

# **ADO.NET**

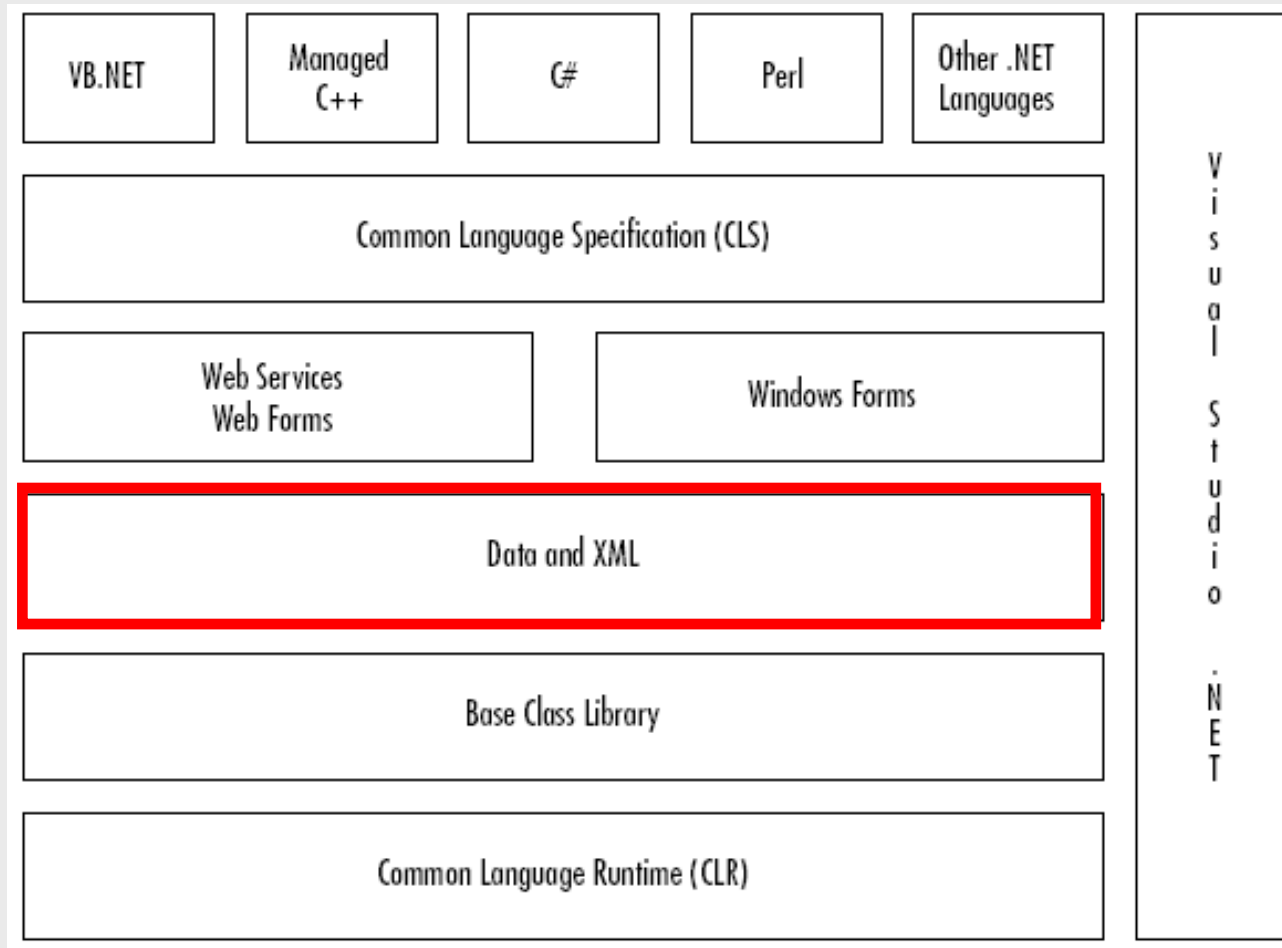
## **ActiveX Data Objects .NET**

---

**6/13**

# ADO.NET i .NET Framework

- ❑ **ActiveX Data Objects .NET (ADO.NET)** je tehnologija koja se unutar .NET Framework koristi za pristup podacima.



```
using
System.Data
System.Data.Common
System.Data.SqlClient
System.Data.OleDb
System.Data.SqlTypes
System.Xml
```

