Università degli Studi di Bologna
Facoltà di Ingegneria

Tecnologie Web L-A
A.A. 2009 – 2010

Esercitazione 5
JSP

Tutor:
Ing. Pasini Samuele *samuele.pasini@unibo.it*

Agenda

- Teoria e pratica
  - fondamentali
- Importazione e modifica di un progetto di esempio
  - class-path a tempo di compilazione ed esecuzione
  - deployment ed esecuzione
  - descrittore *web.xml*
  - interazione con l'applicazione
  - servlet e mantenimento dello stato
- Per approfondire
  - ulteriori esempi

Pagine JSP: primo impatto

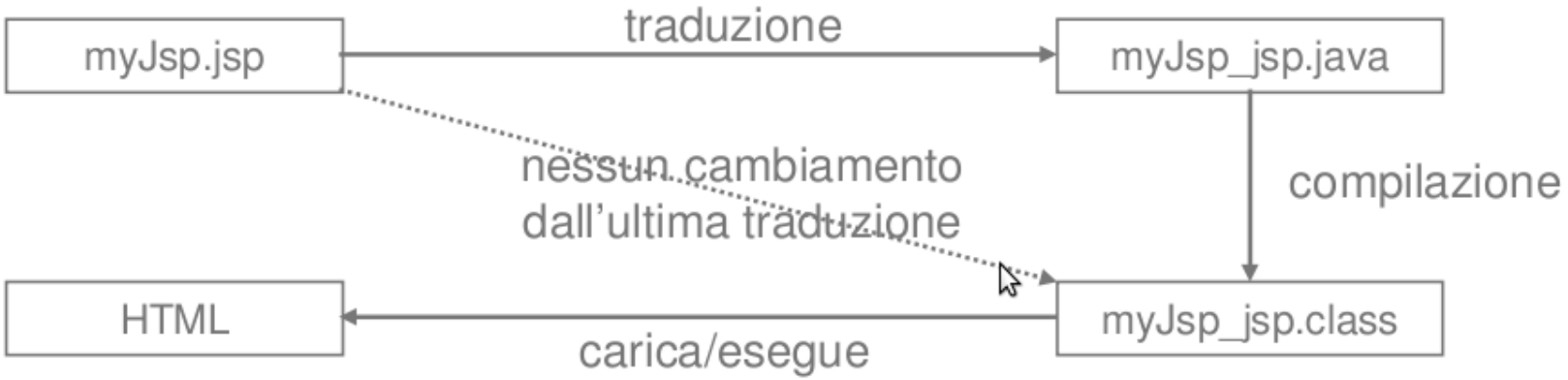
- Pagine HTML con estensione *.jsp* che includono codice Java
  - Trasformate dal Servlet Container in classi Java che estendono javax.http.HttpServlet

    (in Tomcat ciò avviene attraverso una particolare Servlet mappata sulle risorse *.jsp, detta JspServlet)
  - Oggetto di *request dispatching*, dopo la compilazione ed istanziazione

- Attraverso l'esecuzione di codice Java, il Web server permette di ottenere
  - contenuto HTML generato dinamicamente
  - side-effect quali esecuzione di logica di business complessa, scritture su database, ecc...

- L'insieme dei blocchi di codice Java all'interno di una pagina JSP deve costituire un insieme di istruzioni ben formato
  - possibile apertura e chiusura di parentesi graffe in blocchi distinti di codice Java, separati da codice HTML
  - effetto *simile* a quello ottenibile attraverso un linguaggio di scripting interpretato...
  - ...in realtà istruzioni compilate lato server, prima della loro esecuzione.

Ciclo di vita e costrutti principali



- **Direttive** <%@ ... %> *oppure* <jsp:directive.*name attribute* />
  - proprietà generali della pagina, importazione di nomi di classe, uso della sessione, ecc...
  - processate a tempo di compilazione della JSP in Servlet
- **Espressioni** <%= ... %> *oppure* <jsp:expression> *java expression* </jsp:expression>
  - trasposizione del risultato della valutazione di espressioni Java direttamente nel codice HTML prodotto dalla pagina
  - *n.b.:* permettono la valutazione di espressioni (che restituiscono un risultato!), *non* di istruzioni (quindi niente ';' finale)
- **Scriptlet** <% ... %> *oppure* <jsp:scriptlet> *java instructions* </jsp:scriptlet>
  - codice Java la cui valutazione procede parallelamente con l'elaborazione del contenuto della pagina JSP al fine di produrre l'HTML risultato...
  - ...ma la cui compilazione avviene ben prima di questo momento (in caso di fallimento, non è possibile mostrare alcun risultato parziale!)
- **Dichiarazioni** <%! ... %> *oppure* <jsp:declaration> *java definitions* </jsp:declaration>
  - definizione di variabli e metodi che potranno poi essere usati all'interno di *scriptlet* ed *espressioni*

