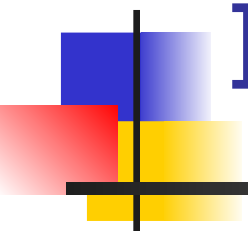


Fondamenti di JSP: Introduzione

Gianluca Moro

gmoro@deis.unibo.it

Dipartimento di Elettronica, Informatica e Sistemistica
Università di Bologna

Sistemi reali in JSP

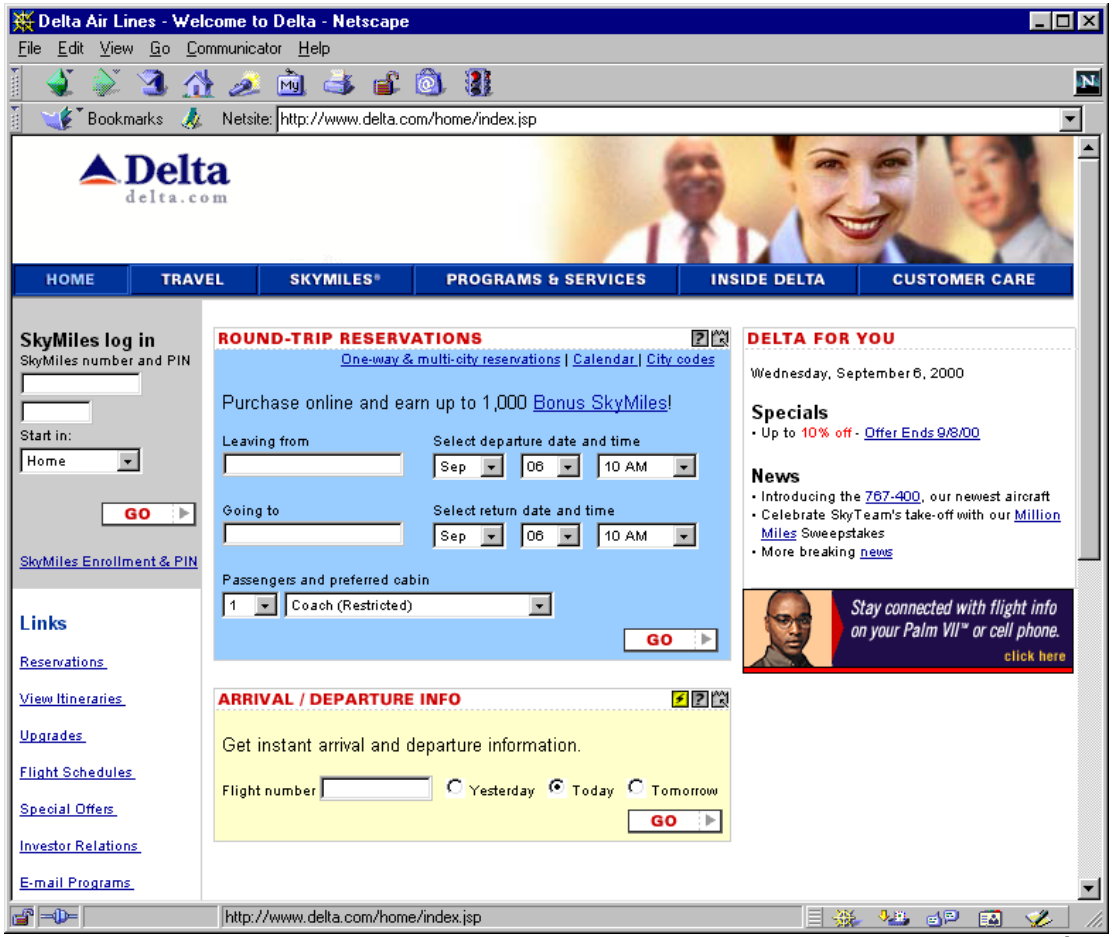
- ofoto.com: stampa e gestisce foto digitali e convenzionali

Sistemi reali in JSP

- Una delle più grandi banche nel mondo per l'emissione di carte di credito e servizi on-line

