Anno Accademico 2007-2008

Corso di Tecnologie Web

Web Application: JSP

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

Java Server Pages

> Java Server Pages rappresentano uno dei due componenti di base della tecnologia J2EE (relativamente alla parte web)

> Le JSP sono una sorta di *template* per la generazione di contenuto dinamico

> Estendono HTML con codice Java custom. In altri termini, quando viene effettuata una richiesta ad una pagina JSP:

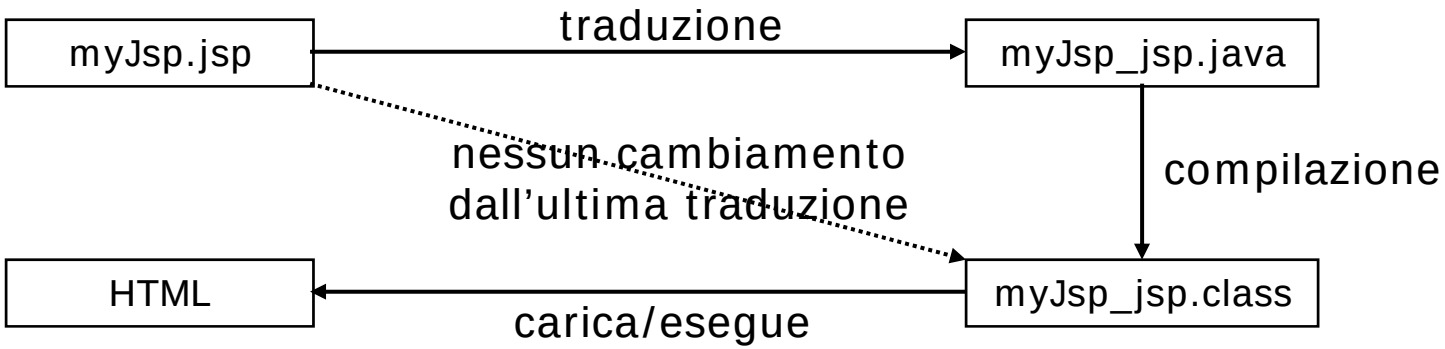
- la parte HTML viene scritta sullo *stream* di *output* senza trasformazione alcuna
- il codice Java viene eseguito sul server per la generazione del contenuto dinamico (si tratta, evidentemente, di contenuto espresso in HTML)
- la pagina HTML così formata (parte statica e parte generata dinamicamente) viene restituita al client

> Le JSP concettualmente sono assimilabili ad un linguggio di scripting. In pratica, tuttavia, esse vengono tradotte e poi compilate in servlet ad opera del web container.

