

JavaServer Pages: Introduzione

Gianluca Moro

gmoro@deis.unibo.it

Dipartimento di Elettronica, Informatica e Sistemistica
Università di Bologna

JavaServer Pages: Introduzione
G. Moro - DEIS Università di
Bologna - a.a. 2002/03

Sistemi reali in JSP

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



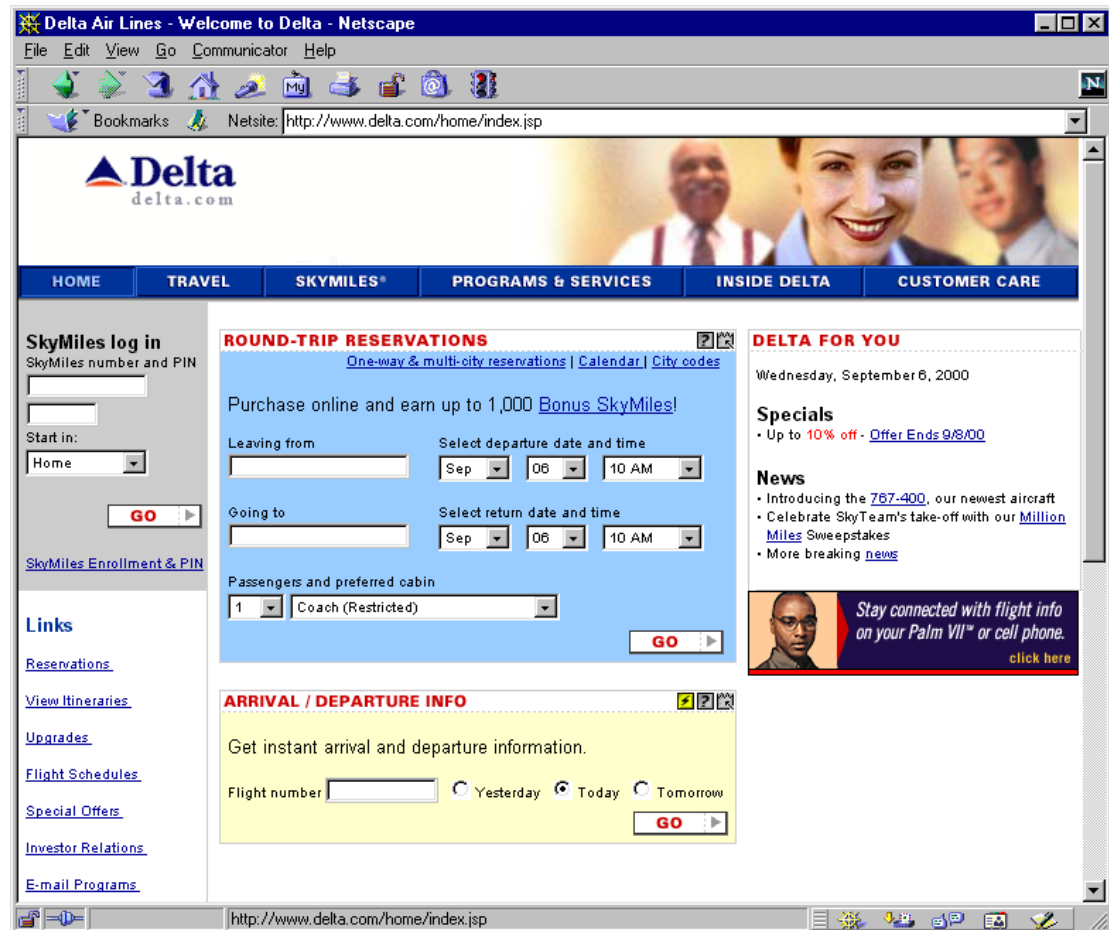
Sistemi reali in JSP

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



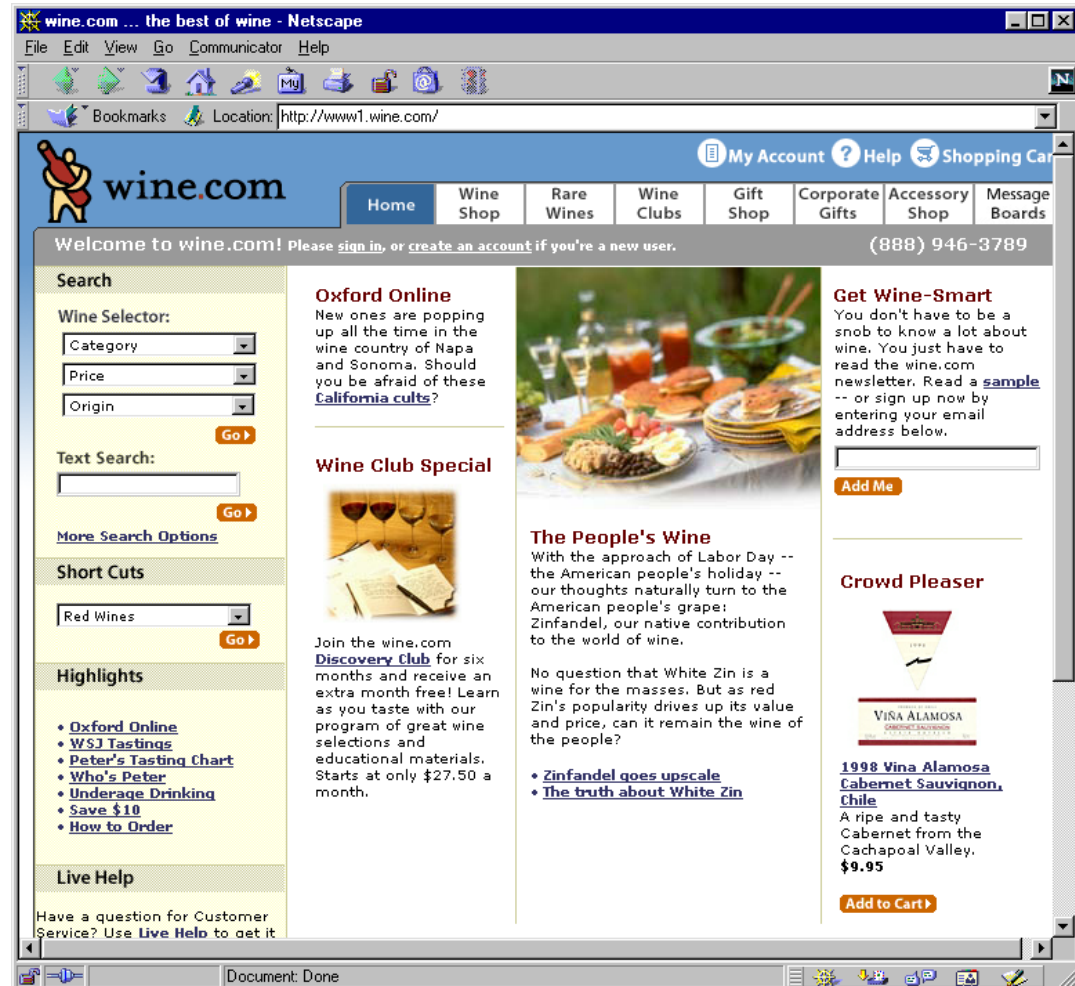
Sistemi reali in JSP

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



Sistemi reali in JSP

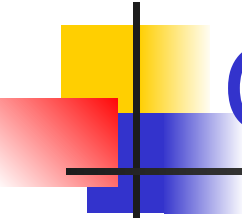
- wine.com: leader nella vendita del vino al dettaglio sulla rete internet



Sistemi reali in JSP

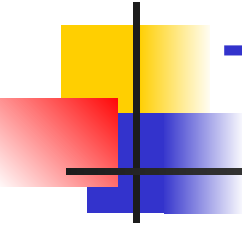
- American Century Investments: più di 70 fondi comuni, 90USD miliardi in gestione, 2 milioni di investitori





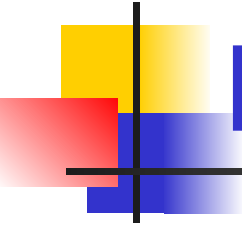
Concetti di base

- 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 ...



