Universita’ degli Studi di Bologna
Facolta’ di Ingegneria

Anno Accademico 2007-2008

Laboratorio di Tecnologie Web

Sviluppo di applicazioni web
primi passi: HTML

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

Just to revise...

> Java SDK

- http://java.sun.com/javase/downloads/index.jsp

> Eclipse IDE

- http://www.eclipse.org/downloads/

> Troubleshooting

- http://www.google.com/

Java EE web applications

> Java EE architecture:

- **business logic is organized into reusable components**
- Java EE servers provide **underlying services** in the form of **a container for every component type** (security, transaction management, naming, persistence, ...)

> Before a component can be executed, it must be...

- compiled in the same way as traditional Java programs
- assembled into a Java EE application (well formed and in compliance with the Java EE specification)
- deployed to its container

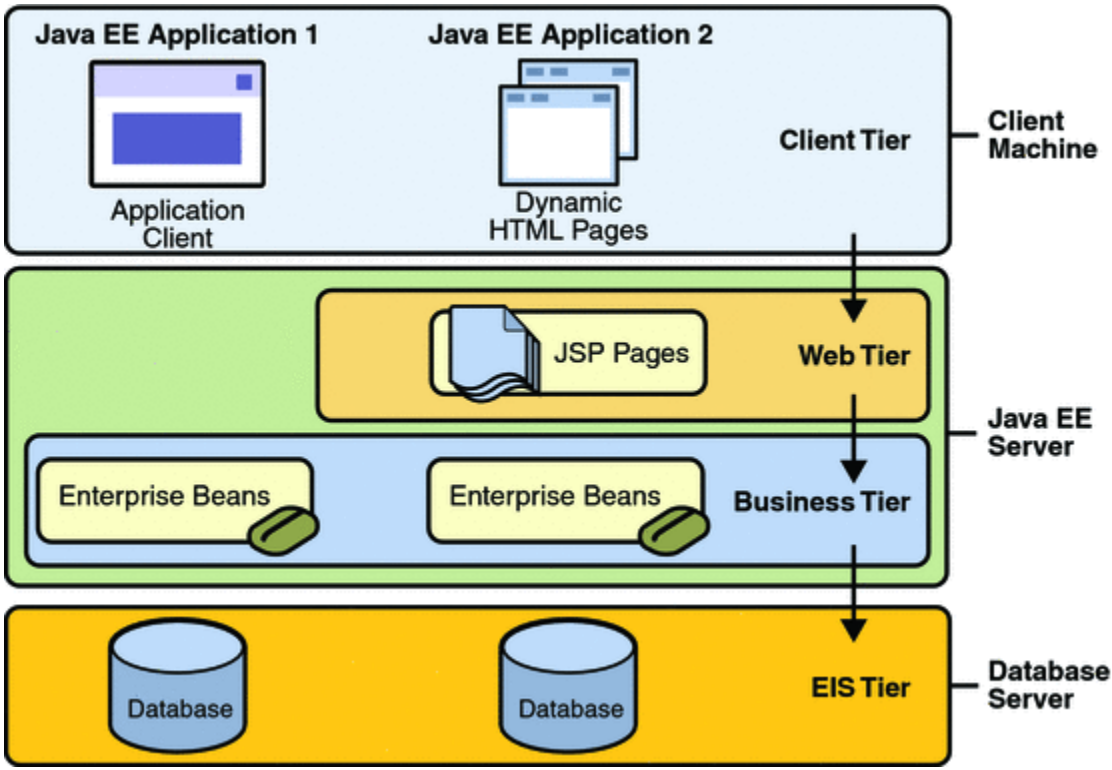
> JEE components are **run and managed by the Java EE server**

- a JEE server realizes the runtime portion of the Java EE framework

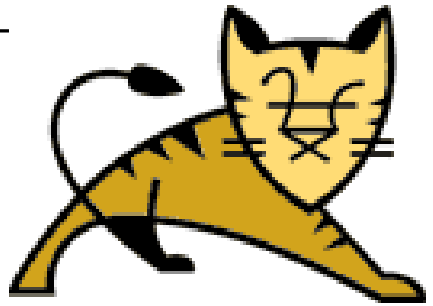
Java EE components

> The Java EE specification defines the following Java EE components:

- **Application clients** and **applets** are components that run on the client.
- **Java Servlet**, **JavaServer Faces** (JSF), and **JavaServer Pages** (JSP) technology components are web components that run on the server.
- **Enterprise JavaBeans** (EJB) components (enterprise beans) are business components that run on the server.



