

SEMINAR 1- teorie slide-uri, Aplicatie Entity Framework/LINQ (conexiune la baza de date directa, prin meniul View din C#), Aplicatie ConsoleWindow, Aplicatie SqlDataReader, SqlDataAdapter.

ADO.NET – slide 1

ADO.NET = set (multime) de clase care permite programatorilor .NET Framework sa interactioneze cu o multime de surse de date – baze de date (SQL Server, Oracle, ...), fisiere XML, (surse de date expuse prin OLE DB si ODBC). Aplicatiile data-sharing consumer pot folosi ADO.NET pentru a se conecta la aceste surse de date si pentru a prelua, manipula și actualiza datele pe care le conțin.

ADO.NET include .Net Framework data provider = class library ce ne permite sa ne conectam la o baza de date, sa executam comenzi pe baza de date si sa returneze rezultatele executiilor acelor comenzi. Aceste rezultate pot fi procesate direct, folosind ADO.NET cu obiecte de tip DataSet. Obiectele DataSet pot fi utilizate si independent de .NET Framework data provider pentru a gestiona datele aplicatiei sau cele din XML.

Clasele ADO.NET se gasesc in System.Data.dll si sunt integrate cu clasele XML din System.Xml.dll. Un posibil cod care permite conectarea la baza de date, preluarea datelor din aceasta si afisarea lor este o aplicatie de tip Console Window (exemplul de la sfarsit).

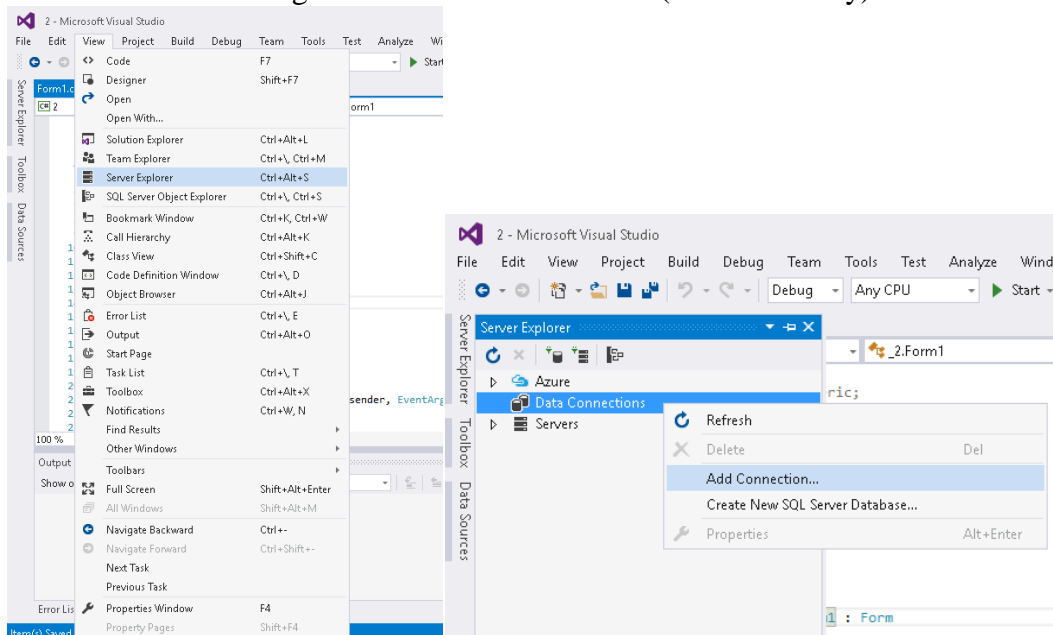
Preluarea datelor din baza de date se poate realiza cu ajutorul tehnologiilor ADO.NET:

- ADO.NET data providers:
 - SqlConnection (System.Data.SqlClient)
 - OleDb (System.Data.OleDb)
 - Odbc (System.Data.Odbc)
 - OracleClient (System.Data.OracleClient)
- ADO.NET Entity Framework:
 - LINQ to Entities
 - Typed ObjectQuery
 - EntityClient (System.Data.EntityClient)
- LINQ to SQL

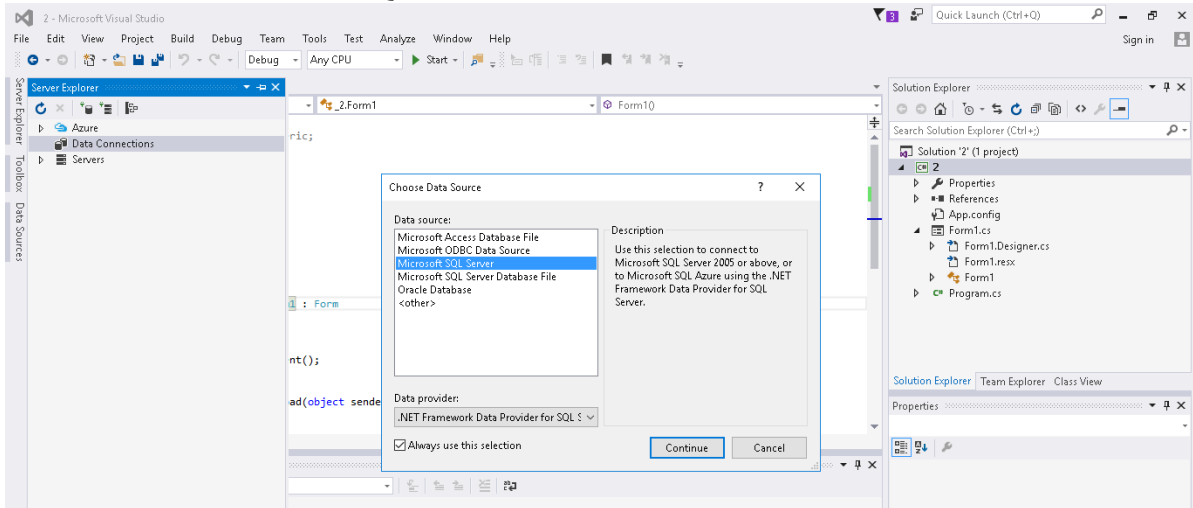
ADO.NET aduce functionalitati programatorilor care scriu cod similar celui component object model (COM) pentru ActiveX Data Objects (ADO). ADO.NET ne da cea mai directa metoda de acces la date cu ajutorul lui .NET Framework.

