# Esercitazione di Fine Settimana – Week 4

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nome | Mauro |
|  |  | Cognome | Abozzi |
|  |  | Data | 8/10/2021 |

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

1. Descrivere le modalità di utilizzo di ADO.NET

Le modalità di utilizzo sono due: connected mode e disconnected mode. La connected mode permette di eseguire comandi SQL e accesso in lettura direttamente sul database, si dovrà quindi aprire una connessione al database, creare ed eseguire i comandi (ad esempio con ExecuteNonQuery della classe Command che esegue il comando SQL) e infine chiudere la connessione. La disconnected mode permette di manipolare i dati in memoria e non nel database, per aggiornare i dati sul database questi andranno riconciliati. La classe DataSet è utilizzata per la gestione dei dati in memoria e conterrà informazioni su righe e colonne delle tabelle. Ad esempio, per cancellare una riga in disconnected mode questa verrà marcata come deleted sul dataset e successivamente verrà riconciliata col database chiamando il metodo Update dell’adapter.

1. Quali sono i metodi di esecuzione della classe DbCommand disponibili e in quali casi vanno utilizzati?

DbCommand è una classe utilizzata per i comandi SQL da eseguire sul database. La classe SqlCommand deriva da DbCommand e i metodi utilizzati sono CreateCommand() della classe DbConnection che restituisce un DbCommand associato alla connessione al db, la proprietà CommandType per definire il tipo di inserimento del comando (ad esmpio Text per una stringa SQL) e la proprietà CommandText per definire la stringa SQL col comando da eseguire.

1. Descrivere l’utilizzo della classe DataAdapter

DataAdapter serve per il collegamento tra il DataSet e il database. Utilizzato per la riconciliazione tra i dati nel dataset e quelli nel database col metodo Fill (adapter.Fill(dataset,”Tabella”)), per definire i comandi SQL (ad esempio in disconnected mode andranno ridefiniti i comandi insert, select e update adapter.SelectCommand = mySeclectCommand), per la riconciliazione dei dati tra dataset e database con il metodo Update (adapter.Update(dataset,”Tabella”).

1. Come si definisce il modello dati in Entity Framework?

Entity Framework definisce il modello dati a partire dal codice (ad esempio C#) in modo da non dover utilizzare script SQL, le query si possono definire con LINQ. Ogni tabella (o entità) è definita nel DbContext come un DbSet (ad esempio public DbSet<Student> Students { get; set; }. Nelle classi di configurazione sono definite le configurazioni delle entità (definendo gli attributi) e delle relazioni, le relazioni sono definite con le navigation properties (ad esempio nelle entità Spesa public Category Category { get; set; } e nell’entità Category public IList<Spesa> Spese { get; set; } rappresentano la relazione uno a molti tra spesa e categoria). Nella classe di configurazione sono definiti anche eventuali vincoli (esempio IsRequired() per not null), relazioni (con i metodi hasOne o hasMany) o altre informazioni come la lunghezza di una stringa di un attributo, che serviranno successivamente per una possibile migration che mapperà entità e relazioni sul database. Il campo definito come Id o <NomeClasse>ID in un’entità viene considerato per convenzione come chiave primaria altrimenti va specificato con annotations o fluent API.

1. Descrivere l’utilizzo delle Migration e i vantaggi che ne derivano

Le migration sono utilizzate per effettuare modifiche sul database direttamente da codice (nel nostro caso C#) senza utilizzare comandi SQL. Nella Package Manager Console il comando Add-Migrations «Migration name» genera una nuova classe con tutte le modifiche effettuate sul db rispetto allo stato precedente, le informazioni verranno prese dalle classi che definiscono le enitità e le configurazioni. Il comando update-database aggiornerà il database eseguendo ciò che è indicato nelle migration. Il file che viene creato per una migration avrà nome TimeStamp + nome migration.cs e contiene i metodi Up e Down per l’aggiornamento del db.

**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*