Creare o Eliminare Record in un oggetto DataTable

Per **Creare un Nuovo Record** in un DataTable, si dichiara un oggetto "record" **R di classe DataRow** e si assegna ad esso un *Record Vuoto* generato utilizzando il **metodo NewRow** del DataTable stesso. Una volta impostati i campi con i valori, si aggiunge R al DataTable usando il normale *metodo Add* del suo insieme Rows.

☞ Un esempio potrebbe essere questo:

```
DataRow R = dt.NewRow;                  ' … creo un "record vuoto" (metodo dt.NewRow) e lo pongo in R (classe DataRow) …
R["idAlunno"] = 241;                    ' … imposto in R la campo Chiave Primaria, ad esempio a 241 …
R["Nominativo"] = "Roger Federer";      ' … imposto in R il campo Nominativo …
dt.Rows.Add(R);                         ' … aggiungo il nuovo record R al DataTable dt …
```

Per **Eliminare un Record,** conoscendone la posizione, basta usare il **metodo Delete** dell'insieme Rows del DataTable.

☞ Volendo eliminare il record in 5° posizione, scrivi: `dt.Rows[4].Delete()`

La classe SqlCommandBuilder e metodo UPDATE del DataAdapter

Se si **Effettuano Modifiche** ai dati presenti localmente nel DataTable memorizzato nella RAM (*inserendo* nuovi record, *modificando* i valori dei campi o *eliminando* record), nasce l'esigenza di "riportare" tali modifiche nel DataBase sul Server, allo scopo di rendere tali modifiche "permanenti".

La creazione di un oggetto della classe **SqlCommandBuilder** è *necessaria per riportare sul Server le modifiche* apportate localmente ai dati del DataTable.

Il **Costruttore della classe SqlCommandBuilder** necessita, come unico parametro, *dell'oggetto DataAdapter* che sta gestendo i dati del DataTable modificato.

☞ L'oggetto *SqlCommandBuilder* si crea **subito dopo** aver creato il DataAdapter, così:

```
SqlConnection cn = new SqlConnection("server=(localdb)\\MSSQLLocalDB;database=scuola;user id=sa;password=abacus");
SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM Alunni", cn);
SqlCommandBuilder cb = New SqlCommandBuilder(da);
DataTable dt = new DataTable();
```

Se l'oggetto di classe *SqlCommandBuilder* è stato creato e sono state *effettuate modifiche* ai dati nel DataTable, è possibile "riportare" tutte le modifiche sul DataBase del Server utilizzando il *metodo UPDATE* del DataAdapter.

Il **metodo UPDATE** necessita, come parametro, *dell'oggetto DataTable* sul quale il DataAdapter aveva precedentemente effettuato il FILL e i cui dati sono stati modificati.

☞ Possiamo ora scrivere il codice completo che mostra come si *caricano, modificano e scaricano* dati da un DataBase:

```
SqlConnection cn = new SqlConnection("server=(localdb)\\MSSQLLocalDB;database=scuola;user id=sa;password=abacus");
SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM Alunni", cn);
DataTable dt = new DataTable();
SqlCommandBuilder cb = New SqlCommandBuilder(da);
da.Fill (dt);
  … visualizzazioni e modifiche varie ai dati presenti nel DataTable dt …
da.UpDate (dt);
```

La **Possibilità di Riportare le Modifiche** sul DataBase originale è possibile SOLO se il *comando SELECT* indicato nel DataAdapter è basato *su un'unica Tabella*.

Il metodo UPDATE, se usato come *Function* in una espressione, restituisce un *Numero Intero* che indica il **Numero di Record che sono stati Modificati con Successo** sul Server.

☞ E' quindi possibile verificare l'esito dello "scarico" di una singola modifica, testando questo valore …

```
Valore = UpDate(dt)
if  ( Valore == 0 )  { MessageBox.Show("Modifica sul Server non riuscita."); }
else                 { MessageBox.Show("Modifica sul Server riuscita."); }
```