

ADO.NET

Seminar 1

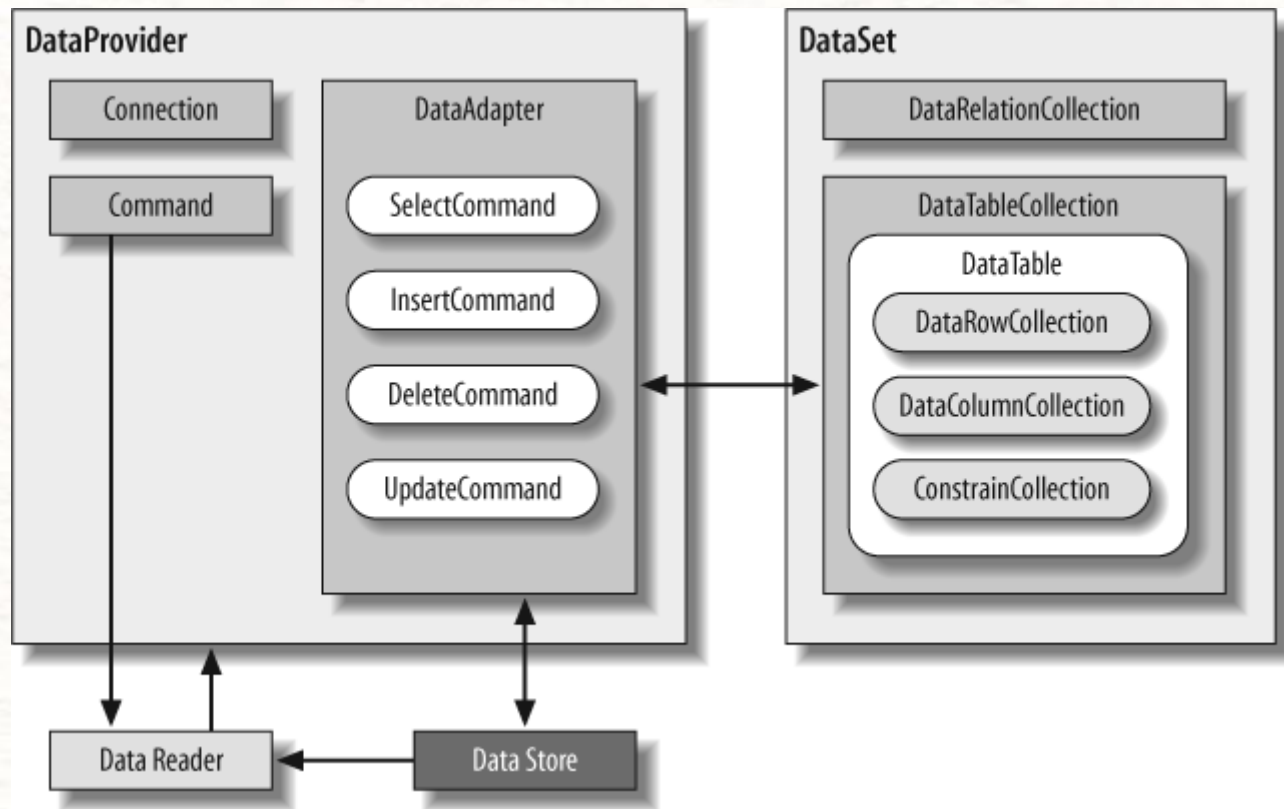
Ce este și ce oferă ADO.NET

- Este un **set de clase** care expun **servicii de acces a datelor** pentru programatorii .NET
- Este o parte **integrantă** a **.NET Framework**, care oferă **acces la date** provenite din **surse de date** cum ar fi **XML** sau **SQL Server**