Tipi di TAG JSP

- Dichiarazioni;
- Espressioni;
- Scriptlet;
- Direttive;



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 --%>

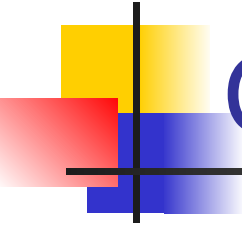
<%= 2002+i+" "+s %>

<%-- questo è uno Scriptlet con codice Java e HTML --%>

<% for (i=1; i<3; i++) { %>

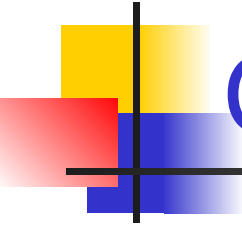
 <H<%=i%>><%=s%></H<%=i%>>

<% } %>



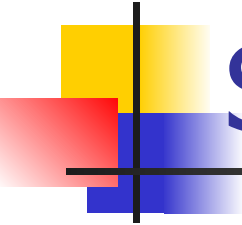
Ciclo di vita di una pagina JSP (i)

- 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**



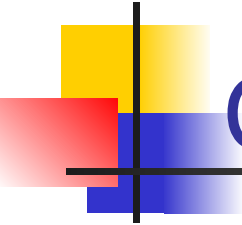
Ciclo di vita di una pagina JSP (ii)

- 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 ...

- sviluppare Servlet richiede maggiore conoscenza di Java rispetto a quanto ne richiede JSP
- con JSP è possibile ottenere risultati analoghi
- JSP facilita la separazione tra codice Java e di presentazione (es: HTML)
- questa unità didattica introduttiva non tratta le Servlet



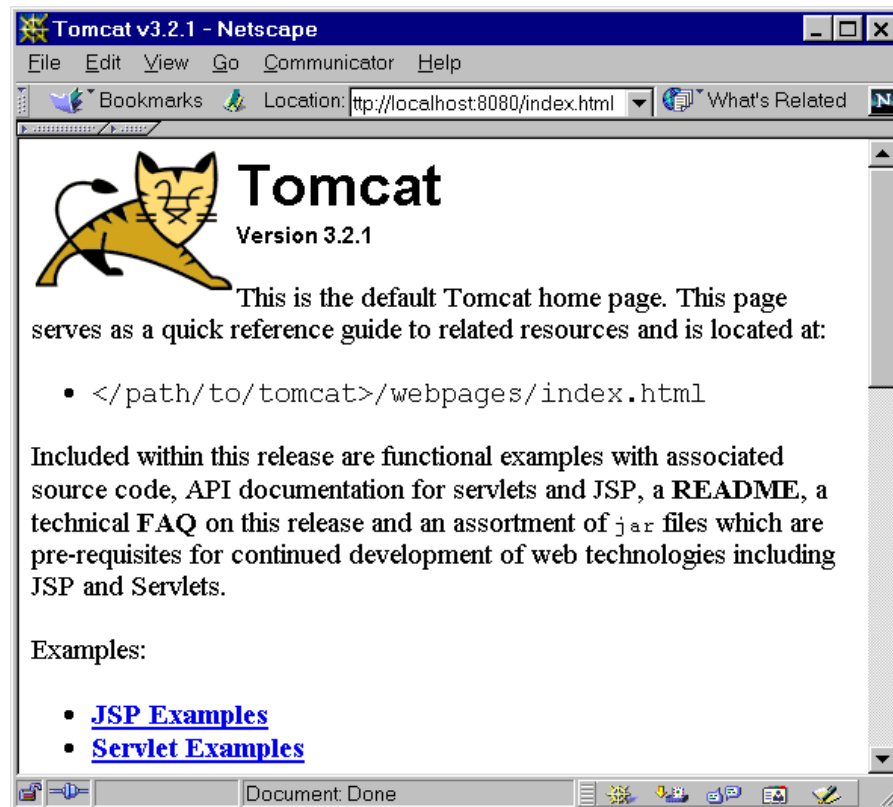
Configurazione per lavorare con JSP

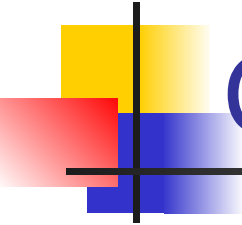
- Necessario un web server JSP compliant:
 - es: Tomcat 3.2 <http://jakarta.apache.org/>
 - scaricare il file compresso con i file compilati
 - l'installazione consiste nel decomprimere il file compresso in una directory (es: c:\tomcat)
 - necessario installare il Java Development Kit
 - definire in Tomcat la propria HOME e quella del JDK
 - 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 questa finestra