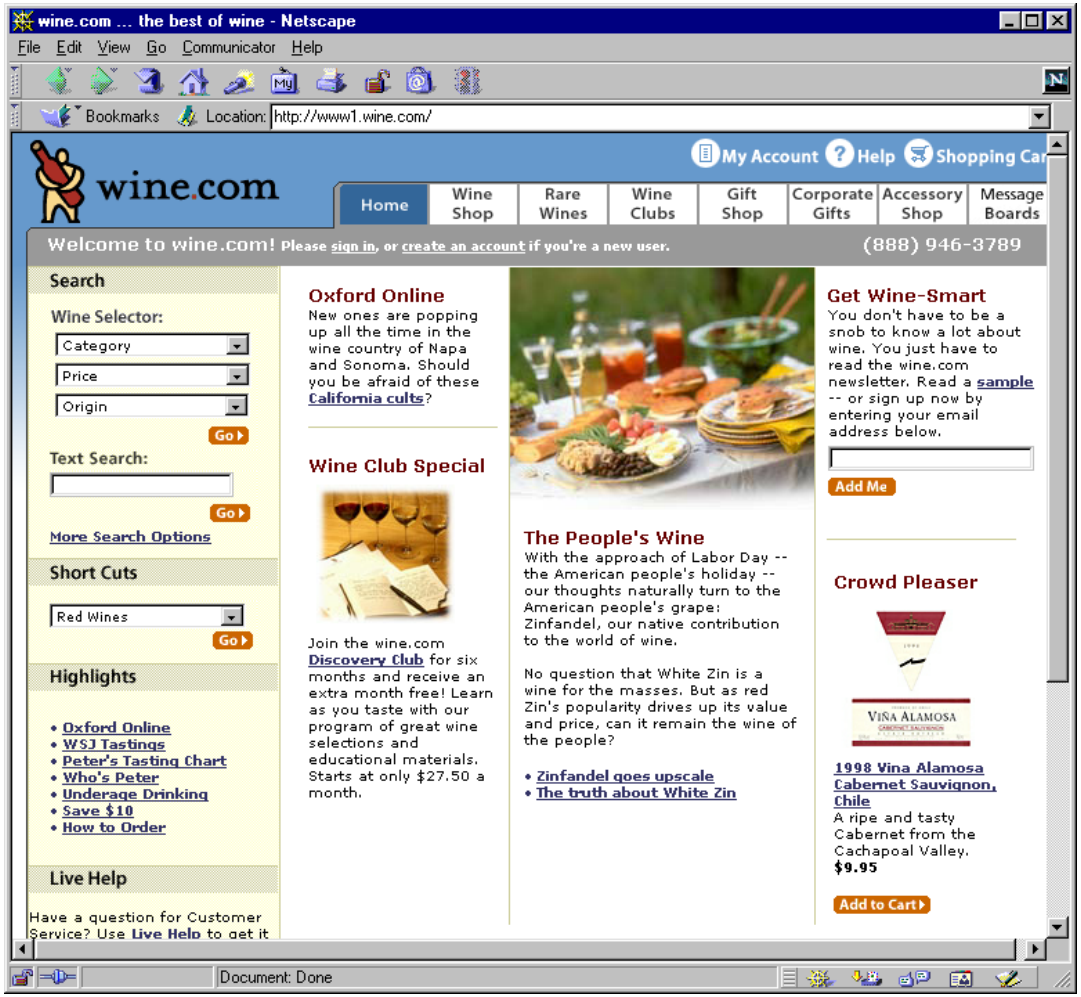
Sistemi reali in JSP

- Delta Airlines: intero sito incluse informazioni real-time sugli orari



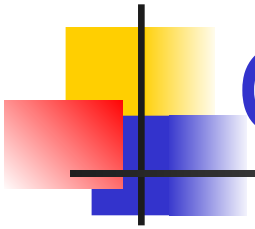
Sistemi reali in JSP

- wine.com: leader nella vendita del vino al dettaglio sulla rete internet



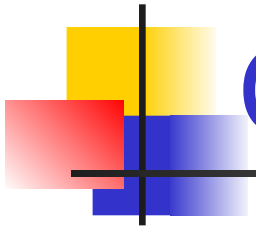
Sistemi reali in JSP

- American Century Investments: più di 70 fondi comuni, 90USD miliardi in gestione, 2 milioni di investitori



Concetti di base

- Una pagina JSP è un'unità di elaborazione server-side ed è strutturata in TAG
- portabile su piattaforme eterogenee senza nemmeno la ricompilazione
- L'esecuzione del codice nella pagina è a carico del Web server o di suoi componenti
  - Tomcat: web server freeware (componente di Apache)
- Il risultato dell'elaborazione può essere:
  - una pagina HTML, XML, WML etc (restituita al browser) e/o operazioni compiute su componenti server-side: oggetti, database ...



