Lezione 9 – JSP

Esempio:

```
<HTML>
<BODY>
<H1>
<
%=request.getParameter("eta")
%>
</H1>
</BODY>
</HTML>
```

Oggetti Predefiniti

Negli scriptlets e nelle espressioni sono definiti una serie di variabile descritte dalla seguente tabella:

variabile	tipo	Descrizione	Scope
out	Writer	Un oggetto wrapper che scrive nello stream di output.	page
request	HttpServletRequest	La richiesta che ha comportato la chiamata della pagina.	request
response	HttpServletResponse	La risposta alla request.	page
session	HttpSession	La sessione creata per il client che ha richiesto la pagina	session
page	Object	equivalente a `this`	page
application	ApplicationContext	`getSerletConfig().getServletContext()`, area condivisa tra le servlet	application
config	ServletConfig	Equivalente al parametro config del metodo `init` di una servlet.	page
pageContext	PageContext	sorgente degli oggetti, raramente usato	page
exception	Throwable	L'eccezione lanciata dalla pagina che ha generato l'errore.	page solo nella errorPage

Esempi

richiesta.jsp

```
<%@ page errorPage="errorpage.jsp" %>
<html>
  <head>
    <title>Richiesta</title>
```

```
  </head>
  <body>

<b>Hello</b>: "<%=request.getParameter("user"))%>

  </body>
</html>
```

sessione.jsp

```
<%@ page errorPage="errorpage.jsp" %>
<html> <head> <title>Sessione</title> </head>
  <body>
    <%
      Integer count = (Integer)
session.getAttribute("count");
      if ( count == null )
        count = new Integer(0);
      count = new Integer(count.intValue()+1);
      session.setAttribute("count", count);
     %>
      <b>Tu hai visitato il nostro sito <
%=count.toString()%>" volte.</b>
  </body>
</html>
```

Esercizi

1. Collegatevi a subito.it ed effettuate una ricerca. Collegatevi al sito chegiochi.it e fate una ricerca e visionate un oggetto. Ritornate a subito.it e ricaricate l'ulima pagina.
2. provare a introdurre degli errori di compilazione nei file richiesta.jsp e sessione.jsp;
3. provare a introdurre degli errori srun-time nelle pagine di sopra;
4. scrivere una servlet che conta il numero di accessi di uno stesso utente indipendentemente dalla sessione.

Direttive

Le direttive sono usate per passare informazione dalla JSP al contenitore della servlet
La sintassi della direttiva è: `<%@ tipo attributo="valore" %>`
Le principali tipi delle direttive sono:

- `page (vedi sotto);`
- `include` (per l'inclusione statica di risorse al momento della compilazione) ;
- `taglib` (per l'inclusione di librerie di tag vedremo più avanti).

Le direttive di pagina sono indipendenti dalla posizione e sono uniche: un attributo non può essere ridefinito ma solo aggiunto (eccetto **import** che può essere usato più volte). I principali attributi delle direttive di pagina sono:

- `<%@ page language="java" %>`: specifica il tipo di linguaggio usato negli scriptlets del resto della pagina (solo java e per default).
- `<%@ page session="true" %>`: indica se usare o meno le sessioni, per default true;
- `<%@ page import="java.awt.*,java.util.*" %>`: analogo alla direttiva **`import`** di Java (ricorsarsi delle virgolette nel caso della JSP).
- `<%@ page isThreadSafe="false" %>`: indica se il codice contenuto negli scriptlets è thread safe, cioè se può essere eseguito concorrentemente da due thread senza conflitti;

- `<%@ page errorPage="URL" %>`: in caso di errore visualizza la pagina specifica;
- `<%@ page isErrorPage="true" %>`: questa è una pagina di errore ed è quindi definita la variabile `exception` (vedi sotto).

Ecco l'elenco completo degli attributi definibile nella direttiva di pagina (dal documento di specifica delle servler):

attribute	Description
language	Definisce il linguaggio di scripting usato nelle scriptlets, espressioni e definizioni. In JSP 1.2, l'unico valore ammesso è “java”.
extends	Il valore è una classe java completa di package e che sarà la superclasse della classe generata dalla trasformazioni di quest pagina JSP. Da usare con cautela.
import	Descrive le classi o anche interi packages che saranno importati e quindi disponibili nell'ambiente di scripting. Es: import="java.util.*,java.lang.*" Default: import=”java.lang.*, javax.servlet.*, javax.servlet.jsp.*, javax.servlet.http.*”
session	Indica se la sessione viene gestita o meno. Nel caso sia “true”, allora l'oggetto session di tipo javax.servlet.http.HttpSession rappresenta la sessione della richiesta attuale. Se “false” la pagina non gestisce la sessione e l'oggetto session non è disponibile. Default: “true”
buffer	Specifica la presenza e dimensione del buffer per loggetto out. Se “none” non c'é buffering e tutto l'output viene direttamente inviato al client. Altrimenti è possibile specificare un numero con il suffisso “kb” che rappresenta la dimensione in kilobyte minima del buffer. A seconda del valore dell'attributo “autoFlush”, quando il buffer è pieno ne viene inviato il contentuto al client o lanciata un'eccezione. Se non specificato viene predisposto un buffer di almeno 8kb.
autoFlush	Se “true” quando il buffer è pieno il suo contenuto debba essere inviato automaticamente al client oppure se “false” viene lanciata un'eccezione. Default “true”. Nota: non si può usare autoFlush = “false” quando “buffer=none”.
isThreadSafe	Se “false” il motore delle servlet deve accodare eventuali richieste concorrenti in modo da invarne una alla volta alla pagina JSP. Se “true” richieste concorrenti possono essere inviate contemporaneamente. Default “true”. Nota: se “false” il programmatore deve in ogni caso prestare attenzione all'accesso concorrente dei contesti session e application in quanto condivisi con altre Servlet/JSP della Web application.
info	Definisce una stringa di descrizione libera che è verrà poi ritornata dal metodo `Servlet.getServletInfo()` creato nella servlet della pagina JSP.
isErrorPage	Indica che la pagina corrente è una destinazione di un attributo “errorPage” di un'altra pagina JSP.

