ACCESUL LA BAZE DE DATE

Pentru realizarea unei aplicaţii care foloseşte baze de date se poate proceda în două moduri:

1. Se foloseşte o aplicaţie de tip sistem de gestiune de baze de date.
2. Se creează baza de date cu ajutorul unei aplicaţii de tip server de baze de date şi se scriu apoi aplicaţiile care accesează baza de date într-un limbaj ce posedă funcţiile necesare accesării serverului.

Platforma.NET

.NET este un cadru de dezvoltare software unitară care permite realizarea, distribuirea şi rularea aplicaţiilor desktop Windows şi aplicaţiilor WEB.

Tehnologia .NET înglobează mai multe tehnologii(ASP- Active Server Pages, XML- Extensible Markup Language, OOP- Object-oriented programming, SOAP- Simple Object Access Protocol, WSDL- Web Services Description Language etc.) şi limbaje de programare(VB, C++, C#, J#) asigurând totodată atât portabilitatea codului compilat între diferite calculatoare cu sistem Windows, cât şi reutilizarea codului în programe, indiferent de limbajul de programare utilizat.

Cele mai importante caracteristici ale acestei platforme sunt următoarele:

- Programatorii au la dispoziţie o serie de limbaje de programare de nivel înalt din care pot alege.
- Pot fi dezvoltate o varietate largă de aplicaţii, de la programe pentru desktop până la aplicaţii pentru dispozitive mobile, aplicaţii şi servicii web sau servicii Windows, de la programe izolate şi până la sisteme distribuite de mari dimensiunie.
- Mediul de execuţie este strict controlat de un motor de execuţie, care oferă o serie de facilităţi ce ridică mult nivelul de calitate al aplicaţiilor (managementul automat al memoriei şi securitatea fiind primele două care ies în evidenţă).

Orice program scris într-unul din limbajele .NET este compilat în Microsoft Intermediate Language (MSIL), în concordanţă cu Common Language Specification (CLS). Acest standard face posibilă colaborarea între componente scrise în limbaje diferite fără eforturi majore din partea programatorilor.

Aplicaţiile realizate cu ajutorul mediului de lucru .NET Framework pot folosi clase şi biblioteci din cadrul de lucru pentru a trimite instrucţiuni SQL către o bază de date relaţională (de obicei Microsoft SQL Server) şi pentru a prelucra rezultatele execuţiei instrucţiunilor SQL de către sistemul DBMS (Database Management System).

ADO.NET

ADO.NET(Active Data Objects) este componenta din .NET Framework formată dintr-o mulţime de biblioteci orientate obiect care permit interacţiunea cu cu sistemele de stocare a informaţiilor de obicei baze de date, dar pot fi şi fişiere XML, fişiere Excel, etc.

ADO.NET este standardizată in sensul că se pot folosi aceleași obiecte pentru accesarea diferitelor tipuri de baze de date : Access, MS SQL Server, Oracle, etc. Pentru realizarea conexiunii cu baza de date si manipularea datelor sunt necesare utilizarea a două namespaces-uri : System.Data si System.Data.SqlClient pentru MS SQL Server, respectiv System.Data si System.Data.OleDb pentru Oracle, Access, Excel.

Furnizori de date(Data Providers)

ADO.Net permite interacţiunea cu diverse tipuri surse de date prin utilizarea bibliotecilor de clase specializate corespuzator fiecarui tip de protocol de comunicare cu sursele de date. Unele sisteme mai vechi folosesc protocolul ODBC, altele mai noi, folosesc protocolul OleDb, etc. Aceste biblioteci de clase sunt denumite Furnizori de date(Data Providers).

Standardul furnizorilor de date impun ca ei să cuprindă componentele Connection, Command, DataReader şi DataAdapter ce permit aplicaţiilor să se conecteze la sursa de date și să aplice diverse comenzi asupra datelor.