Contenuti

- Dichiarazioni;
- Espressioni;
- Scriptlet;
- Direttive (alcune);

Esempio di pagina JSP

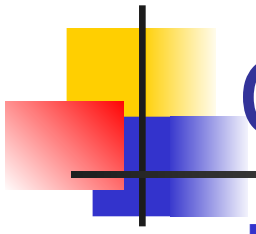
<%-- questo è un commento: segue una direttiva di pagina --%>
<%@ page import="java.util.*" info="Login" errorPage="error.jsp" **%>**

<%-- questa è una direttiva di inclusione di file --%>
<%@ include file="banner.html" **%>**

<%-- dichiarazione che può contenere variabili e metodi --%>
<%! int i = 0; String s="hello world";**%>**

<%-- segue un'espressione java di cui viene visualizzato il risultato --%>
<%= 3+2+i+" "+s **%>**

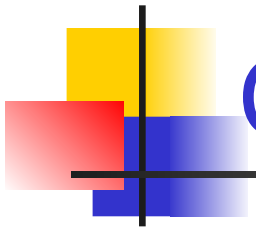
<%-- questo è uno Scriptlet con codice Java e HTML --%>
<% for (i=1; i<3; i++) { **%>**
<H**<%=**i**%>**><%=s%></H**<%=**i**%>**>
<% } **%>**



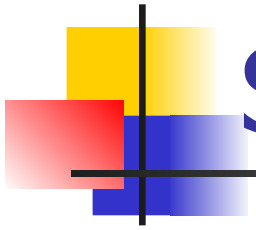
Ciclo di vita di una pagina JSP (i)

- La pagina viene salvata in una cartella pubblica del server web
- alla prima richiesta ricevuta dal Web server la pagina JSP è automaticamente:
  - tradotta in un sorgente Java chiamato Servlet
  - compilata come programma Java
  - caricata in memoria ed eseguita
- nelle chiamate successive la pagina JSP (i.e. la servlet corrispondente) viene solo eseguita

Ciclo di vita di una pagina JSP (ii)



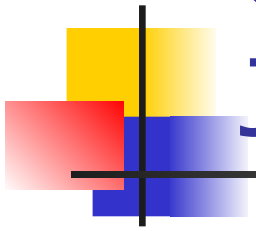
- ad ogni invocazione Il server web verifica se la pagina JSP è più recente della corrispondente Servlet
- se lo è, perché ad esempio la pagina JSP è stata modificata, allora la pagina viene di nuovo tradotta, compilata, caricata e eseguita
- feature molto comoda in fase di sviluppo ma da disattivare al rilascio dell'applicazione perché è costosa



Servlet … no grazie!

- sviluppare Servlet richiede maggiore conoscenza di Java rispetto a quanto ne richiede JSP
- con JSP è possibile ottenere risultati analoghi
- inoltre JSP consente la separazione tra codice Java e di presentazione (es: HTML)
- essendo il corso introduttivo non lavoreremo direttamente con le Servlet

