Tecnologie Web T
3 Febbraio 2016 – Compito

Tempo a disposizione: 3 ore

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

Pooler.zip	file zip contenente il sorgente java/class e pagine Web per punto 1
Shopping.zip	file zip contenente il sorgente java/class e file xml/xsd per punto 2
Partite.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 usi tecnologie Java servlet, JSP e JSON, e che permetta di inserire credenziali utente, un nome logico (stringa *nome*) corrispondente al nome assoluto di un file locale e un numero naturale uguale a 1 o 2. Il contenuto del file dovrà essere stampato server-side sulla stampante virtuale *prn0* o su quella *prn1* (emulate tramite scritture su file diversi server-side), o in entrambe nel solo caso in cui il numero inserito sia uguale a 2 (numero di copie). L'applicazione deve inoltre essere in grado di fare l'accounting delle stampe, permettendo di visualizzare tutte le stampe già svolte in precedenza (nome file, lunghezza in byte e username corrispondente).
In particolare, l'applicazione dovrà essere costituita da una pagina Javascript **start** che controlli che *nome* non contenga caratteri proibiti (ovvero *, ? e simboli di punteggiatura) e che il numero inserito sia uguale a 1 o 2; superati i controlli, la pagina Javascript deve invocare una JSP **pooler.jsp** che sorteggia un numero a caso (0 o 1) e sulla base del risultato dell'estrazione comanda la stampa alla prima servlet (scrittura solo su *prn0*) o alla seconda servlet (scrittura solo su *prn1*), inviando il contenuto del file come parametro di ingresso (di una richiesta GET o POST?). Le due servlet devono restituire come risultato la storia delle operazioni precedenti (nomi file stampati e dimensione) per quell'utente; le risposte delle servlet devono necessariamente essere restituite alla pagina iniziale **start**.
Inoltre, l'applicazione deve includere una pagina **stats.jsp**, accessibile solo da un amministratore (username=admin; password=admin), che mostri quante operazioni di stampa e per quale ammontare complessivo di byte sono state svolte negli ultimi 30 minuti.

ESERCIZIO 2 (11 punti)

Si progetti una grammatica **XML Schema**, e un suo **documento XML** di esempio, per la descrizione delle informazioni di un **sito Web per lo shopping on-line di abbigliamento femminile**, nel rispetto delle seguenti specifiche:

- Ciascun documento XML modella una sessione di "visita" di un utente registrato al sito Web. Nel dettaglio, si tiene traccia di (i) informazioni utente quali email (di *tipo email* e obbligatorio) e password (di *tipo password* e obbligatorio) (ii) scelte dell'utente dal menù di navigazione; in particolare, le scelte rientrano nell'insieme {"*abiti*", "*camice*", "*giacche*", "*gonne*", "*pantaloni*", "*ultimi arrivi*"} e sono non esclusive. Si noti che è obbligatoria almeno una scelta di visita. Il tipo password è una stringa di 8 caratteri che deve contenere almeno un numero e un carattere speciale tra {?!*$}; il tipo email è una stringa formata da due sottostringhe separate dal carattere speciale "@" e in cui la seconda sottostringa contiene un "."
- Ogni scelta di visita, è modellata a sua volta come un insieme di "oggetti" complessi (da 1 a N), ognuno formato dalla tripla: *fotografia capo abbigliamento* (rappresentata dal nome file jpeg), *descrizione capo* (di tipo testuale) e *prezzo di vendita* (espresso in Euro). Tutti e tre i campi sono obbligatori. Durante la navigazione, l'utente può poi selezionare da 0 a N capi d'abbigliamento in base al suo interesse.

Si realizzi quindi l'**applicazione Java "Shopping"** che, facendo uso del **parser SAX**, esponga il metodo `getScelte()`, unitamente a suo un `main` di prova, in grado di restituire la lista di scelte effettuate dall'utente durante la sessione di navigazione, ognuna delle quali caratterizzata dagli oggetti selezionati dall'utente. Il tutto, a fine d'indagine di mercato e analisi di preferenze utente.

ESERCIZIO 3 (11 punti)

Con riferimento allo schema relazionale di seguito riportato relativo alla gestione di partite di calcio

```
Stadio(CodStadio, Nome, Citta)
Squadra(CodiceSquadra, Nome, Categoria, Girone)
Partita(SquadraCasa,SquadraOspite, Stadio, Data)
      FOREIGN KEY Stadio REFERENCES Stadio
      FOREIGN KEY SquadraCasa REFERENCES Squadra
      FOREIGN KEY SquadraOspite REFERENCES Squadra
```

si realizzi l'**applicazione Java-JDBC "Partite"** 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 "Partite"** 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 stadi nella tabella "Stadio" e due o più squadre nella tabella "Squadra"; (ii) di inserire alcune tuple relative alle partite giocate negli stadi nella tabella "Partita", **previa verifica preventiva** (a) del **rispetto dei vincoli di FK presenti nel database** (b) del fatto che **entrambe le squadre** (casa e ospite) **di una tupla della tabella "Partita" siano iscritte alla tessa categoria e allo stesso girone**. In caso di esito negativo (anche solo di uno dei vincoli sopra elencati), il metodo deve rifiutare l'operazione di persistenza richiesta producendo un opportuno messaggio di errore sul file **Partite.txt**.

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