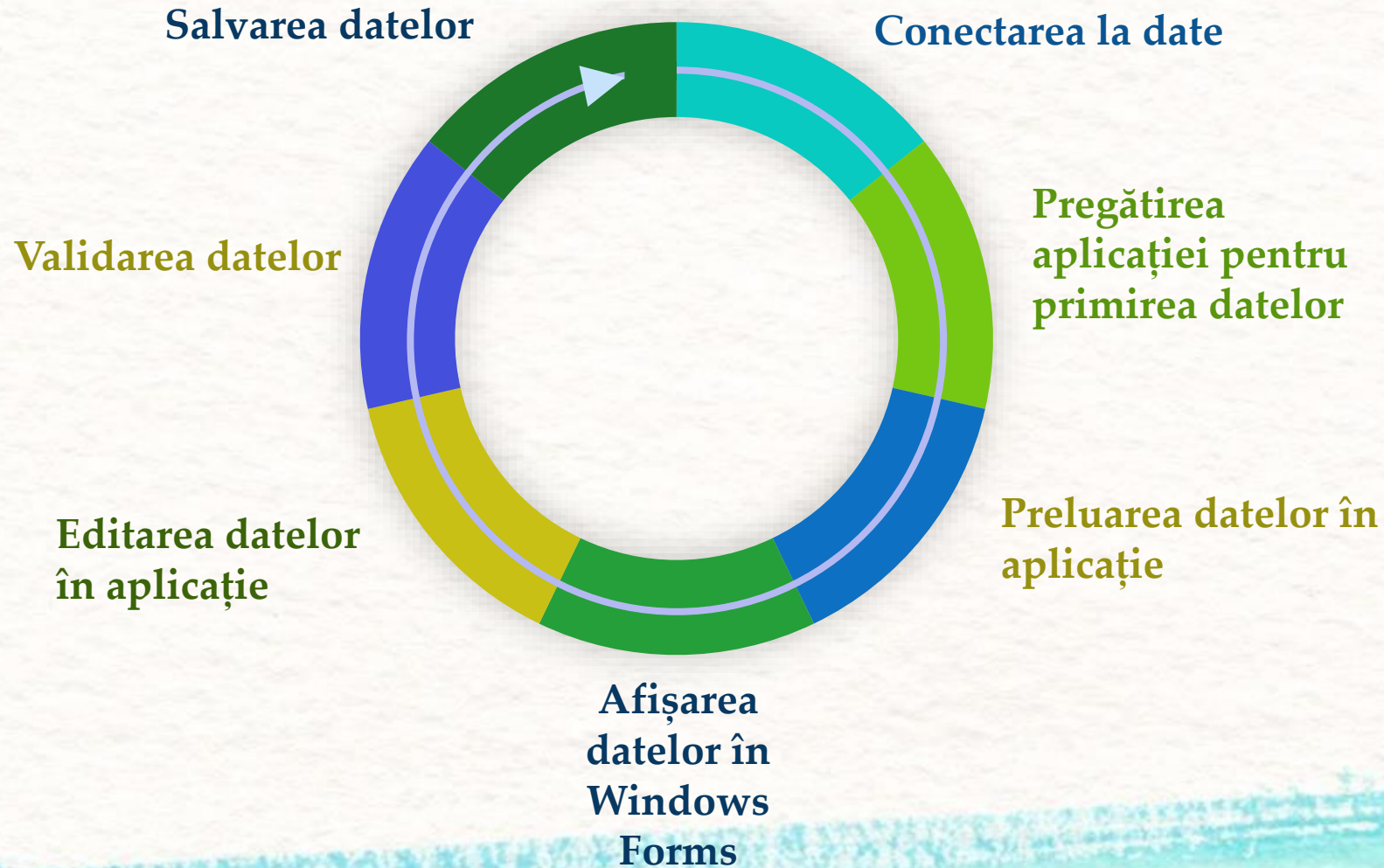
Ce este și ce oferă ADO.NET

- Oferă un set bogat de **componente** pentru **crearea aplicațiilor distribuite**, care **partajează date**
- Include **.NET Framework data providers** pentru **conectare la baza de date, execuție comenzi și returnare a rezultatelor**
- Namespace-ul **System.Data.SqlClient** este **.NET Framework data provider** pentru **SQL Server**
 - Pentru versiunile noi ale framework-ului (.NET 5, .NET 6, etc.), pachetul **Microsoft.Data.SqlClient** este **data provider** pentru **SQL Server**

Diagrama arhitecturii ADO.NET



Ciclul datelor



Ciclul datelor

- **Conectarea la date:** stabilește o comunicare bidirecțională între aplicație și server-ul de baze de date
- **Pregătirea aplicației pentru primirea datelor:** când se utilizează un model deconectat, anumite obiecte stochează datele temporar (*dataset-uri, entități, obiecte LINQ to SQL*)
- **Preluarea datelor în aplicație:** execuția interogărilor și a procedurilor stocate

Ciclul datelor

- **Afișarea datelor în Windows Forms:** se utilizează controale conectate la date (*data-bound*)
- **Exemple de controale conectate la date:**
 - *DataGridView*
 - *TextBox*
 - *Label*
 - *ComboBox*

Ciclul datelor

- **Editarea și validarea datelor în aplicație:**
 - adăugarea/modificarea/ștergerea înregistrărilor
 - verificarea noilor valori (acestea din urmă **trebuie** să îndeplinească cerințele aplicației)
- **Salvarea datelor:** persistarea modificărilor în baza de date
 - *Update(DataSet, String)*
 - *ExecuteNonQuery()*