Configurazione per lavorare con JSP

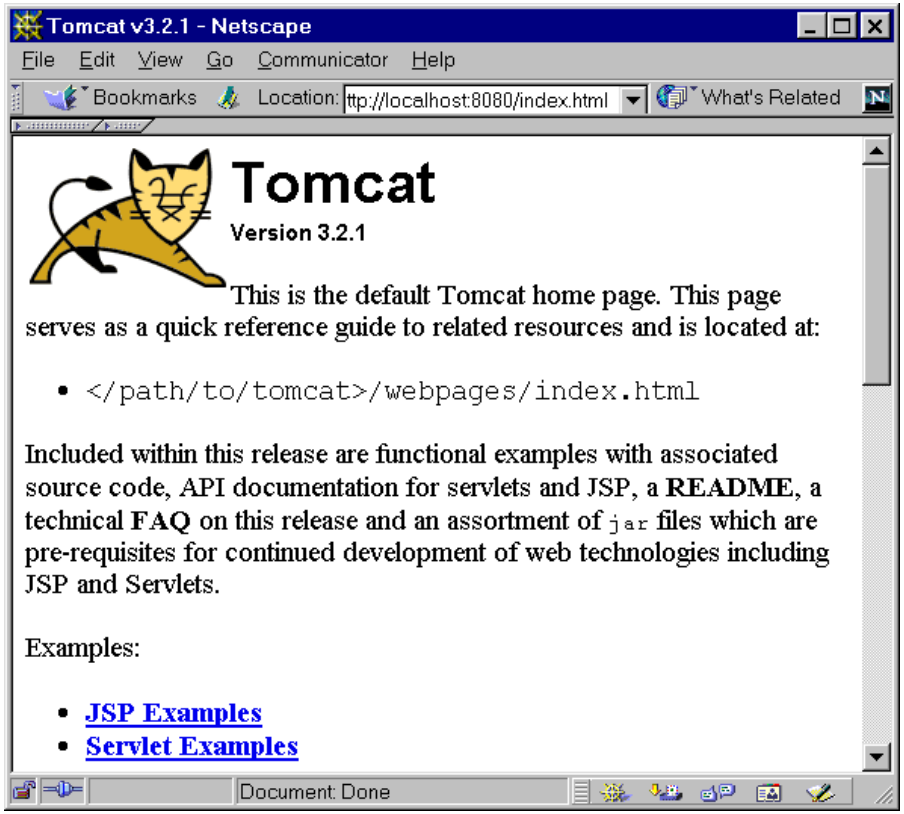


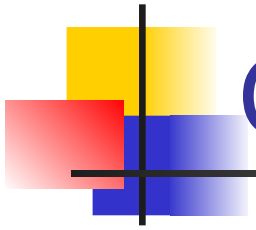
- Necessario un web server JSP compliant:
  - Tomcat 3.2 http://jakarta.apache.org/
  - scaricare il file compresso con i file compilati
  - l'installazione consiste nel decomprimere il file compresso in una directory (es: c:\tomcat)
  - necessario installare il Java Development Kit
  - segnalare a Tomcat la directory del JDK e la sua
    - variabile JAVA_HOME nel file ...\bin\tomcat.bat
    - variabile TOMCAT_HOME nel file ...\bin\startup.bat
  - pagine HTML e JSP inserite in directory contenute in ...\webapps

Avvio e test dell'ambiente Tomcat

- Avviare Tomcat eseguendo …\bin\startup.bat
- accedendo a http://localhost:8080/
  deve comparire

- Per verificare l'installazione JDK eseguire uno degli esempi disponibili
  - link JSP Examples





Contesto dell'applicazione JSP (i)

