

# Adatbázisok I.

## 9

Jánosi-Rancz Katalin Tünde

[tsuto@ms.sapientia.ro](mailto:tsuto@ms.sapientia.ro)

327A

# Adatbáziskezelés C#-ban

- A .NET környezet lehetőséget biztosít számos adatbázis-típushoz való kapcsolódásra, ezekhez különböző motorokat biztosít.
- Többféle osztály használata szükséges:
  - *Kapcsolat*: ez adatbázis-típusfüggő osztály, amelynek segítségével a program kapcsolódik egy adatbázishoz (pl. OleDbConnection, OracleConnection, stb.)
  - *Adattárolók*: olyan osztályok, amelyek az adatbázisból betöltött adatot tárolják
  - *Adat adapterek*: az adatokat mozgatják az adatbázis és a tárolók között
  - *Parancs*: ez az osztály az adatok manipulációját teszi lehetővé (pl. úgy, hogy értelmez és alkalmaz egy SQL utasítást)
  - *Navigáció*: az adatok közötti mozgást és manipulációt megkönnyítő osztály

# Adatbázisok elérése

- *Közvetlen elérés* - minden adatbázismotorhoz külön függvénygyűjtemény
- *Absztrakciós rétegen keresztül*
  - *Open DataBase Connectivity* (ODBC), közös függvényhalmaz, amivel minden DB elérhető. Az ODBC rétegben levő driverek lefordítják.
  - *Object Linking and Embedding* (OLE DB) –kifele táblázatos formában mutatja az adatokat. Az adatforrások OLE DB provider-eken keresztül érhetők el COM objektumok segítségével. ODBC-t is elér.
  - *ActiveX Data Objects* (ADO) – egy vékony réteg az OLE DB felett, a magas szintű nyelvek számára elérhetővé teszi azt.
  - *ADO.NET* – ADO továbbfejlesztett, felügyelt változata

# ADO.NET

- A .NET adatbáziskezelési osztálygyűjteménye
- Névter: System.Data
- Relációs adatok egyszerű elérése:
  - osztályok oszlopok, sorok, táblák, adatbázis leírására
  - Managed provider (adatszolgáltató):
    - ❖ SQL, OleDb, ODBC, Oracle, MySql stb.
  - Kapcsolatfelvétel, parancs futtatás, stb.
- Adatbázis független interfészek, absztrakt osztályok
  - IDbConnection, IDbCommand, IDbDataAdapter, stb.

## SQL provider:

SqlConnection

SqlCommand

SqlDataAdapter

SqlDataReader

# Kétféle adatelérési modell

<p>Kapcsolat alapú (Direct Access)</p> <p>pl: <b>DataReader</b>, <b>DataCommand</b></p>	<p>Kapcsolat nélküli (Disconnected Access)</p> <p>pl: <b>DataSet</b> (és a többiek)</p>
<ul style="list-style-type: none"><li>- Egyirányú, csak olvasható</li><li>- Ha az adatokat azonnal feldolgozzuk</li></ul>	<ul style="list-style-type: none"><li>- Lokális másolat az adatokról</li><li>- Ha az adatok közt navigálunk</li><li>- Ha az adatokat módosítjuk is</li></ul>
<p>Előnyök</p> <ul style="list-style-type: none"><li>- Egyszerűbb konkurencia kezelés</li><li>- Az adatok mindenhol a legfrissebbek</li></ul> <p>Hátrányok</p> <ul style="list-style-type: none"><li>- Folyamatos hálózati kapcsolat</li></ul>	<p>Előnyök</p> <ul style="list-style-type: none"><li>- Nem szükséges folyamatos hálózati kapcsolat</li></ul> <p>Hátrányok</p> <ul style="list-style-type: none"><li>- Ütközések lehetségesek</li><li>- Az adatok nem mindenhol a legfrissebbek</li></ul>

# Kapcsolat alapú adatelérés

- Select
- Tárolt eljárás

## Olvasás

Command

DataReader

Connection

## Írás

Command

- Insert
- Update
- Delete
- Stored procedure

Adatbázis

## Kliens alkalmazás

Form1

address	au_fname	au_id	au_lname	city	contract
10932 Bigge	Johnson	172-32-1176	White	Menlo Park	<input checked="" type="checkbox"/>
309 63rd St.	Marjorie	213-46-8915	GreenNew	Oakland	<input checked="" type="checkbox"/>
589 Darwin L.	Cheryl	238-95-7766	Carson	Berkeley	<input checked="" type="checkbox"/>
22 Cleveland	Michael	267-41-2394	O'Leary	San Jose	<input checked="" type="checkbox"/>
5420 College	Dean	274-80-9391	Straight	Oakland	<input checked="" type="checkbox"/>
10 Mississippi	Meander	341-22-1782	Smith	Lawrence	<input type="checkbox"/>

Load Save Changes in the Database Delete

## Windows Form

Product List - Microsoft Internet Explorer

Address http://localhost/ado/ProductDataGrid.aspx

Product ID (Click for Detail)	Product Name	CategoryID	Unit Price	Discontinued	Editing
1	Chai	1	18	<input type="checkbox"/>	Edit
2	Chang	1	19	<input type="checkbox"/>	Edit
3	Aniseed Syrup	2	10	<input type="checkbox"/>	Edit