Componenta Connection

Clasele din categoria Connection (SqlConnection, OleDbConnection, etc.) conţin date referitoare la sursa de date (locaţia, numele şi parola contului de acces, etc.), metode pentru deschiderea/închiderea conexiunii, pornirea unei tranzacţii etc. Aceste clase se găsesc în subspaţii (SqlClient, OleDb, etc.) ale spaţiului de nume System.Data. În plus, ele implementează interfaţa IDbConnection.

Proprietăţi

ConnectionString

O instanță a obiectului SqlConnection are la bază o listă de parametri necesari conectării memorați într-un string de conectare sub forma parametru=valoare, separaţi prin ;.

Stringul de conectare se transmite fie prin parametrul (de tip string) al constructorului, fie prin intermediul proprietății **ConnectionString**(String), cu accesori de tip **get** şi **set**.

Parametru	Descriere
Provider	Specifică tipul furnizorului de date pentru conectarea la sursa de date. Acesta trebuie precizat doar dacă se foloseşte OLE DB .NET Data Provider, şi nu se specifică pentru conectare la SQL Server.
Data Source	Identifică serverul, care poate fi local, numele unui calculator sau o adresa IP.
Initial Catalog	Specifică numele bazei de date. Baza de date trebuie să se găsească pe serverul dat în Data Source
Integrated Security	Dacă Integrated Security = SSPI sau *true* logarea se face cu user-ul configurat pentru Windows.
User ID	Numele unui user care are acces de logare pe server
Password	Parola corespunzătoare ID-ului specificat.

Exemple de conectare

Ex.1) conectare la o sursă de date SQL Server

```
using System.Data;
using System.Data.SqlClient;
String dataBase=”DbStudenti”;
SqlConnection conexiune=null ;
conexiune = new SqlConnection(@"Data Source=VIOREL-PC\SQLEXPRESS;”
          +”Initial Catalog=" +dataBase+
            ";Integrated Security=SSPI"); // SSPI sau True
Conexiune.Open();
```

Ex.2) conectare la o sursă de date SQL Server

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

```
SqlConnection co = new SqlConnection();
co.ConnectionString = “Data Source=localhost; Initial Catalog=Orar”;
User ID=profesor;pwd=info;
co.Open();
```

Ex.3) conectare la o sursã de date SQL Server

```
using System.Data.SqlClient;
SqlConnection co =new SqlConnection(@”Data Source=serverBD; Database=scoala;User
ID=elev;Password=madonna”);
co.Open();
```

Ex.4) conectare la o sursã de date Access

```
using System.Data.OleDb;
OleDbConnection co =
new OleDbConnection(@”Provider=Microsoft.Jet.OLEDB.4.0;
Data Source=C:\Date\scoala.mdb”);
co.Open();
```

Ex.5.1) conectare la o sursã de date EXCEL

```
using System.Data.OleDb;

OleDbConnection connExcel;
connExcel=new OleDbConnection ("Provider=Microsoft.Jet.OleDB.4.0;”
     + “Data Source="+ Agenda +
      "; Extended Properties=\"Excel 8.0;HDR=Yes;\"");
 connExcel.Open();
```

Ex.5.2) conectare la o sursã de date EXCEL xlsx

```
static string strConectareExcel=
          @"Provider=Microsoft.ACE.OLEDB.12.0;" +
            @"  Data Source=d:\scoala\ProjectsC#\Candidati.xlsx;
              Extended Properties=" + (char)34 + "Excel 12.0;" + (char)34;

       public OleDbConnection con;
       con=new OleDbConnection(strConectareExcel);
```

