



# Laboratorio di Programmazione di Rete

Lezione del 17 Maggio 2010

Docente: Novella Bartolini



## Un primo esempio di pagina JSP

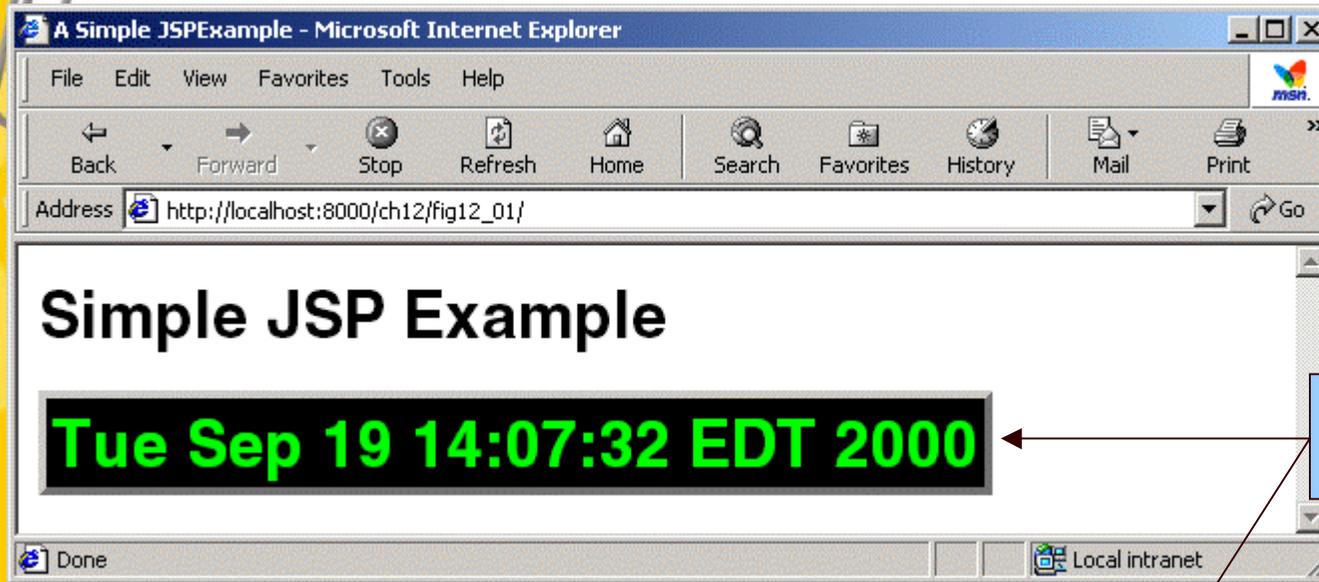
### Esempio in cui

- Viene creata la struttura della pagina attraverso markup XHTML
  - Viene creato un oggetto java (**java.util.Date**)
  - Viene effettuata la conversione automatica di un'espressione JSP in un'oggetto **String**
  - Viene usato un **meta**-elemento per ricaricare la pagina a specifici intervalli di tempo
- ➔ Si noti alla prima invocazione di **clock.jsp** il ritardo con cui
- Il JSP container traduce la pagina JSP in una servlet
  - Il JSP container compila la servlet
  - Il JSP container esegue la servlet
- ➔ Le successive richieste della pagina JSP non sperimentano questo ritardo.