## Web Forms

# Mi az a DataSet?

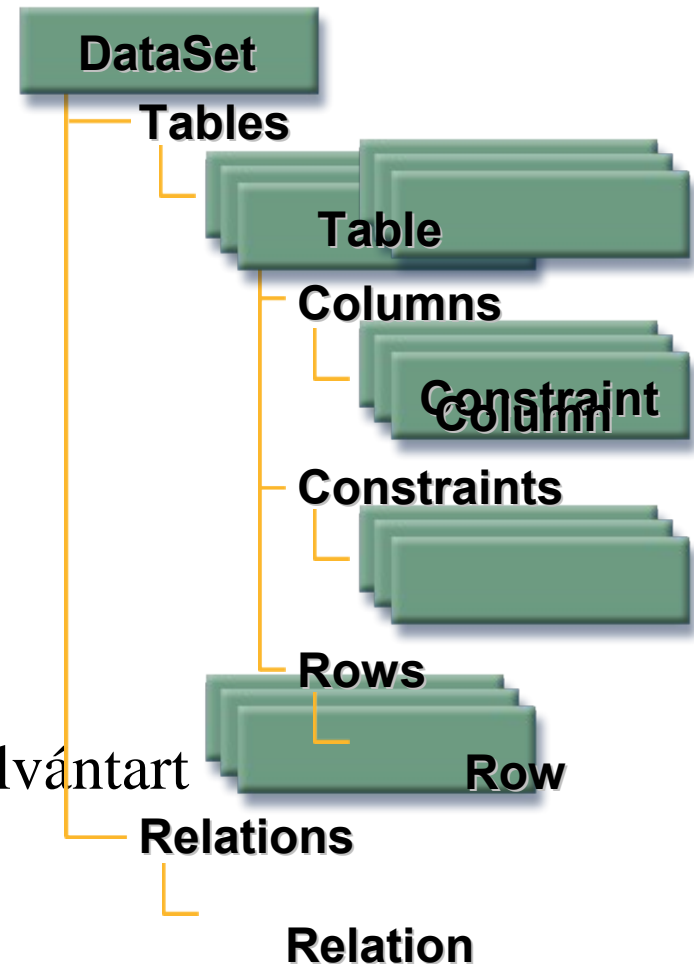
## ■ Adatbázis a memóriában

## ■ Tartalma

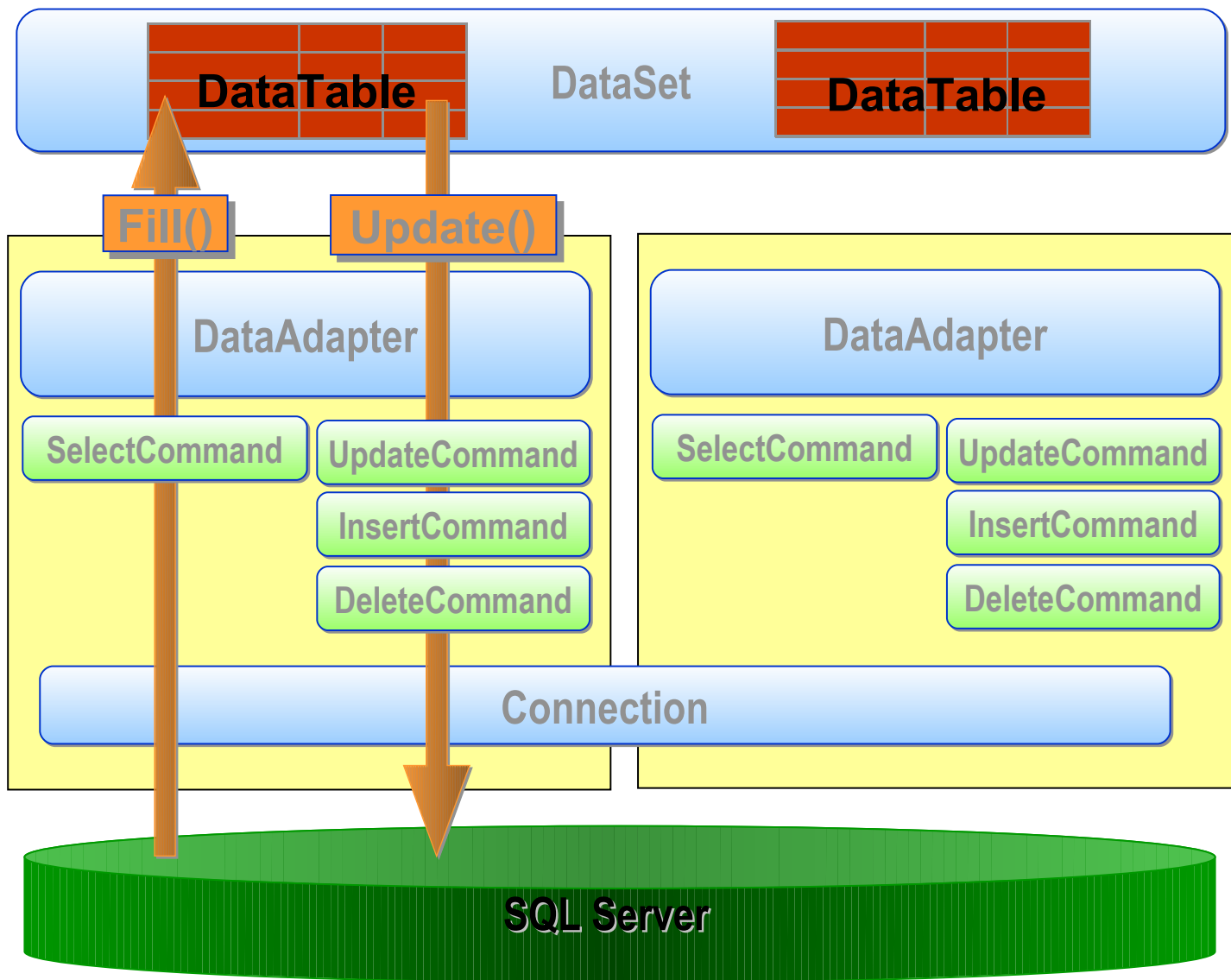
- Táblák, mezők, sorok
- Nézetek, kényszerek
- Kapcsolatok (szülő-gyermek)

## ■ Lehetőségei

- Feltöltése (alapállapot)
- XML támogatás (adat, séma)
- Szerkeszthető: minden változást nyilvántart



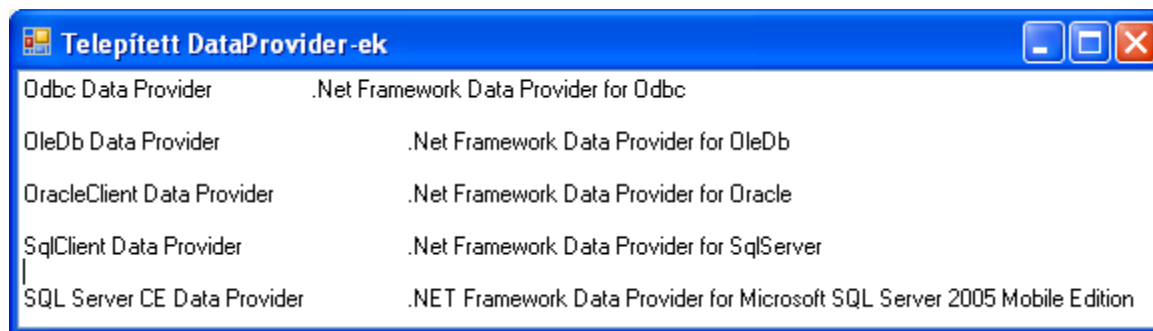
# Kapcsolat nélküli adatelérés





# Data Provider

- ▶ Híd az alkalmazás és az adatforrás között, ezen keresztül mozognak az adatok az alkalmazás és az adatbázis között
- ▶ Microsoft DP-k
  - ★ SqlClient (MS SQL Server)
  - ★ OracleClient (Oracle)
  - ★ OleDb (Access)
  - ★ Odbc
- ▶ Más cégektől
- ▶ Telepített DP-k



