Applicazioni Web: Linguaggi e Architetture

Relazione di Laboratorio

Bertoldi Noemi

Musetta Garion

1) ISTRUZIONI PER L'AVVIO:

- Assicurarsi che il servizio SQL SERVER (SQLEXPRESS) sia attivo, prima di far partire l'applicazione

- Modificare eventualmente la stringa di connessione che si trova in ../AppWebProj/appsettings.json, sostituendo i campi relativi a "User ID" e "Password" con le proprie credenziali SQL Server Authentication.
  Potrebbe essere necessario sostituire "User ID" e "Password" con "Trusted_Connection=Yes" nel caso si usasse Windows Authentication.

- Eseguire lo script di creazione del database riportato nel file ddl.sql

- Aprire lo script di popolamento delle tabelle (dml.sql),sostituire il percorso riportato nella clausola `DECLARE @ROOT VARCHAR(500)` con il percorso su cui si è salvata la cartella del progetto ed eseguire lo script, ad esempio:
  `'C:\Users\(user)\...\appwebcs-master';`

- Aprire la soluzione con Visual Studio (potrebbe essere necessario aprirlo come amministratore) eseguire la BUILD del progetto ed infine eseguire il programma.

2) SCELTE IMPLEMENTATIVE

- Si è scelto di non usare framework come Kendo e AspNetAuthentication per cercare di capire meglio il funzionamento di ASP.NET a più basso livello, per questo motivo sono state implementate manualmente le pagine che mostrano i risultati delle query e il meccanismo di autenticazione, rispettivamente,usando delle semplici tabelle HTML econ il controllo del ruolo dell'utente salvato in sessione. Quest'ultimo punto permette di bloccare l'accesso alle pagine riservate da parte di utenti non autorizzati.

- Per poter offrire ugualmente la funzione di filtraggio sulle tabelle dei risultati, in assenza della Kendo Grid, sono stati implementati degli extension methods (Src/Extensions) che applicano dei query operators Where, al fine di filtrare la query. In questi metodi si sono sperimentati concetti chiave del C# come Extension Methods, Generics, Query Operators, Delegates, Lambdas. E' stato applicato il design pattern Strategy.

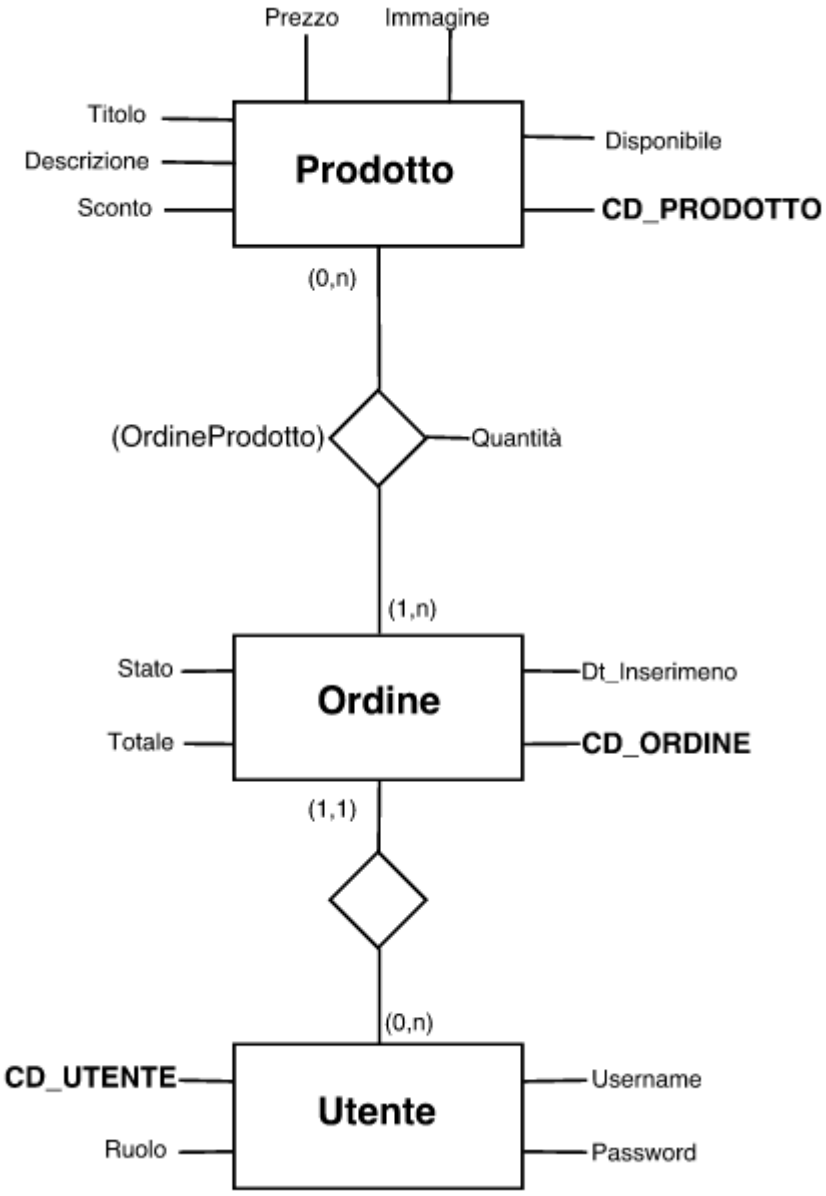
- In modo analogo, per poter aggiungere e cancellare delle righe di tabelle, in assenza della Kendo Grid, si è fatto uso di form e bottoni.