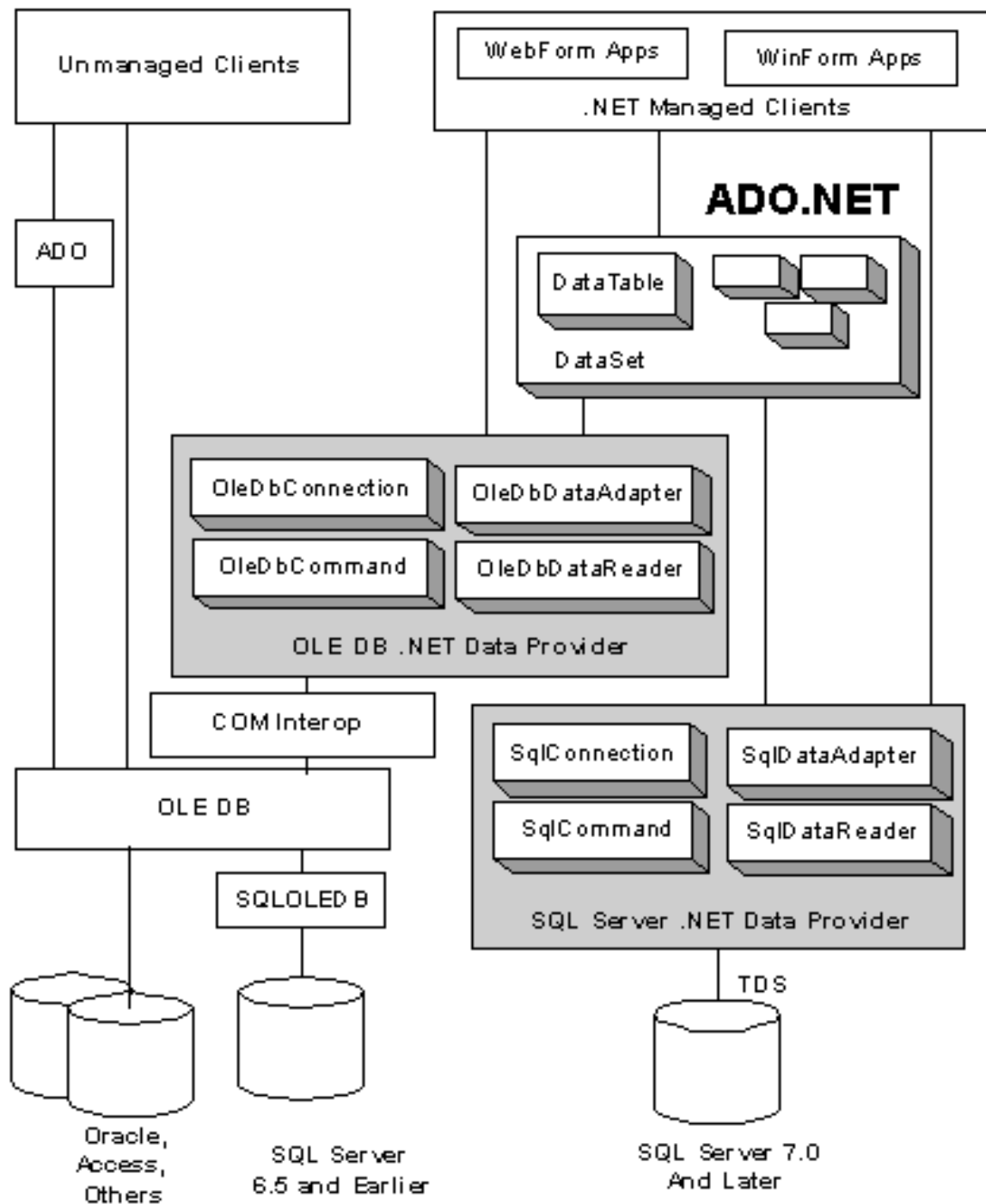
# Pohrana podataka

## ❑ Podrška različitim tipovima pohrane – spremištima (data storage)

- ADO.NET omogućuje pristup bazama podataka, ali i drugim spremištima podataka, za koje postoji odgovarajući davatelj (provider)
- Nestrukturirani podaci
- Strukturirani, nehijerarhijski podaci
  - Comma Separated Value (CSV) datoteke, Microsoft Excel proračunske tablice, ...
- Hijerarhijski podaci
  - XML dokumenti
- Relacijske baze podataka
  - SQL Server, Oracle, Access, druge

