TECNICHE DI PROGRAMMAZIONE

04/06/19

Tema d’esame “Serie A” del 22/01/2018.

In questo problema ci vengono date 3 tabelle:

1. Stagioni (esempio “2003-2004” o “2004-2005”).
2. Squadre.
3. Partite: Ci dice, per ogni partita giocate, in quale stagione è stata giocata, tra quali coppie di squadre e tutti i dati relativi a cosa è successo in quella partita.

Il punto 1 del problema è il seguente:

Immagine che contiene screenshot

Descrizione generata automaticamente

Immagine che contiene screenshot

Descrizione generata automaticamente

Risolviamo il punto A e il punto B insieme in quanto sono collegati.

Iniziamo dal Modello in cui immaginiamo di avere un oggetto Team chiamato “squadraSelezionata” e una Map che, per ogni stagione, mi dia il punteggio (per il punto B). Inolter, aggiungiamo una lista di Team per gestire l’elenco delle squadre e una lista di Season. Alla creazione del modello, potrebbe già leggere le squadre e le stagioni dal Database. Nel nostro caso siamo fortunati in quanti ci viene dato un DAO che contiene i metodi “listAllSeasons” e “listTeams” che ci permettono di ottenere dal Database quello che ci serve.

Quindi, all’avvio del Modello (costruttore), creiamo già un collegamento al DAO. Così facendo, ci calcoliamo le squadre e le stagioni con i metodi “listTeams” e “listAllSeasons”. Di sicuro, le squadre ci servono perché il Controller deve popolare la tendina. Creiamo quindi un metod o “getSquadre” (del Model) nel quale andiamo a ritornare i valori delle squadre. Grazie a questa parte il Modello permette già al Controller di popolare la tendina.

Immagine che contiene screenshot, testo

Descrizione generata automaticamente

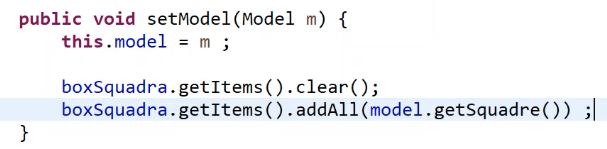
Passiamo quindi alla gestione del Controller. Sicuramente dovremo andare a modificare il contenuto della ComboBox che dovrà avere dei “Team”. Come seconda cosa, andiamo a memorizzare un riferimento al Modello creando l’istanza del model nella classe Controller e utilizzando il metodo “setModel”.

Immagine che contiene screenshot

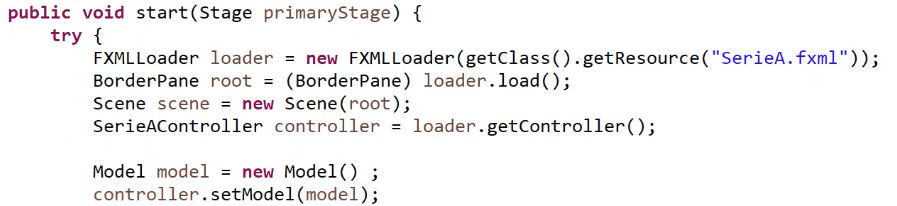
Descrizione generata automaticamente Immagine che contiene screenshot

Descrizione generata automaticamente

Appena ho il riferimento al modello, dato che il modello conosce quale sono le squadre, vado subito a popolare la tendina.



**Ovviamente il Modello va creato e passato al Controller nel Main.**

****

Noi sappiamo che quando facciamo “new Model” partiranno le prime query al Database per caricare le squadre e quando facciamo setModel, verrà impostata la tendina che verrà poi visualizzata sotto con il comando “primaryStage.show()”.

Facendo questo, il punto “A” è terminato e funziona correttamente.

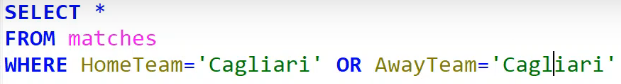
Ritorniamo nel Modello e proviamo a risolvere il punto “B”.

