



Il Web, HTML e Java

Corso di Laurea in Ingegneria Informatica
Progetto S.C.E.L.T.E.

*Università di Bologna
Facoltà di Ingegneria*

Bologna, 08/02/2010

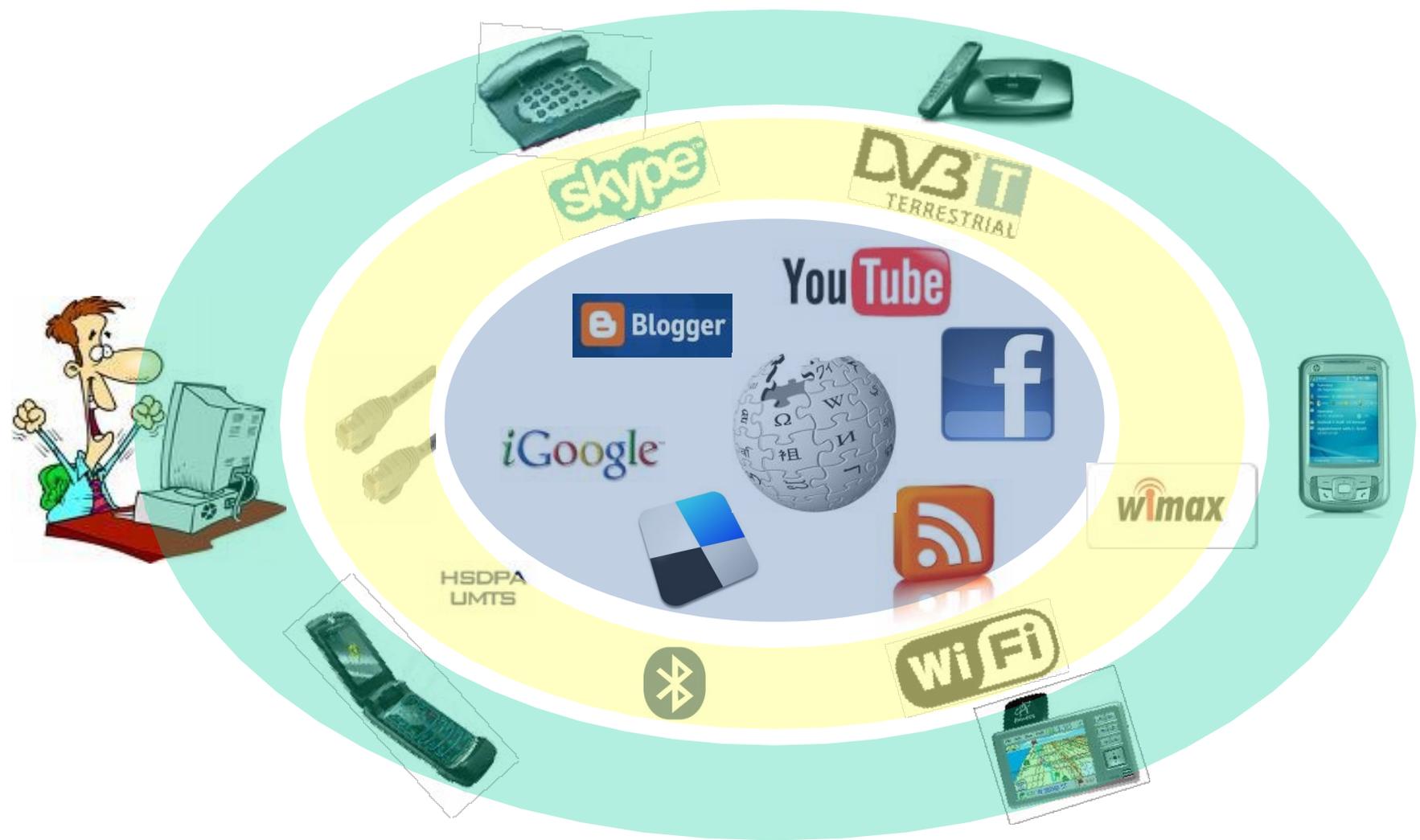


Outline

- Da applicazioni concentrate a distribuite
- Modello Web e HTML
- HTML e Java
- Parte pratica – Prima parte
 - Strumenti
 - Tool di sviluppo (Eclipse IDE)
 - Server Tomcat
 - Una prima pagina HTML di esempio
- Parte pratica – Seconda parte
 - Java Server Pages
 - Prima Pagina Web Dinamica