Tomcat – A JEE *Web* server



> Provides a **Servlet** and **JSP** container.

> Just download and explode it

- http://tomcat.apache.org
- http://www-lia.deis.unibo.it/Courses/TecnologieWeb0708/materiale/laboratorio/applicazioni/apache-tomcat-5.5.20.zip

> Tomcat structure

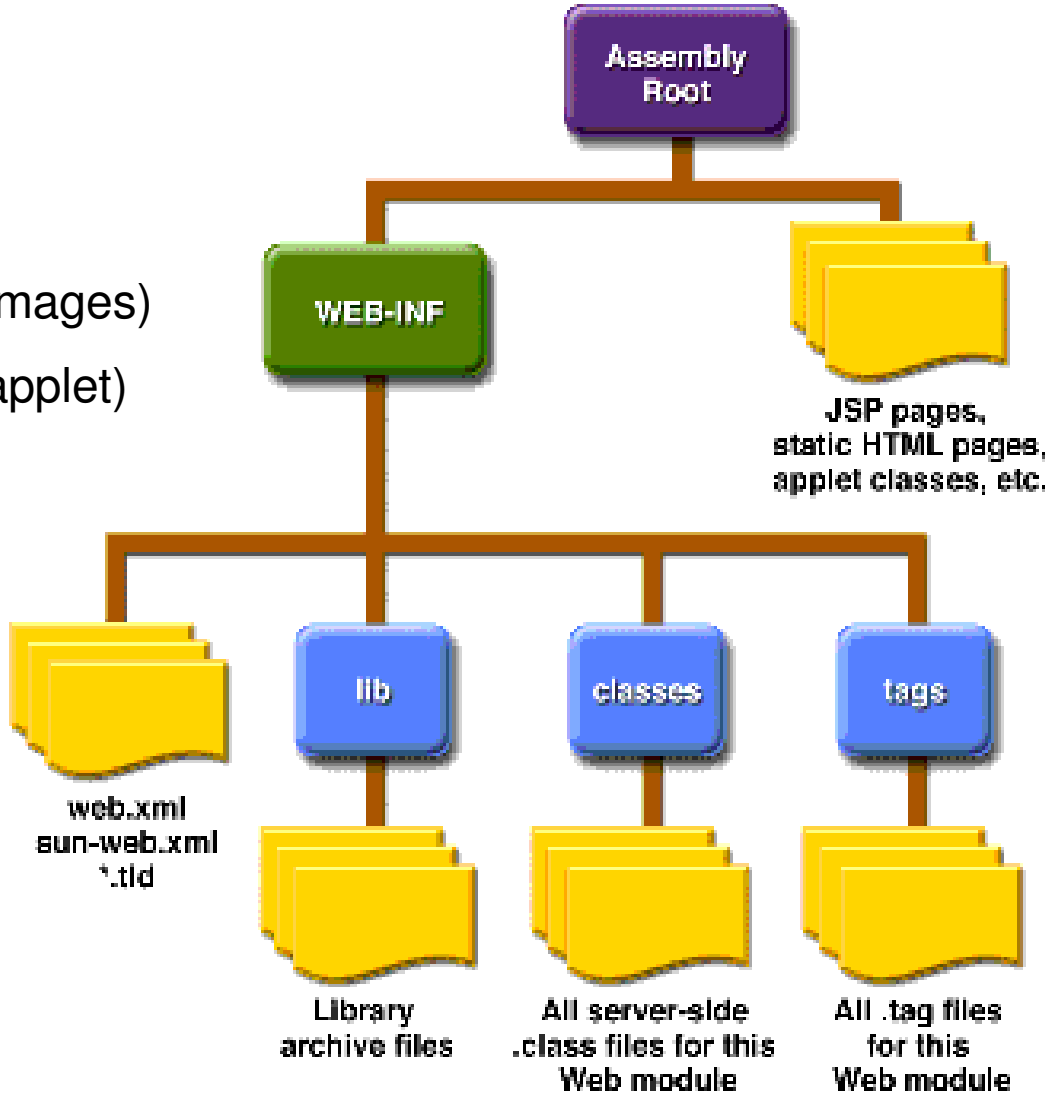
- **webapps**: where to deploy your web applications
- **conf:** configuration files (e.g., configure tomcat users: the admin)
- **bin:** launch and shutdown scripts
- **logs:** log files (e.g., *catalina.out*) will be placed here
  - create this directory if it is not there
  - inspect catalina.out file in case of troubles

