



Introduzione al linguaggio Java: Servlet e JSP

Corso di Gestione della Conoscenza
d'Impresa
A. A. 2006/2007
Dipartimento di Informatica
Università degli Studi di Bari

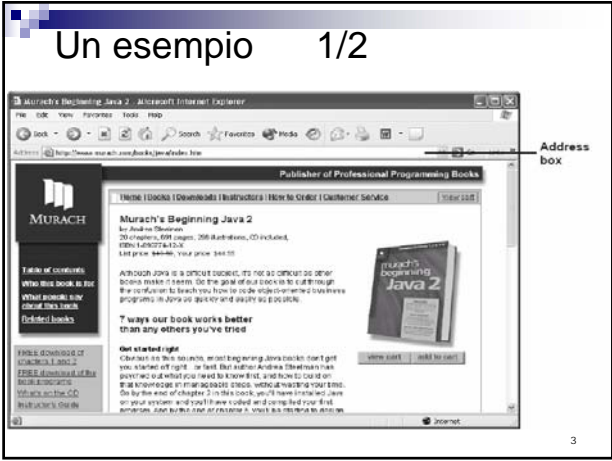
1

Servlet e JSP: il contesto

Un'applicazione web è un'applicazione client/server a cui è possibile accedere mediante un browser web. Essa è quindi caratterizzata da un insieme di pagine web generate in risposta alle richieste degli utenti. Esistono diversi tipi di applicazioni web: motori di ricerca, negozi online, aste online, siti giornalistici, etc...

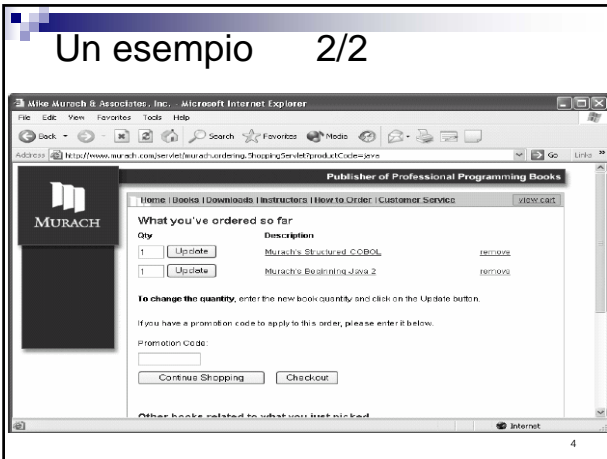
2

Un esempio 1/2



3

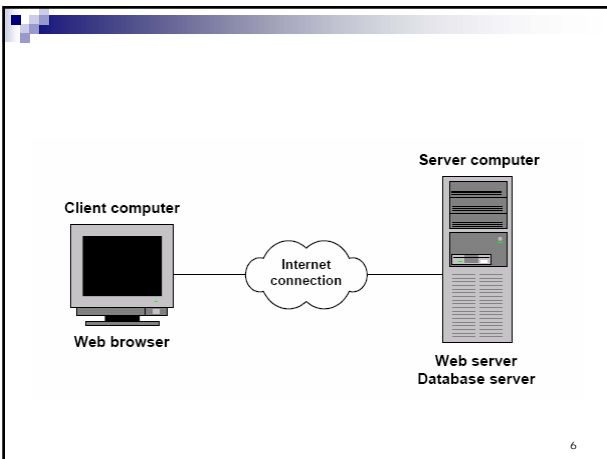
Un esempio 2/2



Le componenti di un'applicazione web

- In un'applicazione web le componenti risiedono sia sul *client* che sul *server*.
- Il browser utilizzato per accedere all'applicazione gira sul client; esso trasforma il codice HTML nell'interfaccia utente.
- L'applicazione vera e propria risiede sul server, qui risiede un *web server* responsabile dell'invio delle pagine web al browser.
- *Apache HTTP Server* della Apache Software Foundation è uno dei web server più diffusi per le applicazioni in Java.
- Molte applicazioni necessitano di dati presenti in database, per tale ragione sui server saranno spesso presenti anche dei *database management system* (or *DBMS*)