Ex.6) conectare la o sursã de date Oracle

```
using System;
using System.Data;
using System.Data.OleDb;

class OleDbConnectionOracle
{
  public static void Main()
  {
    string connectionString = "provider=MSDAORA;data source=ORCL;”
                        + “user id=SCOTT;password=TIGER";
    OleDbConnection myOleDbConnection =
         new OleDbConnection(connectionString);

    OleDbCommand myOleDbCommand = myOleDbConnection.CreateCommand();

    myOleDbCommand.CommandText =
       "SELECT empno, ename, sal FROM emp WHERE empno = 7369";

    myOleDbConnection.Open();

   OleDbDataReader myOleDbDataReader = myOleDbCommand.ExecuteReader();

    myOleDbDataReader.Read();
```

```
   Console.WriteLine("myOleDbDataReader[\" empno\"] = "
                + myOleDbDataReader["empno"]);
   Console.WriteLine("myOleDbDataReader[\" ename\"] = "
                + myOleDbDataReader["ename"]);
   Console.WriteLine("myOleDbDataReader[\" sal\"] = "
                 + myOleDbDataReader["sal"]);

   myOleDbDataReader.Close();
   myOleDbConnection.Close();
  }
}
```

ConnectionTimeout (int, cu accesor de tip **get**): specifică numărul de secunde pentru care un obiect de conexiune poate să aştepte pentru realizarea conectării la server înainte de a se genera o excepţie. (implicit 15). Se poate specifica o valoare diferită de 15 și în ConnectionString folosind parametrul Connect Timeout, Valoarea Timeout=0 specifică aşteptare nelimitată.

```
SqlConnection con = new SqlConnection(".\\SQLEXPRESS;"+
          "Initial Catalog =SALARII; Connect Timeout=30;"+
         " Integrated Security=SSPI");
```

Database (string, read-only): returnează numele bazei de date la care s–a făcut conectarea. Este necesară pentru a arăta unui utilizator care este baza de date pe care se face operarea.
Provider (de tip string, read-only): returnează furnizorul de date
ServerVersion (string, read-only): returnează versiunea de server la care s–a făcut conectarea.
State (enumerare de componente *ConnectionState,* read-only): returnează starea curentă a conexiunii. Valorile posibile: *Broken*, *Closed*, *Connecting*, *Executing*, *Fetching*, *Open*.

Metode

Open(): deschide o conexiune la baza de date
Close() şi **Dispose()**: închid conexiunea şi eliberează toate resursele alocate ei.
BeginTransaction(): pentru executarea unei tranzacţii pe baza de date; la sfârşit se apelează **Commit()** sau **Rollback()**.
ChangeDatabase(): se modifică baza de date la care se vor face conexiunile. Noua bază de date trebuie să existe pe acelaşi server ca şi precedenta.
CreateCommand(): creează o comandă (un obiect de tip *Command*) validă asociată conexiunii curente.

Scopul instanţierii unui obiect de tip SqlConnection este ca alte obiecte ADO.NET să poată lucra cu baza de date. Alte obiecte, cum ar fi SqlDataAdapter şi SqlCommand, au constructori care primesc obiectul conexiunii ca parametru. Atunci când se lucrează cu o bază de date trebuie urmaţi paşii:

1. instanţierea unui obiect SqlConnection;
2. deschiderea conexiunii;

3. trimiterea conexiunii ca parametru altor obiecte ADO.NET;
4. realizarea operaţiilor asupra bazei de date;
5. închiderea conexiunii.

Componenta Command

Clasele din categoria *Command* (*SqlCommand*, *OleDbCommand*, etc.) conţin date referitoare la o commandă SQL (SELECT, INSERT, DELETE, UPDATE) şi metode pentru executarea unei comenzi sau a unor proceduri stocate.

Instanţierea unui obiect de tipul *SqlCommand* sau *OleDbCommand* se face luând ca parametru un string, care este comanda ce va fi executată, şi o referinţă la un obiect de tipul *SqlConnection* respectiv *OleDbConnection*.

