Tecnologie Web T
11 Febbraio 2019 – Compito

Tempo a disposizione: 3 ore

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

Sessione.zip	file zip contenente il sorgente java/class e pagine Web per punto 1
Ping.zip	file zip contenente il sorgente java/class e pagine Web per punto 2
Intolleranza.zip	file zip contenente il sorgente java/class e txt 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 uno strumento di session management per una semplicissima applicazione Web di ricerca indirizzi, basato principalmente su tecnologia Java servlet e JSP.
L'applicazione Web di ricerca indirizzi accetterà sia utenti connessi dopo autenticazione (tramite username e password) sia senza autenticazione. Dovrà semplicemente consentire di inserire la stringa *cognome* (si faccia controllo locale che la stringa non contenga caratteri numerici e sia di lunghezza compresa fra 2 e 20) e restituire come risultato il set di indirizzi registrati con cognome uguale a quello inserito. I risultati restituiti saranno parte dello stato utente fino alla terminazione automatica della sua sessione di interazione dopo 5 minuti circa di inattività.
Inoltre lo strumento di session management deve consentire all'amministratore dell'applicazione (username=admin; password=admin), tramite una servlet, di visualizzare le statistiche relative alle sessioni correnti (ovvero ancora non terminate) e recenti (terminate non più di un giorno prima). Le statistiche devono includere: ora di inizio e di eventuale fine per ogni sessione; lista di risultati forniti in ogni sessione; numero totale di sessioni correnti di utenti non autenticati; numero totale di sessioni recenti di utenti non autenticati.
Inoltre, lo strumento di gestione deve permettere all'amministratore, sempre tramite una servlet, di salvare lo stato di una sessione corrente su memoria persistente lato servitore in formato JSON. Inoltre, si risponda, come commento al codice sorgente della servlet:

- se uno stesso utente (sia autenticato, sia no) cerca di aprire più sessioni correnti, che cosa succede? Problemi di concorrenza e corse critiche?
- se più amministratori cercano di effettuare concorrentemente salvataggi su memoria persistente verso lo stesso file, ci sono problemi di concorrenza e corse critiche?

Tecnologie Web T
11 Febbraio 2019 – Compito

ESERCIZIO 2 (11 punti)

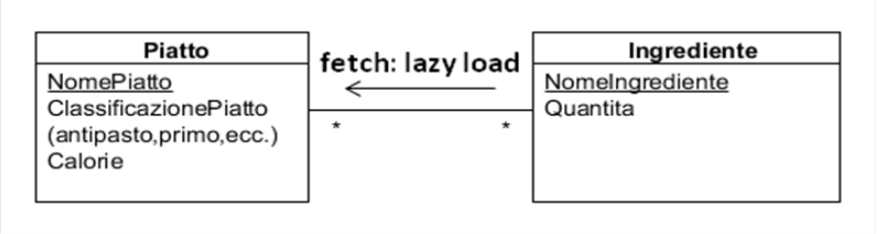
Si sviluppi una applicazione Web, basata su Javascript, AJAX e JSON, che realizzi un semplice ping verso un server Web attivo.

In particolare, l'applicazione Web deve consentire all'utente di inserire un numero naturale *n* fra 1 e 5, e una stringa *s* alfanumerica di lunghezza compresa fra 10 e 100 (si effettuino localmente gli opportuni controlli). Al termine dell'inserimento di una stringa *s* corretta, l'applicazione deve mandare concorrentemente *n* volte la stringa *s* verso il server Web, ottenendo come risultato il ritorno della medesima stringa *s* più un timestamp preso localmente al server al momento dell'invio della risposta. L'invio dei dati, sia dal cliente al servitore che dal servitore al cliente, deve essere effettuato in formato JSON.

Inoltre, per evitare sovraccarichi e rallentamenti, si faccia in modo che, se gli utenti con sessioni attive sono in numero superiore a 10, uno stesso utente non possa usare l'applicazione Web descritta sopra per più di 3 volte all'interno di una sessione.

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 Ricetta e Ingredienti 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 la realizzazione di **un metodo che a partire da un valore specifico di NomeIngrediente (es. *Aglio*), restituisce la lista dei nomi di piatto in cui tale ingrediente non è presente**.

Si crei poi un **main di prova** in grado di:

(i) inserire due o più tuple nelle tabelle di interesse;
(ii) fare uso corretto del metodo realizzato al punto precedente al fine di produrre la stampa del risultato sul file **Intolleranza.txt**.

N.B. L'implementazione del **Pattern DAO deve limitarsi** al solo **DBMS DB2**. La soluzione **deve sfruttare il mapping N-M specificato 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.