5



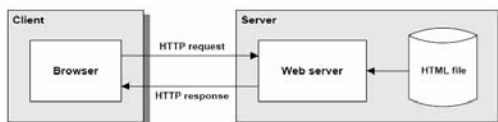
6

Pagine statiche e dinamiche

- Il linguaggio HTML (*Hypertext Markup Language*) viene utilizzato dal browser per produrre l'interfaccia utente.
- Alcune pagine web sono statiche, ossia non cambiano al variare delle richieste, altre invece sono dinamiche.
- In entrambi i casi, il browser usa un protocollo HTTP (*Hypertext Transfer Protocol*) per inviare una richiesta nota come *HTTP request* al server. Quando il web server riceve la richiesta dal browser esso soddisfa la richiesta e invia una risposta nota come *HTTP response*.

7

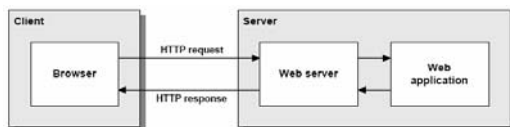
Pagine statiche



- Le pagine web statiche hanno nomi con estensione .htm o .html.
- Se si richiedono pagine web statiche la HTTP request include il nome del file html richiesto
- Il web server risponde con una HTTP response che include il file HTML

8

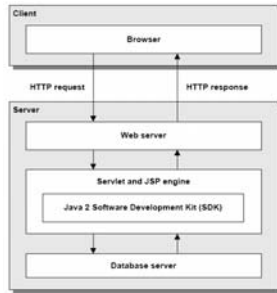
Pagine dinamiche



- Una pagina *web dinamica* è un file HTML generato da una web application. Spesso queste pagine cambiano in base a dei parametri che vengono inviati dal browser alla web application.
 - Quando un web server riceve la richiesta di una pagina web dinamica, il server inoltra la richiesta alla web application che genera una risposta sotto forma di file HTML.
- Il web server invia al browser il documento HTML generato in una HTTP response. Il browser ignora se il file HTML è stato generato dinamicamente o meno.

9

Java Web Application



10

Motore servlet/jsp

- In una web application Java è previsto un *motore servlet/JSP*, o *servlet/JSP container*, che consente al web server di lavorare con servlet and JSP.
- •Java 2 Platform, Enterprise Edition, o J2EE, specifica la modalità di interazione tra web server e motore servlet/JSP. Tomcat è uno dei più diffusi motori servlet/JSP. E' stato sviluppato nell'ambito del Jakarta project della Apache Software Foundation.
- Affinché un motore servlet/JSP funzioni esso deve accedere al Java Software Development Kit (SDK), che è parte del Java 2 Platform, Standard Edition (J2SE). SDK contiene classi Java, il compilatore Java, e il Java Runtime Environment (JRE).

11

Servlet

- Le Servlet sono classi Java eseguite lato server per esaudire le richieste provenienti dai web server. Esse non sono legate ad un particolare protocollo per la comunicazione tra client e server, anche se più comunemente si utilizza il protocollo HTTP ed infatti si parla di HTTPServlet.
- Per scrivere una Servlet si fa uso delle classi del package javax.servlet che è il framework di base per la scrittura delle servlet e del package javax.servlet.http che è l'estensione del framework di base, per la comunicazione via http. Utilizzando un linguaggio portabile come Java, le Servlet consentono di realizzare soluzioni, per estendere le funzionalità dei server web, indipendenti dal sistema operativo su cui esse vengono eseguite.

12

A cosa servono le servlet

Le Servlet vengono più comunemente utilizzate per:

- elaborare e salvare dati provenienti da form HTML;
- provvedere alla creazione di pagine HTML dinamiche, ad esempio utilizzando i dati contenuti in un database;
- gestire informazioni con stato, ad esempio la gestione di un sistema di commercio elettronico, dove un certo numero di clienti fa degli acquisti contemporaneamente e dove occorre quindi mantenere in modo corretto le informazioni sui clienti e sui loro acquisti (il classico carrello della spesa);