JSP *request processing*

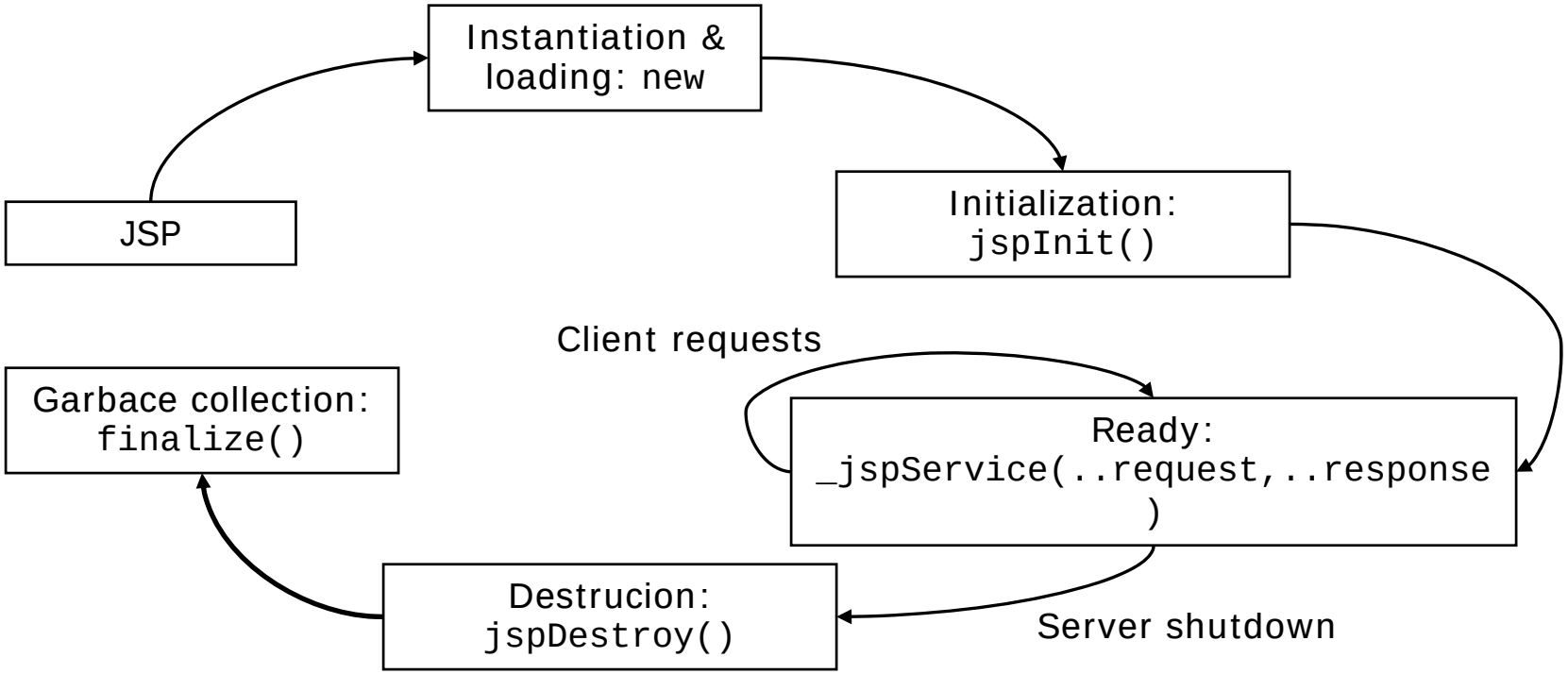
> Le richieste verso JSP sono gestite da una particolare servlet (in Tomcat **`JspServlet.`** cfr. `/conf/web.xml` sotto la directory d'installazione di Tomcat) che effettua le seguenti operazioni:

- traduzione della JSP in una servlet
- compilazione della servlet risultante in una classe
- esecuzione della JSP .



JSP: il ciclo di vita

> Poiché le JSP sono compilate in servlet, il ciclo di vita delle JSP è controllato dal web container



JSP: esempio

> Esempio di invocazione:

- http://myHost/myWebApp/helloWord.jsp -> Hello, Word
- http://myHost/myWebApp/helloWord.jsp?name=me -> Hello, me

```
<html>
  <body>
  <% String visitor=request.getParameter("name");
     if (visitor == null) visitor = "World";
  %>
     Hello, <%= visitor %>!
  </body>
</html>
```

JSP: scripting-oriented tag VS XML-based tag

> Scripting-oriented tag: sono definiti da delimitatori entro cui è presente lo scripting (sono *self-contained*):

- <%! %>
- <%= %>
- <% %>
- <%@ %>

> XML-oriented tag: seguono la sintassi XML. Sono presenti XML tag equivalenti ai delimitatori sopra riportati, rispettivamente:

- <jsp:declaration> *declaration* </jsp:declaration>
- <jsp:expression> *expression* </jsp: expression>
- <jsp:scriptlet> *java_code* </jsp:scriptlet>
- <jsp:directive.*dir_type dir_attribute* />

JSP: dichiarazioni

> Si usano **<%!** e **%>** per la dichiarazione di variabili e metodi

> Variabili e metodi dichiarati possono poi essere referenziati in qualsiasi punto del codice JSP

```
<%! Sring name = “Paolo Rossi”;
    double[] prices = {1.5, 76.8, 21.5};
    double getTotal() {
       double total = 0.0;
       for (int i=0; i<prices.length; i++)
         total += prices[i];
       return total;
    }
%>
```

