Tecnologie di Sviluppo per il Web

Applicazioni Web J2EE:
Java Server Pages (JSP)
Parte 2

versione 3.2

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

Sommario

- Tag di Azione
- Tag di Inoltro e Inclusione
- Tag per l'Utilizzo dei JavaBeans
- Expression Language
- Standard Tag Library

Tag di Azione

- Negli esempi visti finora
  - ⇨applicazioni con archit. base-Modello 1
- Caratteristiche del codice
  - ⇨jsp come modo rapido per scrivere servlet
  - ⇨le pagine jsp implementano vista e controllo
  - ⇨le pagine jsp contengono necessariamente blocchi di codice java e tag HTML assieme

Tag di Azione

- Ma...
  - ⇨l'esperienza dice che le pagine JSP con molto codice Java diventano presto difficili da manutenere
- "Buone pratiche" suggerite dalla Sun
  - ⇨limitare al minimo l'utilizzo di codice Java nelle pagine JSP
  - ⇨a questo scopo la sintassi fornisce speciali tag per effettuare alcune operazioni ricorrenti

Tag di Azione

- La vista ha bisogno di eseguire istruzioni
  - ⇨esempio: accesso ai bean forniti dal controllo per la produzione degli schermi
  - ⇨come fare per tenere il codice separato dai tag ?
- Soluzione: tag di azione
  - ⇨le pagine jsp forniscono tag speciali la cui funzione è eseguire istruzioni

Tag di Azione

- Tag di Azione
  - ⇨tag predefiniti
  - ⇨tag definiti dall'utente (>>)
- Tag di azione predefiniti
  - ⇨due funzioni principali
  - ⇨inoltro delle richieste e inclusione delle risposte
  - ⇨utilizzo dei JavaBeans

Tag di Azione

- Nota
  - ⇨i tag di azione devono stare tra i tag HTML
  - ⇨fuori da scriptlet, espressioni, dichiarazioni e direttive
  - ⇨in sintesi, si tratta di un modo per eseguire operazioni ricorrenti senza dover programmare in Java

Tag di Inoltro e Inclusione

- Inoltro delle richieste
  - ⇨tag <jsp:forward page="uri" />
  - ⇨es: <jsp:forward page="fine.jsp" />
  - ⇨semantica: il tag viene tradotto nel servlet con il seguente blocco di codice Java

```
out.clear();
RequestDispatcher rd = application.getRequestDispatcher("/fine.jsp");
rd.forward(request, response);
return;
```

Tag di Inoltro e Inclusione

- Inclusione delle risposte
  - ⇨tag <jsp:include page="uri" />
  - ⇨es: <jsp:include page="intestazione.jsp" />
  - ⇨semantica: il tag viene tradotto nel servlet con il seguente blocco di codice Java

```
RequestDispatcher rd =
application.getRequestDispatcher("/intestazione.jsp");
rd.include(request, response);
```


Tag di Inoltro e Inclusione

- Un modo alternativo per effettuare l'inoltro
  - ⇨pageContext.forward(*uri*);
  - ⇨es: pageContext.forward("/errore.jsp");
- Comodo
  - ⇨per effettuare gli inoltri negli scriptlet piuttosto che scrivere tutto il codice sul RequestDispatcher

Tag di Inoltro e Inclusione

- Nota
  - è possibile anche specificare parametri da aggiungere alla query string associata alla richiesta
  - tag <jsp:param name="*nomepar*" value="*valore*" />
  - la coppia nomepar=valore viene aggiunta alla query string della richiesta inoltrata o inclusa

Tag di Inoltro e Inclusione

>> aciM1

- Un utilizzo tipico delle inclusioni
  - produrre un file di intestazione (es: intestazione.jsp) unico per tutte le pagine con prologo XHTML, head, foglio di stile
- Problema
  - come fare in modo da variare il titolo da pagina a pagina ?
  - il titolo viene specificato attraverso un parametro aggiuntivo per la include

Tag per l'Utilizzo dei JavaBeans

- Tre tag di azione per manipolare beans
  - ⇨jsp:useBean: consente di istanziare un bean e/o salvarlo come attributo in una delle mappe condivise
  - ⇨jsp:getProperty: consente di prelevare il valore di una proprietà del bean
  - ⇨jsp:setProperty: consente di impostare il valore di una o più proprietà del bean