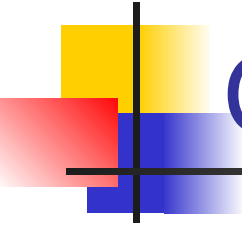
- Per verificare l'installazione JDK eseguire uno degli esempi disponibili
 - [link JSP Examples](#)





Contesto dell'applicazione JSP (i)

- Ogni applicazione JSP può avere il proprio contesto così 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"
docBase="webapps/myApp"
defaultSessionTimeout="30"
debug="1"
reloadable="true"
trusted="false" >

</Context>
```

path. è il prefisso nell'URL che indica a Tomcat quale applicazione usare per elaborare la richiesta. E' obbligatorio

docBase. La *document root* directory dell'applicazione. Può essere un path relativo (rispetto alla dir di Tomcat) o assoluto. E' obbligatorio

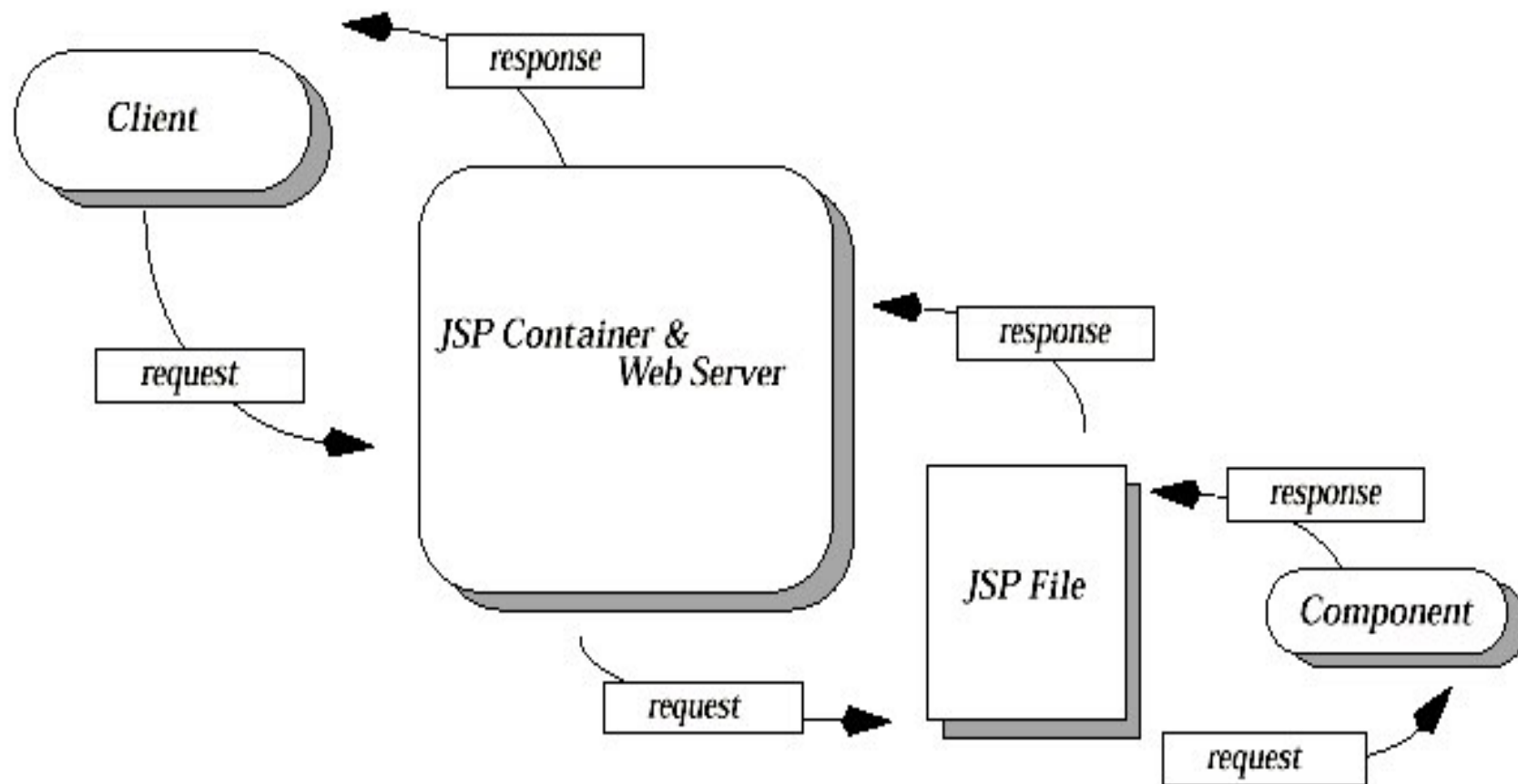
E' il tempo di inattività (timeout) massimo in minuti oltre il quale la sessione utente scade perdendo tutti gli oggetti in essa contenuti. E' possibile cambiare il timeout da una pagina JSP con `HttpSession.setMaxInactiveInterval(int interval)`

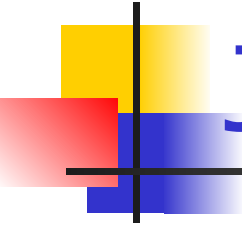
debug. Definisce il livello di verbosità/dettaglio del debugging (da "0" a "9"). Il default è il minimo (i.e. "0")

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. "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.

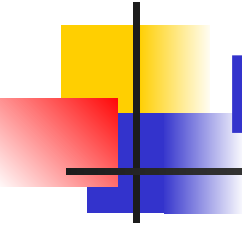
Schema di funzionamento





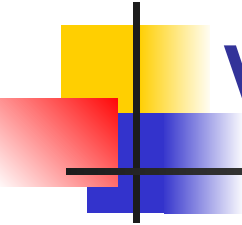
Java come linguaggio per JSP

- In linea di principio la tecnologia JSP è pensata per permettere lo sviluppo con qualsiasi linguaggio:
 - `<%@page language="...">`
- Il linguaggio attualmente supportato è Java:
 - 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

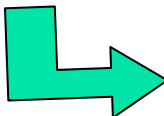


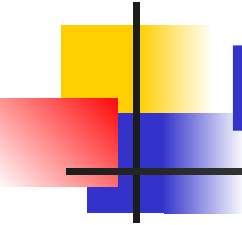
Dichiarazioni di variabili condivise

- `<%! ... %>`
- 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 il web server **istanzia un oggetto** dalla classe
- **ogni richiesta** (browser request) viene soddisfatta avviando un **thread sull'oggetto istanziato**
- **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 $\langle \% = \dots \% \rangle$

- questo TAG ospita un'espressione Java

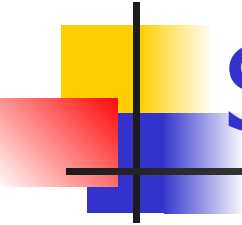
- esempio:

$\langle \% = s + j + " = " + fatt(j++) \% \rangle$

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
- l'esempio calcola alla n-esima richiesta il fattoriale di n-1





Scriptlet

- Il TAG `<% ... %>` permette di ospitare codice con sintassi e semantica Java
- esempio:
 - alla richiesta j-esima calcola tutti i fattoriali da 0 a j-1

