Esercitazione di Fine Settimana – Week 5

Nome Laura

Cognome Martines

Data 30/07/2021

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 usata in modalità Connected o Disconnected. Nel primo caso, la connessione con il DataBase viene mantenuta per tutta la durata delle operazioni, mentre in modalità Disconnected la connessione viene utilizzata solamente per scaricare i dati dal DB e per caricarci quelli modificati, mentre le operazioni sui dati vengono eseguite in locale, in una sorta di modalità offline. La modalità connected ha lo svantaggio di mantenere occupata una connessione del DataBase per un tempo più lungo, rendendola di fatto inaccessibile per altri utenti. Cosa che viene superata nel caso della modalità disconnected, in cui gli utenti restano collegati solamente per il tempo strettamente necessario. Per contro, in disconnected mode bisogna considerare la gestione di eventuali conflitti, che possono avvenire durante la riconciliazione del DB locale con quello in remoto, se per esempio più utenti hanno cercato di agire in locale su uno stesso record apportando modifiche diverse. ADO.NET ha una serie di modi per gestire tali conflitti.
   Altra differenza tra le due modalità sono le classi utilizzate. Infatti usando la disconnected mode, il sistema locale deve crearsi una sorta di modello del database, che poi viene popolato con i record scaricati da remoto per permettere di lavorarci sopra. Per fare ciò, si usa una classe DataSet, che è indipendente dall'origine dei dati e viene utilizzato da tutti i provider. Dataset contiene una serie di raccolte tra cui le DataTables, che contengono le tabelle, DataRows e DataColumn che sono righe e colonne, e tutte le informazioni del dataBase come la presenza di chiavi, vincoli e relazioni. Tale DataSet viene popolato con il comando "Fill".

2. Quali sono i metodi di esecuzione della classe `DbCommand` disponibili e in quali casi vanno utilizzati?
   Il DbCommand rappresenta un'istruzione SQL da eseguire sui dati; nello specifico, noi utilizziamo il SqlCommand, che rappresenta un'istruzione in t-SQL, ma il concetto è lo stesso: queste classi forniscono poi delle classi specifiche del database che ne rappresentano i comandi. Tramite DbCommand possiamo settare la connessione da usare, inserire query e definirne i parametri e così via.

3. Descrivere l'utilizzo della classe `DataAdapter`

Il DataAdapter permette di eseguire comandi e query SQL sui dai, di caricarli e scaricarli dal database remoto. Una istanza di DataAdapter contiene tutti i comandi per eseguire il CRUD dei dati (create, read, update e delete), e anche dei comandi specifici: è quindi necessario creare una istanza DataAdapter per ogni tabella sulla quale si desidera operare.

4. Cos'è una Interfaccia in C#? Come può essere utilizzata?
   L'interfaccia in c# è una sorta di contratto che le classi che vogliono implementarla devono rispettare, e tale contratto si esplica nella ridefinizione di membri dell'interfaccia all'interno delle classi che la implementano. L'interfaccia in C# si comporta come una classe astratta, ossia contiene solamente membri astratti; è effettivamente come una classe base, in quanto supporta il polimorfismo (posso associare istanze di classi che implementano l'interfaccia a oggetti di tipo interfaccia e creare liste di oggetti di classi che implementano l'interfaccia, come si fa con le classi basi), ma mi permette di superare il vincolo della ereditarietà singola in quanto ogni classe può implementare più interfacce (mentre può ereditare da una singola classe base). In aggiunta, le interfacce sono cross-gerarchia, quindi anche classi che derivano da classi base diverse possono implementare la stessa interfaccia. Una interfaccia si definisce come una classe, ma il termine class è sostitutito da Interface, e il suo nome inizia per convenzione con una i maiuscola seguito dal nome definito dall'utente. Anche visual studio implementa di per sé tantissime interfacce. L'interfaccia è priva di modificatori di accesso, è di default pubblica.

5. Cos'è un Costruttore? E come si utilizza la keyword `this` nella definizione di un Costruttore?
   Un costruttore è un metodo particolare di una classe, che viene chiamato quando si utilizza la keyword "new" per creare una nuova istanza della classe stessa; il nome di questo metodo è lo stesso della classe/del tipo relativo, e non è possibile inserire un tipo restituito. Ogni classe può avere più di un costruttore (è possibile fare l'overload del costruttore), e questo permette al programmatore di scegliere quali valori assegnare durante la creazione dell'istanza e quali invece lasciare con il valore di default (che è possibile assegnare dentro il costruttore). Se nessun costruttore viene esplicitamente definito per una classe, essa utilizza il costruttore di default, che non ha parametri in ingresso; appena però un costruttore viene definito in modo esplicito (anche se con una firma diversa da quella del costruttore di default), il costruttore di default smette di esistere per quella classe, e per utilizzarlo diventa necessario esplicitarlo.
   La keyword this è un riferimento all'istanza corrente della classe. All'interno del costruttore si può utilizzare per distinguere tra nome di una proprietà e di variabili, se essi sono uguali (quindi scritti allo stesso modo). In aggiunta, la keyword this ci permette di richiamare i costruttori "in cascata", quindi ci permette di partire dal costruttore più esteso (ossia quello in cui vengono assegnati tutti o il maggior numero di proprietà della classe) e poi generare costruttori più semplici assegnando valori di default ai membri non inizializzati tramite la keyword "this".

Ad esempio, se il costruttore più esteso della classe Persona gli passa i valori

```
public Persona(string nome, string cognome, int age)
    {
            Nome=nome;
            this.cognome=cognome;
        Age=age;
    }
```

Posso costruire un costruttore più semplice “in cascata” come:

```
public Persona(string nome, string cognome) :this(nome, cognome, 16
    {
      Nome = nome;
      Cognome = cognome;
      Age = 16;
    }
```

In cui se non gli passo un valore di Age, il costruttore metterà 16 di default.

Esercitazione Pratica

Realizzare un sistema di gestione di un magazzino merci che si basa su:

- Un database **Magazzino** (SQL Server), costituito dalla tabella
  - **Prodotti**
    - *Id* (int, PK, auto-incrementale)
    - *Codice Prodotto* (varchar(10), UNIQUE)
    - *Categoria* (varchar(20) (Alimentari, Cancelleria, Sanitari,))
    - *Descrizione* (varchar(500))
    - *Prezzo Unitario* (numeric(10, 2))
    - *Quantità Disponibile* (int)
- Una **Console app** che consenta di:
  - Elencare i Prodotti
  - Inserire / Modificare / Cancellare Prodotti
  - Mostri
    - l'elenco dei prodotti con giacenza limitata (Quantità Disponibile < 10)
    - Il numero di Prodotti per ogni Categoria

VINCOLI TECNICI

- Utilizzare ADO.NET con Connected OPPURE Disconnected Mode (a scelta) per la parte di CRUD