Applicazioni distribuite



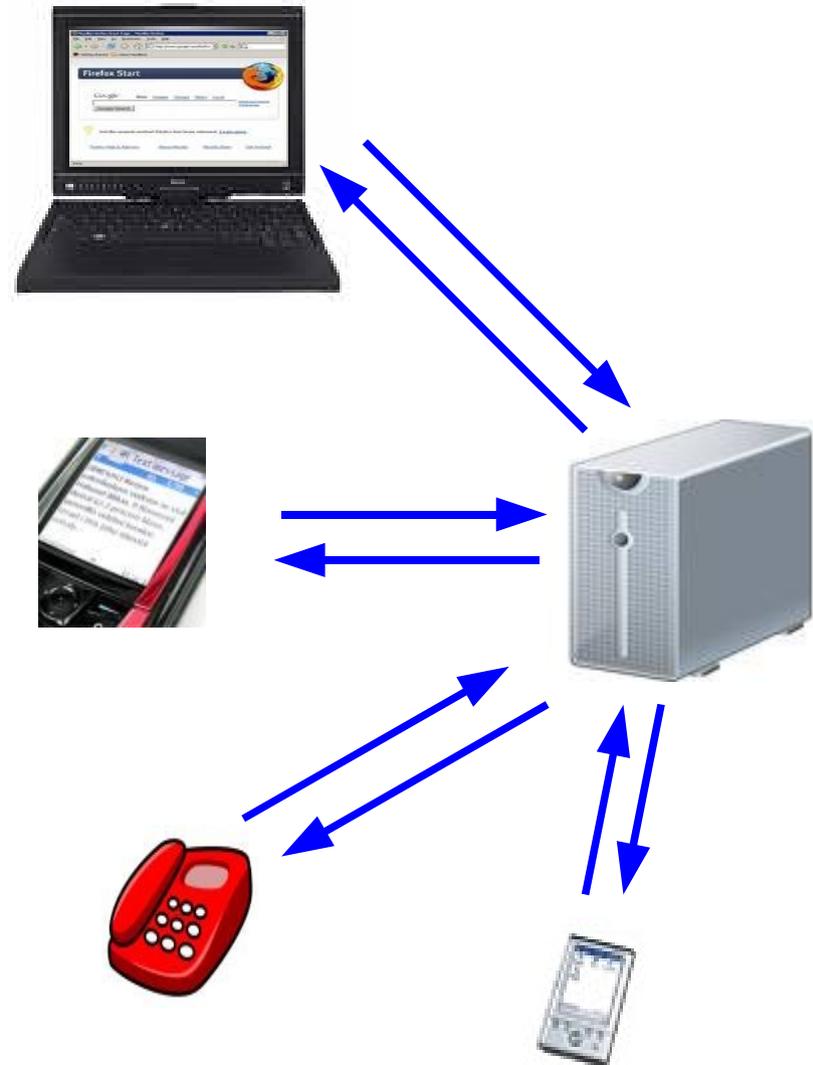


Modello Client/Server

Modello di collaborazione nel
distribuito

- **Client** richiede esplicitamente al server
 - Servizi, contenuti, ecc...
- **Server** remoto prepara ed invia la risposta

Modello asimmetrico (molti
client, un server)





// Web

WEB

- URL: come **identificare univocamente** risorse Web
- HTML: come **descrivere** risorse Web
- HTTP: come **accedere** a risorse Web

Tipico modello C/S

- Client Browser (Firefox /IE) utente richiede una risorsa Web
 - URL per indicare la risorsa
 - HTTP per comunicare col server (richiesta e risposta)
- Server WEB (es. APACHE Web Server)
 - Contenitore di pagine HTML
 - trova la risorsa HTML richiesta
 - manda indietro al client il codice HTML della risorsa

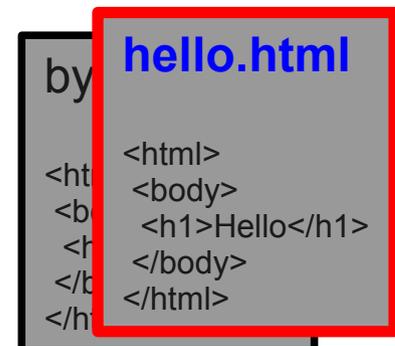
<http://somesite.com/hello.html>



```
<html>
<body>
<h1>Hello</h1>
</body>
</html>
```



<http://somesite.com/>





Linguaggio HTML

HTML: HyperText Markup Language

- definisce
 - **contenuto** Web
 - (in parte) formattazione/**visualizzazione** del contenuto
- documento **strutturato**
 - insieme di marcatori (**tag**) che definiscono proprietà del contenuto
 - ciascun marcatore delimita (apre e chiude la definizione) del contenuto
 - `<nometag>`: tag di **apertura**
 - `</nometag>`: tag di **chiusura**

Es: `<h1>Testo da formattare</h1>`



Linguaggio HTML

Struttura di una pagina HTML

<html>

Tag HTML

- delimita l'intero documento

<head>

<title>A study of population dynamics</title>

...

</head>

Tag HEAD

- racchiude informazioni opzionali
- es. titolo della pagina

<body bgcolor="white" text="black" link="red"

alink="fuchsia" vlink="maroon">

.....

</body>

</html>

Tag BODY

- racchiude corpo del documento
- diversi attributi possibili
 - *bgcolor*: colore dello sfondo
 - *text*: colore del testo
 - *link*: colore dei link



Linguaggio HTML

Pagina HTML di esempio

Welcome to Facebook - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://www.facebook.com/

facebook

Keep me logged in Forgot your password?

Email Password Login

Facebook helps you connect and share with the people in your life.

Sign Up
It's free and anyone can join

First Name:

Last Name:

Your Email:

New Password:

I am: Select Sex:

Birthday: Month: Day: Year:

Why do I need to provide this?