Constructori pentru obiecte de tip SQLCommand:

SqlCommand()
SqlCommand(string CommandText, SqlConnection con)

Constructori pentru obiecte de tip OleDbCommand:

OleDbCommand()
OleDbCommand(string CommandText, OleDbConnection con)

Proprietăţi

CommandText (String): conţine comanda SQL sau numele procedurii stocate care se execută pe sursa de date.
CommandTimeout (int): reprezintă numărul de secunde care trebuie să fie aşteptat pentru executarea comenzii. Dacă se depăşeste acest timp, atunci se generează o excepţie.
CommandType (enumerare de componente de tip *CommandType*): reprezintă tipul de comandă care se execută pe sursa de date. Valorile pot fi: *StoredProcedure* (apel de procedură stocată), *Text* (comandă SQL obişnuită), *TableDirect* (numai pentru OleDb)
Connection (System. Data. [Provider].PrefixConnection): conţine obiectul de tip conexiune folosit pentru legarea la sursa de date.
Parameters (System.Data.[Provider].PrefixParameterCollection): returnează o colecţie de parametri care s-au transmis comenzii.
Transaction (System.Data.[Provider].PrefixTransaction): permite accesul la obiectul de tip tranzacţie care se cere a fi executat pe sursa de date.

Metode asociate obiectelor de tip SQLCommand și OleDbCommand:

Cancel() opreşte o comandă aflată în executare.
Dispose() distruge obiectul comandă.
ExecuteNonQuery() execută o comandă de tip INSERT, UPDATE, DELETE și returnează numărul de înregistrări afectate.

Exemplu:

```
SqlCommand cmd = new SqlCommand("DELETE  tANGAJATI WHERE nume
= 'PREDA'",co);
cmd.ExecuteNonQuery();
```

ExecuteReader() execută comanda ce furnizează un set de rezultate (de tip Select) şi returnează un obiect de tip *DataReader*.

Componenta DataReader

Tipul *DataReader* (*SqlDataReader, OleDbDataReader*, etc.) este folosit pentru a citi date în cea mai eficientă metodă posibilă. Nu poate fi folosit pentru scriere. Odată citită o informaţie nu mai poate fi citită încă o dată. *DataReader* citeşte secvenţial date.

Datorită faptului că citeşte doar înainte (forward-only) permite acestui tip de date să fie foarte rapid în citire.

Un obiect *DataReader* nu are constructor, ci se obţine cu ajutorul unui obiect de tip Command şi prin apelul metodei *ExecuteReader*.

Conexiunea cu baza de date este activată pe toată durata lucrului cu un obiect de tip *DataReader*. Toate clasele *DataReader* (*SqlDataReader*, *OleDbDataReader*, etc.) implementează interfaţa *IDataReader*.

Utilizarea acestui tip de obiecte oferă: accesul conectat, performanţele bune la citire, consum mic de resurse şi tipizarea puternică.

Dacă într-o aplicaţie este nevoie doar de informaţii care vor fi citite o singura dată, sau rezultatul unei interogări este prea mare ca sa fie reţinut în memorie(caching) *DataReader* este soluţia cea mai bună.

DataReader implementează şi indexatori. Valoarea indexului trebuie să fie numărul (începând cu 0) sau numele coloanei din tabelul rezultat. Indiferent că se foloseşte un index numeric sau unul de tip string indexatorii întorc totdeauna un obiect de tipul *object* fiind necesară conversia.

Proprietăţi:

IsClosed (boolean, read-only)- returneză true dacă obiectul este deschis şi fals altfel
HasRows (boolean,read-only)- verifică dacă reader-ul conţine cel puţin o înregistrare
Item (indexator de câmpuri)
FieldCount-returnează numărul de câmpuri din înregistrarea curentă

Metode:

Close() închidere obiectului şi eliberarea resurselor; trebuie să preceadă închiderea conexiunii.
GetBoolean(), **GetByte()**, **GetChar()**, **GetDateTime()**, **GetDecimal()**, **GetDouble()**, **GetFloat()**, **GetInt16()**, **GetInt32()**, **GetInt64()**, **GetValue()**, **GetString()** returnează valoarea unui camp specificat, din înregistrarea curentă
GetBytes(), **GetChars()** citirea unor octeţi/caractere dintr-un câmp de date binar
GetDataTypeName(), GetName() returnează tipul/numele câmpului specificat
IsDBNull() returnează true dacă în câmpul specificat prin index este o valoare NULL
NextResult()determină trecerea la următorul rezultat stocat în obiect
Read() determină trecerea la următoarea înregistrare, returnând false numai dacă aceasta nu există; de reţinut că iniţial poziţia curentă este **înaintea** primei înregistrări.
DataReader obţine datele într-un stream secvenţial. Pentru a citi aceste informaţii trebuie apelată metoda **Read**; aceasta citeşte un singur rând din tabelul rezultat. Metoda clasică de a citi informaţia dintr-un **DataReader** este de a itera intr-o bucla while.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Data.SqlClient;
using System.Data.OleDb;



namespace ConectareLaSurseDeDate
{
    public class DbSqlServer
    {
        static string stringConectare =
             @"Data Source=FUJITSU_AMILO; Initial Catalog=dbFacturare;
            Integrated Security=SSPI";
        public static SqlConnection con;


        public DbSqlServer()
        {
            try
            {
                con = new SqlConnection(stringConectare);
                con.Open();
                con.Close();
                Console.WriteLine("Conectare cu succes!");
            }
            catch { Console.WriteLine("Esec conectare!"); }
        }

        public void ListareClienti()
        {
            SqlDataReader rdr;

            try
            {
                SqlCommand cmd = new SqlCommand();
                cmd.CommandType = CommandType.Text;// implicit
                cmd.Connection = con;
                cmd.CommandText = "select * from tClienti";
                con.Open();
                rdr = cmd.ExecuteReader();

            }
            catch { Console.WriteLine("esec rdr"); return; }

            Console.WriteLine("Lista Clientilor");

            Console.WriteLine("{0,-10}{1,-20}", "Cod ", "Nume ");
            while (rdr.Read())
            {
                Console.WriteLine("{0,-10}{1,-20}", rdr["CodClient"],
rdr["NumeClient"]);
            }
            rdr.Close();
            con.Close();
        }
```

```
        public void InserareClient(string CodClient, string numeClient, string
localitate)
        {
            string strInsert =
              "insert into tClienti(CodClient,NumeClient,Localitate) values ("
            + "'" + CodClient + "', '" + numeClient + "', '" + localitate +
"')";

            try
            {
                SqlCommand cmd = new SqlCommand(strInsert, con);
                con.Open();
                cmd.ExecuteNonQuery();

            }
            catch (Exception)
            { Console.WriteLine("Err: Inserare esuata\n" + strInsert); }
            con.Close();
        }

        public void DeleteClient(string codClient)
        {
            string strDelete = "delete tClienti where codClient=" + codClient;
            // +"'";
            int n;
            try
            {

                SqlCommand cmd = new SqlCommand(strDelete, con);
                con.Open();
                n = cmd.ExecuteNonQuery();
                con.Close();
                if (n == 0) throw new Exception("Cod negasit");
            }
            catch (Exception e)
            {
                Console.WriteLine("Err: Stergere esuata\n"
                                  + strDelete + "\n" + e.ToString());
            };
        }


        public void UpdateDenumireClient(string codClient,
                                         string numeClient)
        {
            string strUpdate = "update tClienti set NumeClient= '" +
                   numeClient + "' where codClient='" + codClient + "'";
            try
            {

                SqlCommand cmd = new SqlCommand(strUpdate, con);
                con.Open();
                int n = cmd.ExecuteNonQuery(); //nr randuri afectate
                if (n == 0) throw new Exception("CodClient inexistent");
            }
            catch (Exception e)
            {
                Console.WriteLine("Err: Modificare esuata\n" +
                                  strUpdate + "\n" + e.ToString());
            }
            con.Close();
        }
```

