Tecnologie Web T
14 Gennaio 2015 – Compito

Tempo a disposizione: 3 ore

La soluzione comprende la **consegna elettronica** dei seguenti file mediante l'apposito applicativo Web **esamix** (**http://esamix.labx**):

Acquisti.zip	file zip contenente il sorgente java/class e pagine Web per punto 1
File.zip	file zip contenente il sorgente java/class e pagine Web per punto 2
Didattica.zip	file zip contenente il sorgente java/class per punto 3

Ogni file .zip consegnato DEVE CONTENERE TUTTI e SOLI i file creati/modificati e/o ritenuti importanti in generale ai fini della valutazione (ad esempio, descrittori, risorse statiche o dinamiche, codice Java e relativi .class, ecc.) e NON dell'intero progetto

N.B. Per superare la prova scritta di laboratorio ed essere ammessi all'orale, è necessario totalizzare almeno 18 punti (su un totale disponibile di 33), equamente distribuiti sui tre esercizi, ovvero almeno 6 punti sul primo esercizio, 6 punti sul secondo esercizio e 6 punti sul terzo esercizio

---

Studenti in debito di Tecnologie Web L-A
Viene richiesto lo svolgimento dei soli esercizi 1 (17 punti) e 2 (16 punti). Tempo a disposizione: 2 ore.
I 18 punti necessari per l'ammissione all'orale sono così distribuiti: almeno 10 punti sul primo esercizio e almeno 8 punti sul secondo

---

ESERCIZIO 1 (11 punti)
Si realizzi un'**applicazione Web** che permetta ad un utente di effettuare acquisti su un sito di e-commerce i cui prezzi siano dinamici (fissati a livello di singola sessione) e dipendenti dall'andamento delle vendite dei prodotti considerati, basata su **tecnologie Java servlet**, **JSP** e **Javascript**.
In particolare, l'applicazione dovrà avere una pagina iniziale **home** che richieda all'utente nome e cognome (considerati identificatori univoci) e che gli permetta di aprire una sessione di interazione. Dopo l'apertura della sessione, verrà visualizzata una lista di prodotti (tutti i prodotti memorizzati lato server in appositi JavaBeans) che include, per ogni prodotto, id unico, nome, descrizione testuale, numero di unità disponibili all'acquisto e prezzo. Il prezzo "personalizzato" è deciso server-side all'apertura della sessione e non varia per tutta la durata della sessione stessa. A questo punto l'utente potrà decidere di selezionare alcuni dei prodotti, specificandone il numero di unità, e di caricarli nel proprio carrello della spesa. Un'altra pagina permetterà di finalizzare l'acquisto e chiudere la sessione di interazione; solo l'azione di finalizzazione genererà l'aggiornamento server-side dei dati relativi a numero di unità disponibili e numero di unità vendute; nel caso di unità richieste non più disponibili, si segnali l'errore all'utente cancellando l'intero ordine di acquisto.
Riguardo alla determinazione del "prezzo di sessione" da utilizzare, il valore personalizzato deve essere determinato server-side sulla base delle vendite già finalizzate al momento dell'apertura della sessione corrispondente, secondo la seguente formula:
prezzo = (prezzo ultima sessione finalizzata) + (prezzo base) * (unità vendute + disponibili) / (unità disponibili)

Inoltre si faccia in modo che una sessione aperta per più di 10 minuti ma non finalizzata venga automaticamente resa invalida, costringendo l'utente a riaprirne una nuova e a ottenere un nuovo prezzo personalizzato.

ESERCIZIO 2 (11 punti)

Si realizzi un'**applicazione Web** per il processamento parallelo AJAX-based di un insieme di file di testo.

In particolare, l'applicazione dovrà avere una pagina iniziale **start.html** in cui l'utente possa inserire un numero x fra 1 e 10 e richiedere all'utente di specificare x nomi assoluti di file differenti (per semplicità, si supponga che tali file siano tutti esistenti nel file system locale e correttamente formattati); si controlli la conformità dell'input alle specifiche (x fra 1 e 10, nomi di file che cominciano per "C:", localmente al Web browser). Ogni file sarà costituito da semplice testo, con un solo numero intero contenuto in ogni riga e 50 righe per ogni file.

Terminato l'inserimento dell'ultimo nome di file, l'applicazione deve invocare x volte, tramite AJAX in modo asincrono e concorrente, la stessa servlet **addizione**, passando in ogni invocazione il contenuto di un file diverso tramite encoding JSON; la servlet si occuperà semplicemente di sommare i 50 numeri interi ricevuti e restituire il risultato, anche questo tramite encoding JSON.

Si commenti nel codice quale è il modello di esecuzione concorrente utilizzato e se possono sussistere problemi di interferenza fra le eventuali istanze multiple/singole della classe servlet messe in esecuzione concorrentemente dalle chiamate AJAX.

ESERCIZIO 3 (11 punti)

Con riferimento allo schema relazionale di seguito riportato relativo alla gestione di corsi universitari

```
Docente(MatricolaDocente, Cognome, Nome)
Corso(CodiceCorso, Nome, CreditiFormativi)
Didattica(Docente, Corso)
      FOREIGN KEY Docente REFERENCES Docente
      FOREIGN KEY Corso REFERENCES Corso
```

si realizzi l'**applicazione Java-JDBC "Didattica"** che esponga i **metodi CRUD per la gestione della persistenza** dei dati contenuti nel relativo database, unitamente ad una classe `main` di prova per gli stessi.

Nello specifico, **l'applicazione "Didattica"** deve provvedere:

- alla creazione delle tabelle dello schema relazionale sopra illustrato all'interno del proprio schema nel **DB TW_STUD** (esplicitando gli opportuni **vincoli di PK** e **FK**);
- all'implementazione dei metodi per il popolamento delle tabelle al punto precedente (passando i dati come parametri di input dei metodi stessi); nel dettaglio, si richiede (i) di inserire due o più docenti nella tabella "Docente" e due o più corsi nella tabella "Corsi"; (ii) di inserire alcune tuple relative alla didattica dei docenti nella tabella "Didattica", **previa verifica preventiva del rispetto dei vincoli di FK presenti nel database**. In caso di esito negativo (anche solo di uno dei due vincoli), il metodo deve rifiutare l'operazione di persistenza richiesta producendo un opportuno messaggio di errore sul file **didattica.txt**;
- alla realizzazione del metodo "**Max**" in grado di determinare *Nome e Cognome dei docenti che hanno tenuto il massimo numero di corsi da 6 crediti formativi* (stampare il risultato sul file **didattica.txt**).

N.B. L'implementazione deve limitarsi al solo **DBMS DB2**