# A kapcsolat - Connection objektum

- A DP egyik komponense
- Kapcsolat az adattárhoz, ezen keresztül kommunikál az alkalmazás az adatbázissal
- `ConnectionString` – a kapcsolat beállításai
- **State** tulajdonság segítségével megtudhatjuk, hogy a csatlakozás milyen állapotában vagyunk: `Closed`, `Connecting`, `Open`, `Executing`, `Fetching`, `Broken`.
- Connection típusok:
  - `SqlConnection`: SQL szerver (pl. Microsoft Data Engine (MSDE))
  - `OdbcConnection`: Open Database Connectivity
  - `OleDbConnection`: Object Linking and Embedding Database (pl. Microsoft Access)
  - `OracleConnection`: Oracle adatbázisok

# A parancs - Command objektum

- Közvetlen hozzáférés a kapcsolt adatbázis adataihoz
- SQL parancsok vagy tárolt eljárások
- Az eredmény adatfolyam, amit DataReader olvashat vagy DataSet-be lehet betölteni
- Parameters tulajdonság: gyűjtemény, az SQL parancsok vagy tárolt eljárások bemenő és kimenő paraméterei
- Command típusok (osztályok)
  - System.Data.SqlClient.SqlCommand
  - System.Data.OleDb.OleDbCommand

# DataReader objektum

- ▶ Adatbázis-specifikus
- ▶ Gyors, csak előrehaladást engedélyező szerver oldali kurzor
- ▶ Sorokat tartalmazó adatfolyamon halad végig
- ▶ A Command objektum ExecuteReader metódusa egy DataReader-t ad vissza
- ▶ Az aktuális sor egyes oszlopaiban tárolt adatokat típusuk szerinti metódusokkal lehet lekérdezni (pl. GetDouble)

# DataAdapter objektum

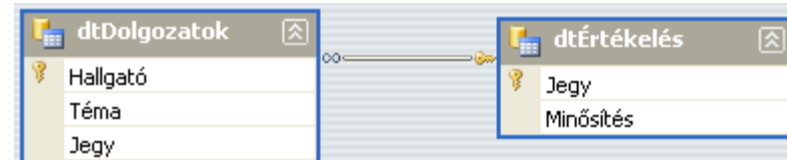
- ▶ Híd a tábla és az adatforrás között
- ▶ Adatbázis parancsok
- ▶ Adatbázis kapcsolatok
- ▶ Alap típusok
  - ★ OleDbDataAdapter
  - ★ SqlDataAdapter – SQL Serverhez

# TableAdapter

- ▶ Egy generált osztály típusos DataSet kezeléséhez
- ▶ Magába foglalja az alábbi objektumokat:
  - ★ DataAdapter
  - ★ Connection
  - ★ Commands
  - ★ Query
  - ★ Parameters
- ▶ Minden táblához külön TableAdapter

# Típusos DataSet

- ▶ További absztrakciós szint
- ▶ Az alap DataSet osztály leszármazottja
- ▶ Típusellenőrzést tesz lehetővé fordítási időben
- ▶ Gyorsabb hozzáférést biztosít az adathalmazban levő táblákhoz és rekordokhoz
- ▶ XML Schema (.xsd) leírás állományokból van generálva az XSD.exe segítségével
- ▶ Grafikusan (.xss, .xcs)
- ▶ Hozzáférés táblákhoz és oszlopokhoz
  - ★ Típus nélküli esetben: `dsNév.Tables("TáblaNév")`
  - ★ Típusos esetben: `dsNév.TáblaNév , dsAdatok.dtDolgozatok.HallgatóColumn`



# Kapcsolat alapú adatbázis elérés

## Lekérdezés

- Az eredmény egy szerver oldali kurzorba kerül, abból olvashatunk soronként DataReader segítségével
- Lépések
- Kapcsolati sztring összeállítása
- Kapcsolat létesítése az adatbázishoz Connection objektum segítségével
- SQL lekérdezés sztring összeállítása
- Kapcsolat megnyitása
- Command objektum létrehozása
- DataReader objektum létrehozása
- Rekordok kiolvasása
- DataReader objektum lezárása
- Kapcsolat lezárása



```

private void tsmiEmailCimekListája_Click(object sender, EventArgs
{ try
{ // Kapcsolat objektum létrehozása
OdbcConnection Kapcsolat = new OdbcConnection();
// Kapcsolat sztring definiálása
Kapcsolat.ConnectionString =
    "STMT=;OPTION=3;DSN=SWT1;UID=root;" +
    "PASSWORD=1234;DESC=MySQL ODBC 3.51 Driver " +
    "DSN;DATABASE=SWT1;SERVER=localhost;PORT=3306";
// SQL parancs objektum létrehozása
OdbcCommand Parancs = new OdbcCommand(
    "SELECT email FROM lista WHERE email IS NOT NULL",
    Kapcsolat);
// Kapcsolat megnyitása
Kapcsolat.Open();
// Adatlekérő objektum létrehozása
OdbcDataReader Olvasó = Parancs.ExecuteReader();
// Létrehozzuk a párbeszédablakot
frmEmailLista fem = new frmEmailLista();
// Töröljük a szövegmező tartalmát
fem.EmailCimek = "";
// Egyenként lekérjük a rekordokat, és mindegyikből az e-mail
// címet bemásoljuk a párbeszédablak szövegmező egy új sorába
while (Olvasó.Read())
{ string EHA = Olvasó.GetString(0);
  fem.EmailCimek += EHA + "\r\n";
}
// Lezárjuk az adatlekérő objektumot
Olvasó.Close();
// Lezárjuk az adatkapcsolatot az adatbázissal
Kapcsolat.Close();
fem.ShowDialog();
}
catch (Exception exc)
{ MessageBox.Show(exc.Message, "Adatbázis hiba",
  MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}

