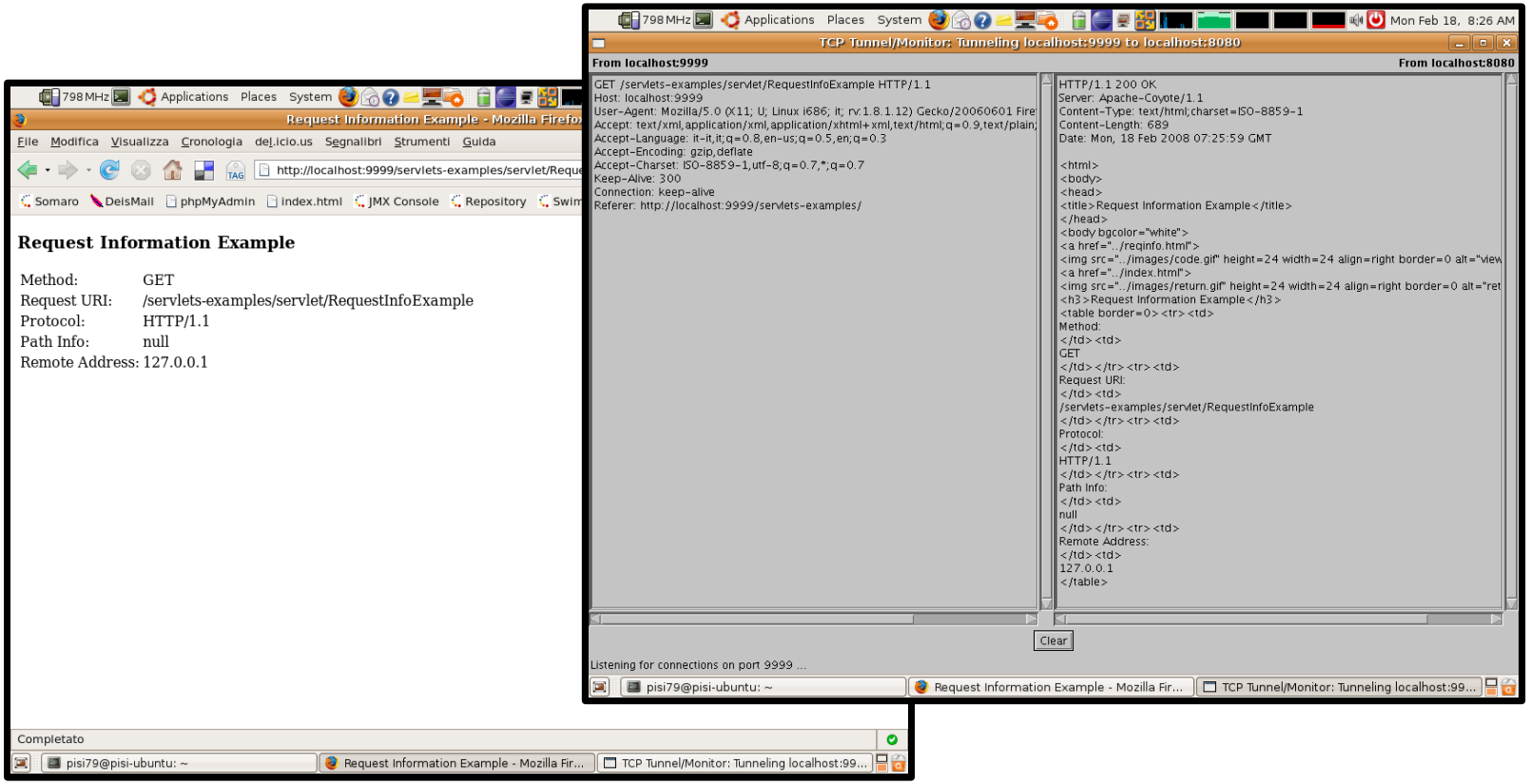


- You can invoke examples on *http://localhost:8080/servlets-examples/* or by following links on your Tomcat local home page
- Source code for the examples is partially accessible via web (through the example pages themselves) and completely available on you file system on *$TOMCAT_HOME/webapps/servlet-examples*

For instance: request info (1/2)

> The 'Request Info' servlet shows information about the request being handled

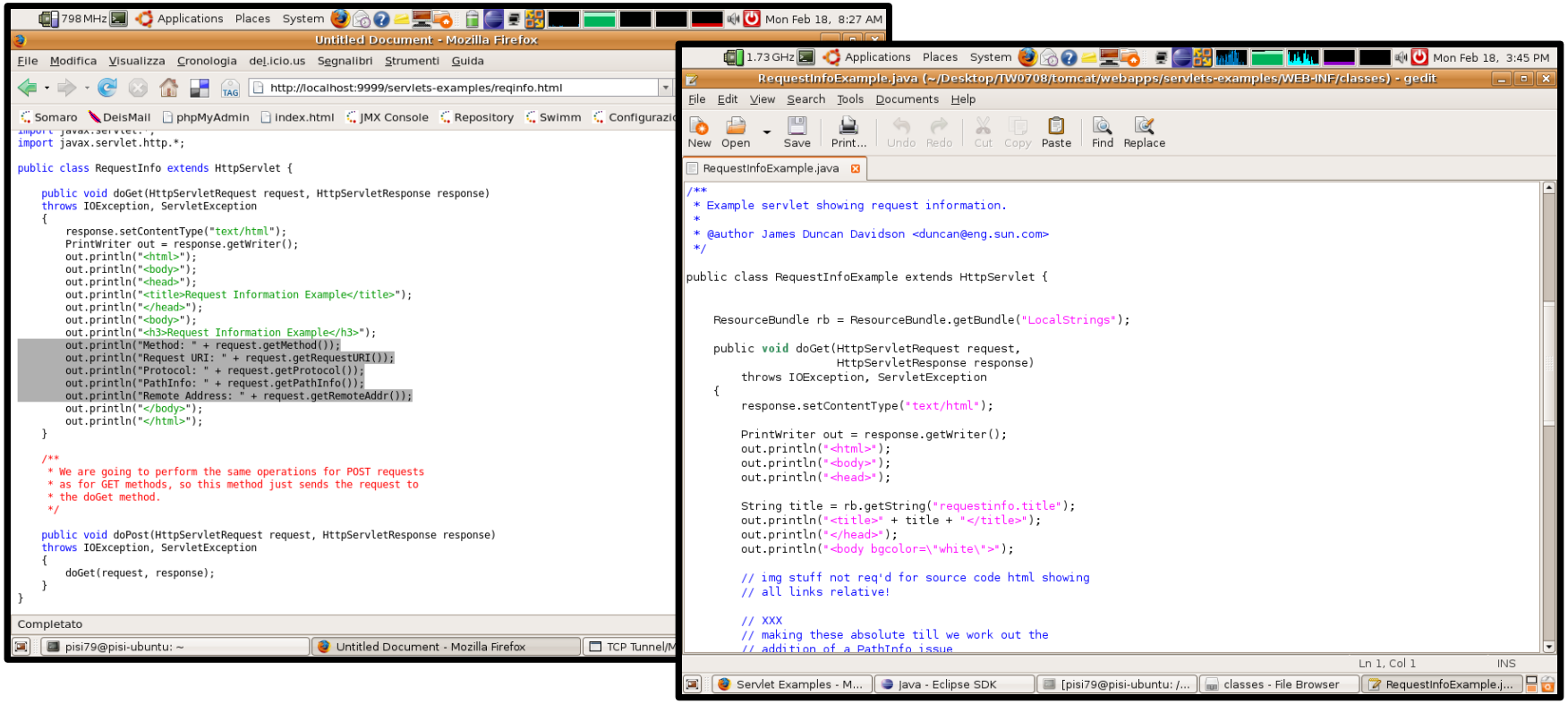
> You can check request arguments and headers against the tunnel GUI



