# Esercitazione di Fine Settimana – Week 4

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nome | Carlotta |
|  |  | Cognome | Colla |
|  |  | Data | 09/04/2021 |

Leggete attentamente ogni domanda e argomentare quanto più possibile fornendo anche degli esempi.

1. Descrivere le modalità di utilizzo di ADO.NET  
   Ado.Net può essere utilizzato in modalità connessa e modalità disconnessa. Nella modalità connessa le modifiche vengono riportate direttamente sul database, se il risultato della query è una tabella utilizza SqlDataReader per leggere i campi, se il risultato è uno scalare utilizza ExecuteScalar, se la query non ha risultato (come un insert o un delete) allora si utilizza ExetuteNonQuery che ritorna il numero di righe modificate.  
   Nella modalità disconnessa vengono definiti i metodi crud che si voglione eseguire (il select deve sempre essere definito) e vengono associati all’adapter che fa da ponte tra il codice e il database. Le tabelle che si vogliono utilizzare vengono caricate nel dataset, in modalità connessa, con il comando fill, poi in modalità disconnessa, vengono eseguite le operazioni sui dati, infine attraverso il comando Adapter.Update le modifiche vengono riportate sul database.
2. Quali sono i metodi di esecuzione della classe DbCommand disponibili e in quali casi vanno utilizzati?  
   I metodi di esecuzione sono 3:  
   1 – command.ExecuteNonQuery() : viene utilizzato nelle query che non ritornano un risultato come Update, Insert e Delete ma ritornano il numero di righe modificate, infatti questo comando ritorna un intero.  
   2 - command.ExecuteScalar() : viene utilizzato quando il risultato di una query è uno scalare, come un SELECT COUNT(\*) che ritorna il numero di record presenti in una tabella.  
   3 – command.ExecuteQuery() : viene utilizzato quando il risultato di una query è una tabella (es un join tra tabelle), ritorna un SqlDataReader che contiene tutte le righe della tabella risultante e viene letto attraverso reader.Read() che ritorna un booleano per sapere se contiene delle righe, se le contiene allora si possono leggere i singoli campi specificandoli tra parentesi quadre (reader[“ID”] legge il valore nel campo ID).
3. Descrivere l’utilizzo della classe DataAdapter  
   Il DataAdapter viene utilizzato nella modalità disconnessa, serve per fare da ponte tra il database e l’applicazione. All’adapter vengono associati i metodi CRUD che servono per eseguire i calcoli desiderati, il comando select deve sempre essere istanziato, serve per popolare il dataset.  
   Una volta associati i comandi viene popolato il dataset con le tabelle richieste dal select e vengono eseguite le modifiche dei dati in modalità disconnessa.  
   Alla fine delle operazioni l’adapter riporta la modifiche effettuate anche sul database, attraverso il comando Adapter.Update(dataset, “NomeTabella”) specificando quali tabelle sono state modificate.
4. Come è possibile andare a definire il modello dati in Entity Framework?  
   Il modello Entity Framework viene definito attraverso:  
   1- Si crea una classe per ogni entità utilizzando la nomenlcatura standard, le Annotations o le Fluent API  
   2 – una classe Context che deriva da DbContext in cui vengono definiti due costruttori (uno vuoto e uno con un parametro options) che richiamano i costruttori della classe base;  
   Viene inserita la stringa di connessione;  
   Vengono aggiunte le tabelle tramite il DbSet  
   Si inseriscono le relazioni tra tabelle attraverso le Flent API  
   3 – Attraverso la Magration viene creato il database
5. Descrivere l’utilizzo delle Migration e i vantaggi che ne derivano  
   Le Migration vengono utilizzate per creare/modificare il database. Per effettuare una migration si deve impostare come progetto di strartup il progetto che contiene la classe context e si apre la package manager console.  
   I comandi per la console sono: Add-Migration Initial, Update-Database per la creazione del database, mentre se si vogliono apportare modifiche si utilizza Add-Migration più il nome di quella modifica.  
   Ogni migration crea una classe nel progetto in cui riporta esattamente come è stato creato/modificato il database.

**Esercitazione Pratica**

Realizzare un sistema di gestione delle spese che si basi su:

* Un database **GestioneSpese** (SQL Server), costituito dalle tabelle
  + **Spese**
    - *Id* (int, PK, auto-incrementale)
    - *Data* (datetime)
    - *CategoriaId* (int, FK)
    - *Descrizione* (varchar(500))
    - *Utente* (varchar(100))
    - *Importo* (decimal)
    - *Approvato* (bit)
  + **Categorie**
    - *Id* (int, PK, auto-incrementale)
    - *Categoria* (varchar(100))
* Una **Console app** che consenta di:
  + Inserire nuove Spese
  + Approvare le Spese esistenti (impostare il campo *Approvato*)
  + Cancellare le Spese esistenti
  + Mostri
    - l'elenco delle Spese Approvate
    - L'elenco delle Spese di uno specifico Utente
    - Il totale delle Spese per Categoria

VINCOLI TECNICI

* Utilizzare Entity Framework
* Utilizzare l'approccio Code-First e attivare le Migrations

**OPZIONALE**: Implementare una o più delle funzionalità utilizzando ADO.NET (Connected o Disconnected mode, a scelta)

*Es. Inserire una nuova spesa*