```

# Kapcsolatalapú adatbáziselérés

## Beszúrás, Módosítás, Törlés

### Lépések

- Kapcsolat objektum létrehozása
- Kapcsolati sztring összeállítása
- Kapcsolat létesítése az adatbázishoz Connection objektum segítségével
- SQL utasítás sztring összeállítása
- Command objektum létrehozása
- Kapcsolat megnyitása
- SQL parancs végrehajtása ExecuteNonQuery
- Kapcsolat lezárása

```

private void tsmiAdatrögzítés_Click(object sender, EventArgs e)
{ // Adatbeviteli párbeszédablak létrehozása
  frmFelvitel ff = new frmFelvitel();
  // Párbeszédablak megjelenítése és OK gomb esetén adatfelvitel
  if (ff.ShowDialog() == DialogResult.OK)
  { try
    { // ODBC kapcsolat objektum létrehozása
      OdbcConnection Kapcsolat = new OdbcConnection();
      // Kapcsolódási sztring definiálása
      Kapcsolat.ConnectionString = "STMT=;OPTION=3;DSN=SWT1; "+
        "UID=root; " +
        "PASSWORD=1234;DESC=MySQL ODBC 3.51 Driver " +
        "DSN;DATABASE=SWT1;SERVER=localhost;PORT=3306";
      // Adatfelviteli SQL parancs definiálása
      OdbcCommand Parancs = new OdbcCommand(
        "INSERT INTO lista (EHA,Nev,email)" +
        "VALUES ('" + ff.EHA + "',''" + ff.Nev + "',''" +
        ff.email + "')",
        Kapcsolat);
      // Kapcsolat megnyitása
      Parancs.Connection.Open();
      // SQL parancs végrehajtása
      Parancs.ExecuteNonQuery();
      // Kapcsolat lezárása
      Kapcsolat.Close();
    }
    catch (Exception exc)
    { MessageBox.Show(exc.Message, "Adatbázis hiba",
      MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
  }
}

```

# Kapcsolat nélküli adatbázis elérés – adatok kiolvasása

- Kapcsolat létesítése az adatbázissal egy Connection objektum segítségével
- Command objektum létrehozása és a Connection objektumhoz kapcsolása
- SQL parancsok összeállítása
- DataAdapter objektum(ok) létrehozása
- DataSet objektum(ok) létrehozása
- Adatok bemásolása a DataSet-be (DataTable) a Fill() metódus meghívásával
- Egy DS-ben több tábla
- Egy DS több különböző forrásból szerezhetsz adatokat
- A DataAdapter vagy TableAdapter objektum Fill() metódusa gondoskodik a kapcsolat megnyitásáról

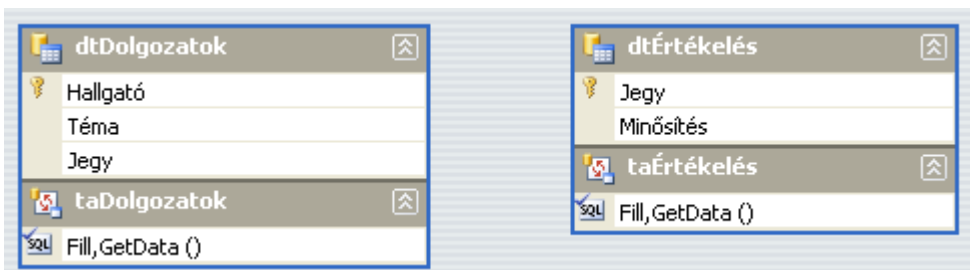
# DataSet feltöltése

```
private void tsmiÖsszesAdat_Click(object sender, EventArgs e)
{
    frmÖsszesAdat foa = new frmÖsszesAdat();
    OdbcConnection Kapcsolat = new OdbcConnection();
    Kapcsolat.ConnectionString =
        "STMT=;OPTION=3;DSN=SWT1;UID=root;PASSWORD=1234; " +
        "DESC=MySQL ODBC 3.51 Driver" +
        " DSN;DATABASE=SWT1;SERVER=localhost;PORT=3306";
    try
    {
        Kapcsolat.Open();
        OdbcDataAdapter Adapter = new OdbcDataAdapter();
        Adapter.SelectCommand = new
            OdbcCommand("SELECT * FROM lista ", Kapcsolat);
        DataSet dataset = new DataSet();
        Adapter.Fill(dataset);
        foa.AdatForrás = dataset.Tables["Table"];
        Kapcsolat.Close();
        foa.ShowDialog();
    }
    catch (Exception exc)
    {
        MessageBox.Show(exc.Message, "Adatbázis hiba",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

# Több táblából álló DS – Kapcsolat létrehozása

- Az ADO.NET nem állítja elő automatikusan az adatbázis táblái közötti kapcsolatokat a DataSet táblái között
- A kapcsolat vizuális eszközökkel Visual Studio 2008-ban is beállítható
- Megoldás programból típusos DataSet esetén

```
 DataColumn dcElsődlegesKulcs = dsAdatok.dtÉrtékelés.JegyColumn;  
 DataColumn dcIdegenKulcs = dsAdatok.dtDolgozatok.JegyColumn;  
 DataRelation drKapcsolat = new DataRelation("Kapcsolat", dcElsődlegesKulcs,  
      dcIdegenKulcs);  
 dsAdatok.Relations.Add(drKapcsolat);
```



# DS-ben tárolt adatok módosítása<sub>1</sub>

- Minden cella közvetlenül írható
- Sor törlése: meghívjuk DataRow objektum **Delete()** vagy **Remove()** metódusát
- A Remove() meghívja az AcceptChanges()-t is
- Beszúrás:
  - Új sorobjektum előállítása
  - Sorobjektum hozzáadása a tábla Rows gyűjteményéhez
- A DS tárolja az eredeti és a módosított adatokat
  - Elfogadás: AcceptChanges()
  - Visszavonás: RejectChanges()

# DataSet-ben tárolt adatok módosítása<sub>2</sub>

- Rekord hozzáadása

```
dsAdatok.dtDolgozatok.AdddtDolgozatokRow("Sorozatfüggő Klementina",  
    "Hogyan csökkenthetjük a tvtől távol töltött időt?",3);
```

- Rekord módosítása