For instance: request info (2/2)

> The 'screwdriver' link brings you to code's relevant excerpts

> Alternatively, you can open the *RequestInfoExample.java* source file on your file system (path: *$TOMCAT_HOME/webapps/servlets-examples/WEB-INF/classes*)



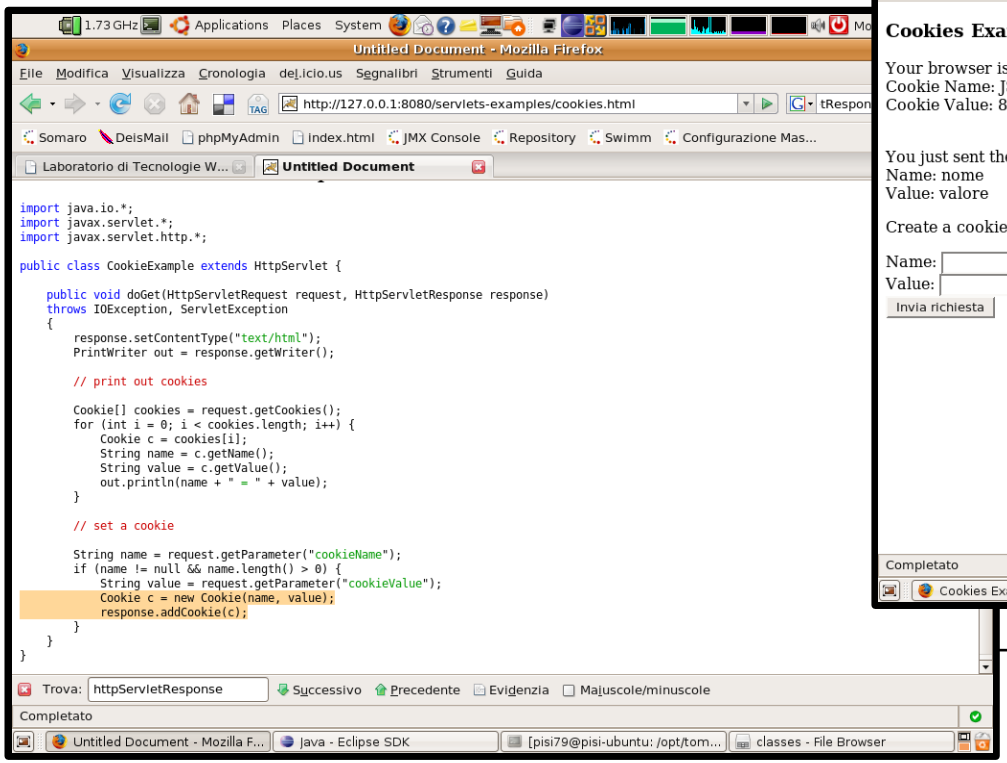
For instance: cookies

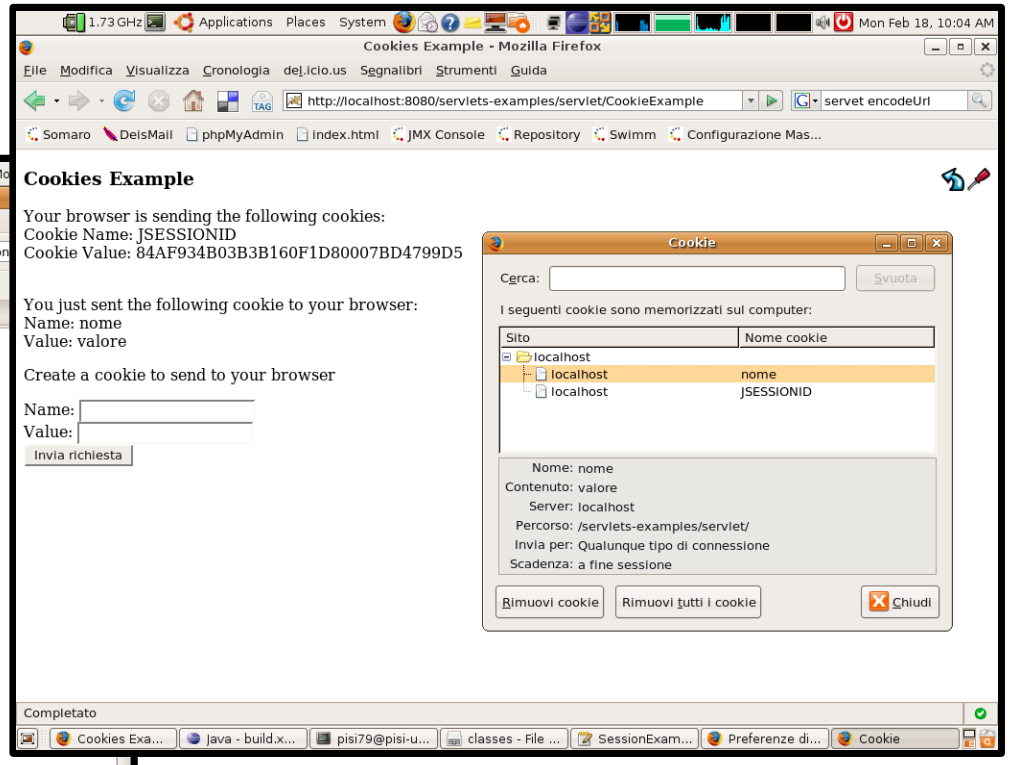
> Cookies are pieces of information, kept client-side