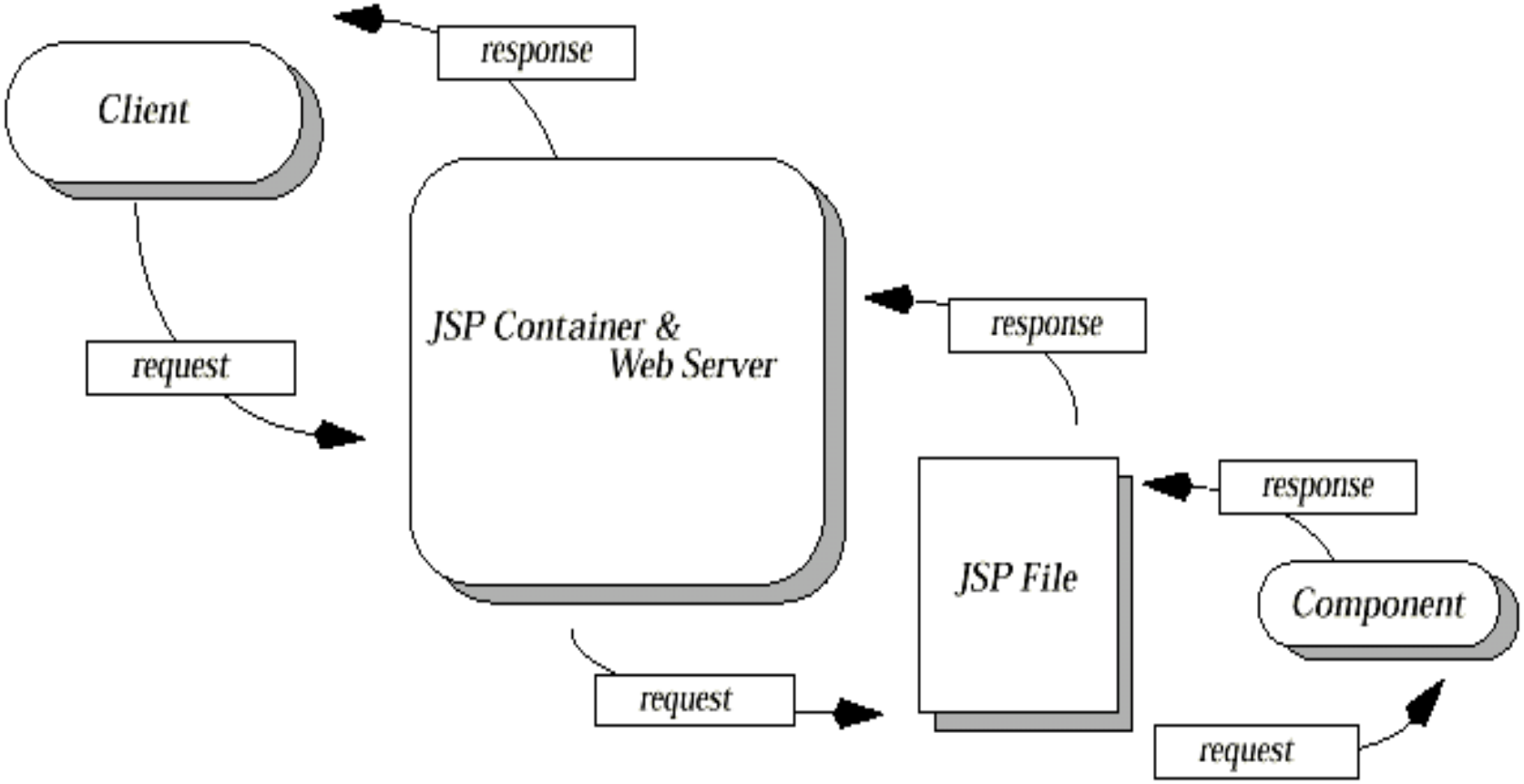
- Ogni applicazione JSP può avere il proprio contesto cosi strutturato:
  - una propria directory contenente i file JSP e HTML
    - esempio: …\webapps\myApp
  - una directory WEB-INF con tutte le risorse della applicazione
    - …\webapps\myApp\WEB-INF\classes
      classi Java che costituiscono l'applicazione
    - …\webapps\myApp\WEB-INF\lib
      classi Java di terze parti usate dall'applicazione, esempio driver JDBC, framework (EJB …)

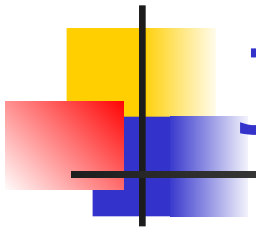
Contesto dell'applicazione JSP (ii)

- Il contesto si definisce nel file …\conf\server.xml aggiungendo quanto segue:

`<Context path="/myApp"`	**path**. è il prefisso nell'URL che indica a Tomcat quale applicazione usare per elaborare la richiesta. E' obbligatorio
`docBase="webapps/myApp"`	**docBase**. La *document root* directory dell'applicazione. Può essere un path relativo (rispetto alla dir di Tomcat) o assoluto. E' obbligatorio
`defaultSessionTimeOut="30"`	E' il tempo di inattività (timeout) massimo in minuti oltre il quale la sessione utente scade perdendo tutti gli oggetti in essa contenuti. E' possibile cambiare il timeout da una pagina JSP con HttpSession.setMaxInactiveInterval(int interval)
`debug="1"`	**debug**. Definisce il livello di verbosità/dettaglio del debugging (da "0" a "9"). Il default è il minimo (i.e. "0")
`reloadable="true"`	**reloadable**. Se vale "true" Tomcat scarica e ricarica l'applicazione automaticamente se rileva delle variazioni nei file in WEB-INF/classes, o file JAR in WEB-INF/lib. Molto utile in fase di sviluppo ma da disattivare al rilascio dell'appl. perché è costosa.
`trusted="false" >`	**trusted**. "true" quando l'applicazione richiede accesso alle classe interne di Tomcat 3.2. Normally, l'accesso è consentito alle sole applicazioni di amministrazione fornite con Tomcat.
`</Context>`	