**IL PROF CONSIGLIA SEMPRE DI ANDARE AVANTI PER PICCOLE ITERAZIONI MA AVERE SEMPRE UN PROGRAMMA CH E FUNZIONA! ALL’ESAME SE NON FUNZIONA IL PROGRAMMA SI È BOCCIATI.**

Ora che conosciamo la squadra, alla pressione del bottone “Seleziona Squadra”, dobbiamo calcolare il punteggio in campionato per ogni stagione.

Proviamo come prima cosa ad impostare la query. Prendiamo come esempio il Cagliari. Questa squadra può aver giocato o in casa o fuori casa, quindi sicuramente nel where avremo un “OR” che considera questi due casi. Fatto questo, possiamo decidere di calcolare i punteggi modificando ancora ulteriormente la query o di lasciare così la query e calcolare i punteggi da Eclipse. Nel nostro caso, mettere della logica di ragionamento nella query non è la tecnica migliore (meglio farlo in codice Java).

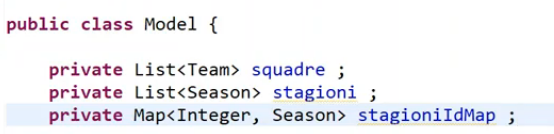
Avremo quindi nel DAO un metodo che ci restituisce una lista di Match dato un certo Team. Questo metodo implementerà la seguente query:



Nel select possiamo anche non portarci tutti i dati dietro (\*), infatti la classe Match di Eclipse ha già tutti i campi. Possiamo quindi scegliere quali tra questi campi scegliere. Una volta modificata la query con i campi che ci servono, la eseguiamo. Apriamo quindi una connessione, creiamo un PreparedStatement e mettiamo il try-catch. Ricordarsi di settare sempre i punti interrogativi (importante l’ordine). Lanciamo poi la query (ResultSet) e costruiamo il risultato che è una lista di Match. Per ogni elemento del ResultSet, mediante il “while”, popolo la lista.

La lista va popolata quindi con oggetti Match (con id,season,div,date,ecc…). **La stagione, è un oggetto Season e a noi non va bene in quanto il database ci restituisce un intero!** Dall’intero noi dobbiamo trovare l’oggetto intero 🡪 IDENTITYMAP!

Andiamo quindi nel Model e aggiungiamo un IdentityMap per le stagione che, dato un intero, ci restituisci l’oggetto Season.

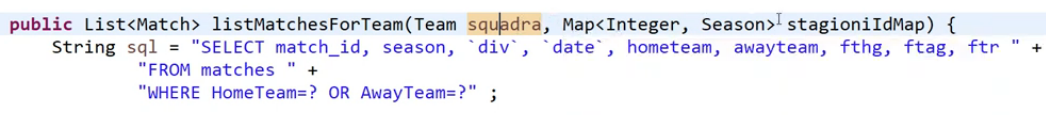


Questa idMap la dobbiamo creare non appena abbiamo letto le stagioni.



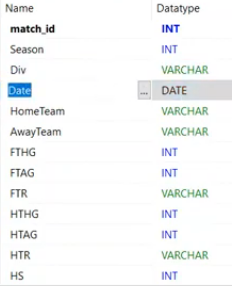
Quindi, grazie al “for”, ho la idMap che mappa l’oggetto intero a partire dall’intero.

Questa mappa dovrà essere un parametro in più per il metodo del modello che stiamo scrivendo . Inizio del metodo “listMatchesForTeam” del DAO:



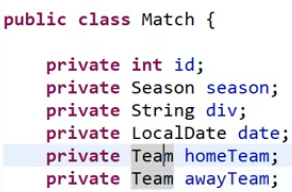
A questo punto posso usare questa mappa nel metodo perché dalla query ottengo il numero e poi, grazie all’idMap, lo uso come chiave per trasformarlo nell’oggetto season corrispondente.

