Verso l'architettura MVC-2
Java Server Pages (JSP)

1

ALBERTO BELUSSI
ANNO ACCADEMICO 2009/2010

JSP

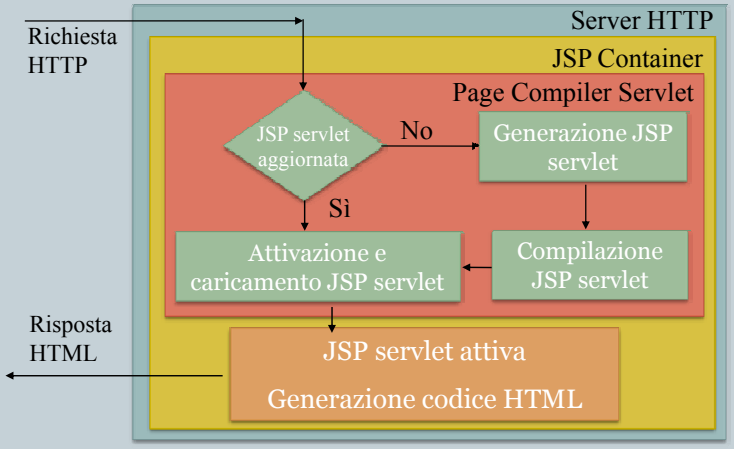
2

- Una pagina JSP può essere vista come uno "schema di pagina Web" dove:
  - le parti statiche sono scritte in HTML e
  - le parti dinamiche sono generate attraverso porzioni di codice Java.
- Le pagine JSP vengono "gestite" da un componente operante sul web server chiamato *JSP container*. Questo componente traduce le JSP in servlet Java, che poi esegue.
- Riferimento: http://java.sun.com/products/jsp

Funzionamento JSP (1)

- Codice sorgente JSP: è scritto dal programmatore dell'applicazione web.
- Si tratta di un file con estensione .jsp contenente:
  - Codice HTML
  - Istruzioni Java (scripting JSP)
  - Marcatori speciali JSP: descrivono il modo in cui generare la servlet associata alla JSP e consentono di gestire oggetti speciali (java data beans, ecc...).

Funzionamento JSP (2)



Sintassi JSP

- Una JSP può essere vista come un documento HTML esteso con alcuni marcatori speciali per "immergere" codice Java nell'HTML. Il "linguaggio" JSP fornisce 4 gruppi principali di marcatori speciali:
  - Direttive (directives)
  - Scripting:
    - Dichiarazione (declarations)
    - Espressione (expressions)
    - Scriptlet
  - Azioni (Actions)
  - Commenti

Direttive (Directives)

- È l'insieme dei marcatori che consentono di definire come il container deve elaborare la JSP.
- Le direttive non influenzano la gestione di una singola richiesta HTTP ma influenzano le proprietà generali della JSP e come **questa deve essere tradotta in una servlet**.
- Una direttiva è introdotta con il tag

  <%@ *tipo* attrib_1="val1"; ... attrib_n="valn" %>

  o il suo equivalente XML <jsp:directive.*tipo* .../>.
- Ad esempio, la seguente direttiva di tipo page stabilisce che il tipo MIME del contenuto della pagina che verrà generata:
  <%@ page contentType="text/html; charset=ISO-8859-1">

Direttive (Directives)

Direttiva PAGE

- <%@page attributi %>
- Attributi significativi
  - ***Import***: consente di precisare classi o package da importare per la compilazione della jsp sevlet
  - ***errorPage***: consente di precisare l'URI di una pagina jsp da invocare in caso di errore
  - ***isErrorPage***: indica se la pagina è una pagina di gestione dell'errore o no. Se true consente l'accesso all'oggetto implicito exception.

Direttive (Directives)

Direttiva INCLUDE

- <%@include attributi %>
- Attributi significativi
  - ***File***: consente di precisare l'URI del file da includere. Si noti che l'include si suppone eseguito prima della compilazione della jsp servlet (**include statico**).