13

Struttura delle servlet

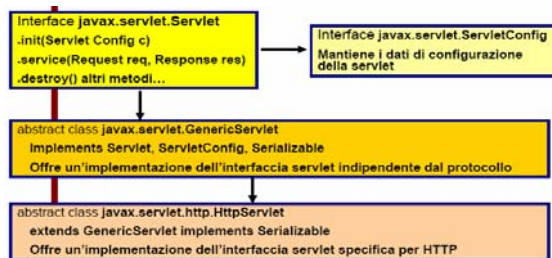
Ogni servlet deve implementare l'interfaccia `javax.servlet.Servlet`

Si estende una delle due classi speciali

- `javax.servlet.GenericServlet` (per servlet indipendenti dal protocollo)
- `javax.servlet.http.HttpServlet` (per HTTP servlet)

14

Gerarchia delle classi



15

Ciclo di vita di una servlet

Ogni servlet attraversa 3 fasi principali:

1. Costruzione ed inizializzazione (metodo *init()*);
2. Servizio di una o più richieste (metodo *service()*);
3. Chiusura e distruzione (metodo *destroy()*).

16

Metodo *init()*

- Effettua le operazioni necessarie per l'inizializzazione della nuova servlet e per il suo corretto funzionamento
- Deriva dalla classe `GenericServlet`
- Viene invocato quando la servlet viene caricata per la prima volta
- Termina prima di qualsiasi altra chiamata fatta alla servlet

17

Metodo *service()*

Dopo l'inizializzazione, la servlet attende una richiesta da parte del client.

Il metodo *service()* realizza la comunicazione bidirezionale con i client per mezzo dei suoi due parametri:

- L'oggetto `HttpServletRequest` che contiene i dati inviati dal client.
- L'oggetto `HttpServletResponse` che rappresenta la risposta della servlet al client.

18

Metodo *destroy()*

- Viene chiamato per consentire alla servlet di eseguire il rilascio di tutte le risorse (es. file aperti, connessioni al database, ecc.) prima che la servlet venga distrutta
- La distruzione di una servlet ed il relativo scaricamento (unloading) dalla memoria avviene solo quando tutte le chiamate a `service()` da parte dei client sono state eseguite, o allo scadere di un timeout specificato

19

Esempio di servlet

```
package slides;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class SimpleHttpServlet extends HttpServlet {
    protected void service(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        out.println("<HTML><TITLE>SimpleHttpServlet</TITLE><BODY>");
        out.println("<H2>ServletAPI Example - SimpleHttpServlet </H2><HR>");
        out.println("<H4>This is about as simple a servlet as it gets</H4>");
        out.println("</BODY></HTML>");
        out.close();
    }
}
```

20

Java Server Pages

- JSP (Java Server Pages) è una tecnologia semplice ma potente, che permette di creare pagine HTML dinamiche lato server.
- Questa tecnologia agli occhi del programmatore viene gestita per mezzo di un linguaggio di script che è in grado di mescolare codice HTML, componenti riusabili (JavaBeans), applicazioni remote (Servlet), codice Java e script Java-like.
- Le pagine JSP sono un'estensione diretta delle Servlet Java e offrono, rispetto ad altre attuali tecnologie Server-Side, il vantaggio e la possibilità di separare la sezione di gestione delle operazioni e di produzione dei contenuti, da quello di visualizzazione vera e propria.

21

Struttura di una jsp

Normalmente una pagina JSP ha una struttura molto simile a quella di un documento HTML. Essa si compone di:

- porzioni di codice HTML, JavaScript, CSS ecc... , e comunemente definiti **blocchi di codice statico**;
- porzioni di codice Java, compilati dal motore Jsp, che prendono il nome di **blocchi di codice dinamico**.

22

Come lavorano le jsp 1

- La prima volta che un Client (di norma un generico browser) effettua la richiesta di una pagina JSP ad un Web Server, questo attiva la compilazione creando un oggetto Servlet che risiederà in memoria; solo dopo questo passaggio l'output viene inviato al Client che potrà interpretarlo come se fosse una semplice pagina HTML.

23

Come lavorano le jsp 2

- Ad ogni richiesta successiva della medesima pagina JSP, il web server controlla se è stata modificata: in caso negativo richiama la Servlet già compilata, altrimenti si occupa di attivare nuovamente la compilazione e memorizzazione della nuova Servlet. Conseguenza di questo meccanismo è che la prima volta che si invoca una pagina JSP il sistema sarà un po' più lento del normale, mentre dalle invocazioni successive si avranno prestazioni di tutto rispetto.