**IMPORTANTE: ogni volta che in un metodo del DAO, dalla query ottengo degli oggetti, uso l’identityMap.**

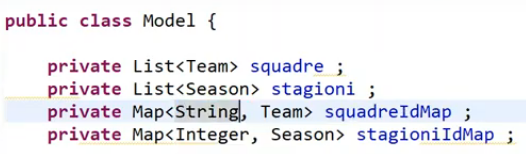


Su Heidi posso sempre controllare su “table” che tipi di dati ci vengono restituiti. Nel caso della data infatti, Heidi mi restituisce una “Date” mentre nella classe “Match” di Eclipse noi abbiamo la data che è un LocalDate. Nel DAO dobbiamo quindi convertirlo in LocalDate.

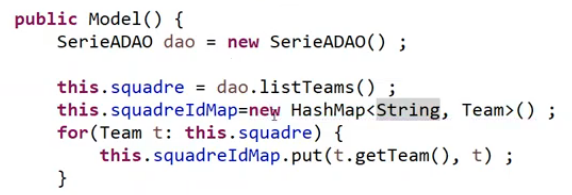
Stessa cosa delle stagioni succede con l’homeTeam e l’awayTeam. Se andiamo nella classe Match, notiamo che sono oggetti Team, mentre su Heidi sono salvati come “String”. **Anche in queso caso sempre l’idMap!**

** **

Nel Model andiamo quindi a creare un'altra idMap che dalle stringa ci restituisce l’oggetto Match.



Anche in questo caso, come nelle stagioni, dopo aver letto i Team li vado a mappare con il for.

(nel costruttore del Model)

Aggiungiamo anche questa Map ai parametri del metodo del DAO.

Una volta definito l’oggetto Match, lo andiamo ad aggiungere alla lista. Ricordiamoci di chiudere empre la connessione alla fine del metodo del DAO e di restituire il risultato finale (nel nostro caso la lista di oggetti Match). Nel “try” del metodo avremo quindi il seguente codice:

Immagine che contiene screenshot

Descrizione generata automaticamente

Questo metodo del DAO lo useremo quando l’utente preme il bottone “doSelezionaSquadra”. Il Controller chiederà al Model la i diversi punteggi della squadra selezionata negli anni.

Abbiamo quindi un metodo nel Model “punteggi” che fornisce al Controller la classifica di una determinata squadra. Questo metodo restituirà una Mappa tra stagione e punteggio.

Il metodo quindi inizializza questa mappa, chiede al DAO di restituirgli l’elenco delle partite e per ogni partita aggiorna questa Mappa. Creiamo quindi al DAO e chiamiamo da esso il metodo.

Immagine che contiene screenshot

Descrizione generata automaticamente

Per ogni Match estraggo la stagione. Creo una variabile “punti”. Se abbiamo un pareggio, aggiungiamo un punto. Nel caso di vittoria, invece, dobbiamo controllare sia le vittorie fuori casa che quelli in casa e dividere i due casi.

Immagine che contiene screenshot

Descrizione generata automaticamente

Ora sappiamo quanti punti dare alla squadra e dobbiamo solo più aggiornare la classifica. Dobbiamo prendere i punti attuali e sommarci questi punti. I punti attuali li abbiamo dentro la Mappa. Come ultima cosa, ritorno i punteggi.

Immagine che contiene screenshot

Descrizione generata automaticamente

Il metodo alla fine è stato chiamato “calcolaPunteggi” per non confondersi. Il metodo è terminato; bisogna solo più collegarlo al Controller (tasto “doSelezionaSquadra”). Il metodo del bottone deve permetterci di prendere il Team selezionato dall’utente nel ComboBox (con controllo se Team=null) e di chiedere al Modello di calcolare il punteggio per questa squadra. Logicamente, il risultato dovrà essere inserito nella mappa “punteggi” che dovrà essere restituita dal metodo.

Immagine che contiene screenshot

Descrizione generata automaticamente

Come ultima cosa, attraverso un “for”, stampo i punteggi per ogni stagione che abbiamo nella mappa. Ricordiamoci che “s” è un oggetto Season quindi per scrivere la stagione (stringa) uso il comando “getDescription”. Per i punteggi, li estraggo dalla mappa “punteggi” per ogni stagione “s”.

Immagine che contiene screenshot

Descrizione generata automaticamente

FINE