Scripting

- È il gruppo dei marcatori che permettono di inserire delle istruzioni di un linguaggio di programmazione all'interno di codice HTML.
- Le istruzioni devono essere scritte nel linguaggio di programmazione designato per la pagina (di default Java!) e vengono eseguite ad ogni richiesta della pagina JSP.
- Un elemento di scripting è introdotto con uno dei seguenti tag: **<%!...%>** o **<%=...%>** o **<%...%>** che individuano rispettivamente i seguenti tipi di scripting:
  - Dichiarazioni
  - Espressioni
  - Scriptlet

Scripting: Dichiarazioni

<%! dichiarazione**;** [dichiarazione**;**]+ ... %>

oppure

<jsp:declaration> dichiarazione**;** [dichiarazione**;**]+ </jsp:declaration>

- Le dichiarazioni consentono di inserire dichiarazioni di variabili di classe (comuni a più istanze) o metodi statici. Tali metodi possono essere chiamati senza richiedere l'accesso ad una istanza, ad esempio: nomeClasse.nomeMetodo()
- Esempio:

<%! static private int x = 4; %>

Scripting: Espressioni

11

<%= espressione %>

oppure

<jsp:expression> espressione </jsp:expression>

- Viene valutata l'espressione JAVA e il risultato viene sostituito al tag <%= …%> nella pagina HTML generata.
- Esempio:

<%= bean.getMatricola() %>

- Questo tag viene sostituito nella pagina HTML con il valore della proprietà Matricola contenuta nel bean.

Scripting: Scriptlet

12

<% codice su una o più linee %>
oppure
<jsp:scriptlet> espressione </jsp:scriptlet>

- Frammento di codice Java che può modificare anche il flusso del codice HTML generato.
- Solitamente gli operatori condizionali (if, ?, ..) ed i cicli (for, while, ..) possono essere utilizzati per produrre dinamicamente porzioni diverse di codice HTML in funzione dei parametri della richiesta HTTP o dei dati estratti dalla base di dati.
- Tale codice diventerà parte dei metodi doGet (doPost) della servlet che viene associata la JSP.

Scriptlet

Gli scriptlet permettono di modificare il flusso del codice HTML prodotto.

Servlet:

```
for (int i=0; i<10; i++) {
  out.println(i+" &egrave; ");
  if (i%2==0) {
    out.println("pari");
  } else {
    out.println("dispari");
  }
  out.println("<br>");
}
```

JSP:

```
<% for (int i=0; i<10; i++) { %>
  <%= i %> &egrave;
  <% if (i%2==0) { %>
        pari
  <% } else { %>
        dispari
  <% } %>
        <br>
<% } %>
```

Azioni

- Questi marcatori permettono di supportare diversi comportamenti della pagina JSP.
- Vengono processati ad ogni invocazione della pagina JSP.
- Permettono di trasferire il controllo da una JSP all'altra, di interagire con i Java Data Beans, ecc.
- Un'azione è introdotta con un tag del tipo: <jsp:*tipoAzione*.../>
- Ad esempio, se si vuole all'interno di una JSP, includere dinamicamente un'altra JSP, è sufficiente inserire nel punto dove si vuole includere l'altra JSP l'azione:
  <jsp:include page="*localURL*" flush="true"/>

Azioni per l'uso dei Java Data Bean

<jsp:**useBean** id="nome_oggetto"
class="nome_classe" scope="context"/>

dove:

- id: definisce un nome univoco da assegnare al bean
- class: specifica la classe del bean
- scope: indica il periodo di vita del bean, può essere:
  - page (default): il bean viene creato ad ogni richiesta della pagina
  - **request**: in questo caso il bean viene recuperato dall'oggetto request (utile nell'approccio MVC)
  - ....