24

Un esempio di jsp 1

```
<head>
<title>Chapter 4 - Email List application</title>
</head>
<body>
<%
String firstName = request.getParameter("firstName");
String lastName = request.getParameter("lastName");
String emailAddress = request.getParameter("emailAddress");
%>
<h1>Thanks for joining our email list</h1>
<p>Here is the information that you entered:</p>
<table cellpadding="5" cellspacing="5" border="1">
<tr>
<td align="right">First name:</td>
<td>%< firstName %</td>
</tr>
<tr>
<td align="right">Last name:</td>
<td>%< lastName %</td>
</tr>
<tr>
<td align="right">Email address:</td>
<td>%< emailAddress %</td>
</tr>
</table>
```

25

Un esempio di jsp 2



26

Componenti di una jsp

- Una Pagina JSP può essere composta da:
- *directive*;
- *script JSP*;
- *oggetti impliciti*;
- *azioni*.

27

Direttive

- Le *direttive* sono elementi JSP che forniscono informazioni globali su un'intera pagina JSP. Un esempio potrebbe essere una direttiva che indichi il linguaggio da utilizzare nella compilazione della Pagina JSP.
- La sintassi delle direttive è la seguente `<%@ direttiva attributo="valore"%>`.
- Volendo indicare che il linguaggio da utilizzare nelle Pagine JSP deve essere Java, è possibile utilizzare la riga di codice seguente:
`<%@ page language="java"%>`.

28

Tipi di direttive

- La **Direttiva Page** definisce informazioni globali riguardanti l'intera Pagina JSP che la contiene. Le impostazioni che modifica influenzano direttamente la compilazione della pagina.
- La **Direttiva Include** serve per inserire testo e codice all'interno di una Pagina JSP al momento della sua traduzione. La sua sintassi è la seguente:
`<%@include file="specificoURL"%>`.
- La **Direttiva Taglib** permette di creare un insieme personalizzato di tag chiamato *tag library*. Questa direttiva dichiara che la pagina intende utilizzare tag personalizzati, indica univocamente la libreria che li definisce e associa loro un prefisso che li distingue da quelli presenti in altre librerie.

29

Script jsp

- Gli script JSP consentono di includere porzioni di codice direttamente all'interno di pagine HTML.
- Esistono tre elementi coinvolti negli script JSP, ciascuno dei quali ha una posizione ben definita all'interno della Servlet generato: Dichiarazioni, Espressioni e Scriptlet.

30

Le Dichiarazioni Jsp

Le **Dichiarazioni** JSP servono per dichiarare variabili e metodi del linguaggio di script usato nelle Pagine JSP.

La sintassi delle dichiarazioni JSP è la seguente: `<#! Dichiarazione %>`

Ad esempio indicheremo:

- `<#! String nome=new String("Andrea"); %>` nel caso in cui si voglia dichiarare una variabile di tipo String contenente un valore iniziale diverso da null;
- `<#! public String getNome() { return nome; } %>` nel caso in cui si voglia definire un metodo che ritorni una variabile, definita in questo caso nella dichiarazione precedente, di tipo String.

31

Le espressioni JSP

■ Le **Espressioni** JSP sono degli elementi del linguaggio di script che vengono valutati logicamente, letteralmente o matematicamente, il cui risultato viene convertito in una `java.lang.String`.

■ Le espressioni sono valutate all'atto della richiesta HTTP. La stringa risultante viene inserita al posto dell'espressione nel file .jsp, nel caso l'espressione non può essere convertita, si verifica un errore di traduzione.

■ La sintassi di un'espressione JSP è la seguente: `<%= Espressione %>`, un esempio `Ciao <%= getName() %>`.

32

Gli Scriptlet JSP

■ Gli **Scriptlet** JSP possono contenere qualsiasi tipo di istruzione a patto che siano valide per il linguaggio specificato nella direttiva `language`. Vengono eseguiti al momento della richiesta e possono far uso di dichiarazioni, espressioni e JavaBeans.