Tag per l'Utilizzo dei JavaBeans

- jsp:useBean
  - ⇨varie funzioni
  - ⇨può essere usato per istanziare un bean
  - ⇨può essere usato per salvare un bean istanziato in una delle mappe condivise (request, session, application, pageContext)
  - ⇨può essere usato per recuperare un bean precedentemente istanziato da una delle mappe condivise

Tag per l'Utilizzo dei JavaBeans

- Sintassi
  - ⇨<jsp:useBean id="*idBean*" [class="*classe*"] [type="type"] [scope=*"mappa*"] />
  - ⇨l'attributo scope può assumere come valori session | application | request | page
  - ⇨è opzionale; il valore implicito è page
  - ⇨la semantica è diversa a seconda che venga specificato class oppure type

Tag per l'Utilizzo dei JavaBeans

- Semantica se viene specificato "class"
  - ⇨il valore di class deve essere una classe concreta
  - ⇨viene cercato un attributo con il nome uguale all'id specificato nella mappa corrispondente al valore dell'attributo scope
  - ⇨se c'è, viene restituito il valore, dopo aver fatto il cast sulla classe specificata (può esserci una ClassCastException)
  - ⇨altrimenti viene creata una nuova istanza del bean e salvata come attributo della mappa

Tag per l'Utilizzo dei JavaBeans

- Esempio:
```
<jsp:useBean id="auto"
             class="it.unibas.acim1.Automobile"
             scope="session" />
```
- Semantica
  - ⇨verifica se nella mappa associata alla sessione c'è una proprietà chiamata auto
  - ⇨se c'è, prova il cast su acim1.Automobile e la associa al riferimento auto
  - ⇨se non c'è la crea, la associa al riferimento auto e la salva nella mappa della sessione

Tag per l'Utilizzo dei JavaBeans

- Attenzione all'id del bean
  - ⇨viene utilizzato con due diverse funzioni
  - ⇨da una parte come chiave per recuperare il riferimento dalla mappa
  - ⇨dall'altra, come nome della variabile riferimento con cui successivamente manipolare il bean

Tag per l'Utilizzo dei JavaBeans

- Nota
  - ⇨il bean deve avere il costruttore standard (vuoto e senza argomenti)
  - ⇨non c'è modo di sapere se il bean è stato istanziato e salvato oppure solo prelevato dalla mappa
  - ⇨quando è necessario saperlo è necessario scrivere esplicitamente la chiamata la metodo getAttribute() oppure usare "type"

Tag per l'Utilizzo dei JavaBeans

- Semantica se viene specificato "type"
  - ⇨il valore di type può essere una classe concreta, astratta o un'interfaccia
  - ⇨viene cercato l'attributo nella mappa specificata attraverso l'attributo scope
  - ⇨se c'è, viene restituito il valore, dopo aver fatto il cast sul tipo specificato (può esserci una ClassCastException)
  - ⇨in questo caso può essere creata una nuova istanza del bean e si genera un'eccezione

Tag per l'Utilizzo dei JavaBeans

- Esempio:
```
<jsp:useBean id="auto"
             type="it.unibas.acim1.Automobile"
             scope="session" />
```
- Semantica
  - ⇨verifica se nella mappa associata alla sessione c'è una proprietà chiamata auto
  - ⇨se c'è, prova il cast su acim1.Automobile e la associa al riferimento auto
  - ⇨se non c'è non può crearla e viene sollevata un'eccezione

Tag per l'Utilizzo dei JavaBeans

- Differenza tra class e type
  - ⇨class serve tipicamente a creare un bean specificando una implementazione
  - ⇨non dovrebbe essere usato per recuperare bean che esistono già; per questo c'è type
  - ⇨type serve esclusivamente a recuperare un bean già salvato utilizzandone l'interfaccia e non l'implementazione

Tag per l'Utilizzo dei JavaBeans

- Un ulteriore esempio:

```
<jsp:useBean id="listaAuto"
             type="java.util.List"
             scope="session" />
```

- Semantica
  - ⇨in questo caso supponiamo che nella mappa della sessione sia stato salvato un riferimento di tipo ArrayList o LinkedList
  - ⇨lo preleviamo e lo manipoliamo come riferimento di tipo List

