Programmazione per il Web
Riassunto della lezione del 09/03/2016

Igor Melatti

Pagine Web Dinamiche: altri esempi

- esempio `slide140`: fare in modo che la pagina con le recommendations contenga un link che rimandi nuovamente alla pagina iniziale
- ispezionare i cookie creati usando le funzionalità del browser
- slide 148-149: manca `setName`, perché tanto lo si setta tramite il costruttore di `Cookie`, e cambiargli il nome dopo aver fatto il retrieve o la `addCookie` non ha senso. Comunque, il metodo esiste.
- sfruttando 148-149, come si può fare in modo che l'esempio `slide140` ordini i cookie secondo la sequenza temporale di scelta nella prima pagina?
- creare una servlet uguale a `slide140`, ma facendo sì che la prima interazione sia con il metodo GET, e la seconda con il metodo POST
- creare una servlet uguale a `slide140`, ma facendo sì che l'informazione scritta nella pagina delle recommendations consista in libri dal titolo possibilmente diverso. Creare 2 versioni: in una il titolo del libro è nel cookie, in un'altra no.
- far sì che, se si usa l'indirizzo `localhost:8080/20160307`, venga visualizzata una lista cliccabile di tutte le web application viste tra lezione 5 e lezione 6
- prendere tutti gli esempi visti finora, e metterli ognuno in una directory diversa di `webapps` (ovvero, creare 5 context root diverse)
- realizzare l'esempio di slide 60 con una servlet
- riconsiderare l'esempio `pag044_complex_form.html` dato nella lezione 2, e le relative modifiche (da considerare tutte insieme) proposte come esercizio in lezione 2. Scrivere una servlet che gestisca l'invio delle informazioni. Tale servlet deve visualizzare una pagina HTML uguale a quella da ottenere nell'esercizio della lezione 2, con i campi contenenti i valori appena

inviati nella form originale. In pratica, cliccando su "Submit Your Entries", dovrebbe vedersi la pagina trasformarsi con i vari elementi che si ridispongono come previsto dall'esercizio nella lezione 2. Cliccando nuovamente su submit (avendo eventualmente modificato nuovamente gli input della pagina), si deve ritornare alla pagina di partenza. A tal proposito, usare una sola servlet.

Seconde specifiche del progetto

- Il progetto consisterà di almeno 3 parti, ognuna delle quali avrà un punteggio
  - la prima parte, descritta qui di seguito, si basa su una configurazione via file XML (previsione di punteggio: 2/12)
  - la seconda parte si baserà su una configurazione via DBMS (previsione di punteggio: 6/12)
  - la terza parte si baserà sulla definizione di tag personalizzati (previsione di punteggio: 4/12)
- Per ora solo con servlet; verrà poi richiesto di realizzare alcune visualizzazioni con pagine JSP
- Le servlet dovranno essere compilabili con Java versione 1.6
- Rendere dinamiche tutte le pagine elencate nel paragrafo "Prime specifiche di progetto" della lezione 3
- Aggiungere delle pagine di amministrazione, accessibili solo agli utenti amministratori del sito
  - per gli utenti amministratori, occorre aggiungere dei link "Modifica" e "Cancella" sia sugli oggetti in vendita che sulle categorie
  - inoltre, occorre aggiungere un singolo link "Aggiungi" sia alla pagina che lista le categorie che a quella che lista gli oggetti di una categoria
  - un click su "Aggiungi" apre una pagina in cui si possono specificare i campi della categoria/oggetto, così come sono elencati nello schema
  - un click su "Modifica" apre una pagina in cui si possono specificare i campi della categoria/oggetto, così come sono elencati nello schema, con i default settati agli attuali valori
  - per "Aggiungi" e "Modifica" devono essere presenti anche dei bottoni che permettano di selezionare un'immagine
  - un click su "Cancella" apre una pagina in cui vengono riportati i dati dell'oggetto, o i dati sintetici della categoria (vedere sotto)

  - cancellare una categoria implica cancellare tutti i suoi oggetti, quindi occorre riportare questi oggetti in modo sintetico: una lista di link con id e titolo, cliccando sui quali si può vedere il dettaglio dell'item (su questa pagina deve essere presente il link "Torna indietro")
  - aggiungere una categoria vuol dire aggiungerla vuota
  - in una pagina separata, possono aggiungere/eliminare/modificare anche gli utenti, con le stesse regole elencate qui sopra (essenzialmente, per gli amministratori esiste una categoria in più, gli utenti)

- Più in particolare: supporre che i prodotti siano memorizzati in un file `market.xml`, che rispetti lo schema `market.xsd`
  - pertanto, questo file contiene una lista di account e di oggetti, organizzati in categorie
  - le modifiche apportate dagli amministratori si dovranno riflettere su questo file
  - supporre, in questo caso, che non ci siano problemi nell'accesso concorrente

- Le pagine che mostrano gli oggetti devono essere paginate con un valore iniziale pari a 10
  - quindi, se ci sono 25 oggetti, ci dovranno essere 3 pagine: la prima va da 1 a 10, l'ultima da 21 a 25
  - l'utente può cambiare il valore della paginazione (immettendo un numero e premendo su un opportuno pulsante)
  - le categorie sono mostrate sempre in una pagina sola
  - l'ordinamento delle categorie segue sempre il tag `id`
  - l'ordinamento degli oggetti è inizialmente sugli `id`; l'utente lo può cambiare a `Price`, `Available` o `Title`
  - un oggetto va mostrato solo se il tag `available` contiene una data anteriore a quella corrente del server *e* l'attributo `show` ha valore true
  - di un oggetto, occorre mostrare tutti gli elementi, nell'ordine descritto nello schema

- Nomi: alcuni saranno fissati
  - ci si abitui a mettere un nome ad ogni form (che ha l'attributo `name`)
  - quello che permette di selezionare una categoria si deve chiamare `selectCategory`
  - quello che permette di selezionare un oggetto si deve chiamare `selectItem`

  - quello che permette di selezionare un ordinamento si deve chiamare `selectOrder`
  - il valore che viene inviato dev'essere uno tra `id`, `Price`, `Available` o `Title`
  - quello che permette di selezionare un valore di paginazione si deve chiamare `selectPageSize`
  - quello che permette di fare login si deve chiamare `login`
  - l'URL iniziale dell'applicazione dev'essere `localhost:8080/ecommerce_2016_matricola/`

- Il progetto andrà consegnato come un file `.zip` contenente l'intero context root della Web application. Il nome del context root deve essere `ecommerce_2016_matricola`. Oltre a contenere le directory richieste per il normale funzionamento, occorre mettere in una directory `WEB-INF/source` tutti i sorgenti delle servlet.

- Per parsare il file XML (che si suppone rispettare lo schema dato) si può usare la libreria Java `Xerces` (scaricabile da `https://xerces.apache.org/xerces-j/`); vedere `http://totheriver.com/learn/xml/xmltutorial.html`