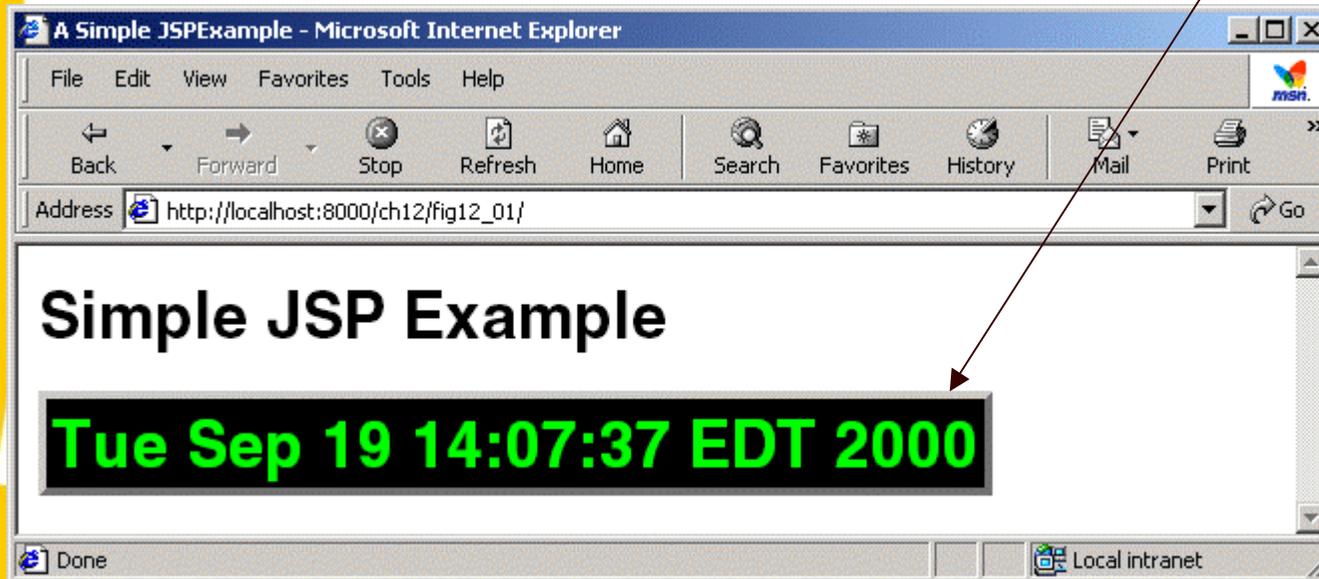
```
4
5
6 <html xmlns = "http://www.w3.org/1999/xhtml">
7
8
9 <head>
10 <meta http-equiv = "refresh" content = "60" />
11
12 <title>A Simple JSP Example</title>
13
14 <style type = "text/css">
15 .big { font-family: helvetica, arial, sans-serif;
16 font-weight: bold;
17 font-size: 2em; }
18 </style>
19 </head>
20
21 <body>
22 <p class = "big">Simple JSP Example</p>
23
24 <table style = "border: 6px outset;">
25 <tr>
26 <td style = "background-color: black;">
27 <p class = "big" style = "color: cyan;">
28
29 <!-- JSP expression to insert date/time -->
30 <%= new java.util.Date() %>
31
32 </p>
33 </td>
34 </tr>
35 </table>
36 </body>
37
38 </html>
```

**meta** element refresh: ricarica la pagina ogni **60** seconds

Crea un oggetto **Date** che viene implicitamente convertito in un oggetto **String**: si tratta di un'espressione



Rappresentazione tramite String dell'oggetto Date





# Aree di visibilità (scope) in una pagina JSP

- ➔ Le pagine JSP possono accedere ad oggetti definiti in diverse aree di visibilità (scope):
  - **Applicazione**
    - Oggetti associati al **contesto servlet** della JSP;
    - Per recuperare tali oggetti si ricorre al metodo `javax.servlet.ServletContext.getAttribute()`
  - **Pagina**
    - Oggetti che sono visibili solo al codice presente sulla stessa pagina
    - Per accedervi si usa `javax.servlet.jsp.PageContext.getAttribute()`
    - Una volta completata la richiesta della pagina il container elimina il riferimento a tali oggetti





## Aree di visibilità (continua)

### – Richiesta

- Visibilità uguale alla richiesta
- Vi si accede con il metodo `javax.servlet.ServletRequest.getAttribute()`
- Viene eliminato il riferimento a tali oggetti quando viene inviata la risposta

### – Sessione

- Oggetti associati ad una sessione utente
- Vi si accede con il metodo `javax.servlet.http.HttpSession.getAttribute ()`
- I riferimenti a tali oggetti vengono eliminati quando la sessione termina (per volere del client o per timeout)



# Oggetti impliciti

Implicit Object	Description
<i>Application Scope</i>	
<b>application</b>	This <code>javax.servlet.ServletContext</code> object represents the container in which the JSP executes.
<i>Page Scope</i>	
<b>config</b>	This <code>javax.servlet.ServletConfig</code> object represents the JSP configuration options. As with servlets, configuration options can be specified in a Web application descriptor.
<b>exception</b>	This <code>java.lang.Throwable</code> object represents the exception that is passed to the JSP error page. This object is available only in a JSP error page.
<b>out</b>	This <code>javax.servlet.jsp.JspWriter</code> object writes text as part of the response to a request. This object is used implicitly with JSP expressions and actions that insert string content in a response.
<b>page</b>	This <code>java.lang.Object</code> object represents the <b>this</b> reference for the current JSP instance.
<b>pageContext</b>	This <code>javax.servlet.jsp.PageContext</code> object hides the implementation details of the underlying servlet and JSP container and provides JSP programmers with access to the implicit objects discussed in this table.