Sign Up

Console HTML CSS Script DOM Net

```
<html id="facebook" class="" lang="en" xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <body class="WelcomePage UIPage_LoggedOut ff3 Locale_en_US">
    <div id="content" class="fb_content clearfix">
      <div class="WelcomePage_Container">
        <div id="menuubar_container">
          <div class="WelcomePage_MainSell">
            <div class="WelcomePage_MainSellCenter clearfix">
              <div class="WelcomePage_MainSellLeft">
                <div class="WelcomePage_MainMessage">Facebook helps you connect and share with the people in your life.</div>
              <div class="WelcomePage_MainMap"> </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </body>
</html>
```

Style Layout DOM

```
182a3096.css (riga 146)
  color: #20336
  font-size: 20
  !important;
  font-weight:
  !important;
  line-height:
  margin: 1px 0
  0 6px;
  padding-
  right: 60px;
  word-spacing:
}
```

Ereditato da body.WelcomePa

6a3druoq.css (riga 9)

Done



Linguaggio HTML

Formattazione dei contenuti

Numerose possibilità per la formattazione di contenuto testuale

Tag ``

- permette di specificare una generica formattazione di testo
- attributi:
 - `size = [+|-]n`: definisce le dimensioni del testo (1-7 o relative)
 - `color` = definisce il colore del testo
 - `face` = definisce il *font* del testo (es. arial, verdana, ecc...)

Tag `heading`: permettono di specificare titoli e sottotitoli a differenti livelli di importanza

`<h1>` Titolo più significativo `</h1>`

`<h2>` Un po' meno significativo `</h2>`

`<h3>` Un po' meno significativo `</h3>`

`<h4>` Un po' meno significativo `</h4>`

`<h5>` Un po' meno significativo `</h5>`

`<h6>` Titolo meno significativo `</h6>`



Linguaggio HTML

Strutturazione dei contenuti

Altri tag per definire contenuti maggiormente strutturati

- tag `<p>`: gestione di paragrafi di testo
- tag `<table>`: creazione di tabelle
 - tag `<tr>` innestati per definire righe delle tabelle
- tag `` e ``: liste di elementi (ordinate e non)
 - tag `` innestati per definire elementi all'interno di una lista
- tag `<div>`: generici raggruppamenti (blocchi) di contenuto



Linguaggio HTML

Interazione con utente - Form

Tag **FORM** contiene elementi di controllo

- Interazione utente (inserimento dati, azioni, ecc...)
- vari elementi di controllo:

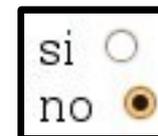
– Bottoni



– CheckBox (Switch on/off)



– Radio Buttons (Switch mutuamente esclusi)



– Menu di selezione (Lista di opzioni)



– Inserimento di testo



– ...



Linguaggio HTML

Esempio di form

```
<html>
```

```
<body>
```

```
<form action="http://localhost" method="GET">
```

Nome:

```
<input type="text" name="firstname"/>
```

```
<br/>
```

```
<input type="submit" name="azione"  
value="invia"/><br/>
```

```
</form>
```

```
</body>
```

```
</html>
```



**firstName=Stefano
azione=invia**



**Web Server
http://localhost**



Problema: HTML è statico!

HTML è stato pensato per la definizione di risorse statiche

Il contenuto di una pagina HTML (semplice) viene definito dallo sviluppatore e non cambia (a meno che lo sviluppatore non intervenga)

Infatti, riprendendo il modello di esecuzione

- utente, tramite browser richiede una pagina HTML
- il server
 - reperisce la pagina HTML
 - risponde (manda al client il contenuto HTML) mediante protocollo HTTP
- il browser visualizza il contenuto della pagina HTML ricevuta

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DIPARTIMENTO
DEIS - ELETTRONICA INFORMATICA

Facoltà

Dipartimento
Dottorati

Monti Stefano

- .. Relazioni(finali/proposte)
- .. Pubblicazioni
- .. Seminari
- .. Tesi di Dottorato

Materiale Scaricabile

- [Nullaosta-tutorato](#)
- [Compatibilità assegno di](#)

Dottotato di Ricerca in Electronics, Compu

Dottorando Monti Stefano, Ciclo XXI

Pagina di Gestione

Ad esempio: come è possibile creare una pagina personalizzata per ciascun utente?