ADO.NET Entity Framework – baza de date adaugata la aplicatie (Console Window/Windows/Windows Forms Application)

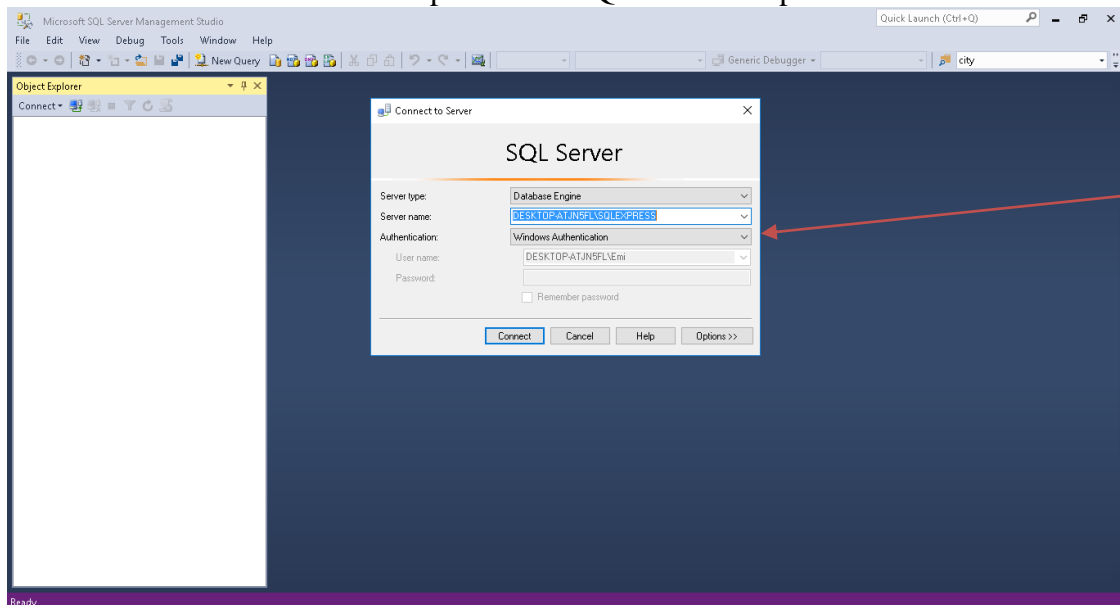
- To add/verify the connection to SQL Server can be use the menu View-> Server Explorer -> Data Connection -> right click -> Add connection (not necessarily)



OR directly from the left menu Server Explorer -> Data Connections -> right click -> Add connection. Choose Microsoft SQL Server -> Continue



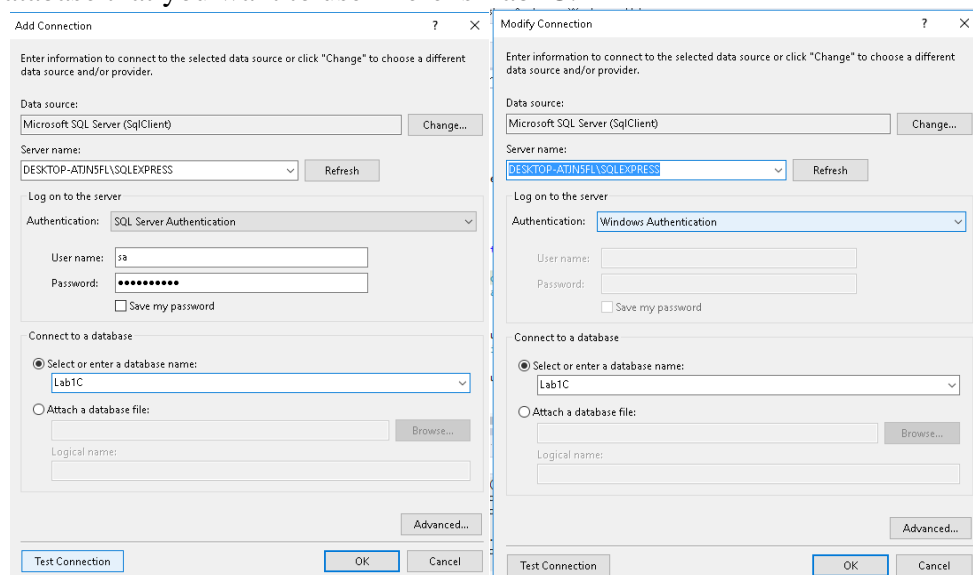
The name of the server is the one that opens when SQL Server is opened.



