Tecnologie di Sviluppo per il Web

Applicazioni Web J2EE:
Java Server Pages (JSP):
Parte 1

versione 3.2

G. Mecca – mecca@unibas.it – Università della Basilicata

Sommario

- Preliminari
  - ⇨Ciclo di Vita
- Struttura della Pagina
  - ⇨Blocchi di Istruzioni (“Scriptlet”)
  - ⇨Espressioni
  - ⇨Dichiarazioni
  - ⇨Direttive
- Corrispondenza con i Servlet

Preliminari

- Java Server Pages (JSP)
  - ⇨sintassi rapida per sviluppare Servlet
  - ⇨pagina di codice HTML misto a codice Java
  - ⇨standard arrivato alla versione 2.0
- API di riferimento
  - ⇨le API dei servlet
  - ⇨una API aggiuntiva: javax.servlet.jsp.*
  - ⇨ridefinisce alcune delle classi per i servlet

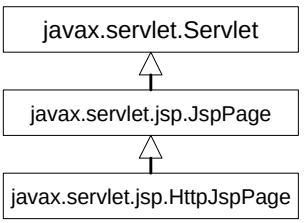
Ciclo di Vita

- Alla prima richiesta, il contenitore
  - ⇨converte la pagina JSP in un servlet
  - ⇨nella cartella %TOMCAT_HOME%\work
  - ⇨compila il servlet
  - ⇨eseguie il servlet secondo il ciclo di vita ordinario
- Successivamente, il contenitore
  - ⇨tiene traccia delle modifiche alla pagina JSP
  - ⇨rigenera, ricompila e ricarica il servlet se nec.

Ciclo di Vita

- Di conseguenza, a differenza dei servlet
  - ⇨non è necessario per l'utente compilare e ricompilare il codice nella pagina
  - ⇨non è necessario configurare il CLASSPATH
  - ⇨non è necessario ricaricare l'applicazione Web in caso di modifiche alle pagine JSP
- Attenzione
  - ⇨è comunque necessario ricompilare e ricaricare i componenti (bean) se cambiano

Ciclo di Vita



- Servlet generato da JSP
  - ⇨generato automaticamente dal contenitore
  - ⇨estende javax.servlet.jsp.HttpJspPage
- Attenzione
  - ⇨il servlet viene inserito in un package privato del contenitore

Ciclo di Vita

- Per Tomcat
  - ⇨il compiltore delle pagine JSP di chiama Jasper
  - ⇨dato <nomeFile>.jsp genera un servlet chiamato chiama <nomeFile>_jsp.java es: tentativo.jsp >> tentativo_jsp.java
  - ⇨nel package org.apache.jsp

Ciclo di Vita

- Metodi ereditati dal servlet generato
  - ⇨il metodo di servizio si chiama
    public void _jspService(HttpServletRequest request,
    HttpServletResponse response)
  - ⇨metodo di servizio unico: viene chiamato sia per le richieste get che per le richieste post
  - ⇨il metoto di inizializzazione si chiama jspInit()
  - ⇨il metodo di distruzione si chiama jspDestroy()

Struttura della Pagina

- Elementi principali
  - ⇨commenti
  - ⇨blocchi di codice HTML
  - ⇨blocchi di istruzioni Java (“scriptlet”)
  - ⇨espressioni Java
  - ⇨dichiarazioni Java
  - ⇨direttive Java

Struttura della Pagina

- Nota
  - ⇨una pagina JSP è in sostanza un servlet
  - ⇨descriviamo una sintassi alternativa per le operazioni fondamentali di un servlet, ovvero:
  - ⇨gestione della richiesta
  - ⇨gestione della risposta
  - ⇨gestione della sessione
  - ⇨gestione del contesto dell’applicazione
  - ⇨inoltro delle richieste

Struttura della Pagina

- Due tipi di commenti
  - ⇨commenti HTML <!-- *testo commento* -->
  - ⇨commenti JSP <%-- *testo commento* --%> non sono riportati nella risposta
- Blocchi di codice HTML
  - ⇨sequenze arbitrarie di codice HTML
  - ⇨i tag HTML vengono stampati identicamente nel corpo della risposta
  - ⇨generano out.print(*tag*) in _jspService()

Oggetti Predefiniti

- In una pagina JSP
  - ⇨all'interno degli "scriptlet" di codice Java è possibile utilizzare una serie di oggetti predefiniti
  - ⇨di cui sono disponibili i riferimenti
  - ⇨questi oggetti consentono di effettuare tutte le operazioni principali della pagina

Oggetti Predefiniti

- Riferimenti a oggetti predefiniti
  - ⇨riferimenti a oggetti preinizializzati e visibili negli scriptlet
  - ⇨HttpServletRequet request
  - ⇨HttpServletResponse response
  - ⇨HttpSession session
  - ⇨ServletContext application: rappresenta il contesto dell'applicazione Web

Oggetti Predefiniti

- Inoltre
  - ⇨JspWriter out: rappresenta il PrintWriter su cui stampare il corpo della risposta
  - ⇨JspWriter estende PrintWriter e fornisce funzionalità di tamponamento delle stampe
  - ⇨è possibile non rispettare l'ordine Codice di stato-Intestazioni-Corpo
  - ⇨il Content-Type predefinito è "text/html"