Soluzione

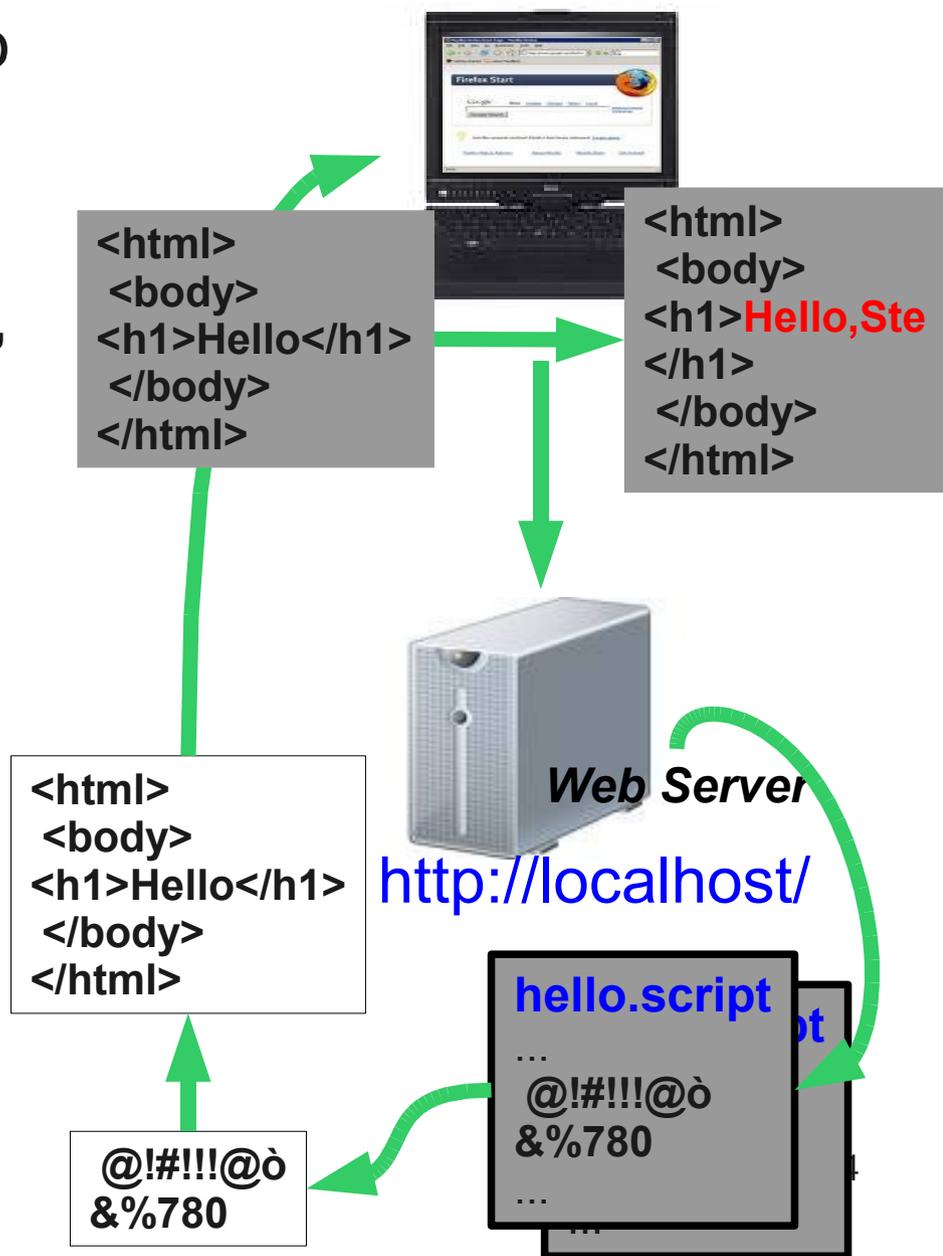
Varie soluzioni e tecnologie (spesso complementari):

- Lato **server**: CGI, PHP, Java Servlet/**Java Server Pages (JSP)**, ...

- Creazione **dinamica** di pagine HTML lato server

- Lato **client**: Javascript, Flash, ...

- Manipolazione del codice HTML ricevuto dal browser (client)





La piattaforma Java

- **Java Standard Edition**
 - applicazioni **locali** (desktop)
 - lanciare una applicazione **esplicitamente**
 - *java nomeClasseMain* (oppure *java -jar mainFile.jar*)
- **Java Enterprise Edition**
 - applicazioni **distribuite** (anche Web)
 - esiste un **server JEE** che ospita le applicazioni
 - gestione caricamento, ciclo di vita, disattivazione
 - niente più **main()** e lancio esplicito!!!
 - funzionalità **standard** di **supporto** per applicazioni complesse distribuite
 - es. protocollo HTTP



Applicazione Web J2EE

- Archivio WAR = archivio JAR + descrittore web (file [web.xml](#))
 - classi Java
 - risorse Web
 - Pagine HTML
 - JSP
 - Servlet
 - ...
- Non c'è più un *main()* !! L'applicazione
 - viene caricata dallo sviluppatore sul server J2EE
 - “lanciata” (resa accessibile agli utenti) automaticamente dal server

```
<?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
  version="2.4">

  <display-name>Progetto Web</display-name>
  <description>
    A sample Web Application
  </description>

  <context-param>
    <param-name>dao</param-name>
    <param-value>it.unibo.DaoFactory</param-value>
  </context-param>

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

  ...

</web-app>
```



Hands on...

Parte prima - Strumenti



Hands on...

Parte prima – Strumenti

- 1) Scaricare Tomcat J2EE Server
- 2) Installare e lanciare Tomcat (unzip...)
- 3) Scaricare Progetto di esempio
- 4) Configurare editor Java (Eclipse IDE)
 - Importare il progetto di esempio nell'editor
- 5) Caricare la prima applicazione Web Java
 - “impacchettare” e caricare la applicazione sul server
 - in questo caso: solo contenuto statico (pagine HTML)



Hands on...

Parte prima – Tomcat

- Scaricare Tomcat Servlet container all'indirizzo:

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

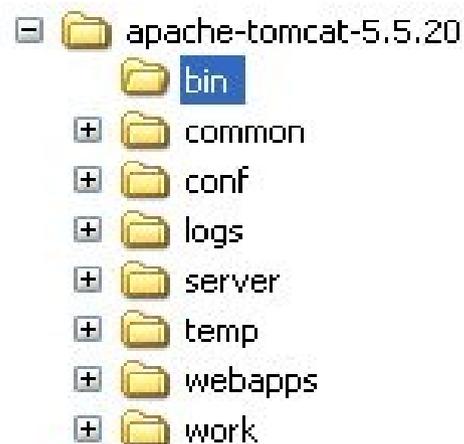
- Installare Tomcat == scompattare l'archivio!

– usare Winzip o WinRAR

- Lanciare Tomcat

– nella sottocartella **bin**

lanciare lo script **startup.bat**





Hands on...