Azioni per l'uso dei Bean

<jsp:**getProperty** name="id_bean"
property="nome_prop"/>

dove:

- name: è il nome del bean da cui leggere la proprietà (assegnato nell'azione useBean attraverso l'attributo id)
- property: è la proprietà da leggere

Azioni per l'uso dei Bean

<jsp:**setProperty** name="id_bean"
property="nome_prop" value="valore" />

dove:

- name: è l'id del bean da modificare (assegnato nell'azione useBean attraverso l'attributo id)
- property: è la proprietà da modificare
- value: è il nuovo valore da assegnare alla proprietà

Commenti

- Questi marcatori permettono di inserire diversi tipi di commenti all'interno delle JSP.
- Ci sono tre tipi di commenti:
  - <!-- commenti di contenuto -->: sono i tipici commenti di HTML, visibili anche dal browser.
  - <% /* commenti di scripting */ %>: sono i commenti all'interno della parte di codice della JSP e sono visibili anche nella servlet equivalente ma non dal browser
  - <%-- commenti JSP --%>: sono i commenti visibili solo nel sorgente della JSP. (La servlet equivalente non conterrà nulla di questi commenti).

Oggetti impliciti

- Ogni pagina JSP rende disponibile un insieme di oggetti che possono essere utilizzati all'interno della pagina.
- Questi *oggetti impliciti* sono accessibili sia attraverso azioni specifiche sia attraverso elementi di scripting.

Oggetti impliciti (2)

Oggetto	Classe o interfaccia	Descrizione
page	javax.servlet.jsp.HttpJspPage	Rappresenta l'istanza della servlet generata da questa JSP. Raramente utilizzato.
config	javax.servlet.ServletConfig	Rappresenta la configurazione dell'istanza della servlet generata da questa JSP. Raramente utilizzato.
request	javax.servlet.http.HttpServletRequest	È l'oggetto request dell'invocazione http.
response	javax.servlet.http.HttpServletResponse	È l'oggetto response dell'invocazione http.
out	javax.servlet.jsp.JspWriter	È l'output stream per il contenuto della pagina.
session	javax.servlet.http.HttpSession	Rappresenta i dati di una sessione specifica.
application	javax.servlet.ServletContext	Rappresenta i dati condivisi fra tutte le pagine JSP.
pageContext	javax.servlet.jsp.PageContext	Rappresenta i dati specifici della pagina utilizzati durante l'esecuzione. Permette un "accesso programmatico" a tutti gli altri oggetti impliciti
exception	javax.lang.Throwable	Permette di gestire gli errori all'interno della pagina JSP.

Esempio di JSP: hello.jsp


```
<%-- hello.jsp stampa il classico saluto --%>
<%! static private String str = "world!";%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/REC-html40/loose.dtd">

<html>
   <head>
         <title>Hello!</title>
   </head>
   <body>
         <h1>Hello world!</h1>
         <b>Hello, <%= str.toUpperCase() %> </b>
   </body>
</html>
```

Dichiarazione

Espressione

L'esecuzione della Servlet hello$jsp.java associata alla jsp produce codice HTML


```
<!DOCTYPE HTML
   PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/REC-html40/loose.dtd">

<html>
   <head>
         <title>Hello!</title>
   </head>

   <body>
         <h1>Hello world!</h1>
         <b>Hello, WORLD!</b>
   </body>
</html>
```

Gestione degli errori

- Quando si verifica un errore nell'esecuzione di una jsp (ovvero nell'esecuzione della servlet equivalente), il container inoltra il controllo ad una specifica pagina JSP alla quale viene fornito l'accesso all'oggetto implicito exception.
- Ogni pagina jsp definisce qual è la pagina di errore (la pagina a cui inoltrare il controllo in caso di errore) tramite la direttiva page:

```
<%@page errorPage="/jsp/error.jsp"
                    isErrorPage="false" %>
```

- Mentre un file JSP che DEVE gestire errori (error.jsp), conterrà la seguente direttiva:

```
<%@page  isErrorPage="true" %>
```

Vediamo alcuni esempi di jsp