> Each cookie is associated to the domain that has saved it

> The 'Cookie Example' servlet shows how to handle cookies

> The browser settings can show you the cookies in your browser

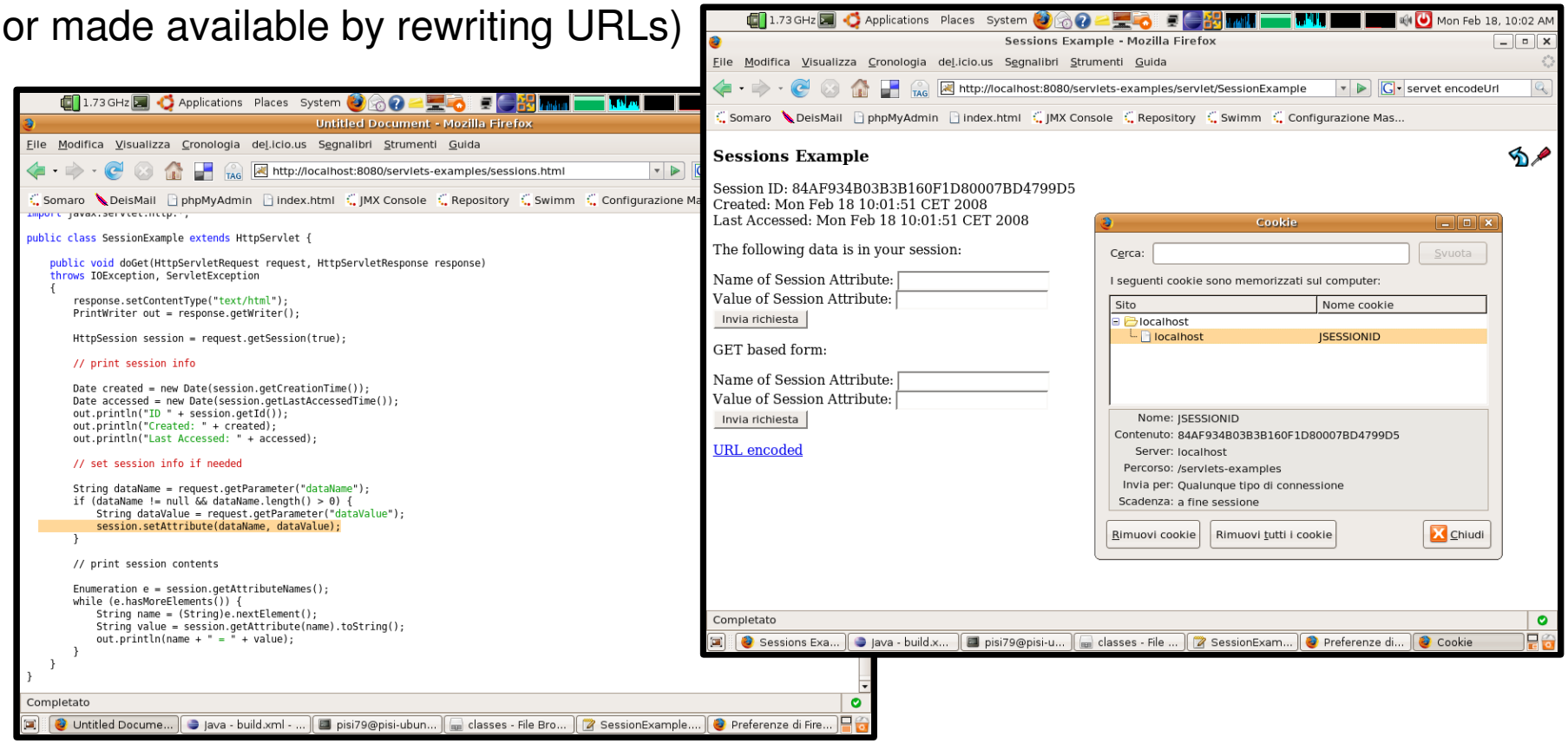




For instance: session

> Session holds pieces of information, kept server-side