```
dsAdatok.dtDolgozatokRow dr = dsAdatok.dtDolgozatok.FindByHallgató("Sorozatfüggő Klementina");  
if (dr != null)  
{  
    dr.BeginEdit();  
    dr.Jegy = 4;  
    dr.EndEdit();  
}
```

```
dr = dsAdatok.dtDolgozatok.FindByHallgató("Törlendő Jónás");  
if (dr != null)  
    dsAdatok.dtDolgozatok.RemovedtDolgozatokRow(dr);
```



# Változások érvényesítése az adatbázisban

- A DataAdapter objektum **Update()** metódusának meghívása
- A DataAdapter elküldi a megfelelő INSERT, UPDATE, DELETE SQL utasításokat
- Az SQL utasításokat
  - a vizuális fejlesztés során automatikusan generálja a fejlesztőrendszer (nem minden adatbázis esetén támogatott, de pl. MS SQL-nél igen)
  - CommandBuilder objektummal állítjuk elő, de csak egytáblás adatbázisnál működik, és csak akkor, ha van elsődleges kulcs
  - a programozó írja meg

# Adatok megjelenítése - adatkötés

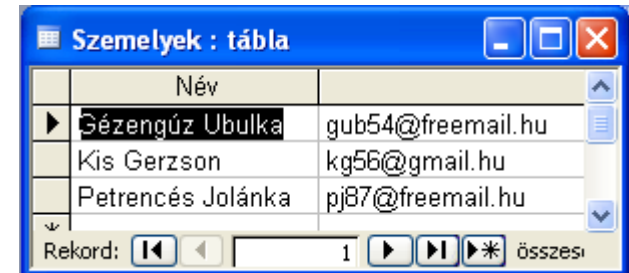
- Egyszerűen és kényelmesen teremt kapcsolatot egy form vezérlői és egy adathalmaz között
- Kötés létesíthető DataSet-hez, tömbhöz, gyűjteményhez vagy más vezérlőhöz (támogatnia kell az indexelt hozzáférést)
- Osztályozás
- Egyszerű kötés (egy rekord egy adata)
- Komplex kötés (több rekord)
  
- Közvetlen kötés
- Közvetett kötés (BindingSource)

# Egyszerű kötés

- Egyszerre egy adatot köt egy vezérlő egy tulajdonságához
- A vezérlő egy tulajdonságába egy DataSet egy táblája egy oszlopának az aktuális mezője kerül.
- Programból:
- Beállítjuk a vezérlő **DataBindings** tulajdonságát az adatforrást megadva

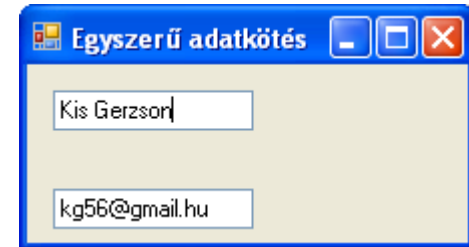
# Példa egyszerű adatkötésre

```
private void frmFőablak_Load(object sender, EventArgs e)
{
    try
    {
        // Kapcsolat objektum létrehozása
        OdbcConnection Kapcsolat = new OdbcConnection();
        // Kapcsolat sztring definiálása
        Kapcsolat.ConnectionString =
            "DSN=Személyek;DATABASE=Szemelyek";
        Kapcsolat.Open();
        OdbcDataAdapter Adapter = new OdbcDataAdapter();
        Adapter.SelectCommand = new
            OdbcCommand("SELECT * FROM Szemelyek WHERE NEV='Kis Gerzson'",
            Kapcsolat);
        DataSet dataset = new DataSet();
        Adapter.Fill(dataset);
        Binding bdKötés1 = new Binding("Text",
            dataset.Tables[0], "Nev", true);
        tbNév.DataBindings.Add(bdKötés1);
        Binding bdKötés2 = new Binding("Text",
            dataset.Tables[0], "email", true);
        tbEmail.DataBindings.Add(bdKötés2);
        Kapcsolat.Close();
    }
    catch (Exception exc)
    {
        MessageBox.Show(exc.Message, "Adatbázis hiba",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```



	Név	
▶	Gézengúz Ubulka	gub54@freemail.hu
	Kis Gerzson	kg56@gmail.hu
	Petrencés Jolánka	pj87@freemail.hu

Rekord: 1 összesi



Kis Gerzson

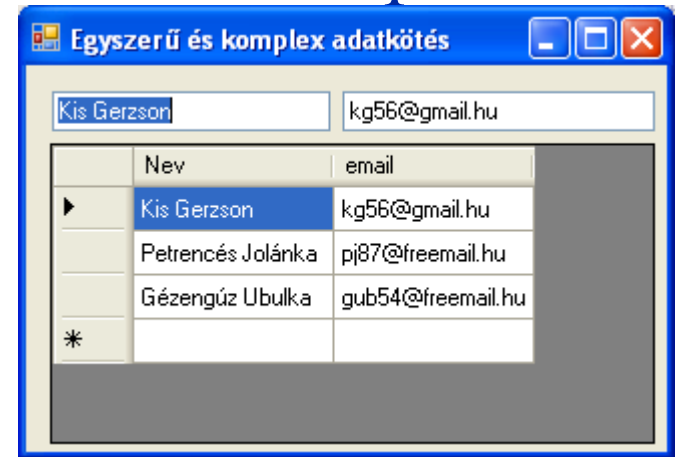
kg56@gmail.hu

# Komplex adatkötés

- Egy listát kötnék egy vezérlőhöz
- Egyszerre egynél több rekord adatai jeleníthetők meg
- **DataSource**
- **DisplayMember** – az adatforrás mely oszlopát kell megjeleníteni