■ La sintassi degli scriptlet JSP è la seguente: `<% Sorgente dello Scriptlet %>`.

■ Il codice contenuto tra i tag `<% %>` viene inserito nella Servlet al momento della compilazione

33

Oggetti Impliciti 1

- Scrivendo Pagine JSP si ha l'accesso ad alcuni oggetti impliciti appositamente creati per essere utilizzati nei documenti JSP senza dover essere dichiarati anticipatamente.
- Gli oggetti impliciti più importanti sono: request, response, pageContext, session, application, out, config, page.

34

Oggetti Impliciti 2

- L'oggetto **request** permette il recupero dell'informazioni legate alla richiesta commissionata dal browser dell'utente come i parametri, le intestazioni, i cookie e così via.
- L'oggetto **response** permette l'accesso al canale di output, per la generazione della risposta HTTP; è possibile comunicare un errore, inviare una particolare intestazione, settare un cookie e altro.
- L'oggetto **pageContext** è molto generico e rappresenta qualsiasi cosa abbia a che fare con il contesto di esecuzione della pagina; da esso è possibile ricavare gli altri oggetti impliciti.

35

Oggetti Impliciti 3

- L'oggetto **session** permette di associare all'utente che richiede la pagina un'informazione recuperabile in ogni altro documento dell'applicazione. E' possibile memorizzare le scelte effettuate dall'utente in una o più variabili di sessione così da poterne tener traccia.
- L'oggetto **application** fornisce un accesso al contesto di esecuzione dell'applicazione Web, cioè offre la possibilità di ottenere informazioni sul Servlet Engine che esegue l'applicazione, recuperare i parametri di inizializzazione del software, registrare dei messaggi accessibili da ogni pagina appartenente al contesto.
- L'oggetto **out** è di tipo *JspWriter*; ad esso corrisponde il canale di output instaurato con il browser dell'utente che ha richiesto la pagina.

36

Oggetti Impliciti 4

- L'oggetto **page** rappresenta la parola chiave `this` dell'istanza corrente della pagina JSP.

37

Azioni

- Le *azioni* rappresentano un livello di astrazione che può essere utilizzato per incapsulare agevolmente compiti comuni. Generalmente vengono impiegate per la creazione o la manipolazione di oggetti, perlopiù JavaBeans.

38

Esempi di azioni 1

- `<jsp:include>`: fornisce un modo per includere risorse aggiuntive, di tipo statico o dinamico, nella Pagina JSP corrente. Gli attributi più importanti legati a questa azione sono:
 - `page` : rappresenta la URL relativa della risorsa da includere;
 - `flush` : questo attributo contiene un valore booleano che indica se il buffer debba o meno essere svuotato.

39

Esempi di azioni 2

- `<jsp:forward>`: consente al motore Jsp l'inoltro, all'atto dell'esecuzione, della risorsa corrente a una risorsa statica, a un Servlet o ad una Pagina JSP. L'unico attributo possibile per questa azione è:
 - `page` : rappresenta la URL relativa all'oggetto da inoltrare.
- `<jsp:param>`: viene impiegata per fornire informazioni a coppie di tag/valore. Gli attributi sono:
 - `name` : rappresenta il nome del parametro referenziato;

40

Servlet o JSP?

Sia le servlet che le jsp possono essere impiegate per la produzione di contenuti web dinamici.

Come e quando conviene utilizzarle?

Anche se usate per gli stessi scopi servlet e jsp presentano sostanziali differenze che suggeriscono un uso ottimale di entrambe.

41

Servlet e JSP

Le servlets sono classi Java per cui andrebbero usate per le elaborazioni richieste da un'applicazione.

Le JSP sono prevalentemente formate da codice HTML per cui ha senso utilizzarle per disegnare le pagine web dinamiche. Come combinarle?

Una buona soluzione affida alle servlet le elaborazioni java necessarie per ciascuna pagina, e alle JSP la semplice visualizzazione .

In tal modo, nelle JSP è ridotta al minimo la presenza di codice Java, col grande vantaggio di sollevare i web designer dalla necessità di interpellare programmatori Java, mentre questi ultimi possono sviluppare Servlet senza preoccuparsi dell'HTML.

42