Tag per l'Utilizzo dei JavaBeans

- jsp:getProperty
  - ⇨serve a prelevare il valore di una proprietà, a convertirla in stringa e a stamparla nella risposta
- Sintassi
  - ⇨<jsp:getProperty name="*idBean*" property="*nomeProprieta*" />
  - ⇨esempio: **<jsp:getProperty name="auto" property="cilindrata" />**

Tag per l'Utilizzo dei JavaBeans

- Semantica
  - esegue il metodo get per accedere la proprietà, e poi esegue una out.print
  - esempio: `<jsp:getProperty name="auto" property="cilindrata" />`

    equivale a: `<%= auto.getCilindrata() %>`
  - è necessario utilizzare il nome di una proprietà esistente
  - se la proprietà è null restituisce la stringa vuota

Tag per l'Utilizzo dei JavaBeans

- jsp:setProperty
  - serve a modificare il valore di una proprietà del bean
- Varie forme per specificare il valore
  - si può fornire un valore costante
  - si può fornire un'espressione Java
  - si può prelevare il valore di uno o di tutti i parametri della richiesta che hanno nomi uguali alle proprietà del bean

Tag per l'Utilizzo dei JavaBeans

- Prima forma
  - <jsp:setProperty name="id*Bean*"
    property="*nomeProprieta*"
    value="*valoreCostante*" />
  - esempio:
```
<jsp:setProperty name="auto"
property="cilindrata"
value="1800" />
```
  - equivale a: `<% auto.setCilindrata(1800); %>`
  - attenzione: avviene una conversione automatica del valore passato

Tag per l'Utilizzo dei JavaBeans

- In particolare
  - il contenitore riceve come parametro una stringa
  - se il tipo della proprietà è diverso da string, applica automaticamente la routine standard di conversione da stringa al valore del tipo
  - es: Integer.parseInt()
  - se il valore fornito è scorretto può essere generata un'eccezione

Tag per l'Utilizzo dei JavaBeans

- Seconda forma
  - ⇨<jsp:setProperty name="*idBean*" property="*nomeProprieta*" value="<%= *espressione* %>" />
  - ⇨esempio:

```
<jsp:setProperty name="auto"
                 property="cilindrata"
   value="<%= request.getParameter("cilindrata")%>" />
```

  - ⇨semantica equivalente alla precedente

Tag per l'Utilizzo dei JavaBeans

- Terza forma
  - ⇨<jsp:setProperty name="*idBean*" property="*nomeProprieta*" />
  - ⇨esempio: `<jsp:setProperty name="auto" property="cilindrata" />`
  - ⇨serve ad associare alla proprietà del bean il valore dell'eventuale parametro fornito con la richiesta con lo stesso nome

Tag per l'Utilizzo dei JavaBeans

- Semantica
  - ⇨viene cercata nella mappa dei parametri associata alla richiesta un parametro con lo stesso nome della proprietà
  - ⇨se il parametro c'è, viene prelevato il valore e si tenta di assegnarlo alla proprietà (con eventuale conversione e possibile eccezione)
  - ⇨se il parametro non c'è, non viene effettuata nessuna operazione

Tag per l'Utilizzo dei JavaBeans

- In sostanza
  - ⇨assume una corrispondenza tra form HTML (attraverso cui si acquisiscono i dati) e bean (in cui vengono mantenuti)
  - ⇨si assume che l'utente utilizzi nelle form controlli con lo stesso nome delle proprietà dei bean
  - ⇨in questo caso la corrispondenza è automatica

Tag per l'Utilizzo dei JavaBeans

- Terza forma, alternativa
  - ⇨<jsp:setProperty name="*idBean*" property="* " />
  - ⇨esempio: `<jsp:setProperty name="auto" property="*" />`
  - ⇨assegna i valori di TUTTI i parametri della richiesta alle proprietà del bean con lo stesso nome, secondo la semantica descritta prima
  - ⇨attenzione anche in questo caso ai valori scorretti forniti dall'utente

Expression Language

- A partire dalla specifica JSP 2.0
  - ⇨il meccanismo dei tag di azione per l'utilizzo dei JavaBeans è stato ulteriormente sviluppato
- Idea
  - ⇨semplificare al massimo la manipolazione dei beans negli schermi
  - ⇨rendendo la sintassi snella e molto semplice