> Launch Tomcat

- Set (and export, in Linux) your JRE_HOME environment variable
- http://localhost:8080. Ok. It works.

Web application structure

- Web pages
  - HTML and JSP pages
  - CSS and Javascript
  - Static resources (e.g., images)
  - Client-side code (e.g., applet)
- Deployment descriptors
  - security constraints
  - component aliases
  - resource mapping
  - ...
- Server-side code
  - classes
  - libraries



Web application structure

> The top-level directory of a web module is the document root of the application.

- web pages, client-side classes and archives, static web resources
- /WEB-INF/ subdirectory:
  - *web.xml*: the web application deployment descriptor
  - tag library descriptor files
  - *classes*: server-side classes (e.g., Servlets, utilities and JavaBeans)
  - tag files, which are implementations of tag libraries (later on...)
  - *lib*: JAR archives of the libraries server-side classes depends on
- application-specific subdirectories (that is, package directories) in either the document root or the /WEB-INF/classes/ directory.

> A web module is portable across different JEE servers and can be either deployed as an unpacked file structure or packaged in a JAR file known as a **web archive** (***WAR***) file.

Web application development

The process for creating creating, deploying, and running a web application is different from that of traditional stand-alone Java classes:

1) Develop the web component code and resources (e.g., HTML pages)

2) Develop the web application deployment descriptor

3) Compile the web application components and helper classes referenced by the components

4) (optionally) package the application into a deployable unit

5) **Deploy** the application into a web container

6) Access a URL that references the web application, via the browser

A simple web module to publish HTML pages

> Download the template project you can find at

http://www-lia.deis.unibo.it/Courses/TecnologieWeb0708/materiale/laboratorio/esempi/TemplateWebapp.zip

> Import the project into your Eclipse workspace

- File → import → Existing project into workspace → Select archive file..
- *index.html* is already in place

> Set up application and environment specific property files

- ${jdk-home} is the path to your JDK (> 1.5) installation
- ${appserver.home} is the path to your Tomcat installation
- properties can be defined in terms of other properties

> ANT targets let deploy the module automatically

- ...provided you set up your environment, of course!
- launch the 'usage' target to verify it

Troubleshooting

> Tomcat won't start

- JRE_HOME is set properly?
- do you have TOMCAT_HOME/**logs** directory?
- what does TOMCAT_HOME/logs/**catalina.out** report?

> My web application is not working

- ***environment.properties*** are set up right? Check with the '*usage*' ANT target
- did you **undeploy** / **deploy.as.xxx** after modifying the app?
- browsers do caching! Especially for static HTML pages: clean up the cache!
- is the **web.xml** file a well-formed XML file? Check it (open it in Eclipse and/or see if *catalina.out* reports parsing errors... yes, it's quite a mess, but you can find errors in it)
- tomcat does caching too! **reload** your web app (if you enabled the manager, see the next slide) or **shutdown** and **startup** tomcat again!