## Oggetti impliciti (cont.)

Implicit Object	Description
<b>response</b>	This object represents the response to the client. The object normally is an instance of a class that implements <b>HttpServletResponse</b> (package <b>javax.servlet.http</b> ). If a protocol other than HTTP is used, this object is an instance of a class that implements <b>javax.servlet.ServletResponse</b> .
<i>Request Scope</i>	
<b>request</b>	This object represents the client request. The object normally is an instance of a class that implements <b>HttpServletRequest</b> (package <b>javax.servlet.http</b> ). If a protocol other than HTTP is used, this object is an instance of a subclass of <b>javax.servlet.ServletRequest</b> .
<i>Session Scope</i>	
<b>session</b>	This <b>javax.servlet.http.HttpSession</b> object represents the client session information if such a session has been created. This object is available only in pages that participate in a session.



## Oggetto Page

- L'oggetto page rappresenta l'istanza corrente della servlet corrispondente alla pagina JSP
- Ha come tipo l'interfaccia HTTPJspPage che discende da JSP page, la quale a sua volta estende Servlet
- Può quindi essere quindi utilizzato per accedere a tutti i metodi definiti nelle servlet

```
<%@ page info="Esempio di uso page." %>  
<p>Page info: <%=page.getServletInfo() %> </p>
```

```
<p>Page info: Esempio di uso di page</p>
```

JSP



HTML



## Oggetto **PageContext**

- 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 attraverso i corrispondenti metodi *get*:

*getPage()*

*getRequest()*

*getResponse()*

*getSession()*

*getServletContext()*

*getException()* ecc.

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



## Oggetto **Application**

- Oggetto che fornisce informazioni sul contesto di esecuzione della JSP (è il **ServletContext**)
- Rappresenta la web application a cui la JSP appartiene
- Consente di interagire con l'ambiente di esecuzione:
  - garantisce l'accesso a risorse server-side
  - permette accesso ai parametri di inizializzazione relativi all'applicazione
  - consente di gestire gli attributi di un'applicazione



## Oggetto **Exception**

- 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 ma solo nelle Error Page (quelle dichiarate con l'attributo `errorPage` impostato a `true`)

```
<%@ page isErrorPage="true" %>
<h1>Attenzione!</h1>
E' stato rilevato il seguente errore:<br/>
<b><%= exception %></b><br/>
<% exception.printStackTrace(out); %>
```



# Componenti di scripting JSP

- ➔ Come specificare componenti JSP
  - Scriptlets (delimitate da `<% and %>`)
  - Commenti (delimitati da `<%-- and --%>`)
  - Espressioni (delimitati da `<%= and %>`)
  - Dichiarazioni (delimitati da `<%! and %>`)



# Scriptlet

- ➔ Delimitate da `<% e %>`
- ➔ Blocchi di codice Java
- ➔ Inserite nel metodo `_jspService` al momento della traduzione
  
- ➔ Scriptlet, espressioni e codice XHTML possono essere intercalati per creare diverse risposte sulla base di informazioni incluse nella richiesta



## Componenti di scripting JSP (seq. di escape)

Literal	Escape sequence	Description
<%	<%	The character sequence <% normally indicates the beginning of a scriptlet. The <% escape sequence places the literal characters <% in the response to the client.
%>	%\>	The character sequence %> normally indicates the end of a scriptlet. The %\> escape sequence places the literal characters %> in the response to the client.
' " \	\' \" \\	As with string literals in a Java program, the escape sequences for characters ' , " and \ allow these characters to appear in attribute values. <b>Remember that the literal text in a JSP becomes string literals in the servlet that represents the translated JSP .</b>