Schema di funzionamento





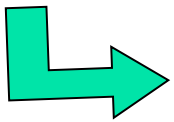
Java come linguaggio per JSP

- Useremo Java come linguaggio per le pagine JSP
- in Java è necessario dichiarare ogni variabile utilizzata
- A differenza di altri linguaggi di scripting lato server, quali VBScript, le variabili devono essere tipizzate
- La sintassi, le regole di visibilità ed i tempi di vita delle variabili sono conformi a Java

Dichiarazioni di variabili condivise

- <%! … %>
- TAG per dichiarare variabili condivise; posto (di norma) all'inizio della pagina
- esempio: dichiarazione di una variabile intera i e una stringa s
  - <%! int j=0; String s = "fattoriale di "; %>
- le variabili così dichiarate sono visibili all'interno della pagina e sono condivise tra tutti gli utilizzatori della pagina

Variabili condivise:precisazione

- Una pagina JSP è una Servlet ossia una classe Java con metodi e attributi
- le variabili dichiarate con <%! ... %> diventano usuali variabili istanza della classe
- Per eseguire una Servlet si istanzia un oggetto dalla classe
- Ogni richiesta (browser request) viene soddisfatta avviando un thread sull'oggetto
- i thread di un oggetto condividono le variabili istanza (ma non quelle locali all'interno dei metodi)
-  variabili istanza = variabili condivise

Dichiarazioni di funzioni

- Con gli stessi TAG <%! … %> si possono dichiarare delle funzioni (i.e. metodi) utilizzabili nella pagina
- Esempio:

```
<%!
int fatt(int n) {
if (n == 0) return 1;
else return n*fatt(n-1);
}
%>
```

- Il metodo così dichiarato diventa un metodo della Servlet

Espressioni

- <%= … %>
- all'interno di questo TAG si inserisce una espressione Java
- esempio:

  <%=s+j+"="+fatt(j++)%>

  notare la terminazione
  senza punto e virgola
- valutazione:
  - l'espressione viene valutata e il risultato viene convertito in stringa
  - la stringa risultante viene posta nella pagina di output inviata all'utente

Scriptlet

- I separatori <% … %> permettono di ospitare codice con sintassi e semantica Java
- esempio:

```
<% j++;
   for (int i=0; i<j; i++) { %>
        <H<%=i+1%>>
            <%=s+i+"="+fatt(i)%>
        </H<%=i+1%>>
<% } %>
```

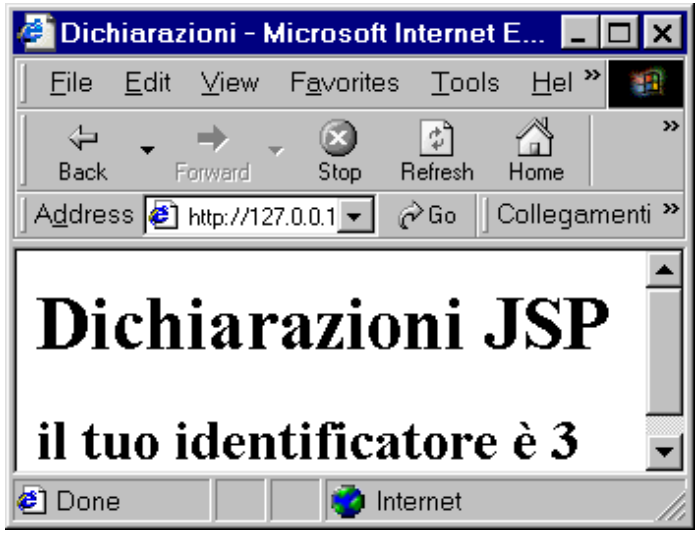
- espressioni e scriptlet vengono inseriti in un metodo della Servlet chiamato _jspService(...)