JSP: espressioni

> Si usano i delimitatori **<%=** e **%>** per valutare Java *expression*

> Il contenuto valido in una qualsivoglia espressione Java viene valutato come stringa

```
<p>Sig. <%=name%>,</p>
<p>l'ammontare del suo acquisto è: <%=getTotal()%>.</p>
<p>La data di oggi è: <%=new Date()%></p>
```

HTML risultante come output

```
<p>Sig. Paolo Rossi,</p>
<p>l'ammontare del suo acquisto è: 99.8.</p>
<p>La data di oggi è: Tue Feb 20 11:23:02 2007</p>
```

JSP: *scriptlet*

> Si usano **<%** e **%>** per aggiungere codice Java eseguibile alla JSP

> Lo scriptlet consente tipicamente di controllare il flusso del programma

> La combinazione di tutti gli scriptlet in una determinata JSP deve definire un blocco logico completo di codice Java

```
<% if (userIsLogged) { %>
        <h1>Benvenuto Sig. <%=name%></h1>
<% } else { %>
        <h1>Per accedere al sito devi loggarti</h1>
<% } %>
```

JSP: direttive

> Le direttive sono comandi JSP valutati a *compile time*

> Le seguenti tre direttive possono cambiare il comportamento delle JSP:

- **`page`**: permette di importare package, dichiarare pagine d'errore, definire il modello di esecuzione della JSP relativamente alla concorrenza, ecc.
- **`include`**: include un altro documento
- **`taglib`**: presiede al caricamento di *custom tag* implementate dallo sviluppatore

> Non producono nessun output visibile

```
<%@ page info="This is a valid set of page directives." %>
<%@ page language="java" import="java.net.*" %>
<%@ page import="java.util.List, java.util.ArrayList" %>

<%@ include file="myHeaderFile.html" %>
```

JSP: *page directive*

> Sintassi: **<%@ page attribute1="value1" attribute2="value2" ..%>**

Attribute	Value	Default	Examples
info	Text string	None	info="Registration form."
language	Scripting language name	"java"	language="java"
contentType	MIME type, character set	See first example	contentType="text/html; charset=ISO-8859-1" contentType="text/xml"
extends	Class name	None	extends="com.taglib.wdjsp.MyJspPage"
import	Class and/or package names	None	import="java.net.URL" import="java.util.*, java.text.*"
session	Boolean flag	"true"	session="true"
buffer	Buffer size, or false	"8kb"	buffer="12kb" buffer="false"
autoFlush	Boolean flag	"true"	autoFlush="false"
isThreadSafe	Boolean flag	"true"	isThreadSafe="true"
errorPage	Local URL	None	errorPage="results/failed.jsp"
isErrorPage	Boolean flag	"false"	isErrorPage="false"

JSP: *include directive*

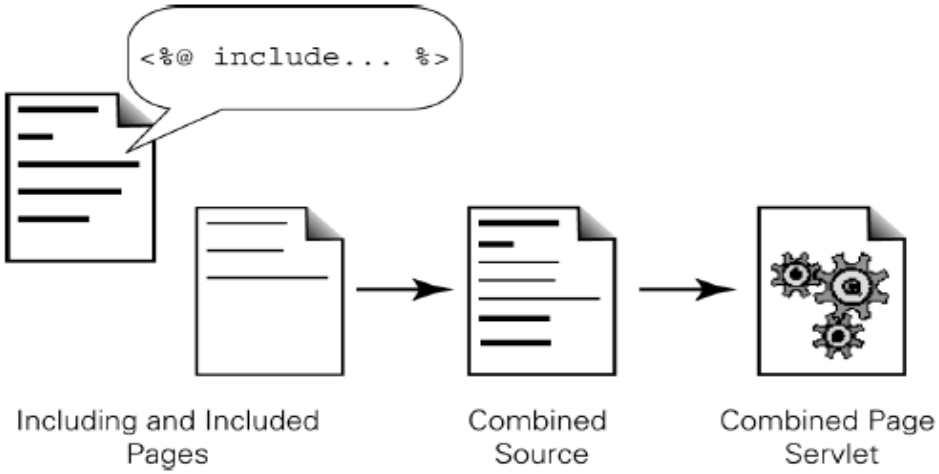
> Sintassi: **<%@ include file = “$localURL”%>**