# Commenti

- ➔ Sono supportati tre tipi di commento:
  - Commento JSP
  - Commento XHTML
  - Commento del linguaggio di scripting



## Commento JSP

- Commento JSP
  - `<%-- --%>`
    - Non si usa all'interno di scriptlet
    - Non è visibile al client



# Commento XHTML

- Commento XHTML
  - `<!-- -->`
    - Non si usa all'interno di scriptlet
    - E' visibile al client



## Commento del linguaggio di scripting

- Commento del linguaggio di scripting
  - Single-line //, oppure Multi-line /\* \*/
    - Si usa esclusivamente all'interno di scriptlet
    - E' visibile al client?



# Espressioni

- ➔ Sono delimitate da `<%=` e `%>`
- ➔ Contengono espressioni Java che vengono valutate quando il client richiede la pagina che le contiene
- ➔ Il container converte il risultato di un'espressione in un oggetto **String** e lo invia in output come parte della risposta (metodo `_jspService()` )



# Dichiarazioni

- ⇒ Delimitate da `<%! e %>`
- ⇒ Consentono la definizione di variabili e metodi attraverso la sintassi di Java
- ⇒ Le **variabili** diventano **attributi della classe** servlet che rappresenta la pagina JSP
- ⇒ I **metodi** così dichiarati corrisponderanno ai **metodi della classe** servlet che rappresenta la pagina JSP
- ⇒ La stessa variabile senza `<%!` diventa var locale di `_jspService()`



## Esempio sulla dichiarazione

- ➔ Scrivere una pagina.jsp contenente una variabile **contatore** e delle istruzioni di incremento del suo valore.
- ➔ Ci sono differenze se la variabile viene utilizzata nei seguenti modi:
  - `<% int counter=0; %>`
  - `<%! int counter=0; %>` ?



Nel primo caso  
scrive sempre:  
1, 2

Nel secondo ?

```
1 <html>
2 <head>
3 <title>
4 Contatore dichiarato come scriptlet
5 </title>
6
7 </head>
8 <body>
9 <%@ page language="java" %>
10 <% int counter=0; %> oppure <%! int counter=0; %>
11 <% counter++; %>
12 <p>Il contatore vale <%= counter %>.</p>
13 <% counter++; %>
14 <p>Il contatore vale <%= counter %>.</p>
15
16
17 </body>
18 </html>
```



## Azione standard `<jsp:include>`

**Reminder:** redirectione attraverso direttiva include

```
<%@ include file="relativeURLspec"%>
```

*specifica il percorso relativo (URL) di un file che deve essere incluso*

### ➔ `<jsp:include ...>`

- Consente l'inclusione di contenuto **dinamico** in una pagina JSP
- Più flessibile della direttiva **include**
  - Richiede maggiore overhead quando il contenuto della pagina cambia frequentemente
  - La **direttiva** include il codice al momento della **traduzione** (l'inclusione corrisponde ad una serie di print), mentre **l'azione standard** include il codice solo al momento dell'**esecuzione**