Modele de date

- DataSet-uri tipizate/netipizate (*typed/untyped*)
- Un **DataSet tipizat** este o clasă care:
 - derivă din clasa *DataSet*
 - moștenește toate metodele, evenimentele și proprietățile ei
 - oferă metode, evenimente și proprietăți puternic tipizate

Modele de date

- Model conceptual bazat pe *Entity Data Model* care poate fi utilizat de *Entity Framework* sau *WCF Data Services*
 - **Entity Framework** este un *object-relational mapper* care permite lucrul cu date provenite dintr-un sistem relațional folosind obiecte specifice domeniului
 - **WCF Data Services** oferă posibilitatea de a crea/utiliza servicii de date pe Web sau intranet

Modele de date

- Clase *LINQ to SQL*
 - Suportă interogări pe un model obiect care corespunde structurii bazei de date relaționale fără a folosi un model conceptual intermediar
 - Transformă interogările *language-integrated* din modelul obiect în Transact-SQL și le transmite bazei de date pentru execuție

Exemplu de clasă LINQ to SQL

```
[Table(Name="Cadouri")]
public class Cadou
{
    [Column(Name="cod_cadou",IsPrimaryKey=true,IsDbGenerated=true)]
    public int CodC;
    [Column(Name="descriere")]
    public string Descriere;
    [Column(Name="posesor")]
    public string Posesor;
    [Column(Name="pret")]
    public float Pret;
}
```


DataSet

- **DataSet**-urile sunt obiecte care conțin **data tables** în care se pot stoca **temporar** date spre a fi **utilizate** în aplicație
- Oferă un **cache local**, în **memorie**, al datelor
- **Funcționează** chiar dacă **aplicația** este **deconectată** de la **baza de date**

DataSet

- Structura unui *DataSet* este similară cu cea a unei baze de date relaționale deoarece conține:
 - Tabele
 - Înregistrări
 - Coloane
 - Constrângeri
 - Relații

DataSet

- **Proprietăți:**

- **Tables** – returnează **colecția de tabele** incluse în *DataSet*
- **Relations** – returnează **colecția de relații** care conectează tabelele și permit **navigarea** dinspre **tabelele părinte** spre **tabelele copil**

DataSet

- **Metode:**

- **Clear()** – șterge toate **datele** stocate în **tabelele** din *DataSet*
- **HasChanges()** – returnează o **valoare** care indică dacă *DataSet*-ul conține **modificări**, inclusiv înregistrări noi, modificate sau șterse

DataSet

- Clasa **DataSet** include:
 - DataTableCollection
 - DataRelationCollection
- Clasa **DataTable** include:
 - DataRowCollection
 - DataColumnCollection
 - ChildRelations
 - ParentRelations

DataSet

- Clasa **DataRow** include proprietatea **RowState**, care are următoarele valori posibile:
 - *Deleted*
 - *Modified*
 - *Added*
 - *Unchanged*

Clasa Console

- Clasa **Console** reprezintă **stream**-urile de **intrare**, de **ieșire** și de **eroare** pentru aplicațiile de tip **consolă**
- **Proprietăți:**
 - **WindowLeft** – returnează sau setează cea mai din stânga poziție a ferestrei consolei relativ la buffer-ul ecranului
 - **WindowTop** – returnează sau setează cea mai de sus poziție a ferestrei consolei relativ la buffer-ul ecranului

Clasa Console

- **WindowHeight** – returnează sau setează înălțimea ferestrei consolei
- **WindowWidth** – returnează sau setează lățimea ferestrei consolei
- **Title** – returnează sau setează titlul afișat în bara de titlu a consolei
- **BackgroundColor** – returnează sau setează culoarea de fundal a consolei

Clasa Console

- **Metode:**

- **Write(...)**

- **Write(String)** – scrie valoarea de tip *string* specificată în stream-ul standard de ieșire

- **WriteLine(...)**

- **WriteLine(String)** – scrie valoarea de tip *string* specificată urmată de *current line terminator* în stream-ul standard de ieșire

Clasa Console

- **Read()** – citește următorul caracter din stream-ul standard de intrare
- **ReadLine()** – citește următoarea linie de caractere din stream-ul standard de intrare
- **ReadKey()** – obține următorul caracter sau tastă function apăsată de către utilizator
- **Clear()** – șterge *buffer*-ul consolei și informația afișată în fereastra consolei

SqlConnection

- Reprezintă o conexiune deschisă la baza de date
- Nu poate fi moștenită
- Dacă iese din domeniul de vizibilitate, **nu** este închisă
(conexiunea **trebuie** închisă în mod **explicit**)