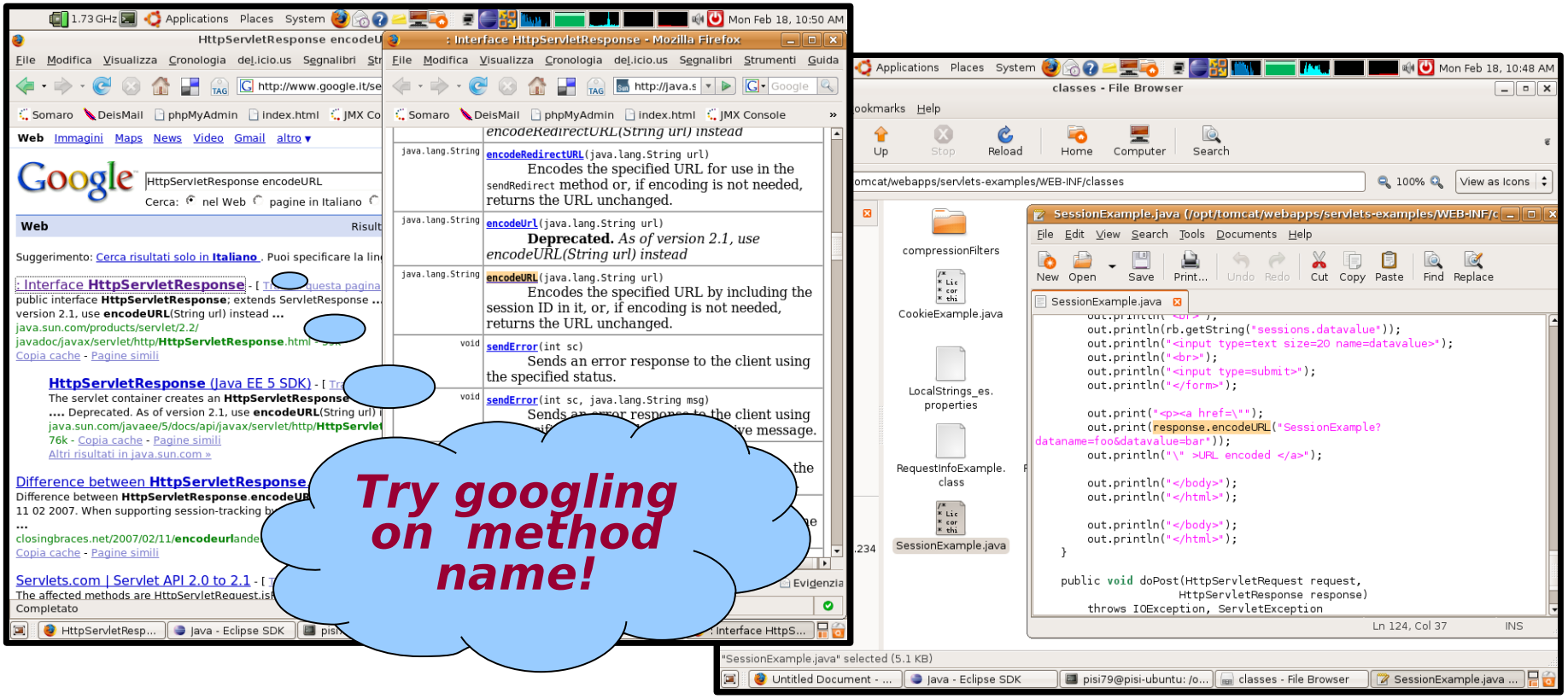
> Clients can retrieve their sessions by means of session identifiers (kept in cookies or made available by rewriting URLs)



For instance: searching for documentation (1/2)

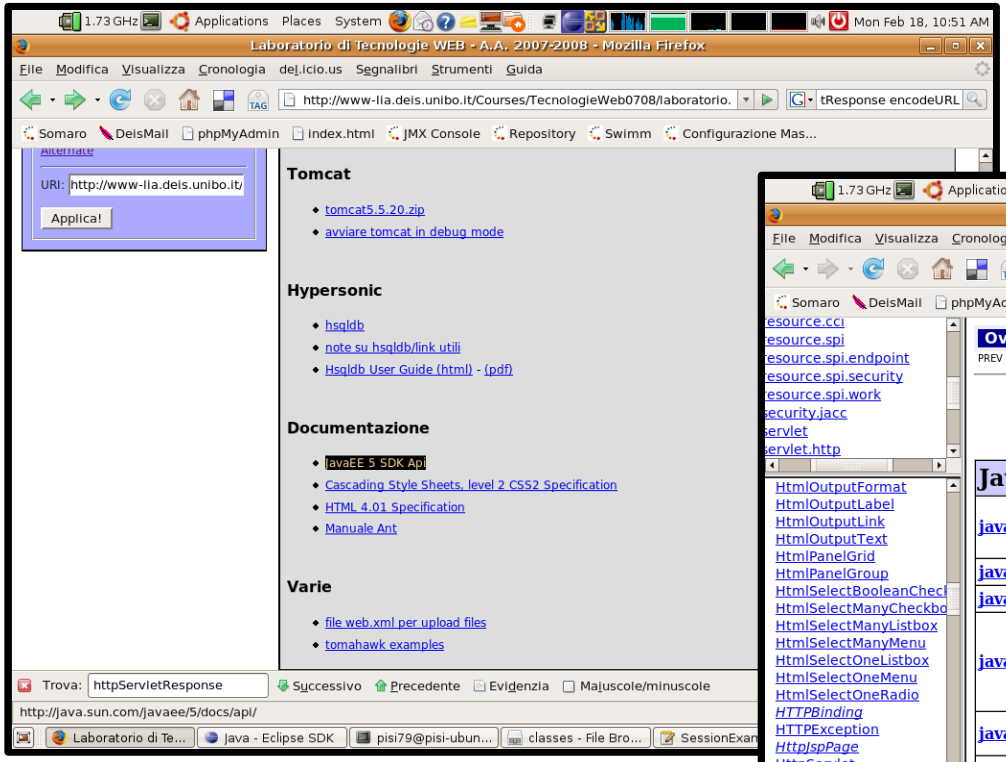
> Do you remember what URL encoding provides for?