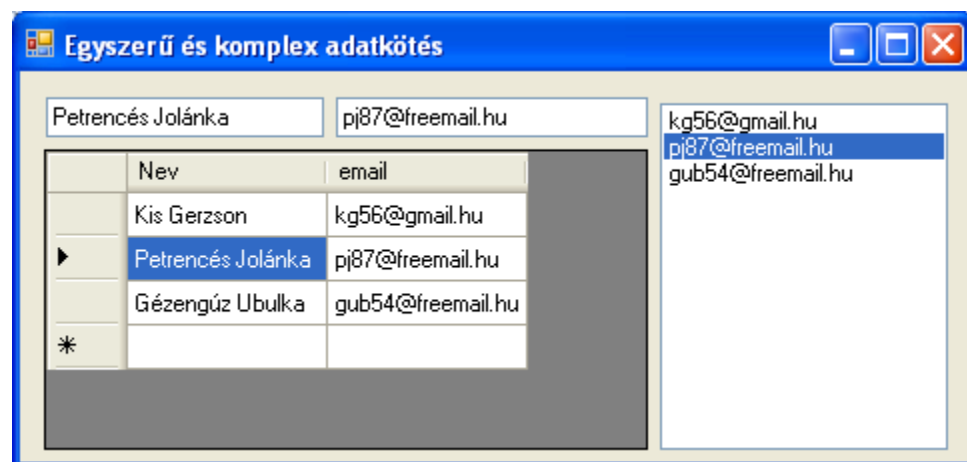
# Példa komplex adatkötésre<sub>1</sub>

```
private void frmFőablak_Load(object sender, EventArgs e)
{
    try
    {
        // Kapcsolat objektum létrehozása
        OdbcConnection Kapcsolat = new OdbcConnection();
        // Kapcsolat sztring definiálása
        Kapcsolat.ConnectionString =
            "DSN=Személyek;DATABASE=Szemelyek";
        Kapcsolat.Open();
        OdbcDataAdapter Adapter = new OdbcDataAdapter();
        Adapter.SelectCommand = new
            OdbcCommand("SELECT * FROM Szemelyek",
                Kapcsolat);
        DataSet dataset = new DataSet();
        Adapter.Fill(dataset);
        Binding bdKötés1 = new Binding("Text",
            dataset.Tables[0], "Nev", true);
        tbNév.DataBindings.Add(bdKötés1);
        Binding bdKötés2 = new Binding("Text",
            dataset.Tables[0], "email", true);
        tbEmail.DataBindings.Add(bdKötés2);
        dgvTáblázat.DataSource = dataset.Tables["Table"];
        Kapcsolat.Close();
    }
    catch (Exception exc)
    {
        MessageBox.Show(exc.Message, "Adatbázis hiba",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```



# Példa komplex adatkötésre<sub>2</sub>

```
private void frmFőablak_Load(object sender, EventArgs e)
{
    try
    {
        // Kapcsolat objektum létrehozása
        OdbcConnection Kapcsolat = new OdbcConnection();
        // Kapcsolat sztring definiálása
        Kapcsolat.ConnectionString =
            "DSN=Személyek;DATABASE=Szemelyek";
        Kapcsolat.Open();
        OdbcDataAdapter Adapter = new OdbcDataAdapter();
        Adapter.SelectCommand = new
            OdbcCommand("SELECT * FROM Szemelyek",
                Kapcsolat);
        DataSet dataset = new DataSet();
        Adapter.Fill(dataset);
        Binding bdKötés1 = new Binding("Text",
            dataset.Tables[0], "Név", true);
        tbNév.DataBindings.Add(bdKötés1);
        Binding bdKötés2 = new Binding("Text",
            dataset.Tables[0], "email", true);
        tbEmail.DataBindings.Add(bdKötés2);
        dgvTáblázat.DataSource = dataset.Tables["Table"];
        lbLista.DataSource = dataset.Tables["Table"];
        lbLista.DisplayMember = "email";
        Kapcsolat.Close();
    }
    catch (Exception exc)
    {
        MessageBox.Show(exc.Message, "Adatbázis hiba",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```



# BindingContext

- Gondoskodik az adatforrás és a vezérlő közötti szinkronizálásról
- Ha több vezérlő kötődik ugyanahhoz az adatforráshoz, gondoskodik az ezek közötti szinkronizálásról
- A vezérlőket tároló elem (pl. form) BindingContext adattagjában van eltárolva a BindingContext-re vonatkozó referencia
- A BC automatikusan jön létre
- A BC Egy gyűjtemény:
  - Listaszerű adatoknál a háttérben egy **CurrencyManager** dolgozik
  - Egyedi adatoknál a háttérben egy **PropertyManager** dolgozik



```

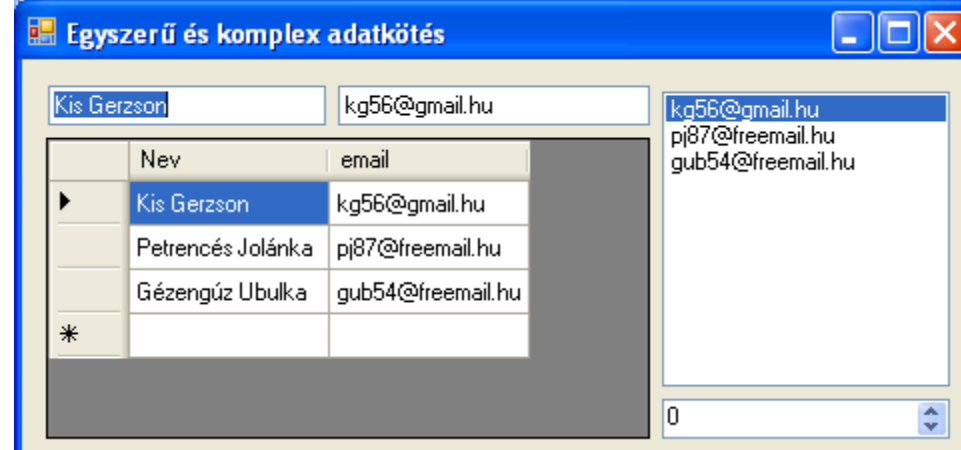
DataSet dsAdatok;
private void frmFőablak_Load(object sender, EventArgs e)
{ try
{ // Kapcsolat objektum létrehozása
OdbcConnection Kapcsolat = new OdbcConnection();
// Kapcsolat sztring definiálása
Kapcsolat.ConnectionString =
    "DSN=Személyek;DATABASE=Szemelyek";
Kapcsolat.Open();
OdbcDataAdapter Adapter = new OdbcDataAdapter();
Adapter.SelectCommand = new
    OdbcCommand("SELECT * FROM Szemelyek",
        Kapcsolat);
dsAdatok = new DataSet();
Adapter.Fill(dsAdatok);
Binding bdKötés1 = new Binding("Text",
    dsAdatok.Tables[0], "Nev", true);
tbNév.DataBindings.Add(bdKötés1);
Binding bdKötés2 = new Binding("Text",
    dsAdatok.Tables[0], "email", true);
tbEmail.DataBindings.Add(bdKötés2);
dgvTáblázat.DataSource = dsAdatok.Tables["Table"];
lbLista.DataSource = dsAdatok.Tables["Table"];
lbLista.DisplayMember = "email";
Kapcsolat.Close();
CurrencyManager cmKezel=
    (CurrencyManager)this.BindingContext[dsAdatok.Tables["Table"]];
nudSorszám.Minimum = 0;
nudSorszám.Maximum = cmKezel.Count - 1;
nudSorszám.Value = cmKezel.Position;
}
catch (Exception exc)
{ MessageBox.Show(exc.Message, "Adatbázis hiba",
    MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}