SqlConnection

- **Proprietăți:**

- **ConnectionString** – returnează sau setează string-ul folosit pentru a deschide o bază de date SQL Server
- **ConnectionTimeout** – returnează timpul de așteptare pentru stabilirea unei conexiuni (la expirare, încercarea de stabilire a conexiunii se termină și este generată o eroare)

SqlConnection

- **Metode**

- **Open()** – deschide o conexiune la baza de date cu setările specificate în *ConnectionString*
 - **Close()** – închide conexiunea la baza de date
- Dacă este generat un *SqlException*, *SqlConnection* rămâne deschisă dacă **nivelul de severitate** al erorii este ≤ 19

SqlCommand

- Reprezintă o **instrucțiune** sau **procedură stocată** Transact-SQL care se dorește a fi executată pe o bază de date SQL Server
- **Proprietăți:**
 - **CommandText** – returnează sau setează instrucțiunea Transact-SQL, numele tabelului sau a procedurii stocate ce urmează să fie executată la nivelul sursei de date
 - **CommandTimeout** – returnează sau setează timpul de așteptare pentru execuția unei comenzi (la expirare, încercarea de execuție a comenzii se termină și este generată o eroare)

SqlCommand

- **Metode:**

- **ExecuteNonQuery()** – execută o instrucțiune Transact-SQL folosind o conexiune și returnează numărul de înregistrări afectate
- **ExecuteScalar()** – execută interogarea și returnează valoarea primei coloane a primei înregistrări din result-set-ul returnat de către interogare (coloanele și înregistrările adiționale sunt ignorate)
- **ExecuteReader()** – construiește un *SqlDataReader*

SqlDataReader

- **SqlDataReader** citește un **stream** *forward-only* de înregistrări dintr-o bază de date SQL Server
- Pentru a crea un *SqlDataReader*, se va apela metoda *ExecuteReader* a unui obiect *SqlCommand* în locul folosirii directe a unui constructor
- În timpul utilizării unui *SqlDataReader*, *SqlConnection*-ul asociat este **ocupat** și nicio altă operație **nu** poate fi aplicată pe *SqlConnection*, în afară de închiderea sa

SqlDataReader

- Singurele **proprietăți** care pot fi **accesate** după ce *SqlDataReader*-ul a fost **închis** sunt *IsClosed* și *RecordsAffected*
- **Modificările** efectuate într-un result-set de către alt **proces** sau **fir de execuție** în timp ce datele sunt **citite** pot fi **vizibile** pentru utilizatorul *SqlDataReader*-ului (totuși, comportamentul exact este **dependent de sincronizare**)

SqlDataAdapter

- Este o **punte** între *DataSet* și SQL Server pentru **obținerea** și **salvarea** datelor
- Atunci când un *SqlDataAdapter* **umple** un *DataSet*, creează **tabelele** și **coloanele** necesare pentru datele **returnate**, dacă acestea **nu există** deja
- *SqlDataAdapter* este folosit **împreună** cu *SqlConnection* și *SqlCommand* pentru a **îmbunătăți** performanța în cazul **conectării** la o bază de date **SQL Server**

SqlDataAdapter

- **Proprietăți:**

- **InsertCommand** – returnează sau setează instrucțiunea Transact-SQL sau procedura stocată folosită pentru a adăuga noi înregistrări în sursa de date
- **UpdateCommand** – returnează sau setează instrucțiunea Transact-SQL sau procedura stocată folosită pentru a actualiza înregistrări în sursa de date
- **DeleteCommand** – returnează sau setează instrucțiunea Transact-SQL sau procedura stocată folosită pentru a șterge înregistrări din setul de date

SqlDataAdapter

- **Metode:**

Fill(DataSet, String) – **adaugă** sau **reîncarcă** înregistrările în obiectul *DataSet*, astfel încât acestea să corespundă celor din sursa de date folosind **numele** *DataSet*-ului și al *DataTable*-ului

Update(DataSet, String) – **modifică** valorile din baza de date, **executând** instrucțiunile INSERT, UPDATE sau DELETE pentru **fiecare** înregistrare **adăugată, modificată** sau **ștearsă** din *DataSet*, folosind **numele** specificat al *DataTable*-ului