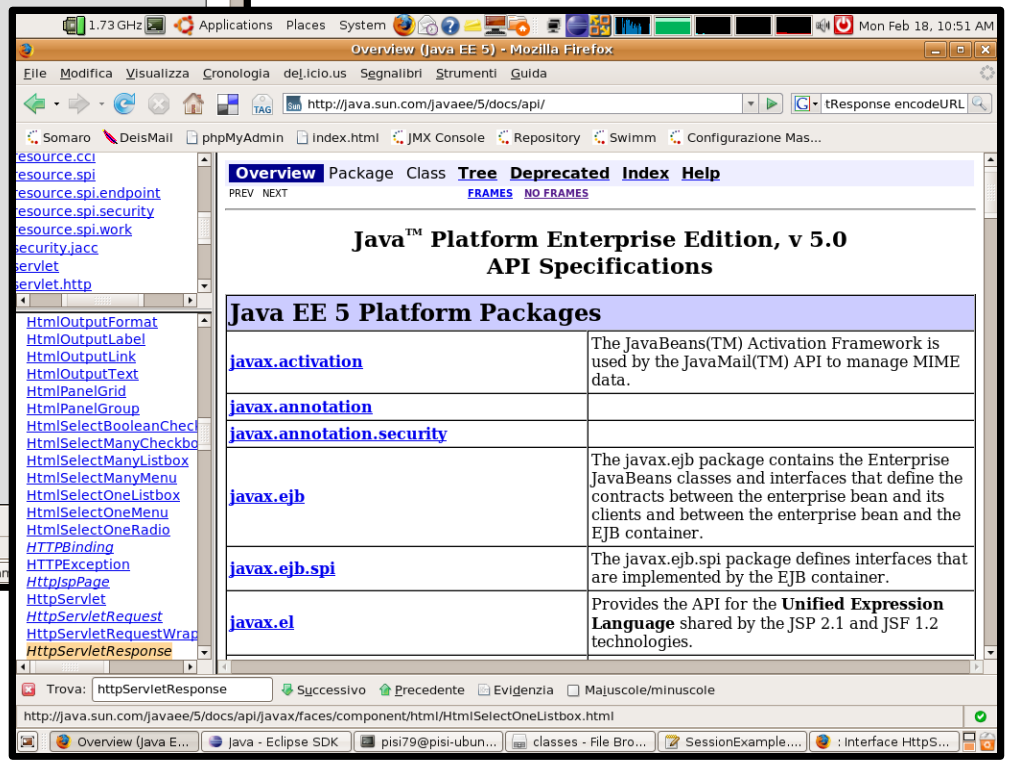
- The code shows it is generated by invoking `response.encodeURL()`
- Where to find more information? → ***Browse the API documentation!!***



For instance: searching for documentation (2/2)

> Besides, you can find a local copy of JEE API documentation on the course site

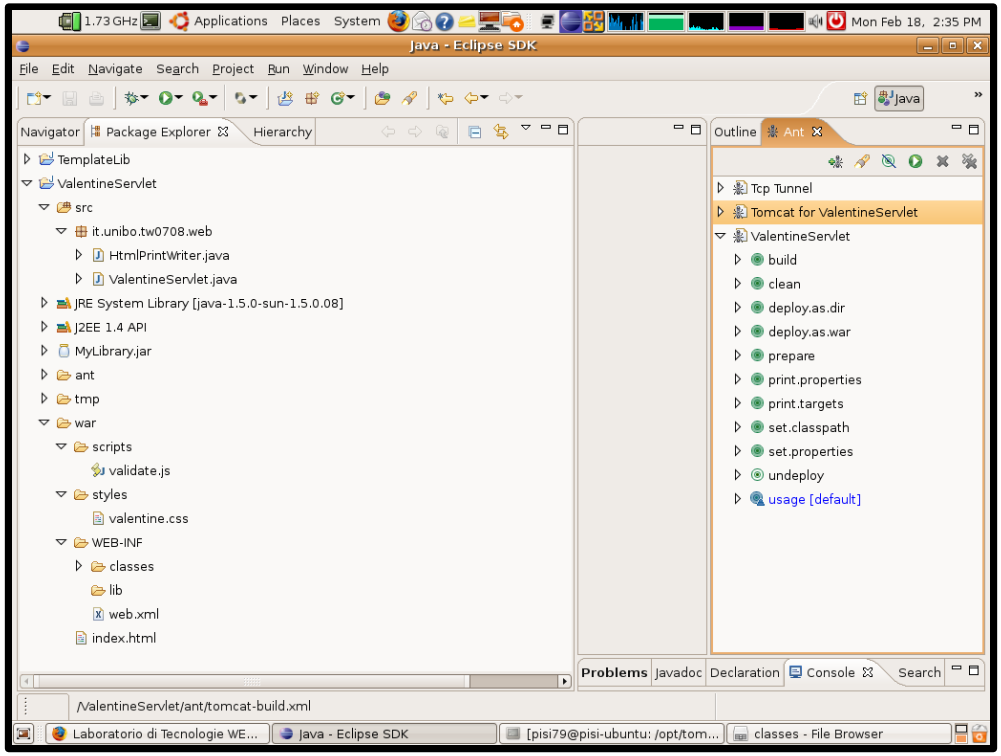


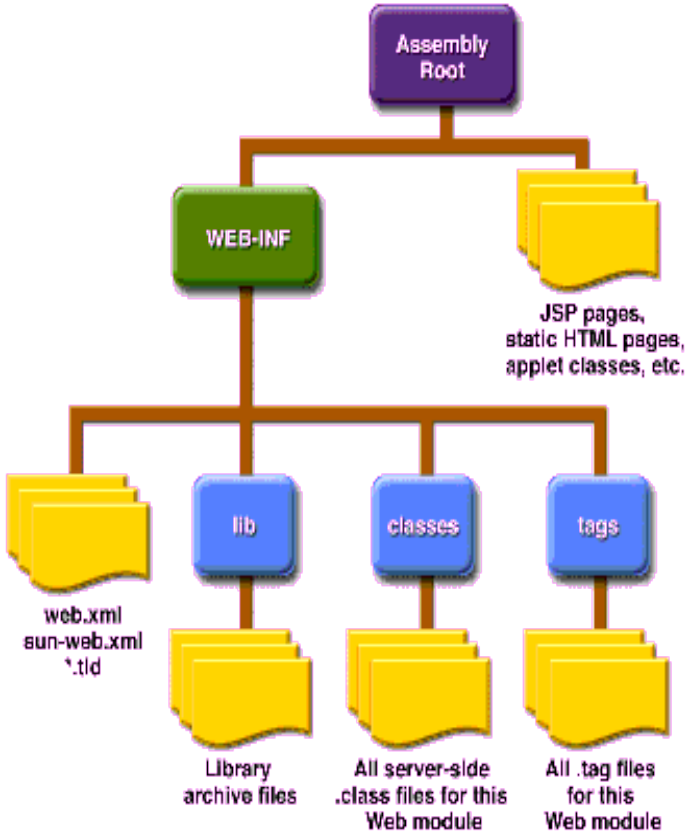


Valentine's servlet: project structure

> Download ValentineServlet.zip project from the course site and import it in Eclipse