## Azione `<jsp:include>`

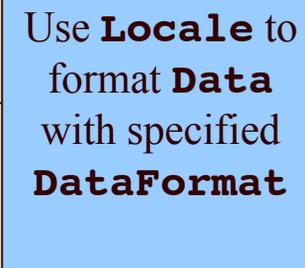
Attribute	Description
<b>page</b>	Specifies the relative URI path of the resource to include. The resource must be part of the same Web application.
<b>flush</b>	Specifies whether the buffer should be flushed before the <b>include</b> is performed. In JSP 1.1, this attribute is required to be <b>true</b> .

```
1 <!-- banner.html      -->
2 <!-- banner to include in another document -->
3 <div style = "width: 580px">
4   <p>
5     Corso di
6     Laboratorio <br /> Internet and
7     World Wide Web Programming Training&nbsp;<br />
8     On-Site Seminars Delivered Worldwide
9   </p>
10
11  <p>
12    <a href = "mailto:novella@di.uniroma1.it">
13      novella@di.uniroma1.it</a><br />
14
15    978.579.9911<br />
16    490B Boston Post Road, Suite 200,
17    Sudbury, MA 01776
18  </p>
19 </div>
```



```
1 <!-- toc.html -->
2 <!-- contents to include in another document -->
3
4 <p><a href = "http://www.uniroma1.it/books/index.html">
5   Publications/BookStore
6 </a></p>
7
8 <p><a href = "http://www.uniroma1.it/whatsnew.html">
9   What's New
10 </a></p>
11
12 <p><a href = "http://www.uniroma1.it/books/downloads.html">
13   Downloads/Resources
14 </a></p>
15
16 <p><a href = "http://www.uniroma1.it/faq/index.html">
17   FAQ (Frequently Asked Questions)
18 </a></p>
19
20 <p><a href = "http://www.uniroma1.it/intro.html">
21   Who we are
22 </a></p>
23
24 <p><a href = "http://www.uniroma1.it/index.html">
25   Home Page
26 </a></p>
27
28 <p>Send questions or comments about this site to
29   <a href = "mailto:novella@di.uniroma1.it">
30     novella@di.uniroma1.it
31   </a><br />
32
33
34 </p>
```

```
1 <!-- Fig. 10.9: clock2.jsp -->
2 <!-- date and time to include in another document -->
3
4 <table>
5 <tr>
6 <td style = "background-color: black;">
7 <p class = "big" style = "color: cyan; font-size: 3em;
8 font-weight: bold;">
9
10 <%-- script to determine client local and --%>
11 <%-- format date accordingly --%>
12 <%
13 // get client locale
14 java.util.Locale locale = request.getLocale();
15
16 // get DateFormat for client's Locale
17 java.text.DateFormat dateFormat =
18 java.text.DateFormat.getDateInstance(
19 java.text.DateFormat.LONG,
20 java.text.DateFormat.LONG, locale);
21
22 %> <%-- end script --%>
23
24 <%-- output date --%>
25 <%= dateFormat.format( new java.util.Date() ) %>
26 </p>
27 </td>
28 </tr>
29 </table>
```



Use **Locale** to  
format **Data**  
with specified  
**DateFormat**

```
1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- include.jsp -->
6
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8
9   <head>
10     <title>Using jsp:include</title>
11
12     <style type = "text/css">
13       body {
14         font-family: tahoma, helvetica, arial, sans-serif;
15       }
16
17       table, tr, td {
18         font-size: .9em;
19         border: 3px groove;
20         padding: 5px;
21         background-color: #dddddd;
22       }
23     </style>
24   </head>
25
26   <body>
27     <table>
28       <tr>
29         <td style = "width: 160px; text-align: center">
30           <img src = "images/logotiny.png"
31             width = "140" height = "93"
32           />
33         </td>
34
```

```
35     <td>
36
37     <%-- include banner.html in this JSP --%>
38     <jsp:include page = "banner.html"
39         flush = "true" />
40
41     </td>
42 </tr>
43
44 <tr>
45     <td style = "width: 160px">
46
47     <%-- include toc.html in this JSP --%>
48     <jsp:include page = "toc.html" flush = "true" />
49
50     </td>
51
52     <td style = "vertical-align: top">
53
54     <%-- include clock2.jsp in this JSP --%>
55     <jsp:include page = "clock2.jsp"
56         flush = "true" />
57
58     </td>
59 </tr>
60 </table>
61 </body>
62 </html>
```





## Azione `<jsp:forward>`

### ➔ `<jsp:forward>`

- Consente ad una pagina JSP di inoltrare la richiesta ad altre risorse

➔ L'azione `<jsp:param>` (annidata nel forward) specifica coppie nome/valore di dati da allegare ad altre azioni

```
1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 10.11: forward1.jsp -->
6
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8
9 <head>
10  <title>Forward request to another JSP</title>
11 </head>
12
13 <body>
14  <% // begin scriptlet
15
16     String name = request.getParameter( "firstName" );
17
18     if ( name != null ) {
19
20  %> <%-- end scriptlet to insert fixed template data --%>
21
22     <jsp:forward page = "forward2.jsp">
23       <jsp:param name = "date"
24         value = "<%= new java.util.Date() %>" />
25     </jsp:forward>
26
27  <% // continue scriptlet
28
29     } // end if
30     else {
31
32  %> <%-- end scriptlet to insert fixed template data --%>
33
```