> Serve ad includere il contenuto del file specificato

> E’ possibile nidificare un numero qualsiasi di inclusioni

> L’inclusione è a *compile time.* Ciò implica che eventuali modifiche al file incluso non determinano una ricompilazione dell’includente (il web container non verifica l’albero delle dipendenze)

> Esempio: **<%@ include file="/shared/copyright.html"%>**



JSP: *taglib directive*

> Sintassi: **<%@ uri="$tagLibraryURI" prefix="tagPrefix"%>**

> L'attributo *uri* fa riferimento alla locazione di un file xml, con estensione tld (acronimo di tag library descriptor), che contiene le informazioni relative alla classe implementante il custom tag. Nell'esempio sotto riportato, nel descriptor associato alla *taglib* sarà presente un elemento **<endProgram>** che definirà la classe Java associata al tag.

> Esempio:

- direttiva: **<%@ taglib uri="/EncomTags" prefix="mcp"%>**
- utilizzo: **<mcp:endProgram/>**

JSP: azioni

> Le azioni sono comandi JSP valutati a *request time.* Sono previsti 6 tipi di azioni definite dai seguenti tag:

- **`useBean`**: istanzia un JavaBean e ne associa un identificativo
- **`getProperty`**: ritorna la property indicata come un oggetto
- **`setProperty`**: imposta il valore della property indicata per nome
- **`include`**: include un file nella JSP
- **`forward`**: cede il controllo ad un'altra JSP o servlet
- **`plugin`**: genera contenuto per scaricare plug-in Java se necessario

> Sono espresse usando sintassi XML

```
<html>
 <body>
  <jsp:useBean id="hello" class="it.unibo.deis.my.HelloBean"/>
  <jsp:setProperty name="hello" property="name" param="name"/>
  Hello, <jsp:getProperty name="hello" property="name"/>!
 </body>
</html>
```

JSP: *built-in objects*

> Le specifiche JSP definiscono 8 oggetti *built-in*, presenti cioè negli sciptlet implicitamente (senza necessità di dichiarazioni ed istanziazioni).

> Questi oggetti rappresentano utili riferimenti agli omologhi oggetti presenti nelle servlet.

> La tabella sottostante elenga gli 8 *built-in objects*

Object	Class or *Interface*	Description
page	*javax.servlet.jsp.HttpJspPage*	Page's servlet instance.
config	*javax.servlet.ServletConfig*	Servlet configuration data.
request	*javax.servlet.http.HttpServletRequest*	Request data, including parameters.
response	*javax.servlet.http.HttpServletResponse*	Response data.
out	javax.servlet.jsp.JspWriter	Output stream for page content.
session	*javax.servlet.http.HttpSession*	User-specific session data.
application	*javax.servlet.ServletContext*	Data shared by all application pages.
pageContext	javax.servlet.jsp.PageContext	Context data for page execution.
exception	java.lang.Throwable	Uncaught error or exception.

JSP: *page object*

> Oggetto legato alla servlet relativa alla pagina JSP

> L'oggetto *page* rappresenta l'istanza della servlet

> Esempio:

```
<%@ page info="Page implicit object demonstration." %>
Page info:
<%=((javax.servlet.jsp.HttpJspPage)page).getServletInfo() %>
```

JSP: *config object*

> Oggetto legato alla servlet relativa alla pagina JSP

> L'oggetto contiene la configurazione della servlet (parametri di inizializzazione)

> Poco usato in pratica (poco usati i parametri di inizializzazione)

> Metodi associati:

Method	Description
`getInitParameterNames()`	Retrieves the names of all initialization parameters.
`getInitParameter(name)`	Retrieves the value of the named initialization parameter.

JSP: *request object*

> Oggetto legato all'I/O della pagina JSP

> L'oggetto *request* rappresenta la richiesta alla pagina JSP

> Consente l'accesso a tutte le informazioni della request:

- l'indirizzo di provenienza
- l'URL richiesto
- tutti gli headers
- i cookie
- i parametri associati alla richiesta

```
<% String xStr = request.getParameter("num");
try {
   long x = Long.parseLong(xStr); %>
   Factorial result: <%= x %>! = <%= fact(x) %>
<%} catch (NumberFormatException e) { %>
Sorry, the <b>num</b> param does not specify an integer value.
<%} %>
```

JSP: *request object*

Method	Description
`getParameterNames()`	Returns the names of all request parameters
`getParameter(name)`	Returns the first (or primary) value of a single request parameter
`getParameterValues(name)`	Retrieves all of the values for a single request parameter.