Ulteriori costrutti (1)

- **Azioni** <jsp:*nomeAzione attributiAzione … /*>

  permettono di effettuare operazioni a tempo di esecuzione della richiesta
  - useBean: istanzia un oggetto conforme alle convenzioni JavaBean e lo rende disponibile al codice che segue tramite un preciso identificativo e un preciso scope di validità
  - getProperty: ritorna in forma di oggetto la property indicata
  - setProperty: imposta il valore della propery indicata
  - include: include a request time (non a compile time, come le direttive) il contenuto di un file nel sorgente della JSP valutato dal server
  - forward: cede la gestione della richiesta a un'altra risorsa
  - plugin: genera il contenuto necessario per scaricare un plug-in Java

- **Oggetti 'embedded'** o **'built-in'**

  risorse immediatamente dichiarabili nel codice della pagina JSP
  - page: proprietà e caratteristiche della vista corrente
  - out: flusso di output su cui riversare l'HTML risultato
  - request: richiesta HTTP ricevuta, suoi attributi e parametri
  - reponse: risposta HTTP da produrre e sue proprietà
  - sessione: stato mantenuto lato server dell'utente associato alla richiesta corrente
  - ...

Ulteriori costrutti (2)

- **Tag libraries**

  Funzionalità aggiuntive rese disponibili sotto forma di librerie di tag

  - utilizzabili all'interno della pagina a seguito della direttiva

    <%@ taglib uri="mnemonico dichiarato dalla tag library"

    prefix="prefisso usato nella pagine per marcare i tag di tale libreria" %>
  - usate per richiamare...
    - …frammenti di HTML piu' complessi
    - …funzionalità fornite da librerie Java (es: JSTL)
    - …funzionalità fornite da propri componenti Java

- **Expression language (EL)**

  Valutazione di espressioni racchiuse all'interno dei caratteri

  ${ … }
  - accesso ai JavaBean dichiarati nella pagina
  - risoluzione delle proprietà di un oggetto attraverso la notazione puntata

    es: ${ miobean.miocampo }
  - risultato…
    - …sostituito direttamente nell'HTML (per la scrittura del contenuto finale della pagina)
    - …assegnato attributi dei tag di una taglib (per la valutazione da parte del processore JSP)

      es: <c:if test="${ miobean.miocampo > 0}"> .... </c:if>

Per cominciare

- All'interno dell'archivio ZIP dell'esercitazione, nel direttorio ***progetti***
  - il file ***05_TemplateJSP.zip*** contiene lo scheletro di un semplice progetto di esempio basato sull'uso di pagine JSP
  - creato con Eclipse, contiene già tutti i descrittori necessari a essere riconosciuto e configurato correttamente

- Importare il progetto come visto nelle precedenti esercitazioni, senza esploderne l'archivio su file system (lo farà Eclipse)
  - *File → Import → General → Existing Projects into Workspace → Next → Select archive file*

- Attraverso i target ANT, compilare, pacchettizzare e pubblicare sul server l'applicazione Web 'AS IS'...

- ...e avviare Tomcat !

index.jsp

- Accedendo al contesto radice...

  http://localhost:8080/TemplateJSP

  ...il Servlet Container seleziona automaticamente la risorsa corrispondente alla pagina *index.jsp* per servicre la richiesta

- A differenza della scorsa esercitazione, tuttavia...
  - ...la pagina JSP presenta un messaggio di attesa...
  - ...ma comanda una redirezione non al browser dell'utente, ma al proprio Servlet Container: prima di produrre e restituire alcuna risposta all'utente!
  - OVVIAMENTE QUESTO NON E' L'EFFETTO DESIDERATO!

- Seguite il gioco dei forward
  - data la complessità che un'applicazione Web può assumere, capita spesso di suddividere la logica necessaria a servire una richiesta su piu' componenti (es: filtri per aprire e chiudere transazioni, servlet per accedere al database, pagine JSP per produrre la vista di risposta, …)
  - si migliora la manutenibilità
  - si evita di replicare parti di codice comuni a piu' operazioni

index.jsp

- Accedendo al contesto radice...

  http://localhost:8080/TemplateJSP

  ...il Servlet Container seleziona automaticamente la risorsa corrispondente alla pagina *index.jsp,* per servire la richiesta

- A differenza della scorsa esercitazione, tuttavia...
  - ...la pagina JSP presenta un messaggio di attesa...
  - ...ma comanda una redirezione non al browser dell'utente, ma al proprio Servlet Container: prima di produrre e restituire alcuna risposta all'utente!
  - OVVIAMENTE QUESTO NON E' L'EFFETTO DESIDERATO!