- Il Carrello dei prodotti è memorizzato interamente in sessione, rendendolo così temporaneo. Il Carrello, infatti, esiste sia per utenti non loggati, sia perutenti che abbiano "user" come ruolo. Questa scelta implica il fatto che una volta "chiusa" la sessione, non sarà possibile recuperare il contenuto del Carrello, pertanto l'acquisto dovrà essere terminato durante la sessione.

- Si è cercato, in generale, di usare molte tecniche diverse per cercare di imparare il più possibile non scegliendo necessariamente la tecnica migliore.

3) DESCRIZIONE MODELLO RELAZIONALE

Lo scherma E-R segue quanto illustrato prima: il carrello non è memorizzato sul database e pertanto sono presenti solo 3 entità: Utente, Ordine e Prodotto. La tabella OrdineProdotto (nel database) è il risultato della relazione n : m fra Ordine e Prodotto.



4) DESCRIZIONE MODELLO A OGGETTI

Per ogni dettaglio più specifico sull'implementazione si faccia riferimento direttamente sul codice, dove classi e metodi sono corredati da opportuni commenti.

Il progetto segue il pattern MVC:

- **Il Model** è il risultato del mappaggio automatico del database in oggetti. Per agevolare l'operazione si è seguito il tutorial:
  http://www.learnentityframeworkcore.com/walkthroughs/existing-database
  Le classi **Utente, Ordine, OrdineProdotto e Prodotto** rappresentano i Model del contesto.

- **I Controllers** modificano e processano i model e li passano alle Views. E' presente un controller per ogni categoria di pagine (uno per ogni sub-folder delle view), nel quale ogni view corrisponde a un metodo di un controller. Quando una pagina è richiesta, il controller ascolta la richiesta, la processa eseguendo operazioni specifiche per ogni pagina, e poi indirizza alla View adatta, passando un Model.

  In ordine alfabetico:
  - **CarrelloController** gestisce le operazioni sul carrello: aggiunta, modifica e rimozione prodotti
  - **CrudController** è una "superclasse" che raccoglie tutti i controller che eseguono operazioni CRUD sui model: questi controllers (OrdineController, ProdottoController e UtenteController) fanno uso dei metodi del CrudController per eseguire le suddette operazioni.
  - **HomeController** gestisce la home page e fornisce la query sulla top 10 dei prodotti venduti questo mese.
  - **OffertaController** pilota la view Offerte, che espone i prodotti in offerta.
  - **OrdineController** permette la creazione (e la modifica, da parte degli admin) di nuovi ordini da parte degli utenti.
  - **PrivateHomeController** è collegato all'area di amministrazione degli utenti admin.
  - **ProdottoController** controlla tutte le pagine relative ai prodotti: ricerche, catalogo e dettagli prodotto. Gestisce anche le modifiche dei dati di un prodotto da parte di utenti admin.
  - **UtenteController** permette ai nuovi utenti di registrarsi e agli admin di modificare i ruoli degli altri utenti.

- Le **Views** rappresentano la parte di front-end e l'esposizione dei dati preparati dai controllers. Permettono anche l'interazione con i controller stessi tramite links e forms, che chiamano alcune funzioni dei controller.

Sono inoltre presenti altre classi:

- Il Folder **Data** contiene le definizioni dei DataSource, gli oggetti che contengono i risultati di query complesse e che devono essere passati alle views.

- **Src** ha due classi: **Extensions** e **FIlterStrategy**.
  Extensions contiene tutti gli Extension Methods definiti per il progetto: ci sono metodi per serializzare e deserializzare oggetti complessi in modo da poterli salvare in Session e i metodi per filtrare le tabelle.
  I primi sono usati da CarrelloController per salvare i prodotti in Session come stringhe, mentre i filtri sono utilizzati da tutti i controller che espongono delle tabelle nelle view; più precisamente, ProdottoController.Advanced (lista dei prodotti che corrispondono alla ricerca), UtenteController.List (riservato agli amministratori) e OrdineController.Index (ordini di uno specifico utente)/ OrdineController.List (espone tutti gli ordini di tutti gli utenti) usano tutti gli extension methods per filtrare.

  I metodi di filtraggio sono stati necessari per rimpiazzare la Kendo Grid e per permettere comunque di filtrare le tabelle secondo alcuni parametri principali.

  FilterStrategy è la strategy usata dal metodo per filtrare: essendonci un unico metodo generale per filtrare parametri diversi, il chiamante specifica anche una strategy su come filtrare: la strategy è composta da una serie di delegates, da istanziare con i metodi specifici dell'oggetto in questione.

  Per ogni chiarimento si faccia riferimento al codice e si guardino i commenti.