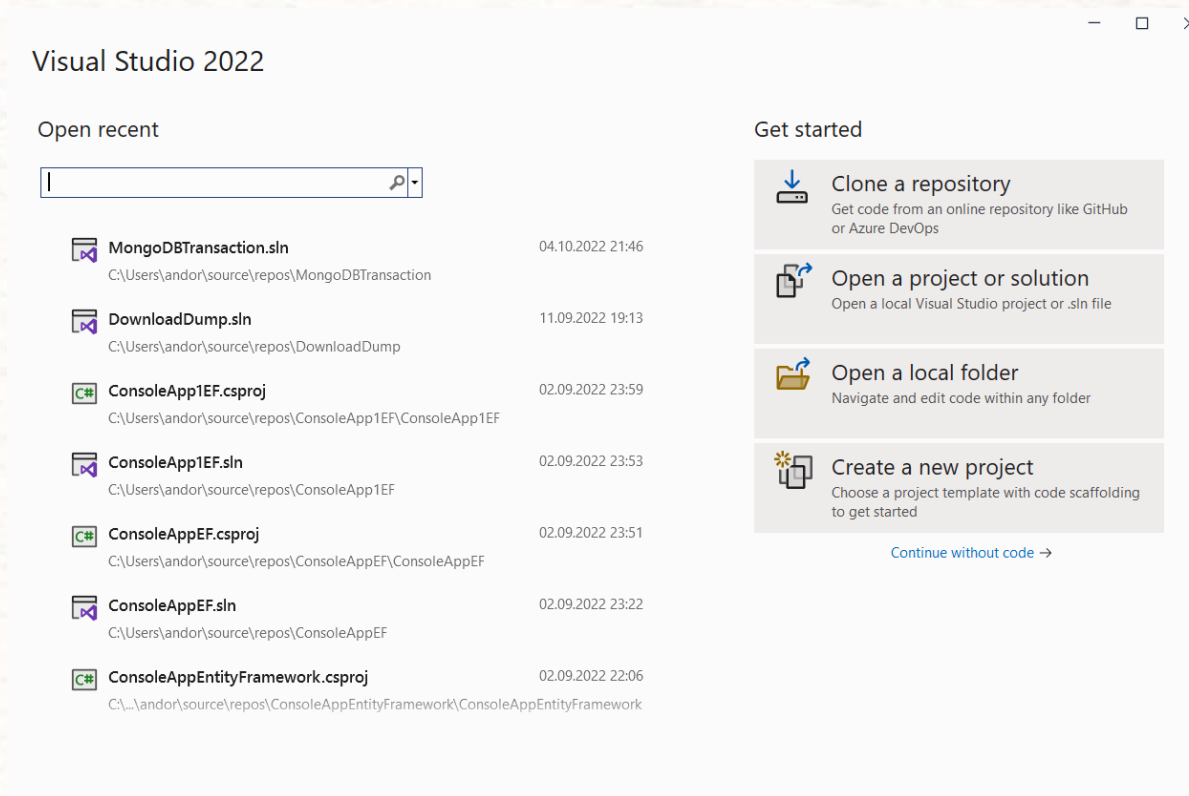
Exemplu – C# Console App

- În SQL Server, vom crea o nouă bază de date numită “SGBDIR”
- După ce baza de date a fost creată, vom crea un tabel nou:

```
CREATE TABLE Cadouri  
(  
    cod_cadou INT PRIMARY KEY IDENTITY,  
    descriere VARCHAR(100),  
    posesor VARCHAR(100),  
    pret REAL  
);
```

Exemplu – C# Console App




- În Visual Studio, vom crea un nou proiect folosind template-ul Console App disponibil în lista de template-uri a Visual C#:



Exemplu – C# Console App

Create a new project

Recent project templates

-  Console App C#
-  Windows Forms App C#
-  Console App C++

ConsoleApp

×


▼

[Clear all](#)

All languages


All platforms

Desktop



An empty Node.js application.


JavaScript Windows Linux macOS Console



Blank Node.js Console Application

An empty Node.js application.


TypeScript Windows Linux macOS Console



Console Application

Run code in a Linux terminal. Prints "hello" by default.


C++ Linux Console



Console App (.NET Framework)

A project for creating a command-line application


C# Windows Console



Console App (.NET Framework)

A project for creating a command-line application

Visual Basic Windows Console



ASP.NET Core Web App (Model-View-Controller)

A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.

Back

Next

Exemplu – C# Console App

– □ ×

Configure your new project

Console App (.NET Framework) C# Windows Console

Project name

SqlConsoleApp1

Location

C:\Users\andor\source\repos

...