Risultato

- j aumenta di 1 ad ogni richiesta poiché è una variabile istanza (condivisa)
- dopo 6 richieste si ottiene questo risultato

Attenzione alla concorrenza

```
<HTML><HEAD><TITLE>Dichiarazioni</TITLE>
</HEAD>
<BODY>
<H1>Dichiarazioni JSP</H1>
<%! int identificatore = 1; %>
<H2>il tuo identificatore è
<%= identificatore %>
<% ++identificatore; %>
</H2>
</BODY>
</HTML>
```



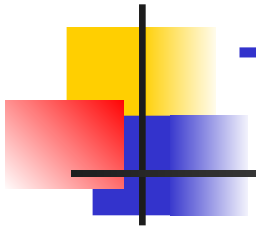
Sincronizzazione

- La modifica di una risorsa condivisa in situazioni di concorrenza può originare anomalie
- Una soluzione è sequenzializzare le sezioni critiche:

```
<% synchronized(this) { %>
  <H2>il tuo identificatore è
  <%= identificatore %>
      <% ++identificatore;
} %>
```

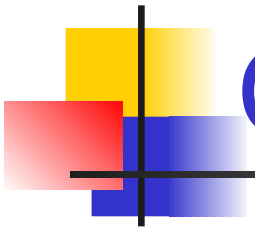
- questo caso (banale) si risolve anche senza:

```
    <H2>il tuo identificatore è
  <%= ++identificatore %>
```



Tipi di direttive

1. Le direttive forniscono indicazioni per l'ambiente di esecuzione e riguardano:
   1. Commenti
   2. Inclusione di file
   3. Reindirizzamento delle richieste
   4. Pagina
   - Altre direttive, quelle deputate all'utilizzo dei bean, verranno introdotte in seguito



Commenti

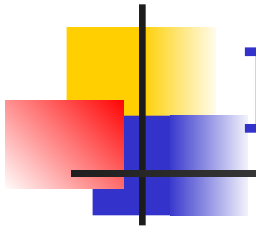
- E’ possibile inserire commenti in una pagina JSP utilizzando la seguente direttiva:

  <%-- commento --%>

- Si noti che questo commento non viene inviato al browser
- Per inserire commenti visibili a livello di codice HTML inviato al cliente, utilizzare i commenti HTML:

  <!-- commento -->

Inclusione a compile-time



- E’ possibile includere a compile-time un file JSP o qualsiasi altro file ad una pagina JSP:

  <%@ include file=“URL del file” %>
- corrisponde a quanto ottenibile con un qualsiasi editor facendo il paste di un file in un altro
- permette di includere porzioni di pagine comuni a più pagine localizzando così le modifiche ad un solo file

Inclusione a request-time

- E' possibile includere un file esterno ogni volta che la pagina JSP viene richiesta:
  - <jsp:include page="URL della pagina" flush="true"/>
- diversamente dalla precedente inclusione, questa permette di includere sempre l'ultima versione del file
- si possono passare parametri alla pagina inclusa aggiungendo righe del tipo:
  - <jsp:param name="name" value="value"/>
  - ...
  - </jsp:include>

Reindirizzamento del client

- E' possibile indicare al browser di richiedere/richiamare un'altra pagina:
  - <jsp:forward page="pagina destinazione"/>
- La pagina contenente la direttiva non può spedire nulla al browser chiamante prima della direttiva stessa
- è possibile passare parametri alla pagina richiamata:
  - <jsp:forward page="pagina destinazione"/>
  - <jsp:param name="name" value="value"/> …
  - </jsp:forward>
- la pagina richiamata riceve i parametri tramite l'oggetto request.getParameter("name")

Attributi di pagina

- Tramite questa direttiva, è possibile definire una serie di attributi relativi all'intera pagina (di supporto all'interprete della pagina):

  <%@ page language = "linguaggio di scripting"
  buffer="dimensione buffer da usare prima di spedire l'output"
  autoFlush="se buffer vale on allora questo vale true"
  info="informazioni descrittive sulla pagina"
  errorPage="URL pagina di gestione degli errori di questa pagina"
  contentType="tipo MIME del contenuto"
  isErrorPage="se vale true allora la pagina gestisce errori" %>

- MIME: Multipurpose Internet Mail Extensions, and refers to an official Internet standard that specifies how messages must be formatted so that they can be sent