```
<% 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
```

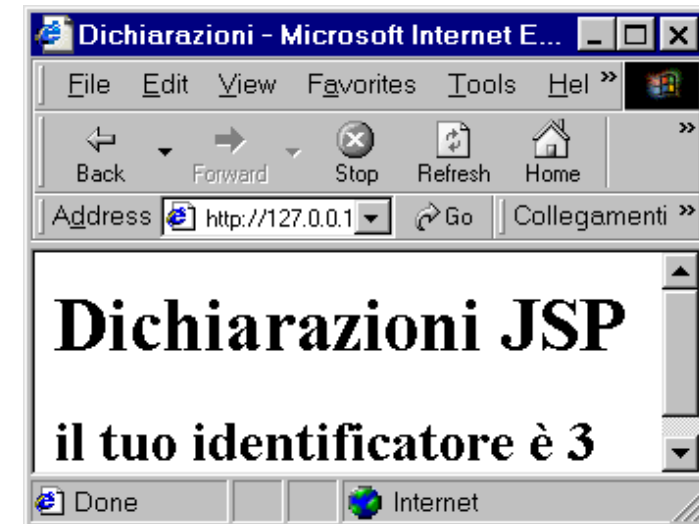
```
<%= identificatore %>
```

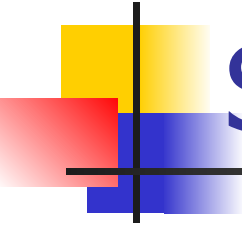
```
<% ++identificatore; %>
```