Il gioco dei fordward

- Seguite il gioco dei forward
  - data la complessità che un'applicazione Web può assumere, capita spesso di suddividere la logica necessaria a servire una richiesta su piu' componenti (es: filtri per aprire e chiudere transazioni, servlet per accedere al database, pagine JSP per produrre la vista di risposta, …)
  - si migliora la manutenibilità
  - si evita di replicare parti di codice comuni a piu' operazioni

- Qui però occorre sistemare...
  - ...il redirect iniziale, affinché avvenga tramite un ordine dato al browser, non al server
  - ...il mapping della classe Servlet a cui viene inoltrata la gestione della richiesta
  - ...il passaggio del parametro di inizializzazione richiesto da tale servlet

- Dopodiché occorre ri-eseguire il deploy (ANT!)
  - le modifiche ai descrittori XML (*web.xml* nel nostro caso) e al codice Java necessitano che il progetto venga ripacchettizzato e ripubblicato sul server
  - attendere che Tomcat riconosca la modifica e comandila ripartenza dell'applicazione

pages/home.jsp

- Se siete arrivati fin qui...
  - esplorate liberamente la struttura della pagina!
    - nell'IDE, per capire come è stato generato l'HTML finale
    - con Firebug, per capire come è strutturato l'HTML finale

- Highlights
  - parti comuni a tutte le altre pagine incluse mediante frammenti JSP esterni
  - ogni pagina è in grado di modificare il colore di sfondo della “tab” ad essa corrispondente mediante l'analisi dell'URL richiesto
  - i soliti giochini con I CSS
  - ...

- Benvenuti nel negozio online!

pages/catalogue.jsp

- Pagina per gestire il catalogo degli articoli in vendita
  - realizzato per mezzo di un bean con scope di applicazione
  - gli articoli e le corrispondenti quantità disponibili sono concetti “unici”, uguali per tutti gli utenti del negozio
- Direttive
  - errori, sessione, bean utilizzati, import di classi Java, uso di una tag library
- Dichiarazioni
  - metodi richiamati nel seguito, per aggiungere/rimuovere oggetti
- HTML, sriptlet, espressioni ed EL
  - analisi dei parametri della richiesta per decidere cosa fare
  - layout a due colonne (tramite attributo float)
    - inserimento di nuovi articoli
    - visualizzazione del contenuto attuale del catalogo (uso di tag JSTL per ciclare)
  - ogni richiesta per *pages/catalogue.jsp* causa una nuova visualizzazione della pagina
- Prendetevi tutto il tempo per analizzare il funzionamento (domande?)

pages/cart.jsp

- Pagina per gestire il carrello degli articoli scelti dall'utente/cliente
  - tale selezione è diversa da cliente a cliente
  - servirà quindi un bean con scope di sessione!

- Ricalcando la struttura di *pages/catalogue.jsp,* riuscite a realizzarla voi?
  - sulla sinistra: tramite JSTL ciclate sugli articoli nel catalogo
  - per ogni riga, introducende un comando per mandare una richiesta di aggiunta al carrello
  - nella pagina, analizzate la i parametri della richiesta per capire come gestirla
  - incapsulate i metodi di utilità dentro le dichiarazioni
  - sulla destra: mostrate il contenuto corrente del carrello (ogni nuova richiesta determina l'aggiornamento della pagina)

- Nella soluzione, oltre a queste funzionalità, un esempio di come dichiarare ed utilizzare *taglib* personalizzate
  - scrittura dell'HTML di una riga del carrello
  - calcolo del totale da pagare

pages/checkout.jsp

- Pagina per concludere l'ordine
  - decrementare le quantità nel catalogo
  - salvare la selezione dell'utente

dove?

  - nel database!

- La prossima volta...

Per approfondire...

- *Tomcat* fornisce out-of-the-box alcuni esempi relativi all'utilizzo delle JSP API, utili come riferimento
  - visionabili a partire da *http://localhost:8080/jsp-examples*
  - funzionamento e estratti del codice sorgente
  - il codice sorgente completo è visibile nei file .JSP su file system