# Arhitektura

- ❑ **Data Consumer – korisnik podataka**
  - komunicira s ADO.NET i obrađuje podatke
- ❑ **DataProvider – dobavljač podataka**
  - rukuje komunikacijom s fizičkim medijem pohrane podataka
  - DataSet – reprezentira stvarne podatke



# Davatelji podataka

- ❑ Postoje dvije osnovne kategorije davatelja prilagođene različitim tehnologijama i smještene u odgovarajuće prostore imena
- ❑ ***System.Data.SqlClient***
  - optimiran za rad s RDBMS MS SQL Server
  - Razredi: *SqlCommand*, *SqlConnection*, *SqlDataReader*, *SqlDataAdapter*
- ❑ ***System.Data.OleDb***
  - generički pružatelj za rad s bilo kojim OLE Database (OLE DB) izvorom
  - npr: Oracle, MS JET, SQL OLE DB
  - Razredi: *OleDbCommand*, *OleDbConnection*, *OleDbDataReader*, *OleDbDataAdapter*
- ❑ Razredi su implementacija zajedničkog sučelja te imaju članove jednakih naziva
- ❑ Skupovi podataka ne ovise o fizičkoj ugradnji davatelja čime se postiže neovisnost aplikacije o fizičkom smještaju podataka

# Davatelj podataka *.NET Data Provider*

## ☐ Connection

- Povezivanje s izvorom podataka

## ☐ Command

- Izvršava naredbe nad izvorom podataka, tj. podacima

## ☐ DataReader

- Rezultat upita nad podcima (forward-only, read-only connected result set)

## ☐ ParameterCollection

- Parametri Command objekta

## ☐ Parameter

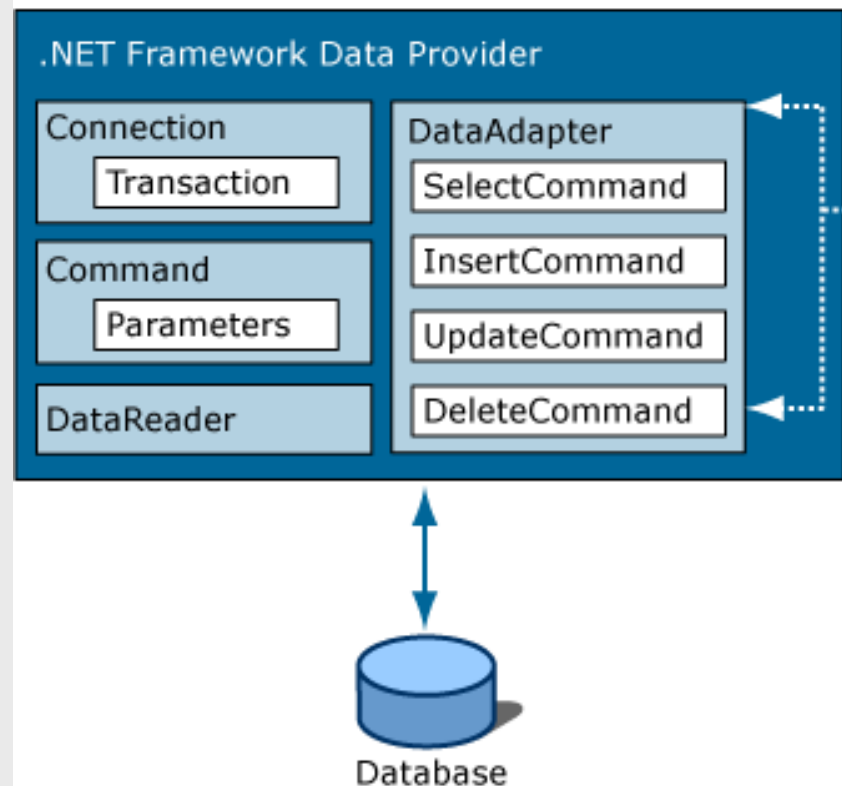
- Parametar parametrizirane SQL naredbe ili pohranjene procedure