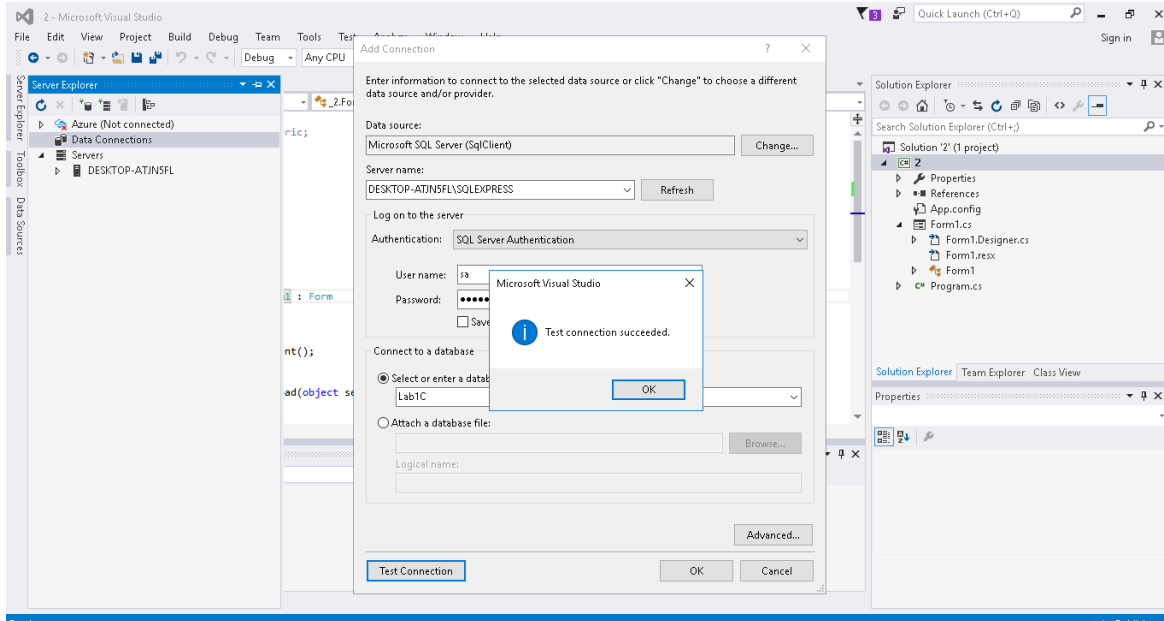
In this case: Server name is: DESKTOP-ATJN5FL\SQLEXPRESS

You can work with SQL Server Authentication or Windows Authentication (recommended).

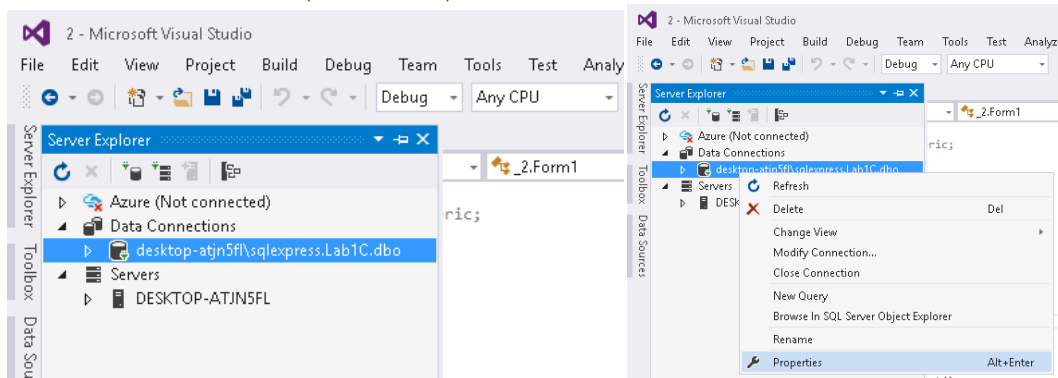
Select the database that you want to use- Here is Lab1C.



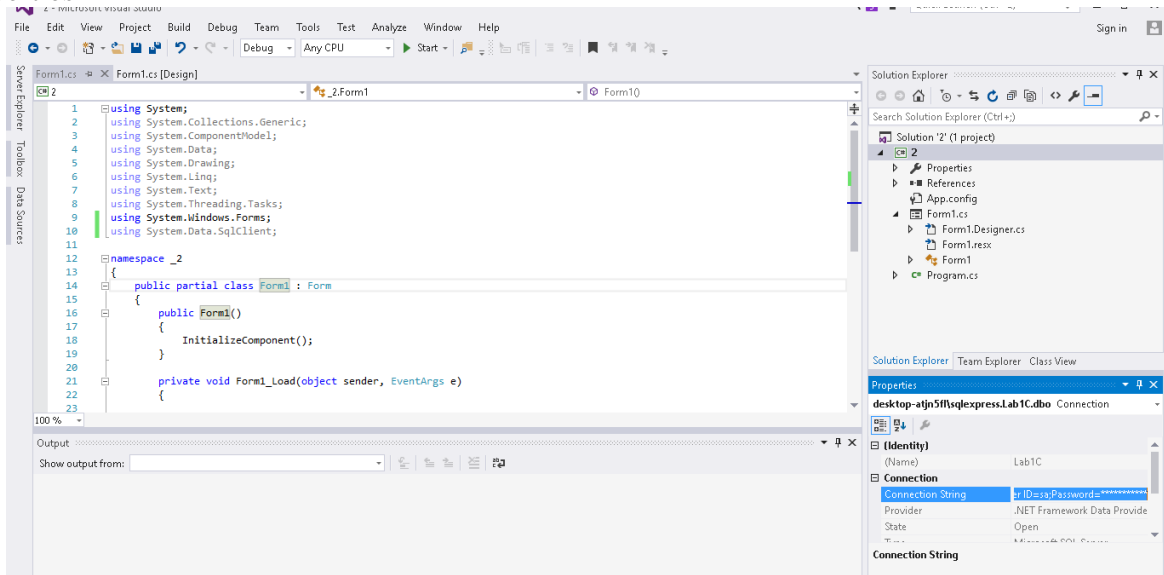
Test Connection -> ok



So, you have a new connection (also tested)



Take the connection string (we use it as a code): right click on the connection established-> Properties



Connection string for the SQL Server Authentication is: Data Source=DESKTOP-ATJN5FL\SQLEXPRESS;Initial Catalog=Lab1C;User ID=sa;Password=*****