Method	Description
`getHeaderNames()`	Retrieves the names of all of headers associated with the request.
`getHeader(name)`	Returns the value of a single request header, as a string.
`getHeaders(name)`	Returns all of the values for a single request header.
`getIntHeader(name)`	Returns the value of a single request header, as an integer.
`getDateHeader(name)`	Returns the value of a single request header, as a date.
`getCookies()`	Retrieves all of the cookies associated with the request.

JSP: *request object*

Method	Description
`getMethod()`	Returns the HTTP (e.g., `GET`, `POST`) method for the request.
`getRequestURI()`	Returns the request URL, up to but not including any query string.
`getQueryString()`	Returns the query string that follows the request URL, if any.
`getSession(flag)`	Retrieves the session data for the request (i.e., the `session` implicit object), optionally creating it if it doesn't already exist.
`getRequestDispatcher(path)`	Creates a request dispatcher for the indicated local URL.
`getRemoteHost()`	Returns the fully qualified name of the host that sent the request.
`getRemoteAddr()`	Returns the network address of the host that sent the request.
`getRemoteUser()`	Returns the name of user that sent the request, if known.
`getSession(flag)`	Retrieves the session data for the request (i.e., the `session` implicit object), optionally creating it if it doesn't already exist.

JSP: *response object*

> Oggetto legato all'I/O della pagina JSP

> L'oggetto *response* rappresenta la risposta che viene restituita al client

> Consente di inserire nella risposta diverse informazioni:

- il content type e l'enconding
- eventuali header di risposta
- URL Rewriting
- i cookie

```
<%response.setDateHeader("Expires", 0);
  response.setHeader("Pragma", "no-cache");
  if (request.getProtocol().equals("HTTP/1.1")) {
          response.setHeader("Cache-Control", "no-cache");
  }
%>
```

JSP: *response object*

Method	Description
`setContentType()`	Set the MIME type and, optionally, the character encoding of the response's contents.
`getCharacterEncoding()`	Returns the character encoding style set for the response's contents.

Method	Description
`addCookie(cookie)`	Adds the specified cookie to the response.
`containsHeader(name)`	Checks whether the response includes the named header.
`setHeader(name, value)`	Assigns the specified string value to the named header.
`setIntHeader(name, value)`	Assigns the specified integer value to the named header.
`setDateHeader(name, date)`	Assigns the specified date value to the named header.
`addHeader(name, value)`	Adds the specified string value as a value for the named header.
`addIntHeader(name, value)`	Adds the specified integer value as a value for the named header.
`addDateHeader(name, date)`	Adds the specified date value as a value for the named header.

JSP: *response object*

Method	Description
setStatus(code)	Sets the status code for the response (for non-error circumstances).
sendError(status, msg)	Sets the status code and error message for the response.
sendRedirect(url)	Sends a response to the browser indicating it should request an alternate (absolute) URL.

Method	Description
encodeRedirectURL(url)	Encodes a URL for use with the `sendRedirect()` method to include session information.
encodeURL(name)	Encodes a URL used in a link to include session information.

JSP: *out object*

> Oggetto legato all'I/O della pagina JSP

> Rappresenta lo stream di output della pagina

> Esempio:

```
<p>Counting eggs
       <% int count = 0;
              while (carton.hasNext()) {
                count++;
                out.print(".");
       } %>
<br />
There are <%= count %> eggs.</p>
```

JSP: *out object*

Method	Description
`isAutoFlush()`	Returns `true` if the output buffer is automatically flushed when it becomes full, `false` if an exception is thrown.
`getBufferSize()`	Returns the size (in bytes) of the output buffer.
`getRemaining()`	Returns the size (in bytes) of the unused portion of the output buffer.
`clearBuffer()`	Clears the contents of the output buffer, discarding them.
`clear()`	Clears the contents of the output buffer, signaling an error if the buffer has previously been flushed.
`newLine()`	Writes a (platform-specific) line separator to the output buffer.
`flush()`	Flushes the output buffer, then flushes the output stream.
`close()`	Closes the output stream, flushing any contents.