	Se “true” allora viene definita la variabile “exception” che contiene l'eccezione sollevata dall'altra pagina. Default“false”
errorPage	Definisce l'URL a una risorsa alla quale viene forwardata in caso di eccezione non catturata nel codice della pagina stessa. Se l'URL è quella di una pagina JSP con isErrorPage=”true” allora in questa pagina di errore viene definita la variabile exception che conterrà l'eccezione sollevata. Default: dipende dall'implementazione del motore delle servlet usato. Nota: il passaggio dell'eccezione da pagina che ha sollevato l'eccezione a pagina di errore avviene tramite attributo “javax.servlet.jsp.jspException” dell'oggetto request. Nota: se autoFlush=true e il buffer della pagina che ha sollevato l'eccezione è già stato svuotato, allora il traferimento alla pagina di errore può non funzionare. Qaundo è specificata una pagina di errore nel file web.xml e anche nella pagina JSP, viene scelta quella presente nella pagina JSP.
contentType	Definisce la codifica dei carattere, il tipo di risposta e il MIME type della risorsa creata dalla pagina JSP. Default “text/html”;
pageEncoding	Definisce la codifica della pagina JSP. Se il CHARSET è specificato nel contentType è usato come default o, altrimenti ISO-8859-1.

Dichiarazioni

Le dichiarazioni sono blocchi di codice Java usati per definire variabili e metodi della classe servlet che verrà generata; Le dichiarazioni seguono esattamente la stessa sintassi che hanno in java.

Sintassi: `<%! dichiarazione%>`.
Esempio:

```
<%! String driver="sun.jdbc.odbc.JdbcOdbcDriver";
public String getDriver() {return driver;} %>
```

Java Bean

JavaBeans è un modello di programmazione a componenti per il linguaggio Java. L'obiettivo è quello di ottenere componenti software riusabili ed indipendenti dalla piattaforma (WORA: Write Once, Run Everywhere).
Inoltre tali componenti (beans) possono essere manipolati dai moderni tool di sviluppo visuali e composti insieme per produrre applicazioni.
Ogni classe Java che aderisce a precise convenzioni sulla gestione di proprietà ed eventi può essere un bean (non è necessario estendere una particolare classe).
La classe non deve avere variabili d'istanza pubbliche e deve avere il costruttore di default (ovvero il costruttore senza parametri).
Una **proprietà** è un singolo attributo pubblico. Le proprietà possono essere in lettura/scrittura, sola lettura o sola scrittura,

Una **proprietà** rappresenta un singolo valore e può essere definita da una coppia di metodi `set`/`get`. Il nome della proprietà deriva dal nome di tali metodi. La sola presenza del metodo `get` indica che la proprietà è in sola lettura, la sola presenza del metodo `set` indica che la proprietà è in sola scrittura, mente la presenza di entrami i metodi indica una proprietà in letture e scrittura.
Ad esempio `setX` e `getX` indicano una proprietà X. La presenza di un metodo `isX` indica che X è una proprietà booleana
Esempio:

```
public class MyBean extends Canvas
{
  String myString="Hello";

  public Prova1(){
   setBackground(Color.red);
   setForeground(Color.blue);
  }

  public void setMyString(String
newString){
   myString = newString;
  }

  public String getMyString() {
   return myString;
  }

  public void print(){
   ...
  }
}
```

MyBean ha una proprietà chiamata myString.

Azioni standard

Le azioni standard sono specificate usando la sintassi dei tag XML. Questi tag influenzano il comportamento al run time delle JSP e della risposta inviata indietro al client.
Sintassi: `<jsp:action attributo="valore" />`
oppure
`<jsp:action attributo="valore">`
...
`</jsp:action>`
L'azione specificata (action) può essere:

- `<jsp:useBean .../>`, `<jsp:setProperty .../>` e `<jsp:getProperty .../>`:

Per usare un Bean in una JSP devo prima di tutto definirlo come nell'esempio seguente:
`<jsp:useBean id="bean" class="MyBean" scope="session" />`
L'`id` è il nome che assegno al bean, la `class` è la classe Java del bean, lo `scope` riguarda il campo di esistenza del bean. Altri scope possibili sono application, request, page. Significato degli scope:

application	l'esistenza del bean è legata alla durata dell'applicazione
session	l'esistenza del bean è legata alla durata della sessione

request	l'esistenza del bean è legata alla durata della richiesta

page	l'esistenza del bean è legata alla durata della pagina (this)

Leggere il valore di una proprietà:

```
<jsp:getProperty name="bean" property="myString" />
```

Impostare il valore di una proprietà:

```
<jsp:setProperty name="bean" property="myString" value="alfa" />
```

In maniera analoga possiamo usare ad esempio `setAttribute` della classe HttpRequest per scambiare dei dati tra una Servlet e la pagina JSP.

Le azioni standard non aggiungono niente alle potenzialità di Java ma sono semplicemente una maniera diversa di scrivere le stesse cose, le seguenti righe di codice JSP:

```
<jsp:useBean id="bean" class="MyBean"
scope="session" />
<jsp:getProperty name="bean" property="costo" />
```

sono equivalenti al seguente codice Java:

```
MyBean bean = session.getAttribute(“bean”);

if (bean ==null){
bean=new MyBean(); //costruttore di default
session.setAttribute(“bean”,bean);
}

out.print(bean.getMyString()); //jsp:getProperty
```