DataSet, DataAdapter, DataGridView

ADO.NET (Active Data Objects) reprezintă o parte componentă a nucleului .NET Framework ce permite aducerea, manipularea şi modificarea datelor.

ADO.NET reprezinta o mulţime de biblioteci de clase ce permit interacţiunea cu sistemele de stocare a informaţiilor. De obicei aceste sisteme sunt reprezentate de bazele de date, dar pot fi şi fişiere text, fişiere XML, fişiere Excel, etc. Lucrul se poate face fie conectat, fie deconectat de la sursa de date. Faptul că se permite lucrul deconectat de la sursa de date rezolvă următoarele probleme:

- menţinerea conexiunilor la baza de date este o operaţie costisitoare. O bună parte a lăţimii de bandă este menţinută ocupată pentru nişte procesări care nu necesită neapărat conectare continuă.
- probleme legate de scalabilitatea aplicaţiei. Se poate ca serverul de baze de date să lucreze uşor cu un nr. rezonabil de conexiuni menţinute, dar dacă numărul acestora creşte aplicaţia poate să reacţioneze extrem de lent.
- pentru unele servere se impun clauze asupra numărului de conexiuni ce se pot folosi simultan.

Folosirea combinată a obiectelor DataAdapter şi DataSet ne ofera posibilitatea efectuarii de operaţii de selectare, ştergere, modificare şi adăugarea datelor asupra unei baze de date.

DataSet

Un DataSet este o reprezentare în memorie a unui "data store" (un sistem de stocare şi obţinere a datelor). Conţine o mulţime de tabele asupra cărora se pot executa diverse operaţii, reţine informaţiile şi nu interacţionează cu sursa de date.

DataSet funcționează în strânsă legătură cu clasa DataAdapter care acționează ca o punte între un DataSet și sursa de date. Remarcabil este faptul că un DataSet poate face abstracție de sursa de date, procesarea datelor desfășurându–se independent de ea.

Marele avantaj al DataSet-ului este faptul că permite lucrul deconectat de la sursa de date, eliminând necesitatea unei conexiuni permanente precum la DataReader. În felul acesta,un server de aplicații sau un client oarecare poate apela serverul de baze de date doar când preia datele sau când se dorește salvarea lor.

Un *DataSet* este format din:

1. Colecția ***Tables*** ce conține 0 sau mai multe obiecte DataTable. Fiecare DataTable este compusă dintr-o colecție de linii și coloane.
2. Colecția ***Relations*** ce conține 0 sau mai multe obiecte de tip DataRelation, folosite pentru marcarea legăturilor părinte–copil.
3. Colecția ***ExtendedProperties*** ce conține proprietăți definite de utilizator.

Clasa **DataTable**

Datele dintr-un DataSet sunt conținute sub forma unor tabele de tip *DataTable*. Aceste obiecte pot fi folosite atât independent, cât și în interiorul unui DataSet ca elemente ale colecției *Tables*. Un DataTable conține o colecție *Columns* de coloane, *Rows* de linii și *Constraints* de constrângeri.

DataColumn

Un obiect *DataColumn* definește numele și tipul unei coloane care face parte sau se adaugă unui obiect DataTable. Un obiect de acest tip se obține prin apel de constructor sau pe baza metodei DataTable.Columns.Add.

Exemplu:

```
DataColumn myColumn = new DataColumn("title", Type.GetType("System.String"));
```

Definirea unei coloane ca fiind de tip autonumber (în vederea stabilirii ei ca și cheie pe o tabelă) se face astfel:

```
DataColumn  idColumn = new DataColumn("ID",Type.GetType("System.Int32"));
idColumn.AutoIncrement = true;
idColumn.AutoIncrementSeed = 1;
idColumn.ReadOnly = true;
```

DataRow

Un obiect de tip DataRow reprezintă un rand dintr-un obiect DataTable.

Orice obiect DataTable conține o proprietate Rows ce da acces la colecția de obiecte DataRow conținută. Pentru crearea unui rand se poate apela metoda NewRow pentru o tabelă a cărei schemă se cunoaște. Mai jos este dată secvența de cod care creează un rand nou pentru o tabelă și o adaugă acesteia:

```
DataRow  tempRow;
tempRow = myTable.NewRow();
tempRow["ID"] = 1;
tempRow["Name"] = "Book";
tempRow["Category"] = 1;
myTable.Rows.Add(tempRow);
```

Constrângeri

Constrângerile sunt folosite pentru a descrie anumite restricții aplicate asupra valorilor din coloane. În ADO.NET există două tipuri de constrângeri:

de unicitate și de cheie străină. Toate obiectele de constrângere se află în colecția Constraints a unei tabele.