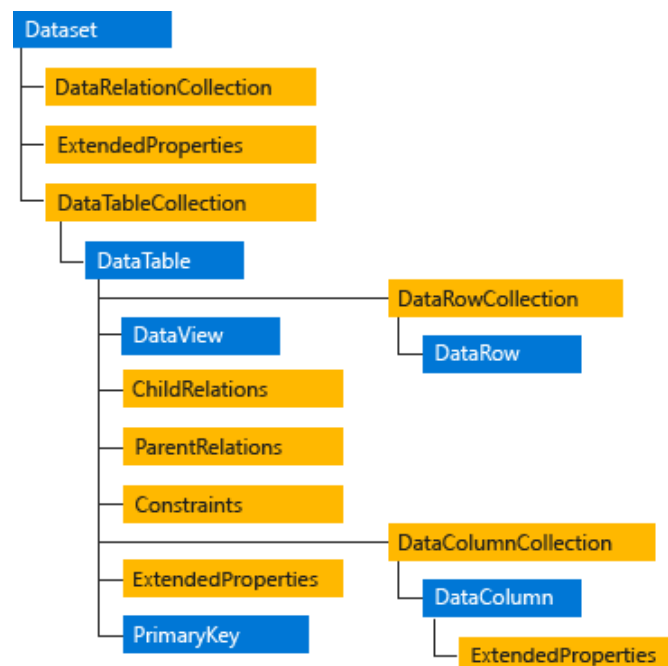
Connection string for Windows Authentication is (we use this): Data Source=DESKTOP-ATJN5FL\SQLEXPRESS;Initial Catalog=Lab1C;Integrated Security=True

CORRECT ONE (with \\): Data Source=DESKTOP-ATJN5FL\\SQLEXPRESS;Initial Catalog=Lab1C;Integrated Security=True

Clasele ce compun providerul sunt SqlConnection, SqlCommand, SqlDataReader, SqlDataAdapter. DataSet se gaseste in namespace System.Data si contine colectie de DataTable si DataRelation. Componente ADO.NET pentru accesare si manipulare date: .NET Framework data providers si DataSet.

1. .NET Framework Data Providers – componente pentru manipularea rapida a datelor, doar transmiterea lor si acces read-only. Obiectul **Connection** se foloseste pentru conectarea la sursa de date. Obiectul **Command** permite accesul la comenzile bazei de date in vederea returnarii datelor, modificarii acestora, executiei procedurilor stocate, trimiterii si primirii de parametrii. Obiectul **DataReader** e folosit la fluxul de date din sursa de date. Obiectul **DataAdapter** functioneaza ca un pod intre obiectele DataSet si sursa de date. DataAdapter foloseste obiecte Command pentru a executa comenzi SQL la sursa de date, atat pentru a incarca DataSet cu date cat si a procesa modificarile realizate asupra datelor din DataSet la sursa de date.
2. DataSet – (ADO.NET DataSet) – se foloseste pentru acces la date independent de sursa lor. Se poate folosi pentru diverse si multiple surse de date (precum date XML) sau se poate folosi pentru a gestiona datele locale ale aplicatiei. Contine o colectie de obiecte DataTable, care au randuri, coloane, chei primare, chei externe, constrangeri si relaționează informațiile despre datele din obiectele DataTable.

Diagrama ilustreaza relatia dintre .NET Framework data provider si DataSet (ADO.NET architecture).



DataReader sau DataSet - Cand realizam o aplicatie si trebuie sa alegem folosirea lui DataReader sau DataSet trebuie sa tinem cont de tipul functionalitatii pe care o doreste aplicatia.

DataSet – se foloseste cand:

- Datele se gasesc in memoria cache locala (astfel incat sa poata fi gestionate). Daca este necesara doar citirea rezultatelor unui query, DataReader este alegerea perfecta
- Avem date la distanta pe nivele diferite sau provin din servicii XML Web
- Interacționează dinamic cu datele, precum ar fi legarea (Bind) la un control Windows Forms sau combina și coreleaza datele din mai multe surse

- Proceaseaza datele fara a necesita o conexiune deschisa la sursa de date (care ingheata conexiunea pentru ceilalti clienti)

Cu `DataReader` datele se returneaza doar intr-o directie read-only.

De asemenea, `DataAdapter` foloseste `DataReader` pentru a popula `DataSet` (utilizand `DataReader`, si ajutand la cresterea performantei – salvare memorie, care ar fi folosita de `DataSet`, si evitarea procesarii cerute de populare pentru `DataSet`).

LINQ to DataSet – ofera posibilitatea interogarii si verificarii timpului de compilare pentru datele stocate in memoria cache a obiectelor `DataSet`. Permite scrierea interogarilor in limbajul .NET Framework (C#, Visual Basic).

LINQ to SQL – accepta interogari chiar daca modelul obiectului este mapat la o structura relationala de baze de date fara a folosi modelul conceptual intermediar. Fiecare tabel este reprezentat ca si o clasa separata legat de modelul schemei relationale a bazei de date. LINQ to SQL traduce interogările în limbajul modelului obiect din Transact-SQL și le trimite în baza de date pentru execuție. Când baza de date returnează rezultatele, LINQ pentru SQL duce rezultatele înapoi în obiecte.

Connect to a Database (LINQ to SQL) - se realizeaza cu ajutorul lui `DataContext` returnand obiecte si triminand schimbarile inapoi. `DataContext` se initializeaza cu o conexiune (connection string) data. Ea traduce cerintele pentru obiecte din interogari SQL si apoi returneaza rezultatele. Permite folosirea LINQ (Language-Integrated Query) prin implementare ca si cel sql standard pentru obiecte (exemplu, where, select)

Exemplu in care `DataContext` se foloseste pentru a se conecta la o baza de date si a returna inregistrările cu anumita proprietate.

```
// DataContext takes a connection string.
DataContext db = new DataContext(@"c:\Northwnd.mdf");
// Get a typed table to run queries.
Table<Customer> Customers = db.GetTable<Customer>();
// Query for customers from London.
var query =
    from cust in Customers
    where cust.City == "London"
    select cust;
foreach (var cust in query)
    Console.WriteLine("id = {0}, City = {1}", cust.CustomerID, cust.City);
// public class
public partial class Northwind : DataContext
{
    public Table<Customer> Customers;
    public Table<Order> Orders;
    public Northwind(string connection) : base(connection) { }
}
```

LINQ to Entities - se refera la instante

ADO.NET Entity Framework – se foloseste pentru a permite programatorilor sa creeze aplicatii de acces la baze de date.