Managing Tomcat

> Tomcat manager reports information on the deployed modules

- but you must authenticate first!
- set up a user for the tomcat manager (modify TOMCAT_HOME/**conf**/***tomcat-users.xml*** by adding *manager* role and corresponding usernames/passwords: see the file in the *tmp* folder of the TemplateWebapp project)
- now you can command Tomcat remotely, also via ANT

> The ***Context*** element represents a web application

- each web application is based on its corresponding Web Application Archive (WAR) file, or the directory containing the corresponding unpacked contents, as described in the Servlet Specification (version 2.2 or later).

> Take a look at what you can do with the manager...

- search for documentation, tutorials, guides, how-tos, etc...
- you don't have to learn everything, but just to learn how/where to find it!

The deployment descriptor

Web applications are configured via the web application deployment descriptor.

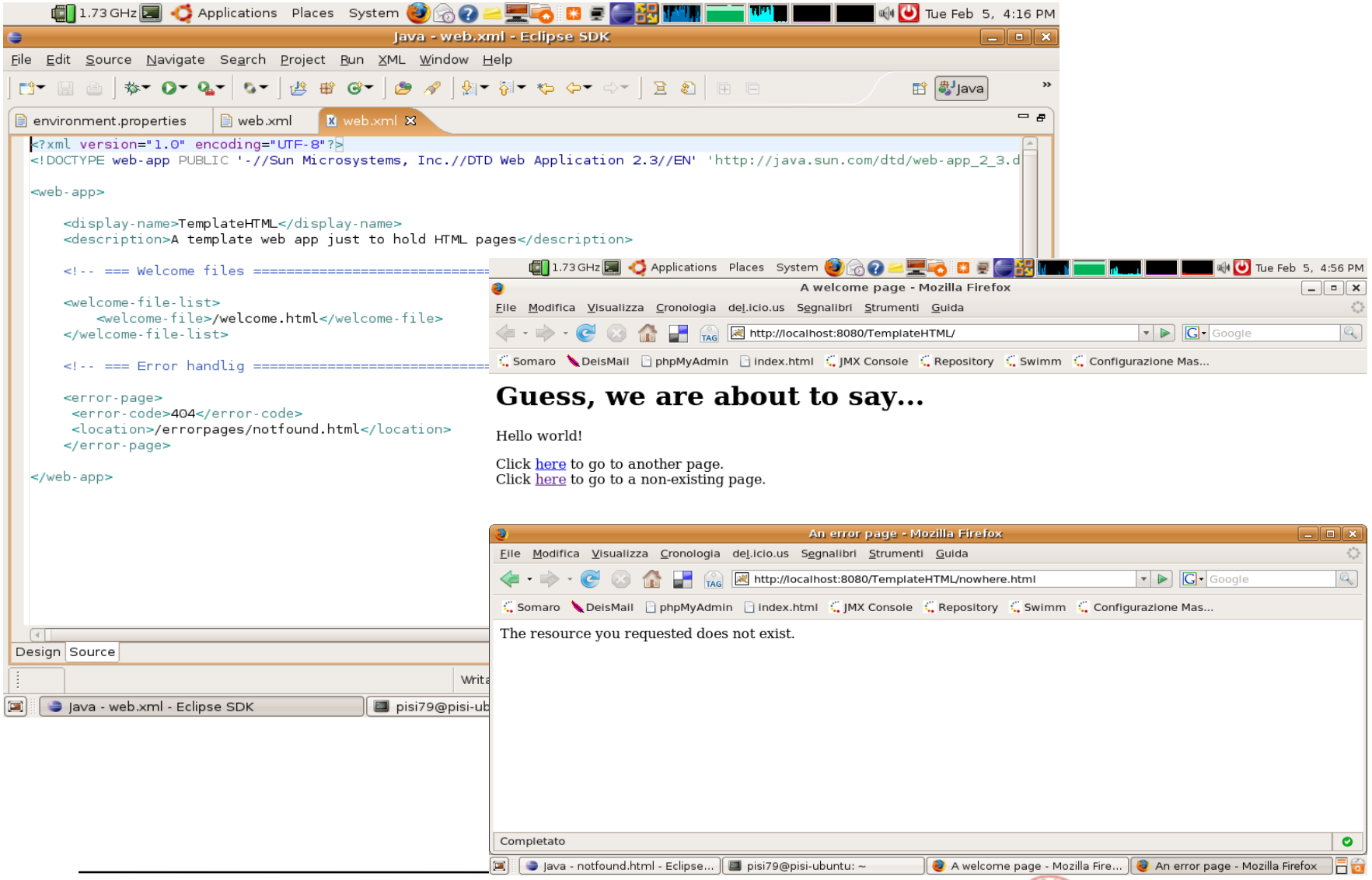
> Declaring welcome files

- The welcome files mechanism allows you to specify a list of files that the web container will use for appending to a request for a (valid) URI that is not mapped to a web component.
- If no welcome file is specified, the Application Server will use a file named index.XXX, where XXX can be html or jsp, as the default welcome file.
- If there is no welcome file and no file named index.XXX, the Application Server returns a directory listing (by default).

> Mapping errors to error screens

- when an error occurs during execution of a web application, you can have the application display a specific error screen according to the type of error.
- you can specify mappings between exceptions or HTTP status codes and error pages