e to  
rds  
or

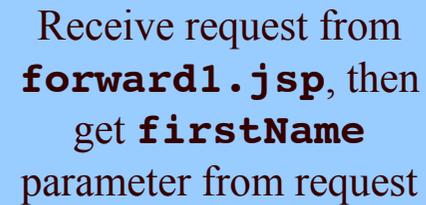
Forward request to  
**forward2.jsp**

```
34 <form action = "forward1.jsp" method = "get">
35 <p>Type your first name and press Submit</p>
36
37 <p><input type = "text" name = "firstName" />
38 <input type = "submit" value = "Submit" />
39 </p>
40 </form>
41
42 <% // continue scriptlet
43
44 } // end else
45
46 %> <%-- end scriptlet --%>
47 </body>
48
49 </html> <!-- end XHTML document -->
```

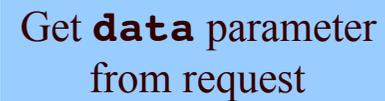


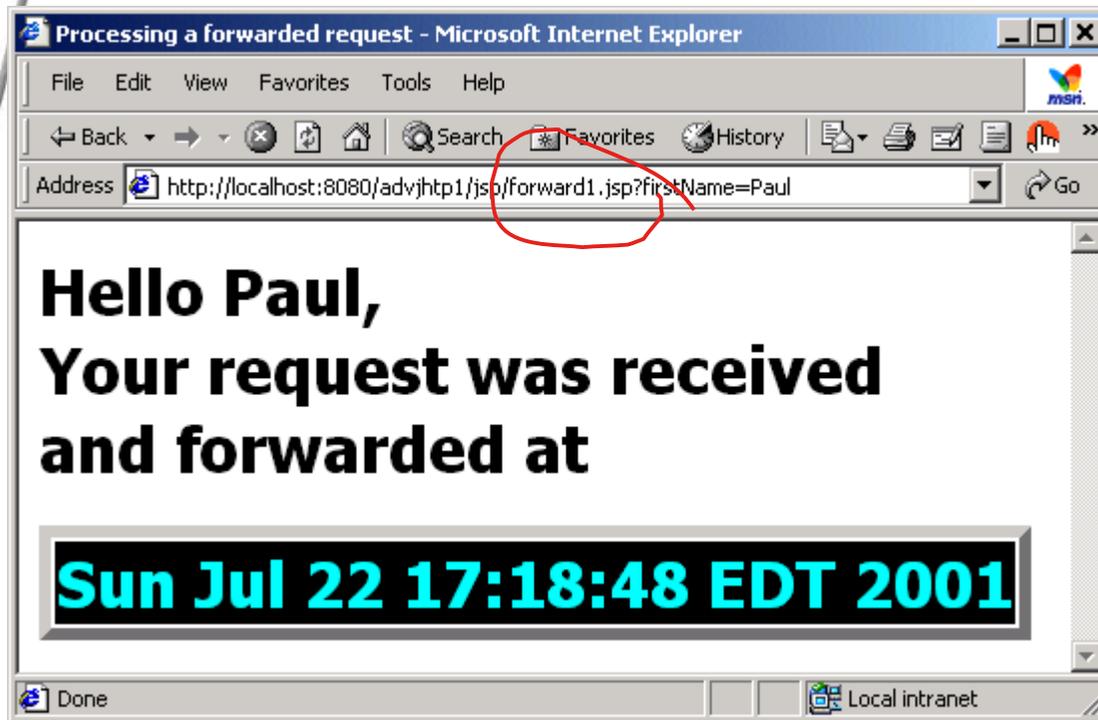
```
1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- forward2.jsp -->
6
7 <html xmlns = "http://www.w3.org/1999/xhtml"v
8
9 <head>
10  <title>Processing a forwarded request</title>
11
12  <style type = "text/css">
13    .big {
14      font-family: tahoma, helvetica, arial, sans-serif;
15      font-weight: bold;
16      font-size: 2em;
17    }
18  </style>
19 </head>
20
21 <body>
22  <p class = "big">
23    Hello <%= request.getParameter( "firstName" ) %>, <br />
24    Your request was received <br /> and forwarded at
25  </p>
26
27  <table style = "border: 6px outset;">
28    <tr>
29      <td style = "background-color: black;">
30        <p class = "big" style = "color: cyan;">
31          <%= request.getParameter( "date" ) %>
32        </p>
33      </td>
34    </tr>
35  </table>
```