WCF Data Services – folosit pentru a implementa servicii de date pentru Web/intranet. Structura datelor este cea a entitatilor si a relatiilor ca si in Entity Data Model.

XML si ADO.NET- tind catre obiecte DataSet. Aceste obiecte se pot popula cu ajutorul surselor XML.

Ciclul datelor – slide 2

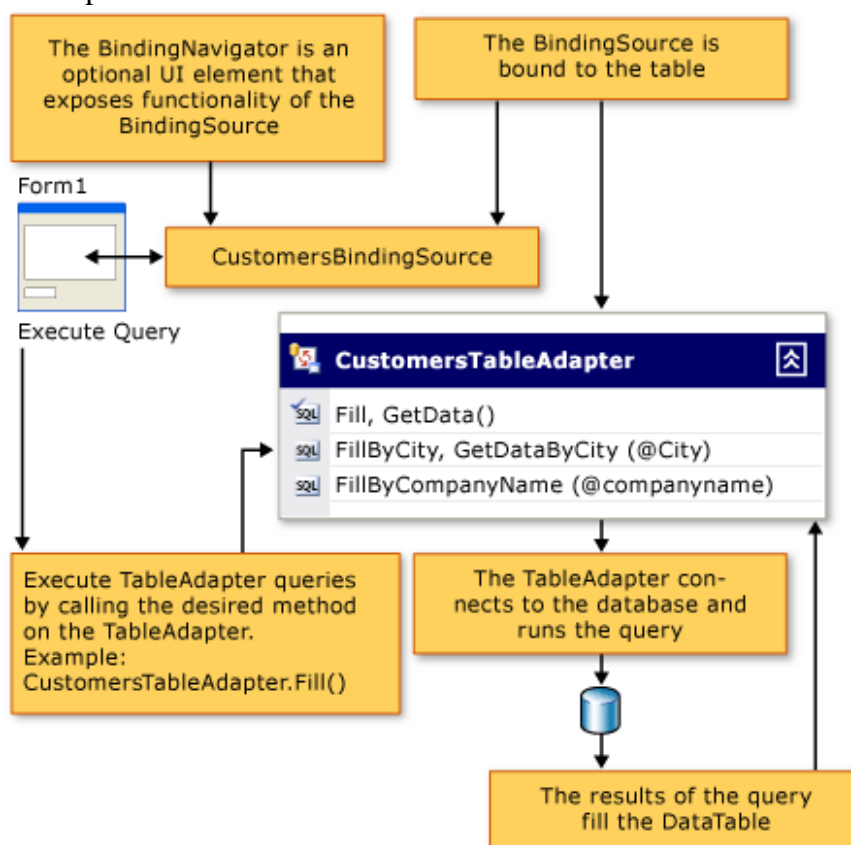
Ciclul de viata pentru o aplicatie cu date:

- Conectarea la baza de date
- Pregatirea aplicatiei pentru primirea datelor (sunt obiecte care retin temporar datele)
- Aducerea datelor in aplicatie (adica, executia procedurilor stocate, a interogariilor sql pentru aducerea locala a datelor)
- Afisarea datelor (cotroale data-bound- pentru legare)
- Editarea si validarea datelor (insert, update, delete)
- Salvarea datelor (propagarea modificarilor in baza de date)

1. Conectarea la date

TableAdapter - componente generate de designer care se conectează la o bază de date, execută interogări sau proceduri stocate și completează DataTable cu datele returnate. TableAdapters trimite datele actualizate din aplicația înapoi în baza de date. Se pot executa oricate interogari pe tabelă atata timp cat datele returnate coincid cu schema tabelii pentru care este asociat TableAdapter.

Interactiune TableAdapters cu bazele de date si obiectele din memorie:



TableAdapters sunt realizati cu DataSetDesigner , iar clasele corespunzatoare nu sunt generate ca si clase imbricate ale DataSet. Sunt localizate diferit pentru fiecare DataSet.

De exemplu: DataSet NorthwindDataSet are asociat ca si numele NorthwindDataSetTableAdapters

```
NorthwindDataSet northwindDataSet = new NorthwindDataSet();
```

```
NorthwindDataSetTableAdapters.CustomersTableAdapter customersTableAdapter =  
    new NorthwindDataSetTableAdapters.CustomersTableAdapter();  
customersTableAdapter.Fill(northwindDataSet.Customers);
```

Metode si proprietati: clasa TableAdapter nu e parte a .NET Framework.

Metoda	Descriere
TableAdapter.Fill	Populeaza TableAdapter cu rezultatul comenzii select
TableAdapter.Update	Trimite modificarile bazei de date si returneaza un numar (int) care reprezinta numarul de randuri modificate la update
TableAdapter.GetData	Returneaza un nou DataTable cu date
TableAdapter.Insert	Creaza un nou rand in tabelul de date
TableAdapter.ClearBeforeFill	Verifica daca tabelul de date este gol inainte de a se apela metoda Fill

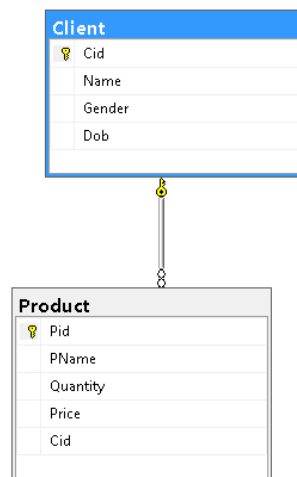
ObjectContext – clasa--- exemplu pe modelul [AdventureWorks Sales Model](#) (construcie obiect).

```
// Create the ObjectContext.
ObjectContext context = new ObjectContext("name=AdventureWorksEntities");
// Set the DefaultContainerName for the ObjectContext.
// When DefaultContainerName is set, the Entity Framework only
// searches for the type in the specified container.
// Note that if a type is defined only once in the metadata workspace
// you do not have to set the DefaultContainerName.
context.DefaultContainerName = "AdventureWorksEntities";
ObjectSet<Product> query = context.CreateObjectSet<Product>();
// Iterate through the collection of Products.
foreach (Product result in query)
    Console.WriteLine("Product Name: {0}", result.Name);
```

DataSet - using.System.Data

- contine colectie de DataTable (DataRow=date, DataColumn=schema tablei, DataConstraint) si DataRelation.

