Tecnologie Web T
08 Giugno 2012 – Compito

Tempo a disposizione: 3 ore

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

Biblioteca.zip	file zip contenente il sorgente java dell'applicazione al punto 1
IMU.zip	file zip contenente il sorgente java dell'applicazione al punto 2
Party.zip	file zip contenente il sorgente java dell'applicazione al 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 7 punti sul primo esercizio, 5 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 (18 punti) e 2 (15 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 (12 punti)
Si deve realizzare una applicazione Web basata su sola tecnologia Java servlet (senza utilizzare Java Server Pages) per la gestione delle richieste di libri da parte degli utenti di un sistema di biblioteche.

L'applicazione Web deve partire da una pagina **richiestaLibri.html** che permetta all'utente di inviare richieste di prelievo di libri. Ogni richiesta deve contenere il codice fiscale dell'utente utilizzato come identificatore unico, il titolo del libro (per semplicità si supponga che non esistano libri diversi con stesso titolo), il cognome dell'autore, data di prelievo desiderata e data di riconsegna desiderata (in formato numerico ggmm); uno stesso utente non può effettuare più prelievi dello stesso libro ma ovviamente può inviare più richieste relative a libri differenti. La pagina **richiestaLibri.html**, tramite opportuni campi, usa la servlet r**ichiesta** per inserire una richiesta di prelievo alla volta (servlet deve verificare che non esistano richieste duplicate per lo stesso libro). Le richieste di prelievo sono momentaneamente associate alla sessione dell'utente; diventano definitive solo quando l'utente preme esplicitamente il pulsante **finalizza** sulla pagina **richiestaLibri.html**; richieste "definitive" devono permanere sistema finché l'applicazione Web rimane attiva (non necessario salvataggio su DBMS).

Inoltre la pagina iniziale deve contenere un link alla pagina **cambiaDate.html** che consenta a un utente di aggiornare le date di prelievo/riconsegna delle proprie richieste di libro. L'identificazione dell'utente deve sfruttare il codice fiscale inserito precedentemente in **richiestaLibri.html** ove possibile (se richiesta in stessa sessione), oppure farlo inserire esplicitamente all'utente. Alla pressione del pulsante **cambia**, **cambiaDate.html** deve inviare una richiesta alla servlet **cambiamenti** che, dopo avere controllato che la distanza fra prelievo e riconsegna sia inferiore a 90 giorni, rende immediatamente definitivi i cambiamenti richiesti.

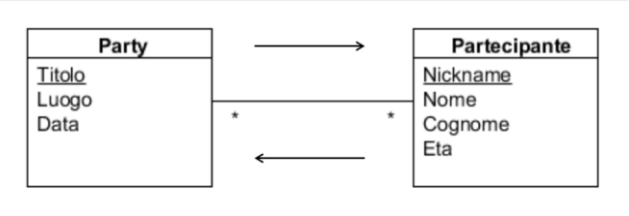
Tecnologie Web T
08 Giugno 2012 – Compito

ESERCIZIO 2 (10 punti)
Realizzare una applicazione Web per il calcolo dell'IMU, basandosi principalmente su tecnologie **Javascript**, **JSP** e **AJAX**.

L'applicazione deve essere costituita da una **pagina HTML** iniziale tramite cui sia possibile inserire il proprio codice fiscale e la rendita catastale dell'immobile. Dopo avere inserito entrambi i dati (controllando che il codice fiscale sia effettivamente composto da 16 caratteri), questi devono essere inviati automaticamente tramite **richiesta AJAX** verso un'opportuna **pagina JSP**. Tale pagina dovrà calcolare l'importo dell'IMU (per semplicità, considerata uguale a rendita catastale * 160 * 0.004); la risposta restituita dovrà contenere sia l'indicazione del codice fiscale (ovviamente invariato) che l'importo IMU calcolato, restituiti tramite **serializzazione JSON**.

La risposta deve essere visualizzata nella **pagina HTML** iniziale, insieme ad altre eventuali risposte calcolate in precedenza, fino a includere un massimo di tre risposte; una volta superate le tre risposte, si può ripartire da capo con pagina vuota e senza "storia" di interazioni precedenti.

ESERCIZIO 3 (11 punti)
Partendo dalla realtà illustrata nel diagramma UML di seguito riportato, si fornisca una soluzione alla **gestione della persistenza** basata su **Hibernate** in grado di "mappare" efficientemente il modello di dominio relativo.
In particolare, la soluzione richiesta deve propendere per una modellazione del problema basata su due **JavaBean**, ovvero **Party** e **Partecipante**, lato Java, e sulle corrispondenti tre **tabelle Party**, **Party_Partecipante** e **Partecipante** contenute nel database **TW_STUD** di DB2.



Nel dettaglio, sfruttando l'efficienza degli **ID surrogati**, dopo aver creato da applicazione le tabelle all'interno del proprio schema nel database **TW_STUD** (esplicitando gli opportuni vincoli di PK e FK), implementato i JavaBean e definito i file XML di mapping Hibernate necessari, si richiede la realizzazione di una classe di prova facente uso delle API Hibernate in grado di:

- inserire almeno due party;
- inserire uno o più partecipanti alle feste;
- restituire, partendo dal nickname di un partecipante, i titoli delle feste a cui ha preso parte;
- restituire, partendo dal titolo di un party, i nickname dei suoi partecipanti

il tutto, mediante opportuna **gestione delle transazioni**.

N.B. Le relazioni **M-N** tra Partecipante e Party e tra Party e Partecipante devono essere specificate esplicitamente nel file XML di mapping.