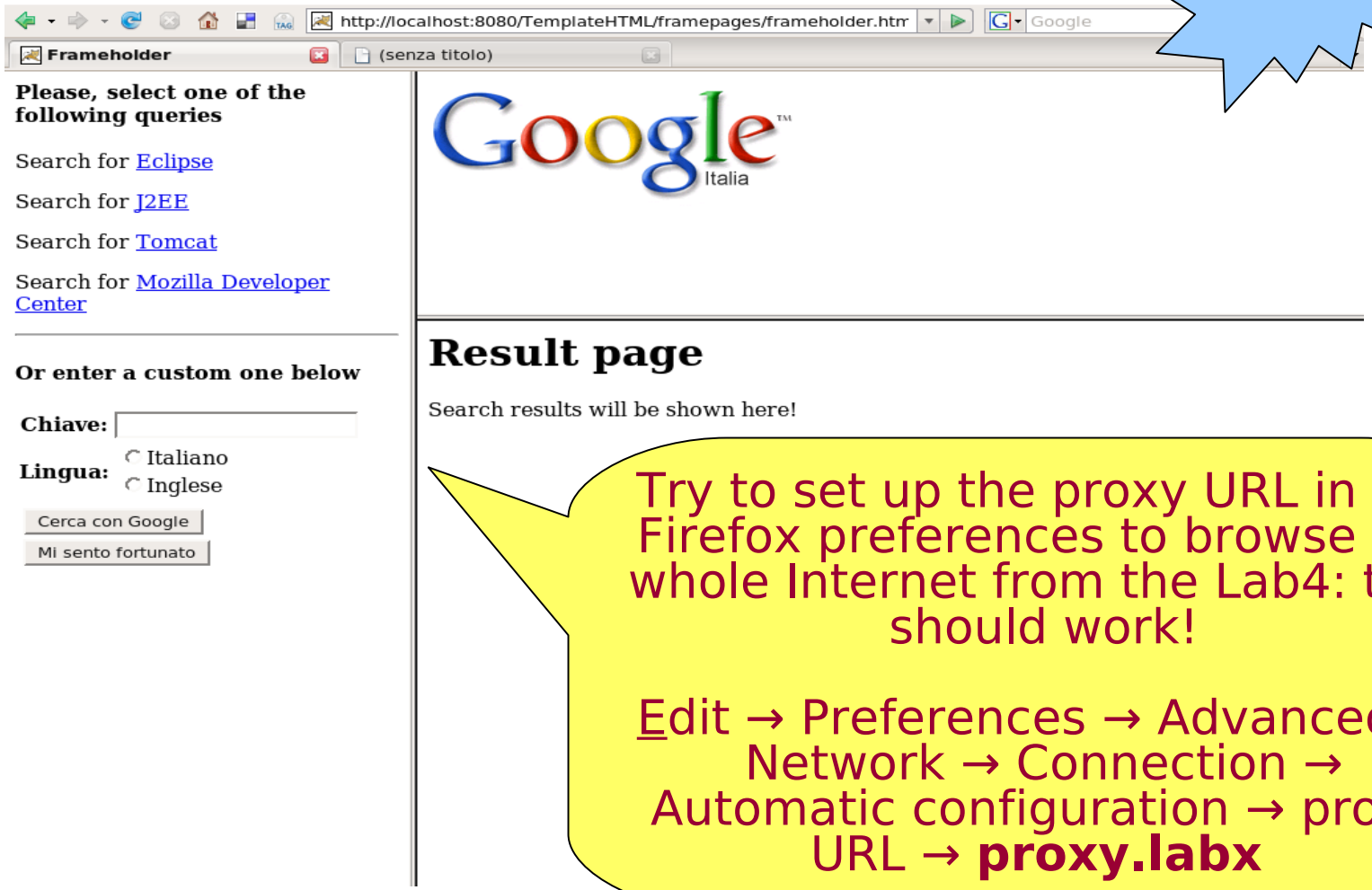
The deployment descriptor



Let's add more HTML pages

extra!

Try to do it your self (see friday class!)...



Try to set up the proxy URL in the Firefox preferences to browse the whole Internet from the Lab4: that should work!

Edit → Preferences → Advanced → Network → Connection → Automatic configuration → proxy URL → **proxy.labx**

The frameset holder page

```
<frameset cols="30%,70%">
    <frame src="search.html"/>
    <frameset rows="30%,70%">
        <frame src="logo.html"/>
        <frame src="results.html" name="results"/>
    </frameset>
    <noframes>
    <h1>Your browser does not support frames.</h1>
    <p>You can find the content by clicking on the following links:
        <ul>
            <li><a href="logo.html">Logo del motore di ricerca</a>;</li>
            <li><a href="search.html">Ricerce predefinite e altro</a>;</li>
            <li><a href="results.html">Risultati della ricerca</a>.</li>
        </ul>
    </p>
    </noframes>
</frameset>
```

The logo page

> There's no 'google.gif' in the project! We are just refererring it elsewhere!

- we sniffed the actual location from Google home page source code!

```
<img alt="Google" height=110 src="/intl/it_it/images/logo.gif" width=276>
```

> Instead, the 'programmer.jpeg' image is a local one.

```
<img src="http://www.google.it/intl/it_it/images/logo.gif"
alt="The google logo" onmouseout="this.style.visibility='visible'"
onmouseover="this.style.visibility='hidden'"/>
      +
<img src="../images/programmer.jpg" alt="The programmer image"
width="25%" height="50%" onmouseout="this.style.visibility='visible'"
onmouseover="this.style.visibility='hidden'"/>
```

> How come images appear and disappear as I move my mouse?

- HTML 4.01 supports events → D(ynamic)HTML
- Events + CSS + Javascript lead to dynamic pages and smart effects, but we can also do something nice just by using DOM !
- It's time to unveil ***FireBug***!