Receive request from  
**forward1.jsp**, then  
get **firstName**  
parameter from request



Get **date** parameter  
from request





Redirezione  
INTERNA



## JSP e Bean

### ➔ Azione Standard **<jsp:useBean>**

- Il codice contenuto negli scriptlet Java può esser inserito in una classe apposita invocata al momento del bisogno
- Necessaria la **conformità agli standard** fissati per i JavaBean (architettura a componenti)

```
// classe JellyBean.java
```

```
package beans; //NOTARE CHE IL BEAN VIENE MESSO IN UN PACKAGE
```

```
public class JellyBean {  
    private String color;
```

```
    public JellyBean() {  
    }
```

```
    public String getColor() {  
        return color;  
    }
```

```
    public void setColor(String newColor) {  
        color=newColor;  
    }  
}
```



# JavaBeans

- ⇒ Ai fini dello sviluppo di servlet e JSP contano i seguenti aspetti della specifica:
- I JavaBean non devono avere nessuna proprietà pubblica (attributi di classe privati).
  - Tutte le proprietà di un JavaBean che devono essere esposte dovranno avere metodi pubblici **get** e **set** secondo necessità.
    - Tali metodi dovranno avere nome **getVariableName** e **setVariableName**
  - Tutti i JavaBean devono avere un metodo costruttore senza argomenti



## Utilizzo di un JavaBean in una JSP

- ➔ Affinchè sia visibile alla web application il bean deve essere incluso nella sotto-directory **classes** di **WEB-INF** con tutto il percorso del suo package
- ➔ Una volta creato il JavaBean, per poterlo utilizzare in una JSP vengono definiti 3 tag:
  - Per individuare o creare un JavaBean nell'area di visibilità specificata
  - Per definire una proprietà di un JavaBean (set)
  - Per leggere una proprietà di un JavaBean (get)



# Caricamento di un JavaBean

➔ Il tag `<jsp:useBean>` ha la seguente sintassi:

– `<jsp:useBean id="nome" scope="page | request | session | application"  
class="nomeClasse" />`

- id è l'etichetta che viene assegnata al bean all'interno dell'applicazione (nome dell'istanza cui si fa riferimento)
- scope definisce le modalità con cui l'istanza del Bean deve essere ricercata

➔ Esempio:

– `<jsp:useBean id="jb" scope="application" class="beans.JellyBean" />`



## Attributi dell'azione <jsp:useBean>

Attribute	Description
<b>id</b>	The name used to manipulate the Java object with actions <code>&lt;jsp:setProperty&gt;</code> and <code>&lt;jsp:getProperty&gt;</code> . A variable of this name is also declared for use in JSP scripting elements. The name specified here is case sensitive.
<b>scope</b>	The scope in which the Java object is accessible — <b>page</b> , <b>request</b> , <b>session</b> or <b>application</b> . The default scope is <b>page</b> .
<b>class</b>	The fully qualified class name of the Java object.
<b>type</b>	The type of the JavaBean. This can be the same type as the <b>class</b> attribute, a superclass of that type or an interface implemented by that type. The default value is the same as for attribute <b>class</b> . A <b>ClassCastException</b> occurs if the Java object is not of the type specified with attribute <b>type</b> .
<b>Attributes of the &lt;jsp:useBean&gt; action.</b>	



## Impostazione di una proprietà JavaBean

➔ Una volta creato il Bean è possibile definirne le proprietà con il tag **jsp:setProperty**

– `<jsp:setProperty name="beanName" property="propertyName" value="propertyValue" />`

*(assegnazione di un valore)*

– `<jsp:setProperty name="beanName" property="propertyName" param="parameterName" />`

*(passaggio di un parametro della richiesta)*

*Si noti che qui "name" è il nome dell'istanza, ovvero l'"id"*

*Specificato nell'azione standard `<jsp:usebean ...>`*





## Impostazione di una proprietà JavaBean

- ➔ Modo rapido per definire le proprietà del Bean attraverso dati provenienti da un form:
  - `<jsp:setProperty name="beanName" property="*">`
- ➔ Aggiungendo “\*” il metodo `setProperty` fa corrispondere **tutti i parametri dell’oggetto richiesta** ai metodi `set` del bean e passa i valori tra di essi.
- ➔ Definizione automatica delle proprietà possibile solo se si rispettano le convenzioni sui nomi dei metodi `set` e `get`