Solution name ⓘ

SqlConsoleApp1

☒ Place solution and project in the same directory

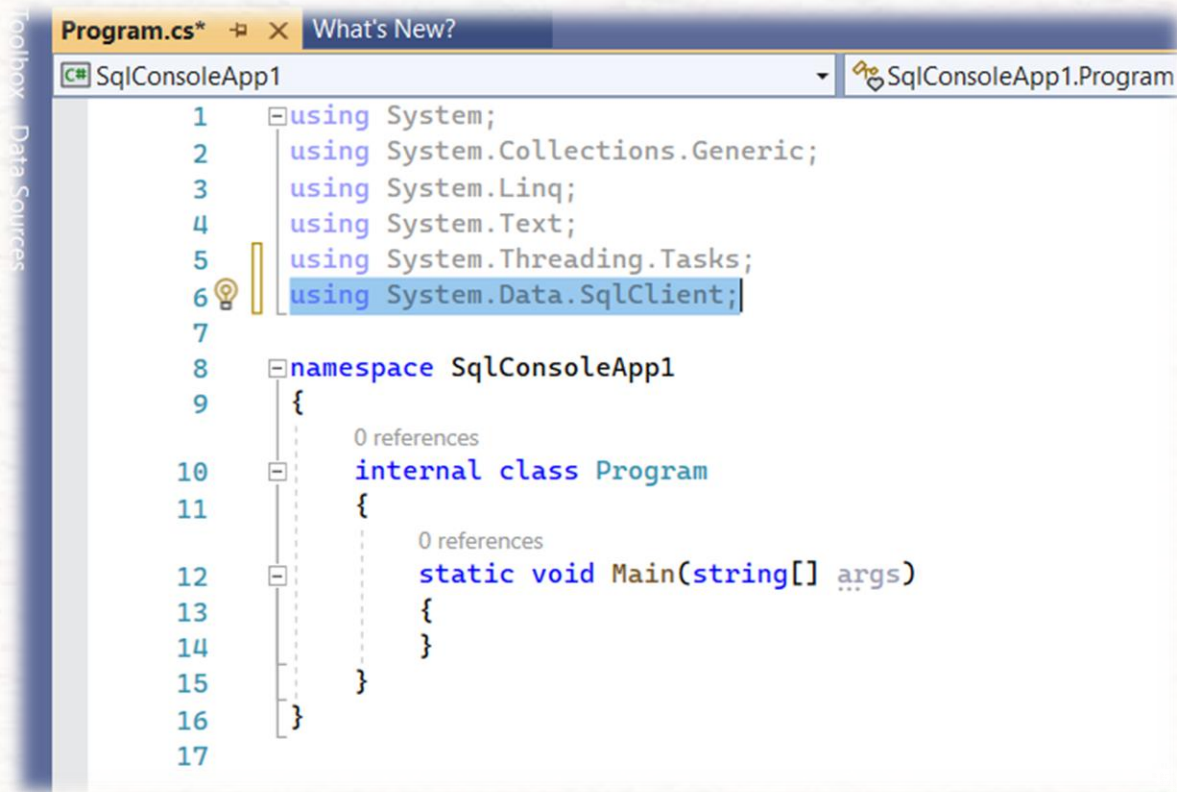
Framework

.NET Framework 4.8

Back Create

Exemplu – C# Console App

- După ce proiectul a fost creat, vom include namespace-ul **System.Data.SqlClient**, care este .NET Data Provider pentru SQL Server:



```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using System.Data.SqlClient;
7
8  namespace SqlConsoleApp1
9  {
10     internal class Program
11     {
12         static void Main(string[] args)
13         {
14         }
15     }
16 }
17
```

Exemplu – C# Console App

- Următoarea secvență de cod (inclusă în fișierul Program.cs) deschide o conexiune la baza de date “SGBDIR” pentru a adăuga, returna, actualiza și șterge înregistrări din tabelul “Cadouri” folosind *SqlCommand* și *SqlDataReader*:

```
using System;

using System.Data.SqlClient;

namespace SqlConsoleApp1
{
    internal class Program
    {
        static void Main(string[] args)
        {
```


Exemplu – C# Console App

```
//Setarea titlului consolei
```

```
Console.Title = "SqlConsoleApp1";
```

```
    string connectionString = @"Server=DESKTOP-01E0F0G\SQLEXPRESS;
```

```
    Database=SGBDIR;Integrated Security=true;TrustServerCertificate=true;";
```

```
try
```

```
{
```

```
    using (SqlConnection connection = new SqlConnection(connectionString))
```

```
{
```

```
        //Deschiderea conexiunii și verificarea stării
```

```
        connection.Open();
```

```
        Console.WriteLine("Starea conexiunii: {0}", connection.State);
```

Exemplu – C# Console App

//Adăugare

```
SqlCommand insertCommand = new SqlCommand("INSERT INTO Cadouri (descriere, posesor, pret) " +  
"VALUES (@desc1, @posesor1, @pret1), (@desc2, @posesor2, @pret2);", connection);  
  
insertCommand.Parameters.AddWithValue("@desc1", "lumanare");  
insertCommand.Parameters.AddWithValue("@posesor1", "Ion");  
insertCommand.Parameters.AddWithValue("@pret1", 2.5F);  
  
insertCommand.Parameters.AddWithValue("@desc2", "bicicleta");  
insertCommand.Parameters.AddWithValue("@posesor2", "Ioana");  
insertCommand.Parameters.AddWithValue("@pret2", 212.5F);  
  
int insertRowCount = insertCommand.ExecuteNonQuery();  
  
Console.WriteLine("Numarul de inregistrari adaugate: {0}", insertRowCount);
```