- *UniqueConstraint* – precizează că într-o anumită coloană valorile sunt unice. Încercarea de a seta valori duplicate pe o coloana pentru care s-a precizat restricția duce la aruncarea unei excepții. Este necesară o asemenea coloană în clipa în care se folosește metoda Find pentru proprietatea Rows: în acest caz trebuie să se specifice o coloană pe care avem unicitate.
- *ForeignKeyConstraint* - specifică acțiunea care se va efectua atunci când se șterge sau modifică valoarea dintr–o anumită coloană. De exemplu se poate decide că dacă se sterge o înregistrare dintr-o tabelă atunci să se șteargă și înregistrările copil. Valorile care se pot seta pentru o asemenea constrângere se specifică în proprietățile ForeignKeyConstraint.DeleteRule și ForeignKeyConstraint.UpdateRule:

– Rule.Cascade - acțiunea implicită, șterge sau modifică înregistrările afectate
– Rule.SetNull - se setează valoare de null pentru înregistrările afectate
– Rule.SetDefault - se setează valoarea implicită definită în bază pentru câmpul respectiv
– Rule.None - nu se execută nimic

Exemplu:

```
ForeignKeyConstraint   custOrderFK=new ForeignKeyConstraint
("CustOrderFK",custDS.Tables["CustTable"].Columns["CustomerID"],
custDS.Tables["OrdersTable"].Columns["CustomerID"]);
custOrderFK.DeleteRule = Rule.None;
//Nu se poate sterge un client care are comenzi facute

custDS.Tables["OrdersTable"].Constraints.Add(custOrderFK);
```

Mai sus s-a declarat o relație de tip cheie străinăîntre două tabele(“CustTable” și “OrdersTable”, care fac parte dintr-un DataSet). Restricțiase adaugă la tabla copil.

Stabilirea cheii primare

O cheie primară se definește ca un vector de coloane care se atribuie proprietății PrimaryKey a unei tabele (obiect DataTable).

```
DataColumn[] pk = new DataColumn[1];
pk[0] = myTable.Columns["ID"];
myTable.PrimaryKey = pk;
```

Proprietatea Rows a clasei DataTable permite căutarea unei anumite linii din colecția conținută dacă se specifică un obiect sau un array de obiecte folosite pe post de cheie:

```
object key = 17;//cheia dupa care se face cautarea
DataRow line = myTable.Rows.Find(key);
```

if (line != null) //proceseaza linia

Relații între tabele

Proprietatea Relations a unui obiect de tip DataSet conține o colecție de obiecte de tip DataRelation folosite pentru a figura relațiile de tip părinte–copil între două tabele. Aceste relații se precizează în esență ca niste perechi de array-uri de coloane sau chiar coloane simple din cele două tabele care relaționează, de exemplu sub forma:
myDataSet.Relations.Add(DataColumn, DataColumn);
//sau
myDataSet.Relations.Add(DataColumn[], DataColumn[]);

concret:
myDataSet.Relations.Add(myDataSet.Tables["Customers"].Columns["CustomerID"],
myDataSet.Tables["Orders"].Columns["CustomerID"]);

Scenariul uzual de lucru cu datele dintr-o tabelă conţine următoarele etape:

- popularea succesivă a unui DataSet prin intermediul unuia sau mai multor obiecte DataAdapter, apelând metoda Fill().
- procesarea datelor din DataSet folosind numele tabelelor stabilite la umplere sau indexarea acestora.
- actualizarea datelor prin obiecte comandă corespunzătoare operaţiilor INSERT, UPDATE şi DELETE. Un obiect CommandBuilder poate construi automat o combinaţie de comenzi ce reflectă modificările efectuate.

DataAdapter deschide o conexiune doar atunci când este nevoie şi o închide imediat când aceasta nu mai este necesară.

DataAdapter

Clasele DataAdapter generează obiecte care funcţionează ca o interfaţă între sursa de date şi obiectele DataSet interne aplicaţiei, permiţând prelucrări pe baza de date. Ele gestionează automat conexiunea cu baza de date astfel încât conexiunea să se facă numai atunci când este imperios necesar.

SqlDataAdapter realizează următoarele operaţii atunci când trebuie să populeze un DataSet:

1. deschide conexiunea;
2. populează DataSet-ul;
3. închide conexiunea.

şi următoarele operaţii atunci când trebuie să facă update în baza de date:

1. deschide conexiunea;
2. scrie modificările din DataSet în baza de date;

3. închide conexiunea.

Între operaţiunea de populare a DataSet-ului şi cea de update conexiunile la data source sunt închise. Între aceste operaţii în DataSet se poate scrie sau citi.

//Crearea unui obiect de tipul DataSet se face folosind operatorul *new*.