Parte prima – Progetto di esempio

- Scaricare Progetto Eclipse di esempio

<http://lia.deis.unibo.it/Courses/TecnologieWeb0809/materiale/altro/CalcolatriceWeb.zip>

- Lanciare ECLIPSE

– Da menu *Start* → *Programmi* → *Eclipse 3.4*

- Importare il progetto in Eclipse

– *File* → *Import* → *Existing Projects into Workspace*
→ *Next*

– Selezionare *Select Archive File*

– Selezionare l'archivio appena scaricato



Hands on...

Parte prima – Progetto di esempio

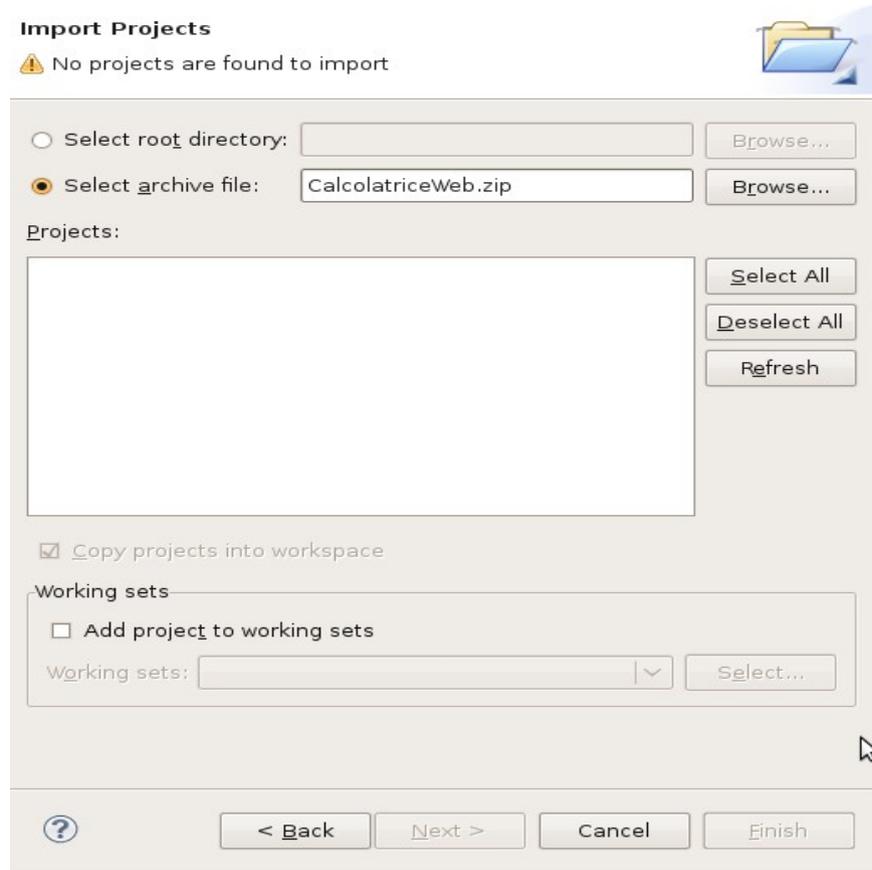
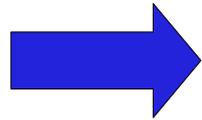
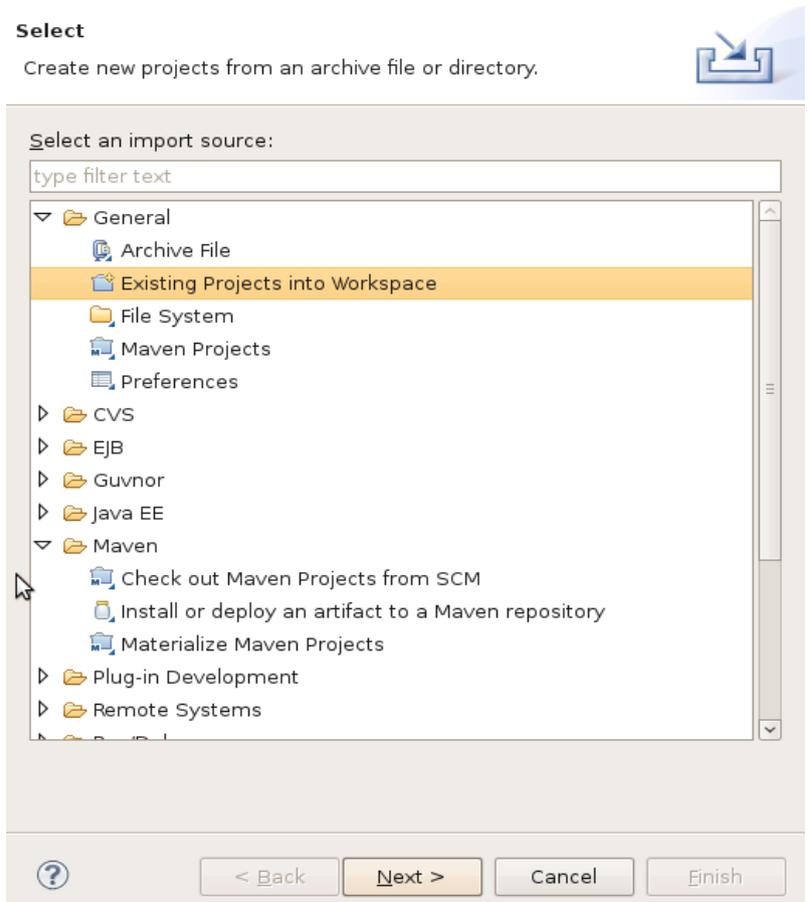
File

- *Import* → *Existing Projects into Workspace*
- *Next*

Selezionare

Select Archive File

Selezionare l'archivio appena scaricato





Hands on...