Exemplu – C# Console App

//Returnare înregistrări

```
SqlCommand selectCommand = new SqlCommand("SELECT descriere, posesor, pret FROM  
Cadouri;", connection);
```

```
SqlDataReader reader = selectCommand.ExecuteReader();
```

```
if (reader.HasRows)
```

```
{
```

```
    Console.WriteLine("Instructiunea SELECT a returnat urmatorul result set:");
```

```
    while (reader.Read())
```

```
    {
```

```
        Console.WriteLine("{0}\t{1}\t{2}", reader.GetString(0),  
        reader.GetString(1), reader.GetFloat(2));
```

```
    }
```

```
}
```

Exemplu – C# Console App

else

```
    Console.WriteLine("Nicio inregistrare returnata");
```

```
reader.Close();
```

//Actualizare

```
SqlCommand updateCommand = new SqlCommand("UPDATE Cadouri SET  
descriere=@descriereModificata WHERE posesor=@posesor;", connection);
```

```
updateCommand.Parameters.AddWithValue("@descriereModificata", "Semn de carte");
```

```
updateCommand.Parameters.AddWithValue("@posesor", "Ion");
```

```
int updateRowCount = updateCommand.ExecuteNonQuery();
```

```
Console.WriteLine("Numarul de inregistrari actualizate: {0}", updateRowCount);
```


Exemplu – C# Console App

//Ștergere

```
SqlCommand deleteCommand = new SqlCommand("DELETE FROM Cadouri WHERE  
posesor=@posesor;", connection);
```

```
deleteCommand.Parameters.AddWithValue("@posesor", "Ioana");
```

```
int deleteRowCount = deleteCommand.ExecuteNonQuery();
```

```
Console.WriteLine("Numarul de inregistrari sterse: {0}", deleteRowCount);
```

//Returnare înregistrări din nou

```
reader = selectCommand.ExecuteReader();
```

```
if (reader.HasRows)
```

```
{Console.WriteLine("Dupa actualizare si stergere, instructiunea SELECT  
a returnat urmatorul result set:");
```

Exemplu – C# Console App

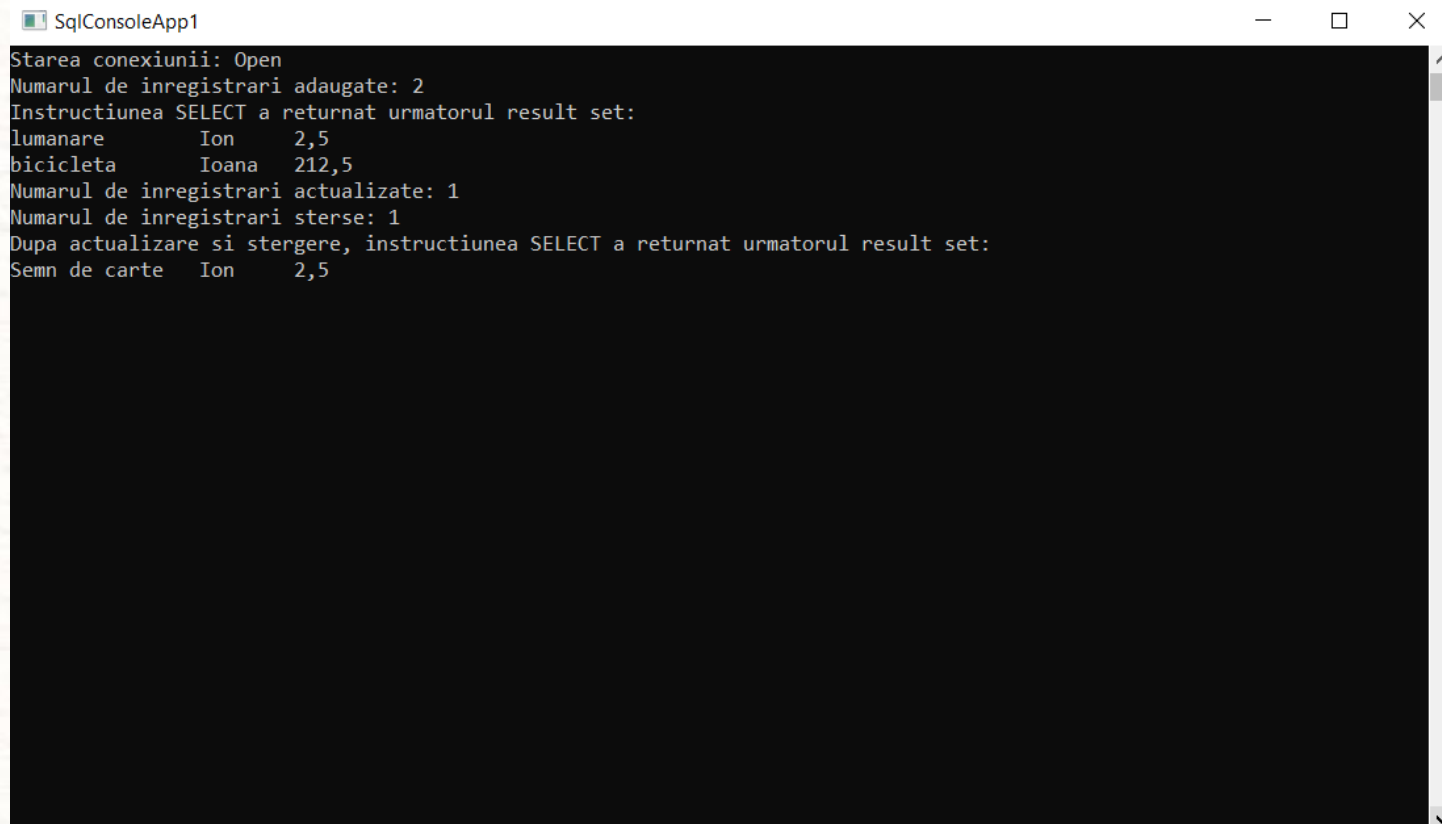
```
while (reader.Read())
{
    Console.WriteLine("{0}\t{1}\t{2}", reader.GetString(0), reader.GetString(1),
        reader.GetFloat(2));
}
}
else
    Console.WriteLine("Nicio inregistrare returnata");
reader.Close();
}
```