Expression Language

- Espression language (EL)
  - ⇨sintassi per l'accesso ai bean presenti nelle mappe condivise all'interno delle pagine JSP
  - ⇨consente di costruire espressioni molto compatte per accedere agli attributi dei bean
- Nota
  - ⇨è disponibile solo sui contenitori conformi a JSP 2.0 (es: Tomcat 5.x e non Tomcat 4.x)
  - ⇨il deployment descriptor dell'applicazione deve essere conforme allo schema 2.4

Expression Language

- Sintassi
  - ⇨${*espressione*}
- Esempi
  - ⇨${partita}
  - ⇨${partita.tentativo}
  - ⇨${automobile.proprietario.codiceFiscale}
  - ⇨${proprietario.listaAutomobili[0].targa}
  - ⇨${numeroGiocatori + 1}

Expression Language

- Elementi dell'espressione
  - un riferimento iniziale, scelto tra una serie di riferimenti predefiniti
  - una serie di identificatori
  - le parentesi quadre per l'accesso alle collezioni
  - una serie di operatori (aritmetici, booleani, di confronto ecc.)

Expression Language

- Riferimenti predefiniti
  - rappresentano il punto di partenza per la ricerca dei bean
  - requestScope: la mappa della richiesta
  - sessionScope: la mappa della sessione
  - applicationScope: la mappa dell'applicazione
  - param, paramValues: la query string
  - pageContext: valore predefinito (si può omettere)

Expression Language

- Identificatori
  - ⇨possono rappresentare id dei bean da prelevare dalle mappe
  - ⇨oppure nomi di proprietà dei bean prelevati attraverso cui “navigare” per raggiungere il valore cercato
- Esempi
  - ⇨${sessionScope.utente}
  - ⇨${partita.tentativo}

Expression Language

- Semantica
  - ⇨viene identificato il riferimento iniziale (se manca è pageContext)
  - ⇨viene identificato l’id del bean (il primo identificatore)
  - ⇨viene cercato un bean con l’id specificato nella mappa corrispondente al riferimento iniziale

Expression Language

- Semantica (continua)
  - per ciascun identificatore di proprietà xyz, viene eseguito sul riferimento ottenuto il metodo getXyz()
  - per ciascun identificatore di collezione del tipo xyz[chiave], viene eseguito il metodo getXyz() e se il risultato è una collezione – lista, array o mappa – viene eseguito get(chiave)

Expression Language

- Durante il processo
  - il contenitore effettua una serie di "coercizioni", ovvero cerca di forzare i tipi effettuando conversioni automatiche in modo da concludere la valutazione dell'espressione
  - es: downcast da Object alle classi dei bean
  - al termine dell'esecuzione il risultato ottenuto viene stampato nella risposta sotto forma di stringa

Expression Language

- Esempio
  - ⇨${sessionScope.utente.nome}
- Equivalente a

```
<%@ page import="it.unibas.aci.modello.Utente" %>
<% Utente utente =
        (Utente)session.getAttribute("utente"); %>
<%= utente.getNome() %>
```


Expression Language

- Esempio
  - ⇨${automobile.proprietario.nome}
- Equivalente a

```
<%@ page import="it.unibas.aci.mod.Proprietario" %>
<%@ page import="it.unibas.aci.mod.Automobile" %>
<% Automobile automobile =
        (Automobile)pageContext.findAttribute("automobile");
   Proprietario proprietario = automobile.getProprietario(); %>
<%= proprietario.geNome() %>
```

Expression Language

- Esempio
  - ⇨${proprietario.listaAutomobili[0].targa}
- Equivalente a

```
<%@ page import="it.unibas.aci.modello.*" %>
<%@ page import="java.util.ArrayList" %>
<% Proprietario proprietario =
        (Proprietario)pageContext.findAttribute("proprietario");
   ArrayList listaAutomobili = proprietario.getListaAutomobili();
   Automobile automobile = (Automobile)listaAutomobili.get(0);
%>
<%= automobile.getTarga() %>
```