JSP: *session object*

> Oggetto che fornisce informazioni sul contesto di esecuzione della JSP

> Rappresenta la sessione corrente per un utente

> La direttiva *session* deve essere valorizzata a *true* affinché la JSP partecipi alla sessione

> Esempio:

```
<%UserLogin userData = new UserLogin(name, password);
   session.setAttribute("login", userData);
%>
<%UserLogin userData=(UserLogin)session.getAttribute("login");
  if (userData.isGroupMember("admin")) {
     session.setMaxInactiveInterval(60*60*8);
  } else {
     session.setMaxInactiveInterval(60*15);
  }
%>
```

JSP: *session object*

Method	Description
`getId()`	Returns the session ID.
`getCreationTime()`	Returns the time at which the session was created.
`getLastAccessedTime()`	Returns the last time a request associated with the session was received.
`getMaxInactiveInterval()`	Returns the maximum time (in seconds) between requests for which the session will be maintained.
`setMaxInactiveInterval(t)`	Sets the maximum time (in seconds) between requests for which the session will be maintained.
`isNew()`	Returns `true` if user's browser has not yet confirmed the session ID.
`invalidate()`	Discards the session, releasing any objects stored as attributes.

JSP: *application object*

- Oggetto che fornisce informazioni sul contesto di esecuzione della JSP
- Rappresenta la web application a cui la JSP appartiene
- Consente di interagire con l'ambiente di esecuzione:
  - fornisce la versione del JSP Container
  - garantisce l'accesso a risorse server-side
  - permette accesso ai parametri di inizializzazione relativi all'applicazione
  - consente di gestire gli attributi di una applicazione

JSP: *pageContext object*

>Oggetto che fornisce informazioni sul contesto di esecuzione della JSP

>Rappresenta l'insieme degli oggetti impliciti di una JSP

>Consente l'accesso a tutti gli oggetti impliciti e ai loro attributi

>Consente il trasferimento del controllo ad altre pagine

>Poco usato per lo scripting, utile per costruire custom tags

JSP: *exception object*

>Oggetto connesso alla gestione degli errori

> Rappresenta l'eccezione che non viene gestita da nessun blocco catch

> Non è automaticamente disponibile in tutte le pagine
(solo nelle Error Page)

> Esempio:

```
<%@ page isErrorPage="true" %>
        <h1>Warning!</h1>
        The following error has been detected:<br/>
        <b><%= exception %></b><br/>
        <% exception.printStackTrace(out);

%>
```

JSP: *exception object*

Method	Description
`getMessage()`	Returns the descriptive error message associated with the exception when it was thrown.
`printStackTrace(out)`	Prints the execution stack in effect when the exception was thrown to the designated output stream.
`toString()`	Returns a string combining the class name of the exception with its error message (if any).