```

```

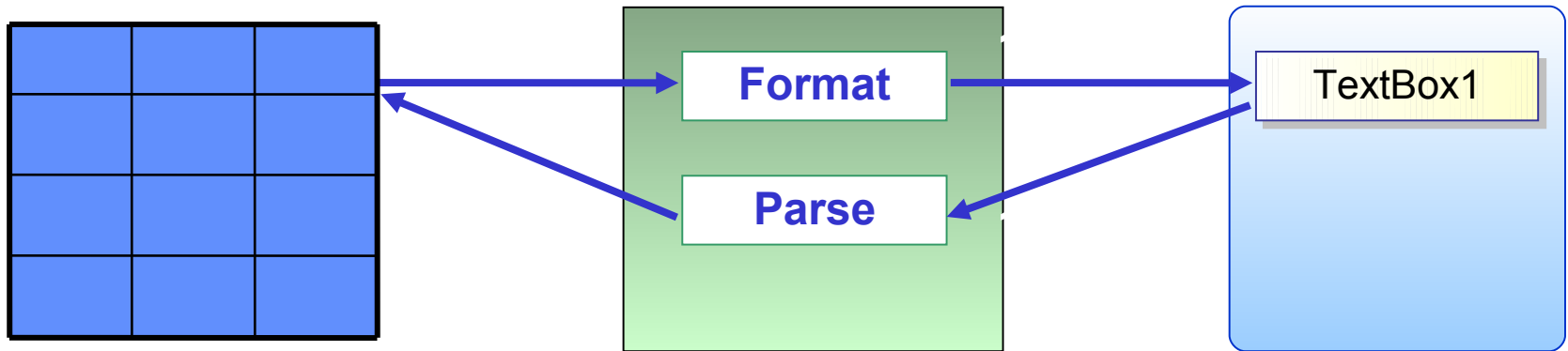
private void nudSorszám_ValueChanged(object sender, EventArgs e)
{ CurrencyManager cmKezel =
    (CurrencyManager)this.BindingContext[dsAdatok.Tables["Table"]];
    cmKezel.Position = (int)nudSorszám.Value;
}

```



Lépegetés  
megvalósítása  
CurrencyManager  
segítségével

# Format és Parse események és az adatmegjelenítés



# Format és Parse

- Az adatforrásból származó adatok típusa gyakran nem egyezik meg a vezérlő által elfogadott típussal
- **Format** esemény: adat kerül az adatforrásból a vezérlőhöz (adatkötés létrehozásakor, Position értékének változásakor, rendezés, szűrés)
- **Parse** esemény: adat mozog a vezérlőtől az adatforrás felé (a vezérlő Validated eseménye után, az EndCurrentEdit metódus meghívásakor, ha változik a Position)

# Példa

Format és Parse használata

	ÁruNeve	EgységÁr	ÁruKód
▶	Fa korcsolya	4512	asd123456
	Kanyarfűró g	9999	axd123456
	Nikkelezett sz	119999	kefe12545

4 512,00 Ft

```
private dsMintaAdatokTípus dsMintaAdatok;
public frmFormatÉsParse()
{ InitializeComponent();

    dsMintaAdatok=new dsMintaAdatokTípus();
    Feltölt();
    dgAdatRács.DataSource=dsMintaAdatok.Termékek;
    tbEgységÁr.DataBindings.Add("Text",dsMintaAdatok.Termékek,
        "EgységÁr");
    Binding bKötés=tbEgységÁr.DataBindings["Text"];
    bKötés.Format+=new ConvertEventHandler(bKötés_Format);
    bKötés.Parse+=new ConvertEventHandler(bKötés_Parse);
}

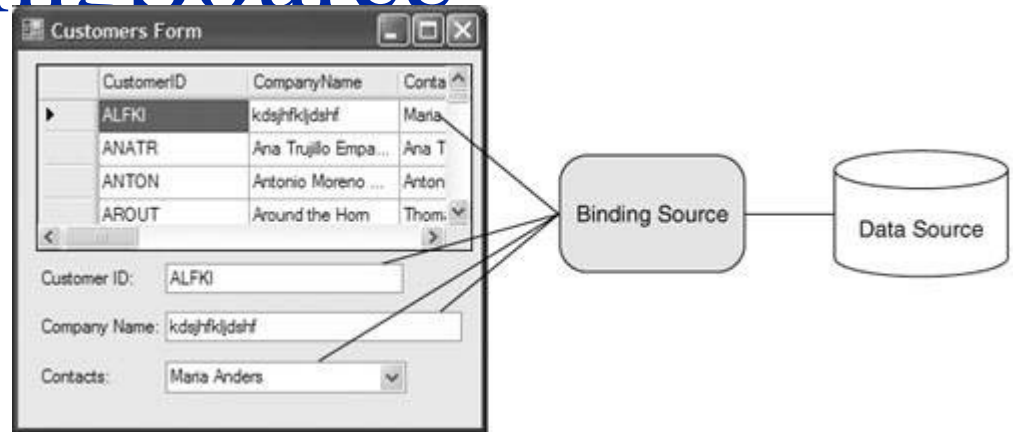
protected void Feltölt()
{ dsMintaAdatok.Termékek.AddTermékekRow("Fa korcsolya", 4512, "asd123456");
  dsMintaAdatok.Termékek.AddTermékekRow("Kanyarfűró gép", 9999, "axd123456");
  dsMintaAdatok.Termékek.AddTermékekRow("Nikkelezett szemmérték", 119999, "kefe12545");
}

private void bKötés_Format(object sender, ConvertEventArgs cea)
{ cea.Value = Convert.ToDecimal(cea.Value).ToString("c");
}

private void bKötés_Parse(object sender, ConvertEventArgs cea)
{ cea.Value = Decimal.Parse(cea.Value.ToString(),
    NumberStyles.Currency);
}
```