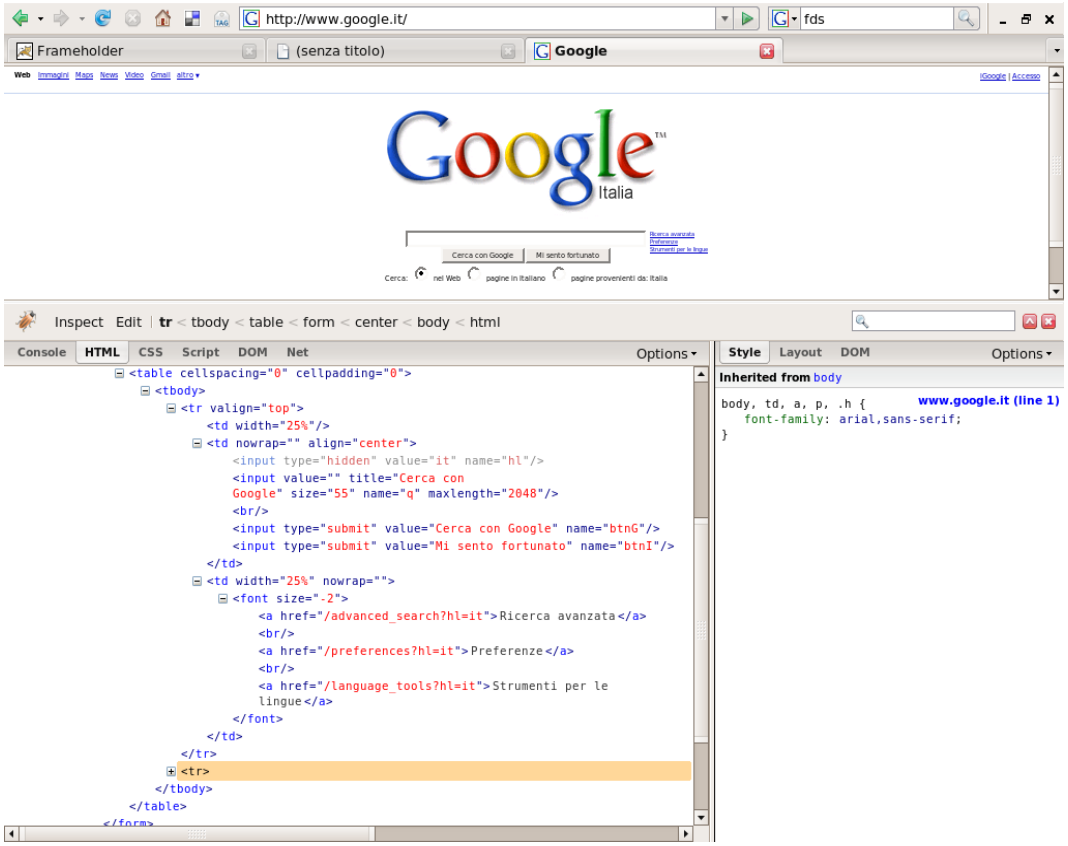
I can see clearly now...

> We are able to command google searches on our own since ***we can see*** what happens when we directly perform searches from the Google page!

- We can command searches just by resembling the format of real Google requests

http://www.google.it/search?hl=it&q=valore&btnG=Cerca+con+Google

- Google **form** is just a form: we can create one on our own
- Form method is '**GET**' → searches are all about requesting ***URIs*** → we can also make links to *URIs* representing Google searches!
- Hey! You don't need firebug to see HTML source, anyway!



Behaviour of the links and form page

- ***Anchor links*** are just hard-coded *URIs*
- ***Form input fields*** resembles Google ones (layout is fine thanks to tables!)

```
<h4>Please, select one of the following queries</h4>
<p>Search for <a target="results" href="http://www.google.it/search?hl=it&q=eclipse">Eclipse</a></p>
<!-- etcetera... -->
<hr/>
<h4>Or enter a custom one below</h4>
<form action="http://www.google.it/search" method="get" target="results">
    <table>
    <tr>  <th>Chiave:</th>  <td><input type="text" name="q" value=""/></td>  </tr>
    <tr>  <th>Lingua:</th>  <td>
        <table><tr><td><input type="radio" name="hl" value="it">Italiano</td></tr>
        <tr><td><input type="radio" name="hl" value="en">Inglese</td></tr></table></td>  </tr>
    <tr>  <td colspan="2">
        <table><tr><td><input type="submit" name="btnG" value="Cerca con Google"/></td></tr>
        <tr><td><input type="submit" name="btnI" value="Mi sento fortunato"/></td></tr></table>
        </td>  </tr>
    </table>
</form>
```