Tecnologie Web T
9 Giugno 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**):

Conteggio.zip	file zip contenente il sorgente java/class e pagine Web per punto 1
Filter.zip	file zip contenente il sorgente java/class e pagine Web per punto 2
Ristorante.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 prevalentemente tecnologie Java servlet e JSP, per il conteggio di occorrenze di caratteri alfanumerici in file mantenuti server-side. L'applicazione deve consentire all'utente di inserire un nome relativo di file (stringa *file*) e un carattere alfanumerico (*car* – ammissibile solo carattere alfabetico o cifra numerica). Una servlet **dispatcher** deve occuparsi di ricevere la richiesta, verificare l'esistenza del file nella cartella di lavoro dell'applicazione e la correttezza di *car*; dipendentemente dalla lunghezza del file, dovrà poi delegare il compito di conteggio a [1, n] slave, ciascuno dei quali lavorerà su 1KB di contenuto. Al termine del conteggio, l'applicazione Web deve essere in grado di mostrare all'utente il numero totale delle occorrenze contate su *file* e i risultati analoghi ottenuti in precedenza per al massimo 3 richieste precedenti di quello stesso utente.
In particolare, l'applicazione dovrà essere costituita da una pagina JSP **start.jsp** che permetta l'inserimento dei dati di ingresso e che invochi la servlet **dispatcher**. **dispatcher** invocherà i suoi slave (si giustifichi nel sorgente la scelta di implementare gli slave come oggetti Java, Javabeans o altre servlet). Problemi in caso di accesso concorrente allo stesso file? Vantaggi/svantaggi nell'uso del modello di concorrenza deprecated per servlet? Al termine dei conteggi parziali, la stessa pagina **start.jsp** deve mostrare il risultato finale complessivo del conteggio avvenuto.
Inoltre, l'applicazione deve includere una pagina **admin.jsp**, accessibile solo da un amministratore (username=admin; password=admin), che mostri quante operazioni di conteggio e con quali risultati sono avvenute negli ultimi 30 minuti.

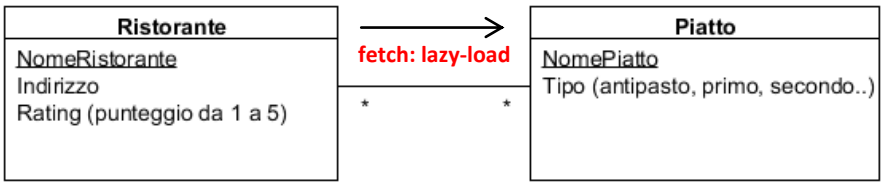
Tecnologie Web T
9 Giugno 2016 – Compito

ESERCIZIO 2 (11 punti)

Si realizzi un'applicazione Web, che usi prevalentemente tecnologie Javascript, AJAX, JSON e Java servlet, per il ritrovamento di occorrenze di parole (sequenze di caratteri terminate da uno spazio bianco) in un file *testo.txt* esistente e mantenuto server-side nella cartella di lavoro dell'applicazione Web stessa. L'applicazione Web deve consentire all'utente di inserire una, due o tre stringhe (*str1, str2, str3*); per ogni stringa inserita si dovrà effettuare una richiesta HTTP asincrona diversa per fare il triggering della operazione di ricerca lato servitore.
In particolare, l'applicazione dovrà essere costituita da una singola pagina **home.html** che permetta l'inserimento delle stringhe di ingresso in 3 campi di input (si controlli che siano costituite solo da caratteri alfabetici e che la loro lunghezza sia almeno di 3 caratteri e al massimo di 20). **home.html** deve poi invocare la servlet **filter** in modo asincrono per il conteggio delle occorrenze delle stringhe inserite, una volta per ogni stringa inserita, in modo asincrono e concorrente. La servlet deve occuparsi di cercare per righe la stringa ricevuta in ingresso nel file *testo.txt*. Problemi in caso di accesso concorrente da parte di più clienti? Vantaggi/svantaggi nell'uso del modello di concorrenza deprecated per servlet? Si risponda a tali domande tramite commenti nel file sorgente.
Infine, ogni invocazione della servlet dovrà restituire al cliente in formato JSON tutte le righe del file in cui è stata trovata almeno una occorrenza della stringa specificata e il numero di tali righe. Quali vantaggi/svantaggi eventuali rispetto alla restituzione delle righe tramite risposta in formato testuale?

ESERCIZIO 3 (11 punti)

Partendo dalla realtà illustrata nel **diagramma UML** di seguito riportato, si fornisca una soluzione alla gestione della persistenza basata su **Pattern DAO** in grado di "mappare" efficientemente e con uso di ID surrogati il modello di dominio rappresentato dai **JavaBean Ristorante e Piatto del diagramma UML** con le corrispondenti **tabelle relazionali derivate dalla progettazione logica del diagramma** stesso.



Nel dettaglio, dopo aver creato da applicazione Java gli **schemi delle tabelle** all'interno del proprio schema nel database **TW_STUD** di **DB2** (esplicitando tutti i **vincoli** opportuni), **implementato** i **JavaBean** e **realizzato le classi** relative al **Pattern DAO** per **l'accesso CRUD** alle tabelle**,** si richiede l'implementazione di **opportuni metodi per il supporto delle seguenti operazioni**:

- per ogni ristorante sito in Bologna, si richiede la lista dei piatti di tipo "primo" offerti nel rispettivo menù; si richiede quanti ristoranti, nella fascia di rating compresa tra 4 e 5, offrono come tipo di secondo piatto "seppie con i piselli".

Si crei poi un **main di prova** che: (i) inserisca due o piu' tuple nelle tabelle di interesse; (ii) faccia uso corretto dei metodi realizzati al punto precedente al fine di produrre una stampa del risultato sul file **ristorante.txt**.

N.B. L'implementazione del **Pattern DAO deve limitarsi** al solo **DBMS DB2**. La soluzione **deve sfruttare i mapping M-N specificati nell'UML e propendere per il caricamento indicato nello stesso**. Ogni ulteriore scelta da parte dello studente deve essere opportunamente giustificata con commenti nel codice.