**Tecnologie Web T**
**13 Gennaio 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**):

| | |
|---|---|
| **Replicaz_file.zip** | file zip contenente il sorgente java/class e pagine Web per punto 1 |
| **Esp.zip** | file zip contenente il sorgente java/class e pagine Web 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 permetta di inserire un nome logico (stringa *nome*) e un contenuto testuale; il contenuto testuale dovrà essere salvato in file binario server-side, trasparentemente per il cliente, o solo nel file *dir1/nome*, o solo nel file *dir2/nome*, oppure in entrambi per motivi di replicazione e fault-tolerance; l'applicazione deve essere in grado di visualizzare tutte le richieste di salvataggio già svolte in precedenza per uno stesso utente e deve essere basata principalmente su tecnologie Java servlet e JSP.
In particolare, l'applicazione dovrà essere costituita da una pagina Javascript **home** che controlli che *nome* non contenga caratteri proibiti (ovvero *, ? e simboli di punteggiatura); superati i controlli, la pagina Javascript deve invocare una JSP **replicaz_file.jsp**; tale JSP deve sorteggiare un numero a caso (0, 1 o 2) e sulla base del risultato dell'estrazione deve passare il valore dei parametri rispettivamente alla prima servlet (scrittura solo su *dir1/nome*), solo alla seconda servlet (scrittura solo su *dir2/nome*), o a entrambe. Le due servlet devono restituire una risposta sintetica di successo/fallimento per l'ultima operazione di salvataggio file e la storia delle operazioni precedenti (nomi file salvati e successo/fallimento) per quell'utente; le risposte delle servlet devono passare necessariamente attraverso la JSP di dispatching descritta sopra prima di essere restituite al cliente.
Inoltre, l'applicazione deve includere una pagina **statistiche.jsp**, accessibile solo da un amministratore (username=admin; password=admin), che mostri quante operazioni di salvataggio e per quale ammontare complessivo di byte sono state svolte negli ultimi 30 minuti.

**Tecnologie Web T**
**13 Gennaio 2016 – Compito**

**ESERCIZIO 2 (11 punti)**

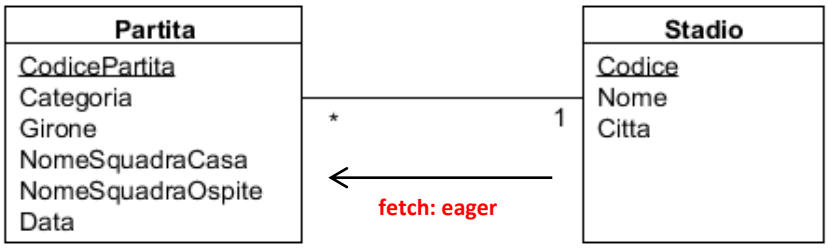
Si realizzi un'applicazione Web che permetta di calcolare in modo efficiente l'elevamento a potenza naturale di un numero reale positivo, basata principalmente su tecnologie Java servlet, Javascript e AJAX.

In particolare, l'applicazione dovrà essere costituita da una pagina Javascript **start** che controlli che i due operandi inseriti siano *x* un numero reale positivo e *y* un numero naturale. Una volta effettuati i controlli, **start** deve invocare in modo concorrente, tramite richieste AJAX parallele, la stessa servlet **esp** due volte, una volta con parametri di ingresso *x* e *trunc(y/2)* e l'altra con parametri di ingresso *x* e *y-trunc(y/2)*. Le due "istanze di invocazione" della servlet **esp** si devono occupare di calcolare il risultato del loro elevamento a potenza senza coordinamento e in modo indipendente; i loro due risultati saranno restituiti a **start**; **start** dovrà infine visualizzare "In attesa di risultato" fino a che entrambi i risultati parziali non sono ricevuti, poi il risultato complessivo derivante dal prodotto dei due risultati parziali.

Il candidato illustri tramite commento nel file sorgente quale modello di esecuzione concorrente delle servlet è il più appropriato per questa applicazione Web, motivi tale scelta e usi tale modello, eventualmente anche se deprecato, nella soluzione proposta.

**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 il modello di dominio rappresentato dai **JavaBean** *Partita* e *Stadio* con le corrispondenti **tabelle relazionali derivate dalla progettazione logica del diagramma** dato.




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 permetta di ottenere:

- Dato uno stadio, il numero totale di partite in esso giocate raggruppate per categoria.

Infine, si crei un semplice *main* di prova che:

- inserisca due o più partite per ogni stadio coinvolto nel campionato;
- faccia uso corretto del metodo realizzato al punto precedente al fine produrre una stampa (opportunamente formattata) del risultato sul file **Partite.txt**.

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