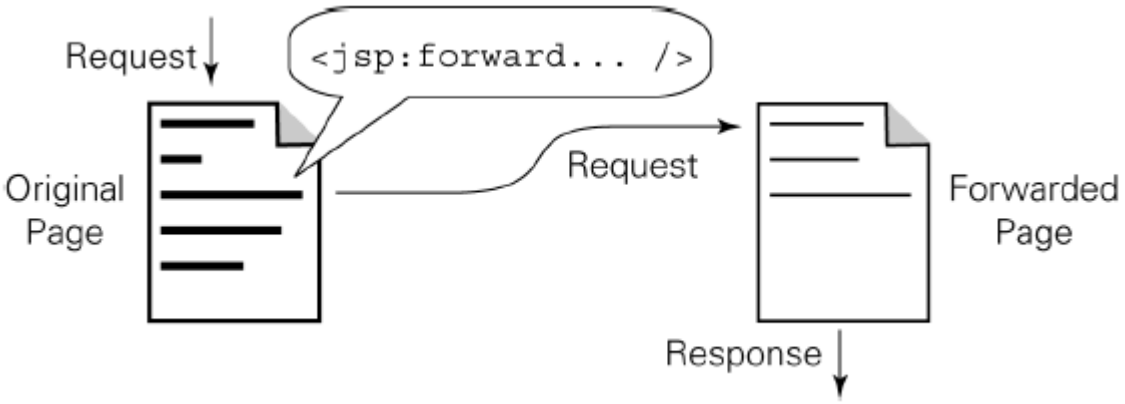
JSP - azioni: *forward action*

> Sintassi: **`<jsp:forward page="$localURL" />`**

> Consente il trasferimento del controllo dalla pagina JSP corrente ad un'altra pagina sul server locale

>L'attributo page definisce l'URL della pagina a cui trasferire il controllo

>La request viene completamente trasferita in modo trasparente per il client



JSP - azioni: *forward action*

> E' possibile generare dinamicamente l'attributo page

```
<jsp:forward page='<%="message"+statusCode+".html"%>'/>
```

> Gli oggetti *request*, *response* e *session* della pagina d'arrivo sono gli stessi della pagina chiamante, ma viene istanziato un nuovo contesto di pagina

> E' possibile aggiungere parametri all'oggetto request della pagina chiamata utilizzando il tag <jsp:param>

```
<jsp:forward page="localURL">
  <jsp:param name="parameterName1" value="parameterValue1"/>
  …
  <jsp:param name="parameterNameN" value="parameterValueN"/>
</jsp:forward>
```

> Il *forward* è possibile soltanto se non è già stato fatto un *flush* del buffer di output o se la chiamata avviene prima della creazione di qualsiasi output

JSP - azioni: *include action*

> Sintassi: **`<jsp:include page="$localURL" flush="true" />`**

>Consente di includere il contenuto generato dinamicamente da un'altra pagina locale all'interno dell'output della pagina corrente

>Questo tag trasferisce temporaneamente il controllo ad un'altra pagina

>L'attributo *page* definisce l'URL della pagina da includere

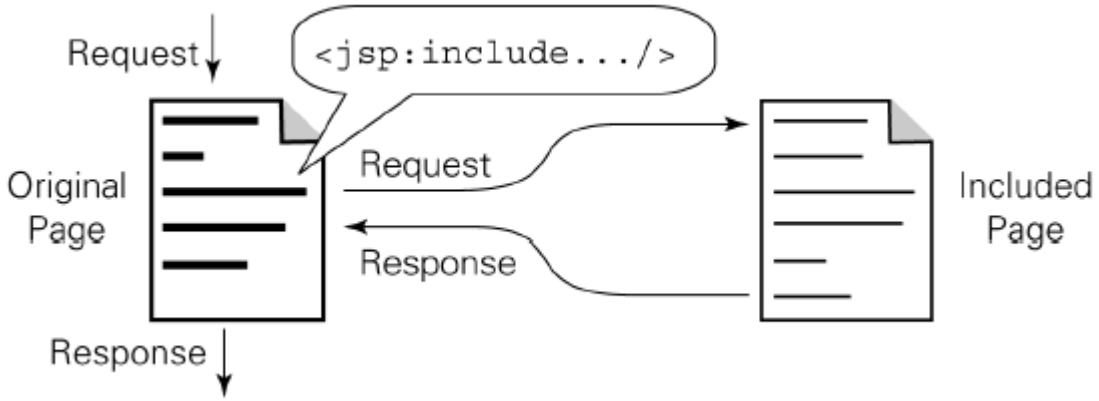
>L'attributo *flush* stabilisce se sul buffer della pagina corrente debba essere eseguito un flush prima di effettuare l'inclusione (in questo caso deve valere *true*)

>Gli oggetti *session* e *request* per la pagina da includere sono gli stessi della pagina chiamante, ma viene istanziato un nuovo contesto di pagina

JSP - azioni: *include action*

> E' possibile aggiungere parametri all'oggetto *request* della pagina inclusa utilizzando il tag `<jsp:param>`

```
<jsp:include page="localURL“ flush=“true”>
  <jsp:param name="parameterName1“ value="parameterValue1"/>
          …
  <jsp:param name="parameterNameN“ value="parameterValueN"/>
</jsp: include >
```