Parte prima – Impacchettare applicazione Web

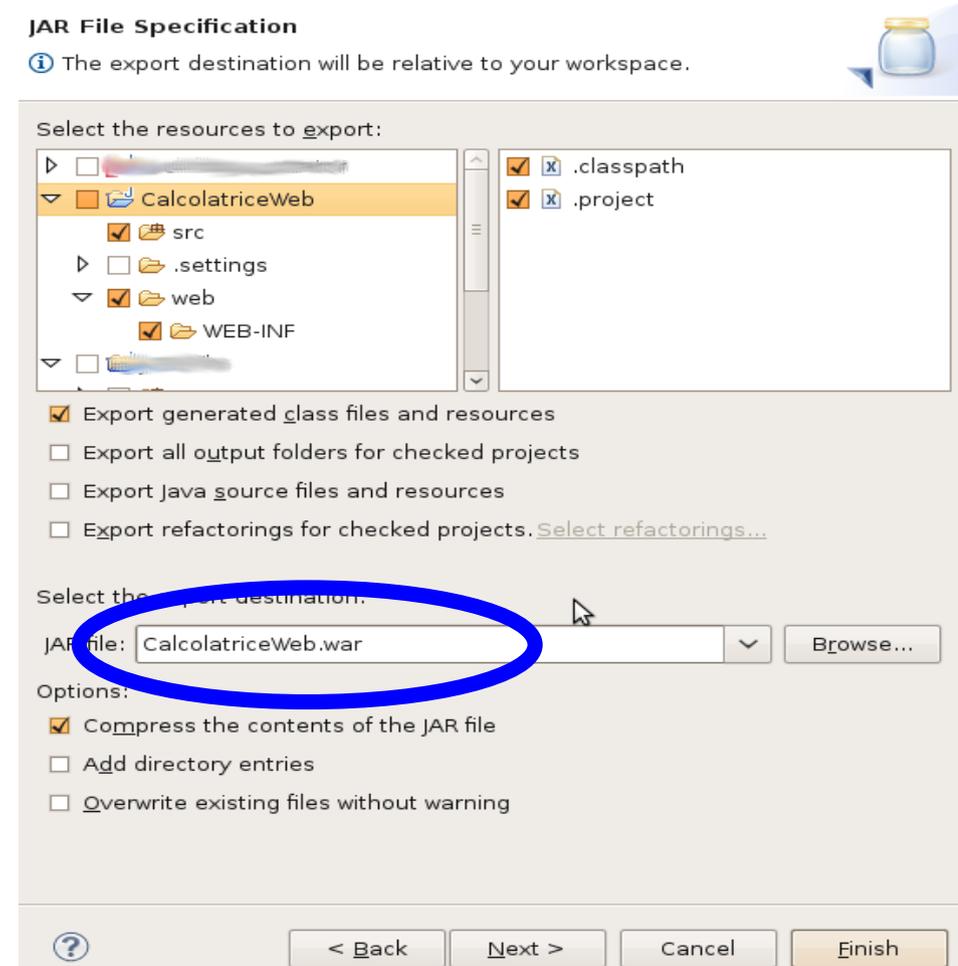
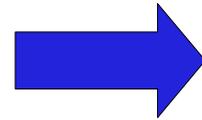
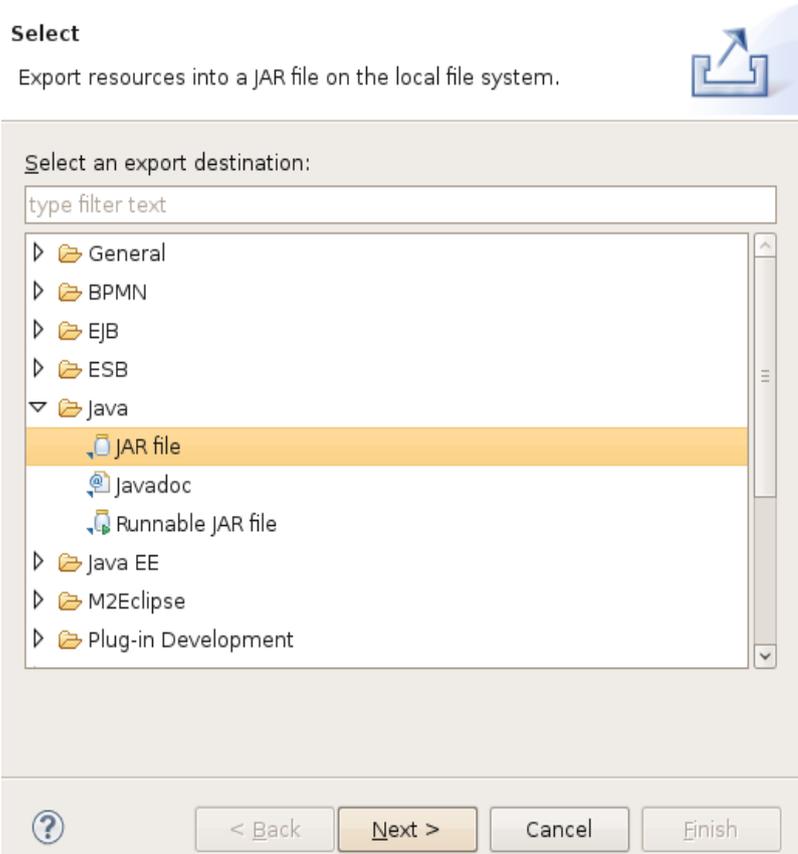
File

→ Export

→ Java → JAR File

→ Next

- Selezionare un percorso dove salvare il file
- Specificare il nome dell'archivio
 - ATTENZIONE: estensione **.war** !!!