```
</H2>
```

```
</BODY>
```

```
</HTML>
```



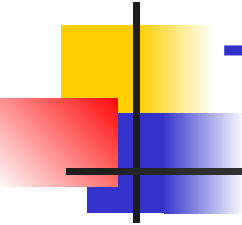


Sincronizzazione

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

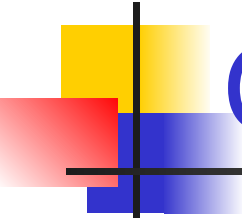
```
<% synchronized(this) { %>  
    <H2>il tuo identificatore è  
    <%= identificatore %>  
    <% ++identificatore;  
} %>
```

- questo caso (banale) si risolve anche senza:
 <H2>il tuo identificatore è
 <%= ++identificatore %>



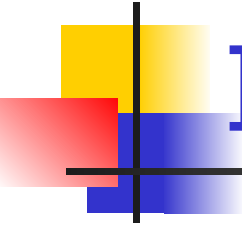
Tipi di direttive

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



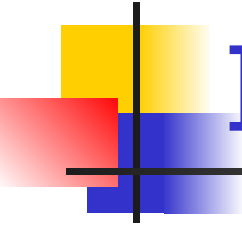
Commenti

- 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 -->`



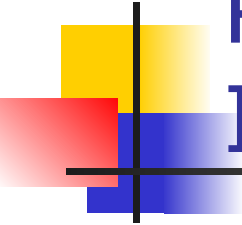
Inclusione a compile-time

- 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>`

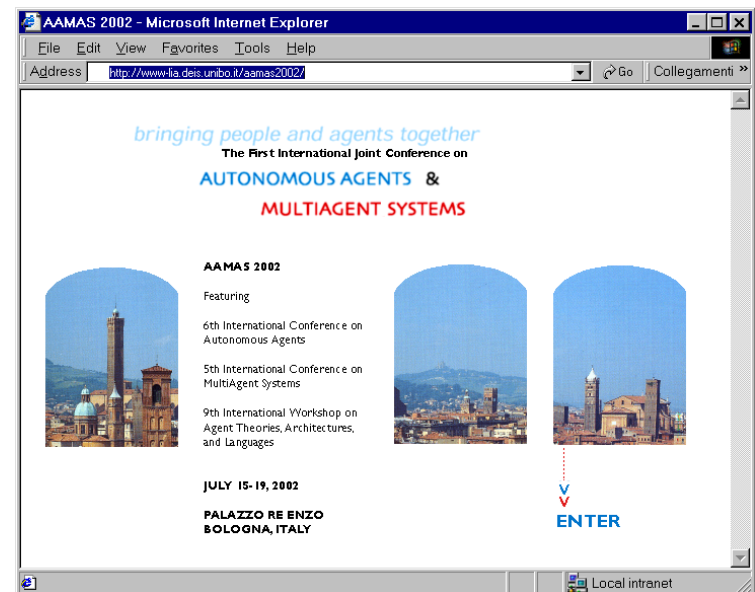


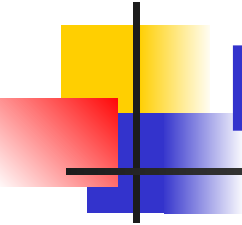
Pagine “Template” attraverso semplici Include parametrici

- ad esempio con un solo file jsp si può realizzare uno schema di pagina HTML
- le sezioni dello schema possono cambiare in base ai param. di inclusione:
 - i contenuti delle pagine di un qualsiasi sito di e-commerce cambiano forse ogni giorno ma lo schema è quasi sempre il medesimo
 - idem per siti di giornali web, siti informativi...
 - la pag. “template” (schema) viene “istanziata” attraverso la sua inclusione in un’altra pagina jsp che stabilisce i contenuti del “template”

Recupero dei parametri nel "template"

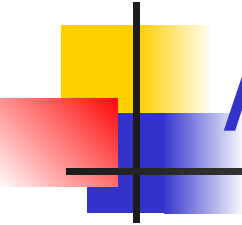
- ad es. in una sezione del template ci può essere un immagine passata come parametro e recuperata così:
- `">`
- si possono anche includere files il cui **nome non è noto a design time**
- `<% String s = request.getParameter("file"); %>`
- `<jsp:include
 page="<%=s %>"
 flush="true">
</jsp:include>`





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 è attivo 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 - standard internet che specifica come i messaggi devono essere formattati per essere spediti