Oggetti Predefiniti

- Struttura implicita di _jspService()

```
public void _jspService(HttpServletRequest request,
                        HttpServletResponse response)
              throws java.io.IOException, ServletException {
    ServletContext application = getServletContext();
    HttpSession session = request.getSession();
    JspWriter out = response.getWriter();
    response.setContentType("text/html");
    ...
}
```


Oggetti Predefiniti

- Oggetto pageContext
  - ⇨classe javax.servlet.jsp.PageContext
  - ⇨un bean per accedere a tutti gli altri oggetti predefiniti e ad altri dati sulla pagina
  - ⇨metodi getRequest(), getResponse(), getOut(), getSession(), getServletContext()
  - ⇨ritornano un Object (è necessario il cast)
  - ⇨utile se ci sono altri metodi a cui è necessario passare vari di questi oggetti

Oggetti Predefiniti

- Nota
  - ⇨pageContext ha una sua mappa associata, analoga a quella di richiesta, sessione e applicazione
  - ⇨mappa degli attributi di pagina
  - ⇨la visibilità è limitata al codice della pagina JSP (è la mappa con il ciclo di vita più breve in assoluto)

Oggetti Predefiniti

- Attenzione
  - ⇨pageContext fornisce funzionalità sofisticate di gestione delle mappe di riferimenti
- Gestione delle mappe con pageContext
  - ⇨la classe javax.servlet.jsp.PageContext fornisce una serie di metodi unificati per l'utilizzo delle mappe di riferimenti (richiesta, sessione, applicazione e pagina)

Oggetti Predefiniti

- Ricerca di un riferimento
  - ⇨pageContext.findAttribute(“chiave”) cerca un riferimento nelle mappe di richiesta, sessione e applicazione
- Eliminazione di un riferimento
  - ⇨pageContext.removeAttribute(“chiave”) rimuove il riferimento da tutte le mappe in cui è contenuto

Oggetti Predefiniti

- Riassumendo
  - ⇨gli attributi si condividono tra servlet/jsp diversi utilizzando mappe di attributi
  - ⇨una associata a session
  - ⇨una associata ad application
  - ⇨una associata a request
  - ⇨una associata a pageContext
  - ⇨pageContext è in grado di manipolare attributi in tutte le mappe

Scriptlet

- Blocchi di istruzioni Java
  - ⇨sintassi: <% *istruzioni* %>
  - ⇨in sostanza compongono (con le stampe dei tag) il metodo _jspService() del servlet corrispondente
  - ⇨è possibile specificare qualsiasi operazione ammessa nel metodo doGet() o doPost() di un servlet

Scriptlet

- Il codice degli scriptlet
  - ⇨si aggiunge al metodo _jspService()
- Esempio:

```
<html>
<body>
<% String nome = request.getParameter(“nome”);
   session.setAttribute(“nome”, nome); %>
<p>Benvenuto, <% out.print(nome);%>. La data e’:
<% Date d= new java.util.Date();
   out.print(d); %> </p>
</body>
</html>
```

Scriptlet

```
public void _jspService(HttpServletRequest request,
                        HttpServletResponse response)
                  throws java.io.IOException, ServletException {
    ServletContext application=getServletContext();
    HttpSession session=request.getSession();
    JspWriter out = response.getWriter();
    response.setContentType("text/html");
    out.println("<html>");
    out.println("<body>");
    String nome = request.getParameter("nome");
    session.setAttribute("nome", nome);
    out.println("<p>Benvenuto,">; out.print(nome);
    out.println("La data di oggi e':");
    Date d= new java.util.Date();
    out.print(d); out.println("</p>");
    out.println("</body>");
    out.println("</html>");
}
```


Scriptlet

- E' possibile
  - ⇨immergere i tag nelle strutture di controllo
- Esempio:

```
<% String nome = request.getParameter("nome");
   if (nome==null) { %>
      <p>Benvenuto, immetti il tuo nome</p>
      <form action="..." method="...">...</form>
   <% } else { %>
      <p>Benvenuto, <% out.print(nome);%> </p>
   <% } %>
```

Scriptlet

- Traduzione in _jspService()

```
String nome = request.getParameter("nome");
if (nome == null) {
    out.println("<p>Benvenuto, immetti il tuo nome</p>");
    out.println("<form action=\"...\" method=\"...\">");
    ...
    out.println("</form>");
} else {
    out.println("<p>Benvenuto," + nome + "</p>");
}
```


Espressioni

- Forma rapida per la stampa di stringhe
  - ⇨sintassi: <%= *espressione* %>
  - ⇨esempio: <%= new java.util.Date() %>
  - ⇨semantica: il risultato dell'espressione convertito in stringa viene stampato su out
  - ⇨del tutto equivalente a
    <% out.print(*espressione.*toString()); %>
  - ⇨es: <% out.print((new java.util.Date()).toString()); %>