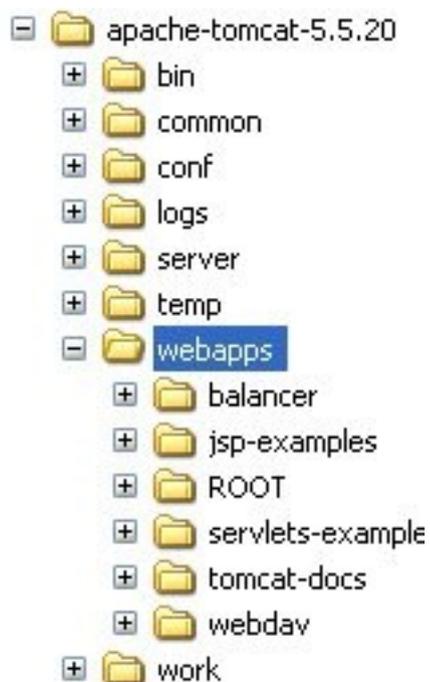


Hands on...

Parte prima – Caricamento applicazione Web

In Tomcat caricamento “a caldo” (server acceso)
della applicazione

- copiare archivio **.war** nella sottocartella **webapps**



Applicazioni attualmente caricate sul server



Hands on...

Parte prima – Pagina Web di esempio

Aprire un browser e digitare la URL

<http://localhost:8080/CalcolatriceWeb/hello.html>

Indirizzo del server

*Nome applicazione Web
(== nome archivio
senza estensione)*

*Nome risorsa
Web*

Esercizi

- ispezionare il contenuto di [hello.html](#) nel progetto
- modificare la pagina a piacere

Riferimenti HTML

- <http://www.w3schools.com/>



Hands on...

Parte seconda – Java Server Pages



Java Server Pages

Documenti HTML con estensione **.jsp**

- includono codice Java
- elaborate “al volo” dal servlet container (Tomcat) a seguito di richieste utente
- generazione dinamica di contenuto HTML

```
<html>
```

```
  <body>
```

```
    <h1> Titolo pagina </h1>
```

```
    <% ... codice Java ben formato ... %>
```

```
  </body>
```

```
</html>
```



Oggetti built-in

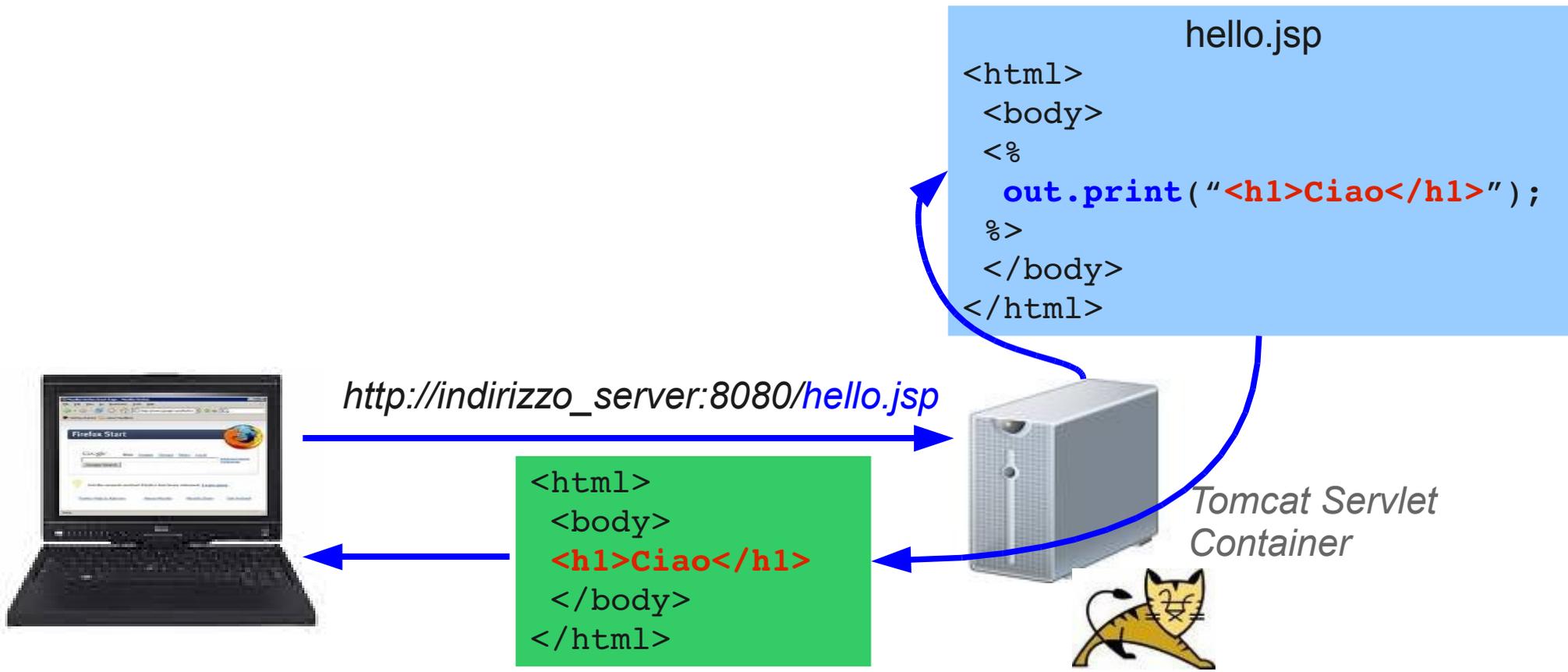
Risorse *automaticamente disponibili* all'interno del codice Java nella JSP e *accessibili per nome*