Exemple si aplicatii: In tot ceea ce urmeaza se va utiliza baza de date Lab1C



SqlConnection: using System.Data.SqlClient;

```
SqlConnection cs = new SqlConnection("Data Source=DESKTOP-ATJN5FL\\SQLEXPRESS;Initial Catalog=Lab1C;Integrated Security=True");
```

OR

```
SqlConnection cs = new SqlConnection(@"Data Source=DESKTOP-ATJN5FL\\SQLEXPRESS;Initial Catalog=Lab1C;Integrated Security=True");
```

OR

```
SqlConnection cs = new SqlConnection(@"Data Source=DESKTOP-ATJN5FL\SQLEXPRESS;Initial Catalog=Lab1C;User Id=sa; Password=YOURPASSWORD");
```

Metode: cs.Open(); ... cs.Close(); - exceptie- metoda Fill() care are inclusa operatia de deschidere si inchidere a conexiunii.

Pid	PName	Quantity
2	cheese	3
4	lemon	5
5	grapes red	12
6	chocolate	5
7	ice cream	2
9	orange	1
13	tiramisu	4
14	tiramisu	4

SqlCommand: CommandText, CommandType, CommandTimeout.

```
SqlCommand cmd = new SqlCommand("select count(*) from Product", cs);
cs.Open();
cmd.CommandType = CommandType.Text; //implicit
string se = cmd.ExecuteScalar().ToString();
textBox5.Text = se;
cs.Close();
```

Metode:

ExecuteNonQuery() – returneaza numarul de randuri afectate

ExecuteScalar() – returneaza valoarea primei coloane a primului rand

ExecuteReader() – construiește un SqlDataReader

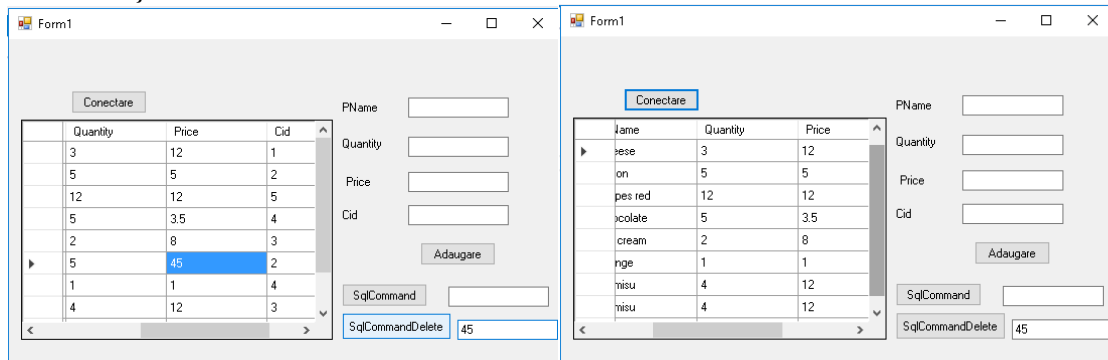
```
string css = "Data Source=DESKTOP-ATJN5FL\SQLEXPRESS;Initial Catalog=Lab1C;Integrated Security=True";
string sql = "DELETE FROM Product WHERE Price =@p";
using (SqlConnection conn = new SqlConnection(css))
{
    SqlCommand cmd = new SqlCommand(sql, conn);
    cmd.Parameters.Add("@p", SqlDbType.Int);
    cmd.Parameters["@p"].Value = Int32.Parse(textBox6.Text);
    try
    {
```



```

        conn.Open();
        Int32 count = (Int32)cmd.ExecuteScalar();
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
        //conn.Close();
    }
}

```

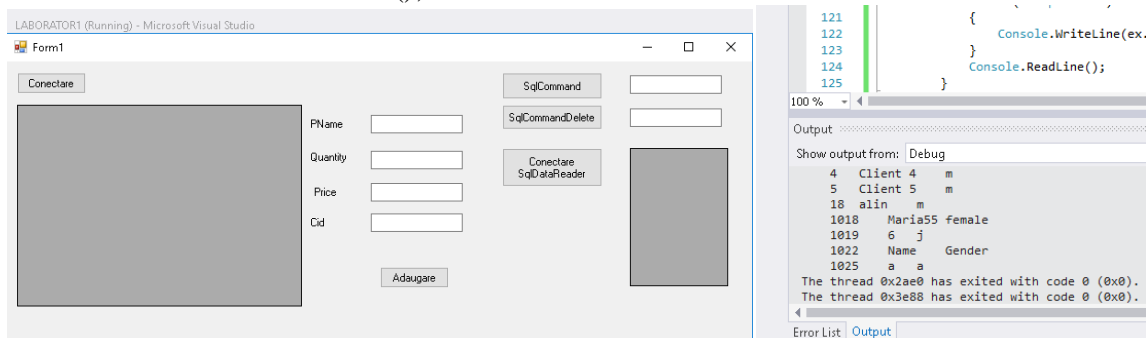


SqlDataReader – flux de date read-only si forward-only (doar intr-o singura directie)

```

SqlCommand cmd = new SqlCommand("SELECT * FROM Client", cs);
cmd.CommandText = "SELECT * FROM Client";
try
{
    cs.Open();
    SqlDataReader sdr = cmd.ExecuteReader();
    while (sdr.Read())
    {
        Console.WriteLine("\t{0}\t{1}\t{2}", sdr[0], sdr[1], sdr[2]);
    }
    sdr.Close();
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
Console.ReadLine();