## Attributi dell'azione `<jsp:setProperty>`

Attribute	Description
<b>name</b>	The ID of the JavaBean for which a property (or properties) will be set.
<b>property</b>	The name of the property to set. Specifying "*" for this attribute causes the JSP to match the request parameters to the properties of the bean. For each request parameter that matches (i.e., the name of the request parameter is identical to the bean's property name), the corresponding property in the bean is set to the value of the parameter. If the value of the request parameter is "", the property value in the bean remains unchanged.
<b>param</b>	If request parameter names do not match bean property names, this attribute can be used to specify which request parameter should be used to obtain the value for a specific bean property. This attribute is optional. If this attribute is omitted, the request parameter names must match bean property names.
<b>value</b>	The value to assign to a bean property. The value typically is the result of a JSP expression. This attribute is particularly useful for setting bean properties that cannot be set using request parameters. This attribute is optional. If this attribute is omitted, the JavaBean property must be of a data type that can be set using request parameters.



## Lettura delle proprietà di un bean

- ➔ Una volta creato il Bean è possibile leggerne le proprietà con il tag **jsp:getProperty**
  - `<jsp:getProperty name="beanName" property="propertyName" />`



## Esempio: definizione dell'attributo di un bean per l'accesso da una pagina jsp

```
<!-- Pagina di scrittura delle proprietà: scrivi.jsp -->

<html>
<head>
  <title> Impostazione del colore di JellyBean </title>
</head>
<&@ page language="java" %>
<jsp:useBean id="jb" scope="session" class="beans.JellyBean" />
<body>
<form action = "leggi.jsp" method = "get">

  <strong>Scegli il colore del bean:</strong><br />
  <label>Rosso
    <input name = "newColor" type = "radio"
      value = "red" checked = "checked" />
  </label>
```



## Esempio: definizione dell'attributo di un bean per l'accesso da una pagina jsp

```
<label>Verde
  <input name = "newColor" type = "radio"
    value = "green" /></label>

<label>giallo
  <input name = "newColor" type = "radio"
    value = "yellow" /></label>

  <input type = "submit" value = "Submit" />
</form>
</body>
```



## Come ottenere una proprietà di un JavaBean

```
<!-- leggi proprietà: leggi.jsp ->

<html>
  <head>
    <title> Ottenere il colore di JellyBean </title>
  </head>
  <%@ page language="java" %>
  <jsp:useBean id="jb" scope="session" class="JellyBean" />

  <body>
    <jsp:setProperty name="jb" property="color" param="newColor" />
    Il colore del bean è stato impostato al valore:</td>
    <jsp:getProperty name="jb" property="color" /></td>
  </body>
</html>
```



## Esercizio:

- ➔ Scrivere semplici esempi di pagine jsp contenenti:
  - La direttiva “include”
  - Le azioni standard
    - include
    - forward
    - useBean (con diversi contesti di visibilità)
    - setProperty (sia il caso in cui viene passato un valore, sia il caso in cui viene preso il valore di un parametro della richiesta)
    - getProperty
  - Descrivere (non codice) come vengono tradotte dal jsp container (consegnare solo quest’ultima parte)



## JSP useBean: contesti di visibilità

➔ request

– `<jsp:useBean id="..." class="..." scope="request" />`

➔ session

– `<jsp:useBean id="..." class="..." scope="session" />`

➔ application

– `<jsp:useBean id="..." class="..." scope="application" />`

➔ page

– `<jsp:useBean id="..." class="..." scope="page" />`

oppure

`<jsp:useBean id="..." class="..." />` (page è il contesto di default)



## Visibilità di un JavaBean

- ➔ Si può accedere ad un bean da più pagine JSP, a condizione che
  - Il bean sia stato dichiarato con scope *session* e sia lo stesso utente ad accedere ad entrambe le pagine
    - Se si osserva il codice generato per le JSP si noterà una chiamata al metodo **pageContext.getSession()**
  - Oppure che il bean sia stato dichiarato con scope *application*
    - Tutti gli utenti avranno accesso alla stessa istanza del bean