8 oggetti estremamente utili

- *page*: la pagina e le sue proprietà
- *config*: dati di configurazione
- ***out***: per scrivere codice HTML nella risposta (analogo a ***System.out*** !!)
- ***request***: richiesta HTTP ricevuta e i suoi attributi, header, cookie, ***parameteri***, ecc...)
- *response*: risposta HTTP e le sue proprietà
- *application*: dati condivisi da tutte le pagine della web application
- *session*: dati specifici della sessione utente corrente
- *exception*: eventuali eccezioni lanciate dal server; utile per pagine di errore
- *pageContext*: dati di contesto per l'esecuzione della pagina



Oggetti built-in out





Oggetti built-in

request

```
<input type="text"
  name="firstname"/>
<input type="submit"
  name="azione" value="invia"/>
```

Nome:

```
hello.jsp
<html>
  <body>
    <%
      String name =
        request.getParameter("firstname");
      out.print("Ciao "+name)
    %>
  </body>
</html>
```



http://indirizzo_server:8080/hello.jsp

```
<html>
  <body>
    Ciao Stefano
  </body>
</html>
```



Tomcat Servlet Container





Oggetti built-in

Un esempio completo

hello.jsp

```
<html>
  <body>
    <form>
      <input type="text" name="firstname" />
      <input type="submit" name="azione"
        value="invia" />
    </form>
    <%
      String name =
        request.getParameter("firstname");
      out.print("Ciao: "+name)
    %>
  </body>
</html>
```



http://indirizzo_server:8080/hello.jsp



Tomcat Servlet
Container



Nome:

Ciao:

```
...
<body>
...
</body>
```



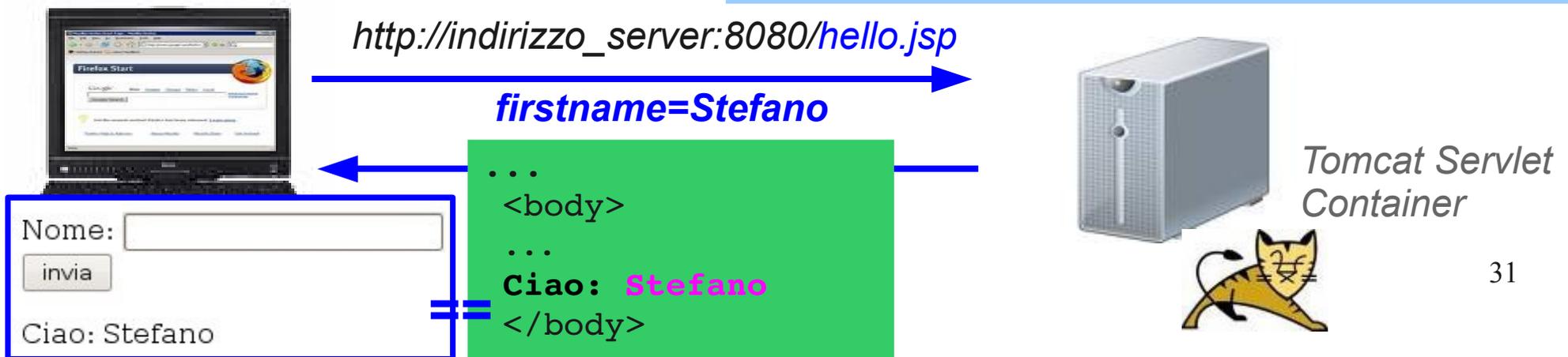
Oggetti built-in

Un esempio completo

Nome:

Ciao:

```
hello.jsp
<html>
  <body>
    <form>
      <input type="text" name="firstname" />
      <input type="submit" name="azione"
        value="invia" />
    </form>
    <%
      String name =
        request.getParameter("firstname");
      out.print("Ciao: "+name)
    %>
  </body>
</html>
```





Hands on...

Parte seconda - JSP

Realizzare una *calcolatrice web*

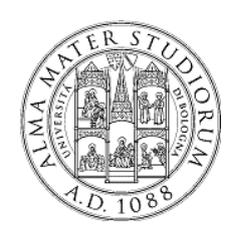
- Vista utente: “schermata” con
 - 3 campi di **input testo**: 2 operandi + operazione da eseguire
 - **pulsante** per inviare la richiesta di calcolo
 - output: risultato dell'operazione

Primo operando (num. intero):

Operazione (+,-,*,/) :

Secondo operando (num. intero):

Risultato :



Hands on...

Parte seconda - JSP

- Lato server: la parte Java deve
 - recuperare i parametri di input della richiesta
 - `request.getParameter(...)`
 - effettuare l'operazione aritmetica mediante la usuale sintassi Java
 - generare una pagina HTML contenente il risultato
 - `out.print(...)`