JSP: JavaBean

>I Componenti solo elementi software autonomi, riusabili che incapsulano un certo insieme di funzionalità

> I Componenti dialogano fra loro e con l'esterno attraverso una interfaccia ben definita

> I **JavaBean** sono componenti software scritti in Java

> I JavaBean hanno delle proprietà che ne definiscono lo stato:

- possono essere read-only, write-only o readable and writable
- possono essere trigger o linked
- possono essere indexed
- possono contenere qualsiasi tipo di dato Java

JSP: Bean tag

> Esistono dei tag per agganciare un bean e le sue proprietà

> Ci sono tre tipi di tag:

- Tag per creare un riferimento al bean
- Tag per settare il valore delle proprietà del bean
- Tag per leggere il valore delle proprietà del bean

JSP: Bean tag

```
<jsp:useBean id="user" class="RegisteredUser" scope="session"/>
<jsp:useBean id="news" class="NewsReports" scope="request">
 <jsp:setProperty name="news" property="category" value="fin."/>
 <jsp:setProprety name="news" property="maxItems" value="5"/>
</jsp:useBean>

<html>
  <body>
    Welcome back
    <jsp:getProperty name="user" property="fullName"/>,
    your last visit was on
    <jsp:getProperty name="user" property="lastVisitDate"/>.
    Glad to see you again!
    <P>
    There are <jsp:getProperty name="news" property="newItems"/>
    new articles available for your reading pleasure. Please
        enjoy your stay and come back soon.
  </body>
</html>
```

JSP: <jsp:useBean>

> Sintassi: **`<jsp:useBean id="beanName" class="class name"/>`**

> Inizializza e crea il riferimento al bean

> Gli attributi principali sono *id* e *class* ma è possibile specificarne altri

Attribute	Value	Default	Example Value
`id`	Java identifier	none	`myBean`
`scope`	page, request, session, application	page	`session`
`class`	Java class name	none	`java.util.Date`
`type`	Java class name	same as class	`com.manning.jsp.AbstractPerson`
`beanName`	Java class or serialized Bean	none	`com.manning.jsp.USCurrency.ser`

JSP: <jsp:getProperty>

> Sintassi: **<jsp:getProperty name="$beanName"**
property="$propertyName"/>

> Consente l'accesso alle proprietà del bean

> Produce come output il valore della proprietà del bean

> Il tag non ha mai body e ha solo 2 attributi:

- name: definisce il nome del bean a cui si fa riferimento
- property: definisce il nome della proprietà di cui si vuole visualizzare il valore

JSP: <jsp:getProperty>

```
<jsp:useBean id="style" class="beans.CorporateStyleBean"/>
<html>
  <body bgcolor="<jsp:getProperty name="style"
       property="color"/>">
    <center>
      <img src="<jsp:getProperty name="style"
       property="logo"/>">
      Welcome to Big Corp!
    </center>
  </body>
</html>
```

```
<html>
  <body bgcolor="pink">
    <center>
      <img src="http://imageserver/logo.gif">
      Welcome to Big Corp!
    </center>
  </body>
</html>
```

JSP: <jsp:setProperty>

> Sintassi: **<jsp:setProperty name="$beanName"**
property="$propertyName"
value="$propertyValue "/>

> Consente di modificare il valore delle proprietà del bean

> Il bean eseguirà determinate operazioni in funzione del valore assegnato alle proprietà

> Esempi:

```
<jsp:setProperty name="user" property="daysLeft" value="30"/>
<jsp:setProperty name="user" property="daysLeft"
                                        value="<%=15*2%>"/>
```

JSP: indexed properties

>I bean tag non supportano le proprietà indicizzate

> Un bean è un normale oggetto Java: è possibile accedere a variabili e metodi.

> Esempio:

```
<b>Tomorrow's Forecast</b>: <%= weather.getForecasts(0)%>
<br/>
<b>The Rest of the Week</b>
<ul>
   <% for (int index=1; index < 5; index++) { %>
        <li><%= weather.getForecasts(index) %></li>
   <% } %>
</ul>
```