Espressioni

- In sostanza
  - ⇨il metodo print potrebbe essere sempre sostituito da espressioni
- Esempio

```
<html>
  <body>
  <% String nome = request.getParameter("nome");
     session.setAttribute("nome", nome); %>
  <p>Benvenuto, <%= nome %>. La data di oggi e':
     <%= new java.util.Date() %> </p>
  </body>
</html>
```


Dichiarazioni

- Finora
  - ⇨tutto il codice scritto nella pagina JSP viene tradotto nel metodo _jspService
  - ⇨ma in alcuni casi è necessario scrivere codice al di fuori di _jspService
- In particolare
  - ⇨metodo jspInit()
  - ⇨metodo jspDestroy()

Dichiarazioni

- Dichiarazioni
  - ⇨servono a questo scopo
  - ⇨producono codice al di fuori di _jspService()
  - ⇨proprietà della classe associata alla pagina
  - ⇨altri metodi
- Sintassi
  - ⇨<%! *istruzioni* %>

Dichiarazioni

- Attenzione
  - ⇨nelle dichiarazioni non sono visibili gli oggetti predefiniti (visibili solo in _jspService())
  - ⇨nel caso in cui nei metodi definiti sia necessario utilizzare i riferimenti predefiniti è necessario passarli attraverso i parametri
  - ⇨per esempio passando pageContext

Dichiarazioni

- Esempio:


```
<%!
private static String fileRecord = “record.txt”;

public void jspInit() {
 ServletContext application = getServletContext();
 String pathReale = application.getRealPath(fileRecord);
 Record record = new Record();
 record.setPath(pathReale);
 application.setAttribute("record",record);}

private int random() {
 return Math.abs(new Random().nextInt() % 100) + 1;
}
%>
```


Direttive

- Servono a definire altri aspetti del servlet
- Sintassi
  - ⇨<%@ *direttiva valori* %>
- Direttiva principale: page
  - ⇨<%@ page *attr1=“val1”, attr2=“val2”, … %>*
- Attributi principali
  - ⇨import
  - ⇨errorPage, isErrorPage

Direttive

- import
  - ⇨specifica i package e le classi da importare
  - ⇨può essercene più di una
  - ⇨normalmente all'inizio della pagina JSP
  - ⇨importazione automatica: javax.servlet.jsp.*
- Esempio

```
<%@ page import="it.unibas.indovina.*" %>
```

  - ⇨nel servlet corrispondente

```
import it.unibas.indovina.*;
```


Direttive

- errorPage
  - ⇨serve a specificare l'URI di una pagina (HTML o JSP) a cui reindirizzare il browser in caso di errori nell'esecuzione della pagina
  - ⇨valore: URI della pagina di errore
- Esempio

```
<%@ page errorPage="errore.jsp" %>
```

  l'impostazione è locale alla pagina

Direttive

- isErrorPage
  - ⇨serve a specificare che una pagina può essere utilizzata come pagina di errore
  - ⇨valore true o false
  - ⇨fornisce funzionalità aggiuntive, tra cui un riferimento predefinito aggiuntivo (exception)
- Esempio

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


Corrispondenza con i Servlet

- Riassumendo
  - ⇨la pagina JSP viene tradotta in un servlet dal contenitore
  - ⇨il servlet estende di javax.servlet.jsp.HttpJspPage
  - ⇨metodi fondamentali: _jspService(), jspInit(), jspDestroy()
  - ⇨il servlet viene ricompilato e re-inizializzato ogni volta che il file jsp viene modificato

Corrispondenza con i Servlet

- Costruzione del servlet
  - ⇨vengono importati i package standard e quelli specificati nella direttiva page (<%@ page import="..." %>)
  - ⇨vengono incluse le proprietà e i metodi definiti nelle dichiarazioni (<%! ... %>)
  - ⇨nel metodo _jspService() vengono inizializzati gli oggetti predefiniti
  - ⇨viene assegnato il Content-Type predefinito

Corrispondenza con i Servlet

- Costruzione del servlet (cont.)
  - ⇨i tag della pagina jsp danno origine a chiamate a out.print("*tag*");
  - ⇨il codice degli scriptlet (<% ... %>) viene incluso ordinatamente nel metodo _jspService()
  - ⇨le espressioni (<%= ... %>) danno origine a istruzioni del tipo out.print*(espressione);*

>>> indovinailnumeroM1 - JSPC
>> %TOMCAT_HOME%/work

Corrispondenza con i Servlet

>>> discussione codice - indovinailnumeroM1

- Nota
  - ⇨il programmatore naturalmente non modifica il codice del servlet generato
  - ⇨ma in alcuni casi può essere costretto a consultarlo
- Caso tipico
  - ⇨errore sintattico nel codice di una pagina JSP
  - ⇨l'errore viene segnalato con riferimento alle linee di codice del servlet corrispondente

Riassumendo

- Preliminari
  - ⇨Ciclo di Vita
- Struttura della Pagina
  - ⇨Blocchi di Istruzioni ("Scriptlet")
  - ⇨Espressioni
  - ⇨Dichiarazioni
  - ⇨Direttive
- Corrispondenza con i Servlet

Termini della Licenza