```
    }




    class DbAccess
    {
        static string strConnAccess = @"Provider=Microsoft.Jet.OleDB.4.0;
                                        Data
Source=D:\Scoala\ProjectsC#\DREPT.mdb";
        public OleDbConnection con;

        public DbAccess()
        {
            try
            {
                con = new OleDbConnection(strConnAccess);
                con.Open();
                con.Close();


            }
            catch { Console.WriteLine("erroare conectare"); };
        }
        public void Listare()
        {
            try
            {
                string strSelect = "select nume,prenume,media from tDosare1
where media>9.50";

                OleDbCommand cmd = new OleDbCommand(strSelect, con);

                try
                {
                    con.Open();
                    OleDbDataReader rdr = cmd.ExecuteReader();
                    Console.WriteLine(" \n LISTA CANDIDATILOR CU MEDII MAI MARI
DACAT 9.50");
                    while (rdr.Read())
                    {
                        Console.WriteLine("{0,-20}{1,-20}{2:##.##}",
                                rdr["Nume"], rdr["prenume"], rdr["media"]);
                    }
                    rdr.Close();
                }
                catch { Console.WriteLine("Esec"); }


            }
            catch (Exception e1) { Console.WriteLine(e1.Message); }
            con.Close();
        }

    }


    public class DbExcel
    {
        static string strConectareExcel =
```

```
            @"Provider=Microsoft.ACE.OLEDB.12.0;" +
              @"  Data Source=d:\scoala\ProjectsC#\Candidati.xlsx;
                Extended Properties=" + (char)34 + "Excel 12.0;" + (char)34;

        public OleDbConnection con;
        public DbExcel()
        {
            try
            {
                con = new OleDbConnection(strConectareExcel);
                con.Open();
                con.Close();


                Console.WriteLine("Conectare cu succes");
            }
            catch { Console.WriteLine("eroare conexiune"); }
        }

        public void Listare()
        {


            string strSelect = "select nume,nota1,nota2,media from
[sheet1$B4:E20]";


            OleDbCommand cmd = new OleDbCommand(strSelect, con);
            OleDbDataReader rdr;

            try
            {
                con.Open();
                rdr = cmd.ExecuteReader();
                while (rdr.Read())
                    Console.WriteLine("{0, -10} {1, 6:#.00} {2, 6:#.00}",
rdr[0], rdr[1], rdr[2]);
                rdr.Close();
                con.Close();
            }
            catch (Exception ex)
            { Console.WriteLine(ex); }
        }

    }


    class Program
    {

        static void Main(string[] args)
        {
            DbSqlServer dbSQL = new DbSqlServer();
            dbSQL.ListareClienti();

            dbSQL.InserareClient("21", "Radu", "Pitesti");
            dbSQL.ListareClienti();

            dbSQL.UpdateDenumireClient("21", "Nicolae");
            dbSQL.ListareClienti();

            dbSQL.DeleteClient("21");
```

```
            dbSQL.ListareClienti();

            // Baza de date ACCESS

            DbAccess dbA = new DbAccess();
            dbA.Listare();


            // Baza de date EXCEL

            DbExcel dbE = new DbExcel();
            dbE.Listare();



            dbE.con.Open();
            string strUpdate = "update [sheet1$b4:e9] set
media=(nota1+nota2)/2";
            OleDbCommand cmdUpdate = new OleDbCommand(strUpdate, dbE.con);
            cmdUpdate.ExecuteNonQuery();
            dbE.con.Close();
            Console.WriteLine("\n");
            dbE.Listare();

            dbE.con.Open();
            string insert = "insert into [sheet1$b4:e10](nume,nota2)
values('ion',6)";
            OleDbCommand cmdInsert = new OleDbCommand(insert, dbE.con);
            cmdInsert.ExecuteNonQuery();
            dbE.con.Close();

            Console.ReadKey();

        }
    }
}
```