Un obiect SqlDataAdapter conţine mai multe obiecte SqlCommand şi un obiect SqlConnection pentru a citi şi scrie date.

SqlDataAdapter conţine mai multe obiecte comandă: câte unul pentru inserare, update, delete şi select.

Prin intermediul constructorului putem instanţia doar comanda de interogare.

Instanţierea celorlalte comenzi se face fie prin intermediul proprietăţilor `InsertCommand`, `UpdateCommand` și `DeleteCommand` pe care le expune `SqlDataAdapter`, fie folosind obiecte de tipul `SqlCommandBuilder`.

Iniţializarea unui `SqlCommandBuilder` se face prin apelul unui constructor care primește ca parametru un obiect de tip `SqlDataAdapter` pe care îl folosește la construirea comenzilor de inserare, modificare și ștergere (`InsertCommand`, `UpdateCommand` și `DeleteCommand`). SqlCommandBuilder are limitări: nu poate construi decât comenzi simple şi care se aplică unui singur tabel.

Metode uzuale ale clasei DataAdapter

1. Constructori de la cei impliciți până la cei în care se specifică o comandă de tip SELECT și conexiunea la sursa de date.

2. **Fill**() – metodă polimorfică, permițând umplerea unui tabel al unui obiect de tip DataSet cu date. Permite specificarea obiectului DataSet în care se depun datele, eventual a numelui tabelului din acest DataSet, numărul de înregistrare cu care să se înceapă popularea (prima având indicele 0) și numărul de înregistrări care urmează a fi aduse. Returnează de fiecare dată numărul de înregistrări care au fost aduse din baza de date. În clipa în care se apelează Fill() se procedează astfel:
(a) Se deschide conexiunea (dacă ea nu a fost explicit deschisă)
(b) Se aduc datele și se populează un obiect de tip DataTable din DataSet
(c) Se închide conexiunea (dacă ea nu a fost explicit deschisă!)

De remarcat că un DataAdapter își poate deschide și închide singur conexiunea, dar dacă aceasta a fost deschisă înaintea metodei Fill() atunci tot programatorul trebuie să o închidă.
Exemplu:

```
SqlConnection con;
DataSet ds;
SqlDataAdapter daCl;

  string strConectare="Data Source=FUJITSU_AMILO; "+
                      "Initial Catalog=dbFacturare;"+
```

```
" Integrated Security=SSPI;";


con = new SqlConnection(strConectare);
ds = new DataSet();

string strSelect = "select * from tClienti";

daCl = new SqlDataAdapter(strSelect, Global.con);

daCl.Fill(Global.ds, "Clienti");
```

3. **Update()** – metodă polimorfică, permițând reflectarea modificărilor efectuate într–un DataSet. Pentru a funcționa are nevoie de obiecte de tip comandă adecvate: proprietățile InsertCommand, DeleteCommand și UpdateCommand trebuie să indice către comenzi valide. Returnează de fiecare dată numărul de înregistrări afectate.

Exemplu de utilizare a obiectului de tip SqlCommandBuilder:

```
SqlCommandBuilder cb = new SqlCommandBuilder(Global.daCl);
DataSet dsChange = ds.GetChanges();
if (dsChange != null)
{
     daCl.Update(dsChange, "Clienti");
     ds.AcceptChanges();//accepta schimbarile din dataset

}
```

Proprietăți

1. SelectCommand, InsertCommand, UpdateCommand, DeleteCommand – de tip Command, conțin comenzile ce se execută pentru selectarea sau modificarea datelor în sursa de date. Măcar proprietatea SelectCommand trebuie să indice către un obiect valid, pentru a se putea face popularea setului de date.

2. MissingSchemaAction – de tip enumerare MissingSchemaAction, determină ce se face atunci când datele care sunt aduse nu se potrivesc peste schema tablei în care sunt depuse. Poate avea următoarele valori:
• Add - implicit, DataAdapter adaugă coloana la schema tablei
• AddWithKey - ca mai sus, dar adaugă și informații relativ la cine este cheia primară
• Ignore - se ignoră lipsa coloanei respective, ceea ce duce la pierdere de date pe DataSet
• Error - se generează o excepție de tipul InvalidOperationException.

Controlul DataGridView

Controlul DataGridView ofera o modalitate puternica si flexibila de a afisa informatii intr-o forma tabelara. Putem folosi controlul DataGridView pentru a afisa vederi read-only ale unui volum mic de date, cat si ale unor seturi de date foarte mari.

Putem extinde controlul DataGridView la numeroase modalitati de customizare a aplicatiei noastre. De exemplu, putem specifica din cod algoritmii proprii de sortare pentru datele din grid, sau ne putem crea propriile tipuri de celule. De asemenea, putem schimba design-ul acestuia prin setarea catorva proprietati.