Exemplu – C# Console App

```
catch (Exception ex)
{
    /*Schimbă culoarea textului din consolă în roșu și afișează
    mesajul erorii*/
    Console.ForegroundColor = ConsoleColor.Red;
    Console.WriteLine("Mesajul erorii: \n{0}", ex.Message);
    Console.ReadKey();
}
Console.ReadKey();
}
}
```

Exemplu – C# Console App

- După execuția aplicației, obținem următorul rezultat:



```
SqlConsoleApp1
Starea conexiunii: Open
Numarul de inregistrari adaugate: 2
Instruciunea SELECT a returnat urmatorul result set:
lumanare      Ion      2,5
bicicleta     Ioana    212,5
Numarul de inregistrari actualizate: 1
Numarul de inregistrari sterse: 1
Dupa actualizare si stergere, instructiunea SELECT a returnat urmatorul result set:
Semn de carte Ion      2,5
```


Bibliografie

- <https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/ado-net-architecture>
- <https://learn.microsoft.com/en-us/sql/connect/ado-net/introduction-microsoft-data-sqlclient-namespace?view=sql-server-ver16>
- <https://learn.microsoft.com/en-us/dotnet/api/system.data.sqlclient.sqlconnection.connectionstring?view=dotnet-plat-ext-7.0>
- <https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/connection-string-syntax>
- <https://learn.microsoft.com/en-us/sql/connect/ado-net/sql-server-data-type-mappings?view=sql-server-ver16>
- <https://learn.microsoft.com/en-us/sql/connect/ado-net/retrieving-modifying-data?view=sql-server-ver16>
- <https://www.oreilly.com/library/view/programming-aspnet-35/9780596156657/ch09s01.html>

Bibliografie

- <https://learn.microsoft.com/en-us/sql/connect/ado-net/retrieve-data-by-datareader?view=sql-server-ver16>
- <https://learn.microsoft.com/en-us/sql/connect/ado-net/commands-parameters?view=sql-server-ver16>
- <https://learn.microsoft.com/en-us/sql/connect/connect-history?view=sql-server-ver16>
- <https://learn.microsoft.com/en-us/dotnet/api/system.console?view=net-7.0>
- <https://learn.microsoft.com/en-us/dotnet/api/system.data.sqlclient.sqldataadapter?view=dotnet-plat-ext-7.0>
- <https://learn.microsoft.com/en-us/dotnet/api/system.data.dataset?view=net-7.0>
- <https://learn.microsoft.com/en-us/dotnet/api/system.data.sqlclient.sqlconnection?view=dotnet-plat-ext-7.0>