# Esercitazione di Fine Settimana – Week 4

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nome | Maria Chiara |
|  |  | 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 è una tecnologia che fornisce l’accesso ai dati in modo coerente a uno o più database, include dei provider di dati per la connessione, l’esecuzione di comandi e il recupero dei dati.  
   Per la connessione si utilizza una stringa di connessione che contiene tutte le informazioni necessarie per connettersi al db, con ADO si ha una connection pool che contiene delle istanza di connessione già create, ogni volta che si chiama Open() sulla connessione si prende una di queste connessioni, se disponibile altrimenti si aspetta che se ne liberi una, e quando si chiama Close() si restituisce la connessione tra quelle disponibili.  
   ADO consente l’accesso ai dati in due modalità:  
   - Connection Mode: accesso in sola lettura con la possibilità di eseguire dei comandi sull’origine dei dati, i principali comandi sono Connection (per stabilire la connessione con il db), Command (per creare un comando), DataReader (per eseguri il comando e leggerlo) e Parameter (possibilità di aggiungere dei parametri ai comandi)  
   -Disconnected Mode: manipolare i dati recuperati e poi rinconciliarli con l’origine, le classi principali sono DataSet (contiene una raccolata di DataTable), DataTable(contiene le tabelle del db), DataColumn(colonne delle tabelle), DataRow(righe delle tabelle), constraint(vincoli) e DataRelation(relazioni tra le tabelle), il DataAdapter fornisce un ponte tra il DataSet e l’origine dei dati.
2. Quali sono i metodi di esecuzione della classe DbCommand disponibili e in quali casi vanno utilizzati?  
   i metodi di esecuizione di un Command sono 3:  
   -NonQuery: non produce risultato, al massi può restituire il numero di righe che sono state aggiunte/rimosse o aggiornate,  
   -DataReader: produce come risultato un insieme di record,  
   -Scalar:
3. Descrivere l’utilizzo della classe DataAdapter  
   Il DataAdapter è presente nella modalità disconnesso di ADO.NET, permette di collegare l’origine dei dati, che può essere più di una, con le classi.  
   Esegue i comandi SQL sull’origine dei dati per caricare il DataSet con tutte le modifiche e riconciliarle all’origine.
4. Come è possibile andare a definire il modello dati in Entity Framework?  
   Esistono diversi approcci per definire il modello dati:  
   -DataBase-First: il modello viene importato da un database già creato,  
   -Model-First: il modello viene creato dal designer di Visual Studio, non favorisce il riuso e la separazione del codice,  
   -Code-First: il modello viene creato dal codice scritto in Visual Studio, la sua implementazione è basata sul codice.
5. Descrivere l’utilizzo delle Migration e i vantaggi che ne derivano  
   Le Migration permetto di creare il database pertendo dal codice scritto in Visual Studio (code-first), le Migration leggono l’intero codice e grazie ai DbSet capiscono quali tabelle si vogliono creare nel nuovo db e con quali colonne (nome, tipo e altre informazioni), controlla le convezioni sul codice per definire degli aspetti aggiuntivi alle colonne delle tabelle, per esempio chiamare la chiave primaria ID o nomeclasseID, e la chiave esterna nomeclasseesternaId.  
   Controlla tutte le DataAnnotations (annotazioni sopra le proprietà delle classi per definire chaivi e vincoli su quei campi specifici) e le fluent API che sono un alternativa o un aggiunta alle DataAnnotations, sono segnate come codice e vengono definite dentro al Context, si possono definire le chiavi, le relazioni e i vincoli sulle varie tabelle.  
   Le Migration danno la precedenza alle Fluent API, poi alle DataAnnotations e in fine alle convenzioni sul codice.  
   Una volta letto il codice per intero, se scritto corretto, le Migration sono in grado di creare un nuovo database con quelle caratteristiche.

**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*