> This silly servlet computes the chances of... love affairs *(..wrote it on Feb 14th ...)*



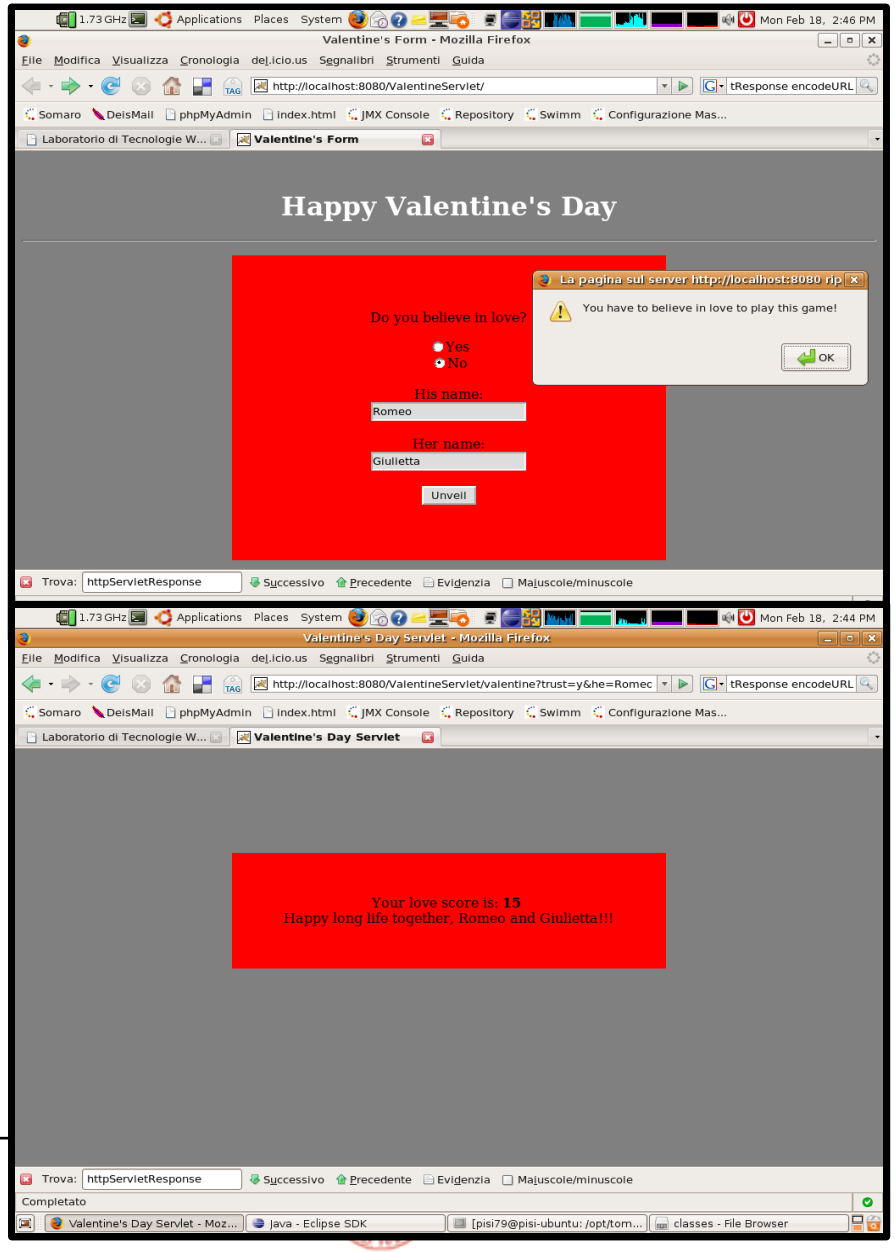


Valentine's servlet: elements

> Just a sample servlet project:

- **main page** displays a form
- **stylesheet** provides for a '*lovely*' layout
- **form** action invokes a servlet, by passing input field values as parameters
- **javascript** validates form on the client-side before performing the request
- **servlet** invokes a **library** function to compute results

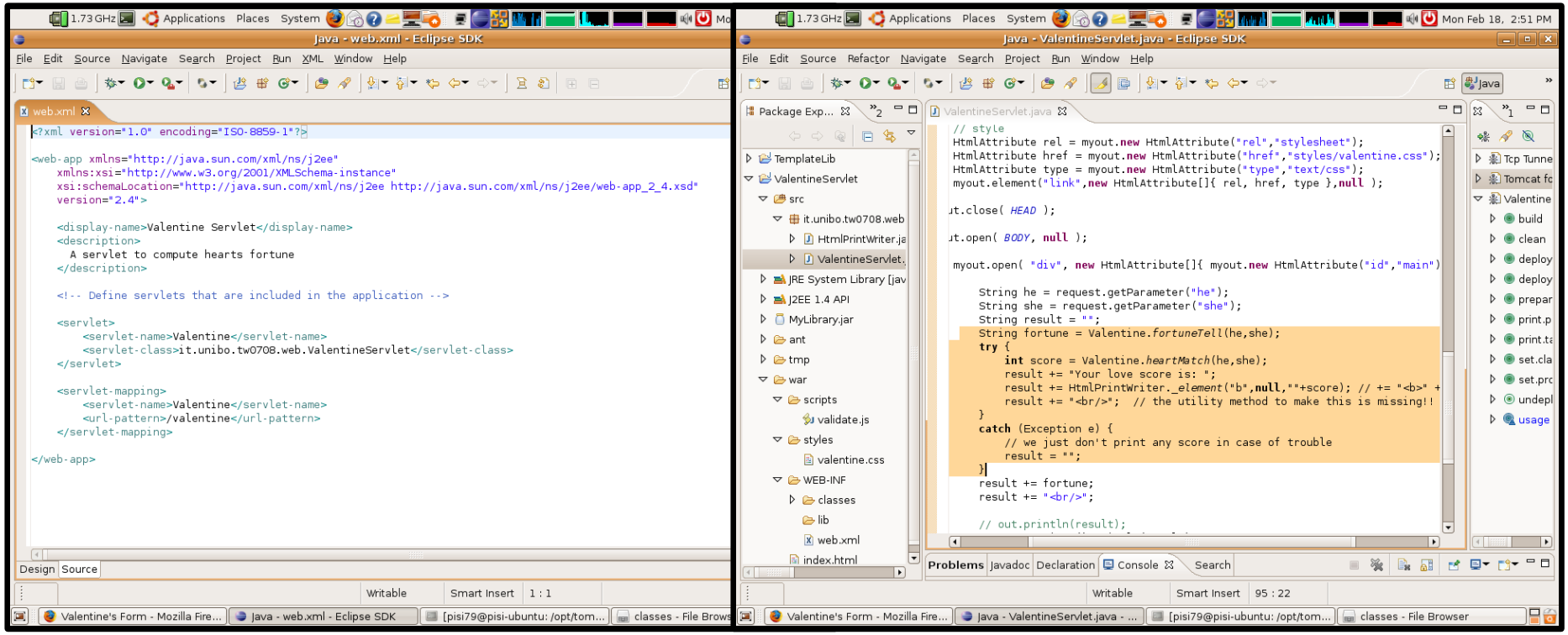
> Look at the project structure and find out elements!



