BALTARIU Alexandra, grupa 1A, anul 3

CHIRILĂ Ștefan Alexandru, grupa 2A, anul 3

PĂRPĂUȚĂ Dan, grupa 2A, anul 3

**COMPARAȚIE**

**ADO.NET vs. Entity Framework *(operații fundamentale)***

**ADO.NET**

* **ADO.NET** clasic presupune folosirea de obiecte de tip **DataReader/ DataAdapter** pentru a prelua date din baza de date și comenzi SQL pentru a executa operații CRUD.
* Obiectele de conectare sunt utilizate pentru a stabili o conexiune la o bază de date, operaţie necesară înainte de a folosi obiectele **DataAdapter** şi **Command** pentru a interacţiona cu baza de date. Clasele acestor obiecte implementează interfaţa **IDbConnection**. Această interfaţă specifică metode comune, precum **Open()** sau **Close()**, pentru toţi furnizorii de surse de date.
* Soluţia folosită în **ADO.NET** pentru lucrul în mod deconectat cu sursa de date este clasa **DataSet**, care este alcătuită din două componente de bază: o colecţie de tabele (**DataTable**) şi una de relaţii (**DataRelation**).
* Altă clasă ce permite lucrul deconectat cu sursa de date este clasa **DataView**, similară cu un view dintr-o bază de date.

**Entity Framework (EF)**

* În **EF** rândurile și coloanele din tabele sunt returnate ca obiecte și nu mai sunt folosite în mod direct comenzi SQL, altfel spus, se translatează datele din forma tabelară in obiecte;
* **EF** folosește un model de date dezvoltat din **Entity Relationship Modeling (ERM)**, numit **Entity Data Model (EDM)**, care creează entități;
* Clasa **ObjectServices** furnizează funcționalitatea necesară de a lucra cu obiecte din entități, construind comenzile Insert, Update și Delete pentru fiecare obiect adăugat, modificat sau șters;
* **ObjectServices** lucrează direct cu obiectele create și furnizează funcționalitatea necesară pentru a genera și a interacționa cu obiectele din modelul conceptual care extrag datele din baza de date, procesând cereri **Linq To Entities (LtE)** și **ObjectQuery** rezultând obiecte.

*Utilizarea obiectelor* ***DataReader*** *şi* ***Command***

// 1. instantiaza un nou obiect conexiune

SqlConnection conn = new qlConnection(@"Data

Source=localhost\SQLEXPRESS;Initial Catalog=northwind;Integrated Security=SSPI");

SqlDataReader rdr = null;

try {conn.Open();

// 2. Deschide conexiunea

// 3. Foloseste obiectul de conexiune la creare unui obiect Command

SqlCommand cmd = new SqlCommand("select \* from employees", conn);

// 4.Executa comanda si preia rezultatele interogarii

rdr = cmd.ExecuteReader();

// afiseaza toate câmpurile

while (rdr.Read()){

for(int i=0; i< rdr.FieldCount; i++)

{ Console.Write(rdr[i]+" "); }

Console.WriteLine("");}}

catch(Exception ex)

{ Console.WriteLine(ex.Message); }

finally{ // închide reader-ul

if (rdr != null){ rdr.Close(); }

// 5. Închide conexiunea

if (conn != null){ conn.Close(); }}}

*Utilizarea obiectelor* ***DataTable****,* ***DataRow*** *şi* ***DataColumn***

//obţin o referinţă la tabelul Orders

DataTable dt = myDataSet.Tables["Order"];

//iterez peste toate coloanele tabelului

foreach (DataColumn dc in dt.Columns)

{Console.Write(String.Format("{0}\t", dc.ColumnName); }

Console.WriteLine(“Inregistrari:”);

foreach(DataRow row în dt.Rows){

foreach(DataColumn dc în dt.Columns)

{ onsole.Write(String.Format("{0}\t", row[dc])); }

Console.WriteLine();}

* Clasa **ObjectContext** furnizează metode pentru lucrul cu date transformate în obiecte, instanțe ale tipurilor entitate, definite în modelul conceptual.
  + O instanță a clasei **ObjectContext** conține o conexiune la baza de date sub forma unui obiect **EntityConnection**, un obiect **MetadataWorkspace** care descrie modelul și un obiect **ObjectStateManager** care gestionează obiectele din cache.

*Utilizarea modeului* ***EDM***

private static void QueryContacts()

{using (var context = new SampleEntities()){

var contacts = context.Contacts;

foreach (var contact in contacts){

Console.WriteLine("{0} {1}",

contact.FirstName.Trim(),

contact.LastName);}}

Console.Write("Press Enter...");

Console.ReadLine();}

*Utilizarea obiectului* ***DataSource***

authorList.DataSource = publishContext.Author;

authorList.DisplayMember = "FirstName";

*Utilizarea* ***Linq-to-Entities (LtE)***

IQueryable<Payroll> payrollQuery =

from p in publishContext.Payroll

where p.Author.AuthorID == selectedAuthorID select p;

List<Payroll> selectedPayroll = payrollQuery.ToList();

if (selectedPayroll != null && selectedPayroll.Count > 0){

currentPayroll = selectedPayroll.First();}

else{currentPayroll = null;}