# Esercitazione di Fine Settimana – Week 4

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nome | Francesco |
|  |  | Cognome | Dell’Ascenza |
|  |  | 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
2. Quali sono i metodi di esecuzione della classe DbCommand disponibili e in quali casi vanno utilizzati?
3. Descrivere l’utilizzo della classe DataAdapter
4. Come è possibile andare a definire il modello dati in Entity Framework?
5. Descrivere l’utilizzo delle Migration e i vantaggi che ne derivano
6. ADO.NET ha due modalità di utilizzo, una detta “connected”, l’altra detta “disconnected”. Nella prima modalità la connected abbiamo l’accesso diretto ai dati, prima si apre la connessione come istanza di SqlConnection, poi si crea il comando (e gli eventuali parametri) come istanza di SqlCommand e SqlParameter, e dopo i dati vengono letti da un’istanza di DataReader, oppure viene eseguito il comando ExecuteNonQuery se ad esempio si tratta di un comando DML. Infine la connessione viene chiusa. La seconda modalità, la disconnected invece è divisa in due parti, prima si esegue il comando in modalità “disconnessa” tramite l’adapter, i dati invece vengono ritornati in un DataSet o un DataTable e solo successivamente sono adattati sul DB.
7. I metodi di esecuzione sono ExecureNonQuery() per eseguire ad esempio un DML command, ExecuteScalar() quando vogliamo che ci ritorni solo un valore scalare ed ExecuteReader() quando vogliamo farci ritornare i dati in forma di oggetto.
8. Il DataAdapter è ciò che fa da tramite tra il DataSet e i dati, quando ci facciamo tornare i dati l’adapter in un secondo momento li adatta su un’istanza di DataSet o DataTable e viceversa .
9. Il modello di dati in EF può essere definito seguendo uno di tre approcci:
   1. DB-First ovvero si crea il db e nell’app si ha una rappresentazione fedele del modello relazionale del db pre-esistente sulle entità
   2. Model-First ovvero il Db viene creato con il db-designer di visual studio ma questa modalità è poco flessibile visto che ogni modifica al db verrà persa sul lato codice
   3. Code-First ovvero si crea il modello di entità e relazioni direttamente lato codice facendo sì che il db segua questo modello
10. Le migration servono per creare effettivamente il modello di relazione dal lato codice al lato db, ovvero con la prima migrazione si “crea” il db a partire dal modello che ho creato con c# e le successive modifiche a questo vengono via via aggiunte tramite ulteriori migrazioni successive. I vantaggi di questo approccio code-first sono molteplici, prima cosa si può fare un controllo più efficace, veloce e completo del source avendo tutto espresso nel lato codice, poi non bisogna effettivamente conoscere SQL o comunque il suo utilizzo non è necessario visto che le relazioni, chiavi types ecc sono tutti specificati nel lato codice con la sintassi di c# ed una sintassi intermedia propria di EF che ci aiuta a definire i constraint del nostro modello ad esempio

**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*