Valentine's servlet: servlet mapping

> *web.xml* descriptor maps *ValentineServlet* class to */valentine* path by means of the servlet's *Valentine* name

> servlet retrieves request parameters and invokes library functions to produce result

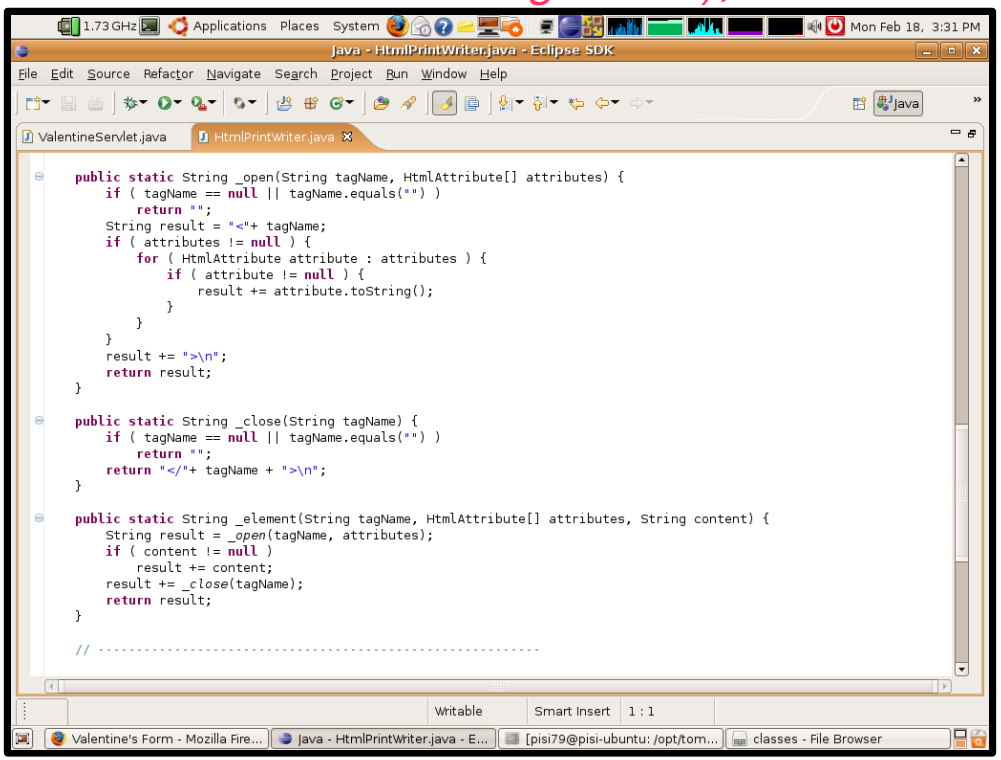


> library is stored in the *WEB-INF/lib/* path of the .war archive (note: Eclipse is not going to show it there, in the '*Package Explorer*' view, because it's part of the build-path!!!)

Valentine's servlet: not to mess up with output

> I just can't stand typing all those “<”, “>”, ”/”' and “\”” in the code

- so I wrote my own utility class (*HTMLPrintWriter.java*) that arranges tag strings on my behalf
- but you can just write *out.println(“ Something ”);* if it makes you fill more confortamble



Valentine's servlet: reusing libraries across projects

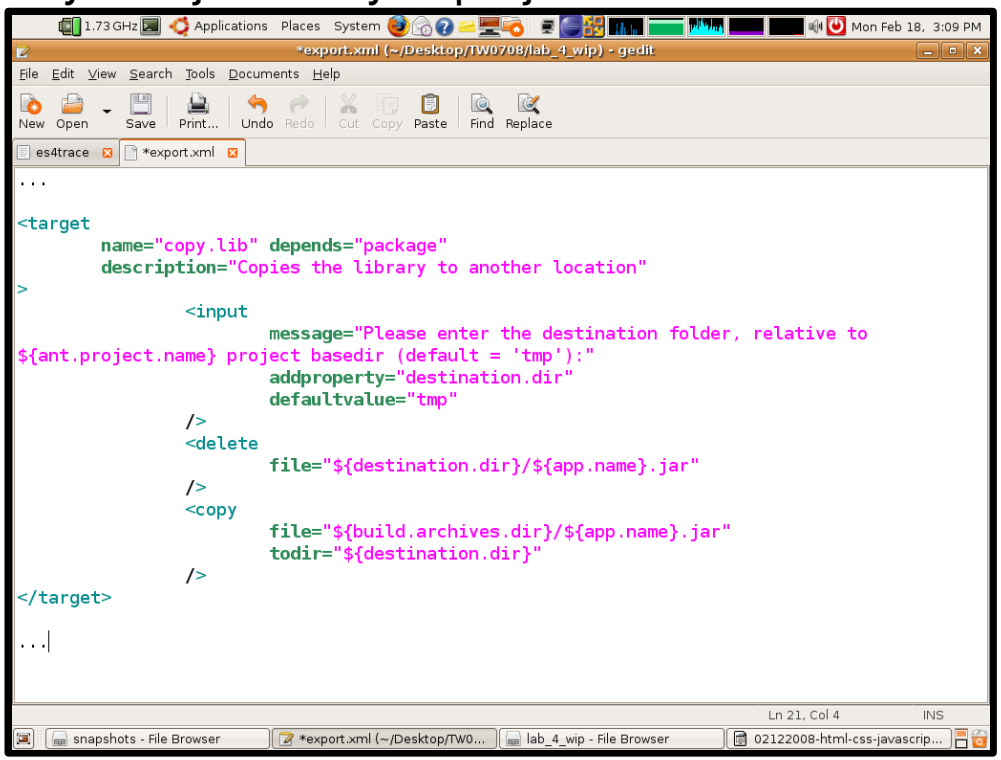
> *MyLibrary* is the result of the *'package'* target in the ANT build file of project *TemplateLib* (that you can download from the course web site and import in Eclipse)