Expression Language

- Attenzione
  - ⇨come tutte le novità sintattiche, per ragioni di compatibilità è possibile disabilitare l'expression language (${} è riservato)
- Nel deployment descriptor

```
<jsp-property-group>
   <url-pattern>*.jsp</url-pattern>
   <el-ignored>true</el-ignored>
</jsp-property-group>
```

Expression Language

- Viceversa
  - ⇨è possibile disabilitare scriptlet, dichiarazioni ed espressioni e utilizzare solo l'expression language
- Nel deployment descriptor

```
<jsp-property-group>
   <url-pattern>*.jsp</url-pattern>
   <scripting-invalid>true</scripting-invalid>
</jsp-property-group>
```


Standard Tag Library

- Utilizzando l'expression language
  - ⇨si riduce drasticamente il codice necessario per l'accesso ai javabeans
- Ma...
  - ⇨restano alcuni casi in cui è comunque necessario utilizzare scriptlet
  - ⇨esempio: iterazione su una collezione per stampare gli elementi

Standard Tag Library

- La soluzione
  - ⇨utilizzare una libreria di tag personalizzati
  - ⇨che si aggiungono ai tag di azione predefiniti
  - ⇨e implementano le operazioni ricorrenti
- Obiettivo
  - ⇨eliminare del tutto gli scriptlet di codice Java dalle pagine JSP

Standard Tag Library

- Libreria di tag personalizzati
  - ⇨package di classi, una classe per ciascun tag
  - ⇨un descrittore della libreria
- Tag library descriptor
  - ⇨file con estensione .tld che associa il nome del tag alla corrispondente classe
- Per utilizzare la libreria
  - ⇨direttiva “taglib” nelle pagine Jsp
  - ⇨specifica URI della libreria e prefisso dei tag

Standard Tag Library

- JSTL ("JSP Standard Tag Library")
  - ⇨la Libreria Standard di Tag JSP Definita nel Java Community Process
  - ⇨implementazione di riferimento nel progetto "taglibs" di jakarta.apache.org
  - ⇨numerosissimi tag; es: gestione della sessione, gestione dell'applicazione, if, cicli, bean, date, espressioni regolari ecc.

Standard Tag Library

- Struttura della libreria
  - ⇨varie categorie

Categoria	URI	Pref	Esempio
Core	http://java.sun.com/jsp/jstl/core	c	<c:tag ...>
XML proc.	http://java.sun.com/jsp/jstl/xml	x	<x:tag ...>
I18N	http://java.sun.com/jsp/jstl/fmt	fmt	<fmt:tag ...>
Database	http://java.sun.com/jsp/jstl/sql	sql	<sql:tag ...>

Standard Tag Library

- Installazione
  - ⇨scaricare lo zip dal sito jakarta.apache.org
  - ⇨decomprimere lo zip
- Per utilizzare la libreria
  - ⇨sono necessari i due jar forniti jstl.jar e standard.jar nella cartella lib dell'applicazione
  - ⇨contengono anche i file .tld

Standard Tag Library

- Nelle pagine JSP
  - ⇨è necessario utilizzare la direttiva taglib relativa alla categoria o alle categorie necessarie
- Esempio
  - ⇨<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
  - ⇨<%@ taglib uri="http://java.sun.com/jsp/jstl/xml" prefix="x" %>

Standard Tag Library

- Esempi di tag della libreria core

```
<c:forEach items="collezione" var="variabile">
    <operazioni>
</c:forEach>
<c:if test="condizione">
    <operazioni>
</c:if>
```

- Nota
  - ⇨ la libreria è basata sull'expression language

Un Esempio

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<html>
<head> <title>JSTL</title> </head>
<body>
<h1>Automobili FIAT</h1>
<c:forEach var="automobile"
            items="${proprietario.listaAutomobili}">
    <c:if test="${automobile.targa == 'FIAT'}">
        ${automobile} <br />
    </c:if>
</c:forEach>
</body>
</html>
```

Standard Tag Library

- Per un elenco completo dei tag
  - ⇨è necessario consultare lo standard prodotto dallo JCP relativo a JSTL
  - ⇨http://java.sun.com/products/jsp/jstl

Riassumendo

- Tag di Azione
- Tag di Inoltro e Inclusione
- Tag per l'Utilizzo dei JavaBeans
- Expression Language
- Standard Tag Library

Termini della Licenza