```



SqlDataAdapter – are Connection

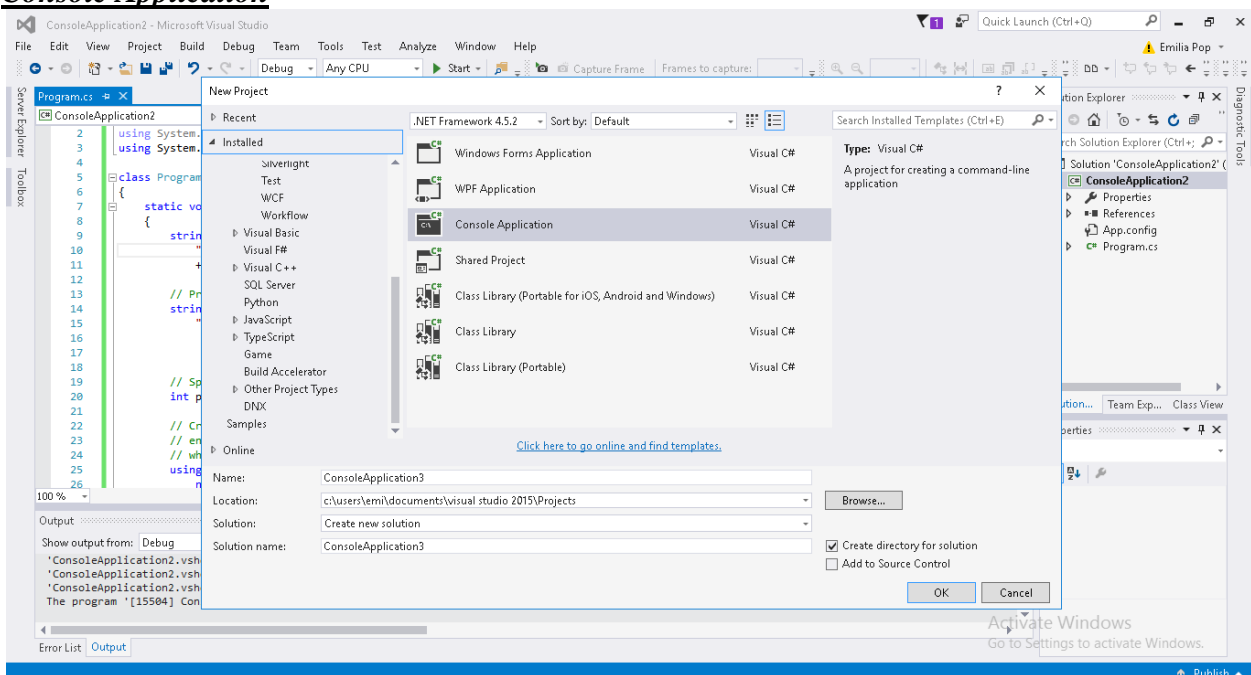
- Metode: Fill(DataSet ds, „nume_tabel”), Update(DataSet ds, „nume_tabel”)
- Proprietate: .SelectCommand, .InsertCommand, .UpdateCommand, .DeleteCommand.

Aplicatie:

```
DataSet ds1 = new DataSet();
SqlDataAdapter da1 = new SqlDataAdapter("Select * from Client", cs);
da1.Fill(ds1, "Client");
dataGridView2.DataSource = ds1.Tables["Client"];
DataRow dr = ds1.Tables["Client"].NewRow();
dr["Name"] = "Mihai";
dr["Gender"] = "male";
dr["Dob"] = "12/12/2012";
ds1.Tables["Client"].Rows.Add(dr);
```

Cid	Name
1018	Maria55
1019	6
1022	Name
1025	a
	Mihai

Console Application



Code:

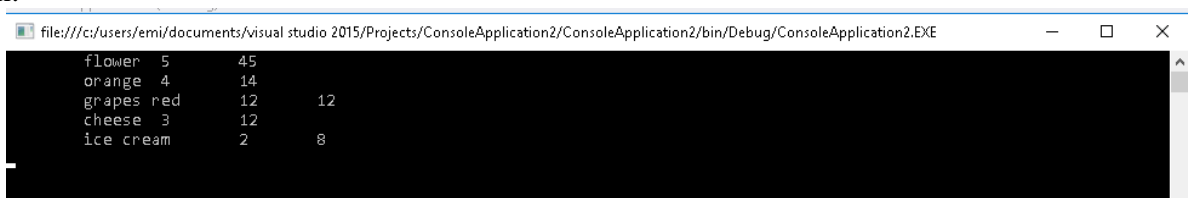
```
using System;
using System.Data;
using System.Data.SqlClient;

class Program
{
    static void Main()
    {
        string connectionString = "Data Source=DESKTOP-ATJN5FL\\SQLEXPRESS;Initial
Catalog=Lab1C;" + "Integrated Security=true";
        // Provide the query string with a parameter placeholder.
```

```

string queryString = "select PName, Quantity, Price from Product "
    + "where Price>@p " + "ORDER BY Price DESC;";
// Specify the parameter value.
int paramValue = 5;
// Create and open the connection in a using block. This ensures that all resources will be
// closed and disposed when the code exits.
using (SqlConnection connection =
    new SqlConnection(connectionString))
{
    // Create the Command and Parameter objects.
    SqlCommand command = new SqlCommand(queryString, connection);
    command.Parameters.AddWithValue("@p", paramValue);
    // Open the connection in a try/catch block.
    // Create and execute the DataReader, writing the result set to the console window.
    try
    {
        connection.Open();
        SqlDataReader reader = command.ExecuteReader();
        while (reader.Read())
        {
            Console.WriteLine("{0}\t{1}\t{2}", reader[0], reader[1], reader[2]);
        }
        reader.Close();
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    Console.ReadLine();
}
}
}
Run:

```

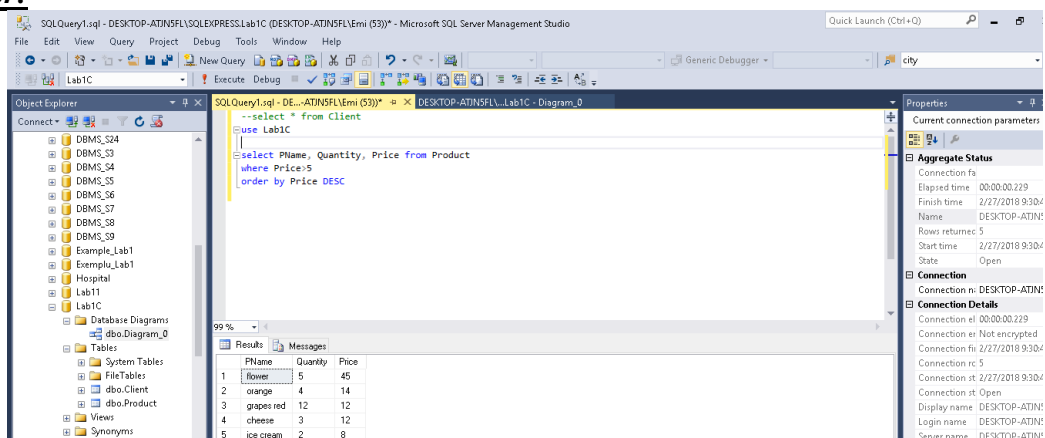


```

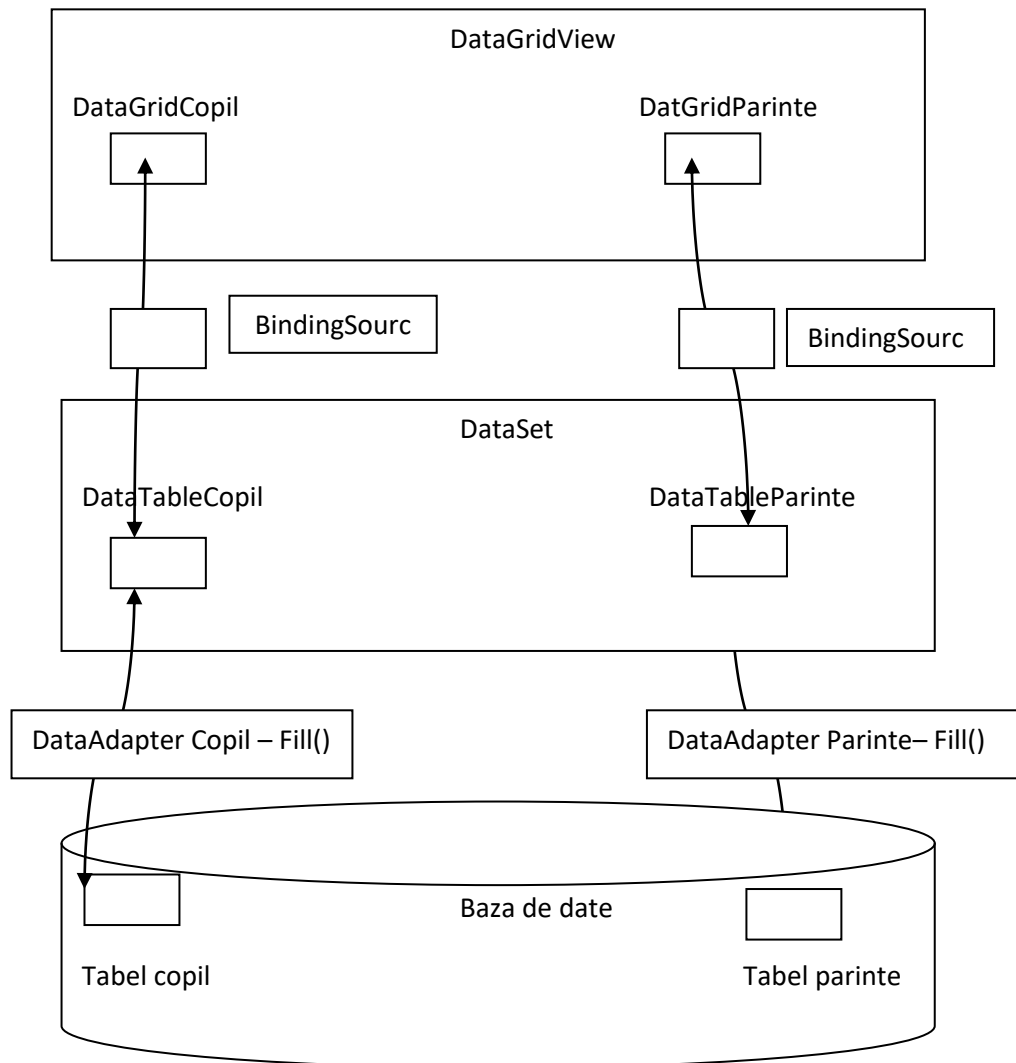
file:///c:/users/emi/documents/visual studio 2015/Projects/ConsoleApplication2/ConsoleApplication2/bin/Debug/ConsoleApplication2.EXE
flower 5 45
orange 4 14
grapes red 12 12
cheese 3 12
ice cream 2 8

```

Sql Server:



SCHEMA SqlDataAdapter



Bibliografy:

<https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ado-net-overview>
<https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ado-net-code-examples>
<https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ado-net-architecture>
<https://msdn.microsoft.com/en-us/library/ms171919.aspx>
[https://msdn.microsoft.com/en-us/library/system.data.objects.objectcontext\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.data.objects.objectcontext(v=vs.110).aspx)
[https://msdn.microsoft.com/en-us/library/bb399375\(v=vs.100\).aspx](https://msdn.microsoft.com/en-us/library/bb399375(v=vs.100).aspx)