> It's just a simple (not webap) project like the *TemplateApp* one of class #1

> But... do I have to copy and paste manually that jar library in projects where I need it?

Try to add a new ANT target do to that!

- you can copy delete, and copy tasks from the *deploy*-related targets of other projects
- you can add a popup message asking for the destination by copying from the target that runs the *TcpTunnel*
- and that's all



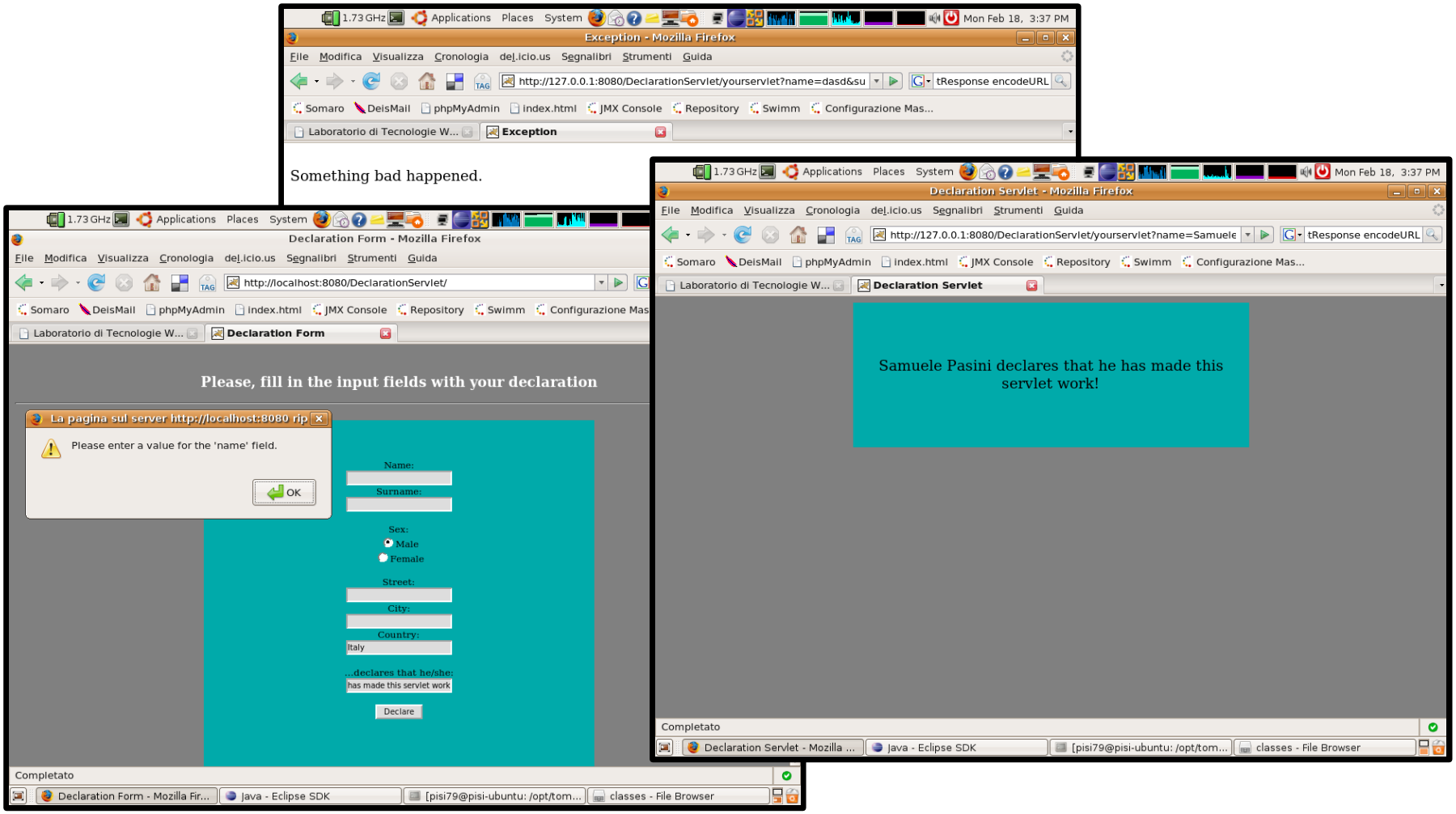
Declaration servlet: it behaves the same

> *MyLibrary.jar* also contains a class to assemble declaration statements
  - want to know how it works without looking at the code? try exporting the Javadoc to the */doc* path in the *TemplateLib* project!

> Try making your own servlet:
  - use the *TemplateServlet* project as a structure skeleton for your project
    - download the zip and import it in Eclipse
    - change project name, project.properties (and build files' project name too, if you want to) and set up your environment.properties file
  - a form that asks for name, surname, address, sex and a declaration
  - javascript code that ensures name and surname fields are non-blank ones
    - you can copy thst from the sample pages in *CSSandJavascript.zip*
  - a servlet that processes the request by invoking library methods & returns results
  - some styles to make things look pretty-good
  - perhaps you can handle exceptions via *web.xml* mapping to error pages

  (see what happens if people from hell perform declarations...)

Declaration servlet: for instance...



> *DeclarationServlet* sample project wil be online after the lab class...