# Közvetett adatkötés - BindingSource

- Az adatforrás és a vezérlő között egy újabb réteg jelenik meg



- Megkönnyíti az adatok közötti navigálást és az adatforrás esetleges lecserélését
- DataSource
- DataMember

# BindingSource

- Metódusok és tulajdonságok az adatmódosításhoz
- Sort, Filter, MoveNext, MoveLast, Remove
- Események az adatforrásban történő változásokhoz

```

BindingSource bsKapocs;
private void frmFőablak_Load(object sender, EventArgs e)
{
    try
    {
        // Kapcsolat objektum létrehozása
        OdbcConnection Kapcsolat = new OdbcConnection();
        // Kapcsolat sztring definiálása
        Kapcsolat.ConnectionString =
            "DSN=Személyek;DATABASE=Szemelyek";
        Kapcsolat.Open();
        OdbcDataAdapter Adapter = new OdbcDataAdapter();
        Adapter.SelectCommand = new
            OdbcCommand("SELECT * FROM Szemelyek",
                Kapcsolat);
        DataSet dataset = new DataSet();
        Adapter.Fill(dataset);
        // BindingSource objektum létrehozása
        bsKapocs = new BindingSource();
        bsKapocs.DataSource = dataset.Tables[0];
        Binding bdKötés1 = new Binding("Text",
            bsKapocs, "Nev", true);
        tbNév.DataBindings.Add(bdKötés1);
        Binding bdKötés2 = new Binding("Text",
            bsKapocs, "email", true);
        tbEmail.DataBindings.Add(bdKötés2);
        dgvTáblázat.DataSource = bsKapocs;
        lbLista.DataSource = bsKapocs;
        lbLista.DisplayMember = "email";
        Kapcsolat.Close();
    }
    catch (Exception exc)
    {
        MessageBox.Show(exc.Message, "Adatbázis hiba",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

/// <summary>
/// Név szerint növekvően rendez
/// </summary>
private void btRendezNövekvő_Click(object sender, EventArgs)
{
    bsKapocs.Sort = "Nev ASC";
}

```

**Egyszerű és komplex adatkötés**

Kis Gerzson      kg56@gmail.hu      kg56@gmail.hu  
 pi87@freemail.hu  
 gub54@freemail.hu

	Nev	email
▶	Kis Gerzson	kg56@gmail.hu
	Petrencés Jolánka	pi87@freemail.hu
	Gézengúz Ubulka	gub54@freemail.hu
*		

Növekvően rendez

**Egyszerű és komplex adatkötés**

Gézengúz Ubulka      gub54@freemail.hu      gub54@freemail.hu  
 kg56@gmail.hu  
 pi87@freemail.hu

	Nev	email
▶	Gézengúz Ubulka	gub54@freemail.hu
	Kis Gerzson	kg56@gmail.hu
	Petrencés Jolánka	pi87@freemail.hu
*		

Növekvően rendez

# C# Projektek és megoldások

- **Projekt („Project”)**
- A projekt egy futtatható programhoz vagy más típusú szoftvermodulhoz tartozó, együtt kezelt szoftverelemek (többségében fájlok) összessége.
  - C# forráskód („source code”) [\*.cs]
  - Hivatkozások („references”)
  - Beállítások („settings”) [\*.settings]
  - Konfigurációs fájlok („configuration”) [\*.config]
  - Egyéb erőforrások („resources”) [\*.resx, \*.rc, \*.resources]

A projekthez tartozó elemek mappák létrehozásával hierarchikus fastruktúrába rendezhetők.

A C# projekteket a Visual Studio \*.csproj kiterjesztésű fájlokban tárolja.

- **Megoldás („Solution”)**

A megoldás több összefüggő projekt együttes kezelését teszi lehetővé. Ezek a projektek virtuális mappák segítségével hierarchikus fastruktúrába is rendezhetők.

A megoldásokat a Visual Studio \*.sln kiterjesztésű fájlokban tárolja.



# Projektek típusai

A legfontosabb projekttypusok

- Grafikus Windows alkalmazás („Windows Application”)
  - Végeredménye egy „exe” kiterjesztésű futtatható program.
- Parancsértelmezőben futó Windows alkalmazás („Console Application”)
  - Végeredménye egy „exe” kiterjesztésű futtatható program.
- Háttérben futó Windows rendszerszolgáltatás („Windows Service”)
  - Végeredménye egy „exe” kiterjesztésű futtatható program.
- Osztálykönyvtár („Class Library”)
  - Végeredménye egy „dll” kiterjesztésű könyvtárfájl.
- Windows vezérlők gyűjteménye („Windows Control Library”)
  - Végeredménye egy „dll” kiterjesztésű könyvtárfájl.
- Webre szánt vezérlők gyűjteménye („Web Control Library”)
  - Végeredménye egy „dll” kiterjesztésű, webkiszolgáló által használt könyvtárfájl.
- Üres projekt („Empty Project”)
  - Ehhez a projekttypushoz kézzel kell a megfelelő elemeket hozzáadni.

# DLL készítése C#-ban és annak felhasználása

- A DLL(**D**ynamic-**L**ink **L**ibrary) elkészítése: *File / New Project / Class Library*.
- . A teljes névtér elérhető lesz, de szokás szerint statikus függvényeket készítenek. Fordítás után a megfelelő kimeneti könyvtárban előáll a DLL.
- A DLL-t nem lehet elindítani, nincs Main függvénye, de le lehet fordítani (BUILD / BUILD SOLUTION), *Console Application* forráskódjának *./bin/Debug* alkönyvtárába kerül
- A C# DLL felhasználása C#-ban
- A projekthez a Projekt/Hivatkozás hozzáadása (Project/Add reference) menüpontban a .NET szerelvény böngésző
- (.NET Assembly Browser) fül alatt tallózni kell a DLL le-t és hozzáadni.
- Ezután elérhető a DLL file tartalma. Érdemes using segítségével hozzáadni a DLL névtérét, majd az egyes statikus metódusokra az osztálynév segítségével hivatkozhatunk.

# Felhasznált forrás:

- **Zs. Csaba Johanyák: Vizualis programozás:  
Adatbázisok elérése**