## ☐ Transaction

- Nedjeljiva grupa naredbi nad podacima

## ☐ DataAdapter

- Most između podataka na izvoru i lokalne pohrane (DataSet i DataTable)



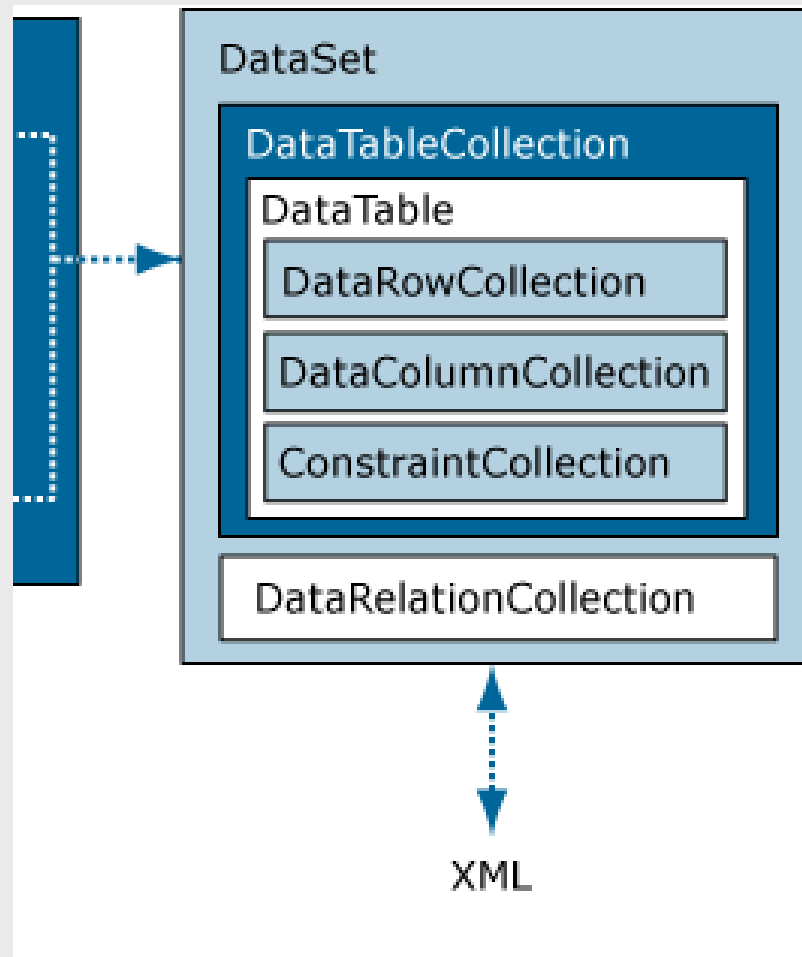
# Struktura i smještaj podataka

## ❑ Podaci se smještaju u dinamički skup podataka (DataSet)

- DataSet može sadržavati više tablica
- Dohvaćanje podataka iz više od jedne tablice ne zahtijeva povezivanje (JOIN)

## ❑ Podržano logičko oblikovanje podataka

- DataSet opisuje podatkovne strukture i veze podataka na vanjskim izvorima
- Veze između podataka (DataRelation) i dalje postoje
- podaci se mogu u potpunosti oblikovati i pohranjivati lokalno – XML schema



# Dinamički skup podataka (*DataSet*)

## ❑ DataSet

- skup(ovi) podataka u memoriji računala
- sadrži kolekciju `DataTable` objekata

## ❑ DataTable

- tablica podataka u memoriji računala
- `DataColumnCollection` – kolekcija atributa
- `DataRowCollection` – kolekcija zapisa
- `ConstraintCollection` – kolekcija ograničenja nad tablicom

## ❑ DataRow

- rukovanje retkom u `DataTable`

## ❑ DataColumn

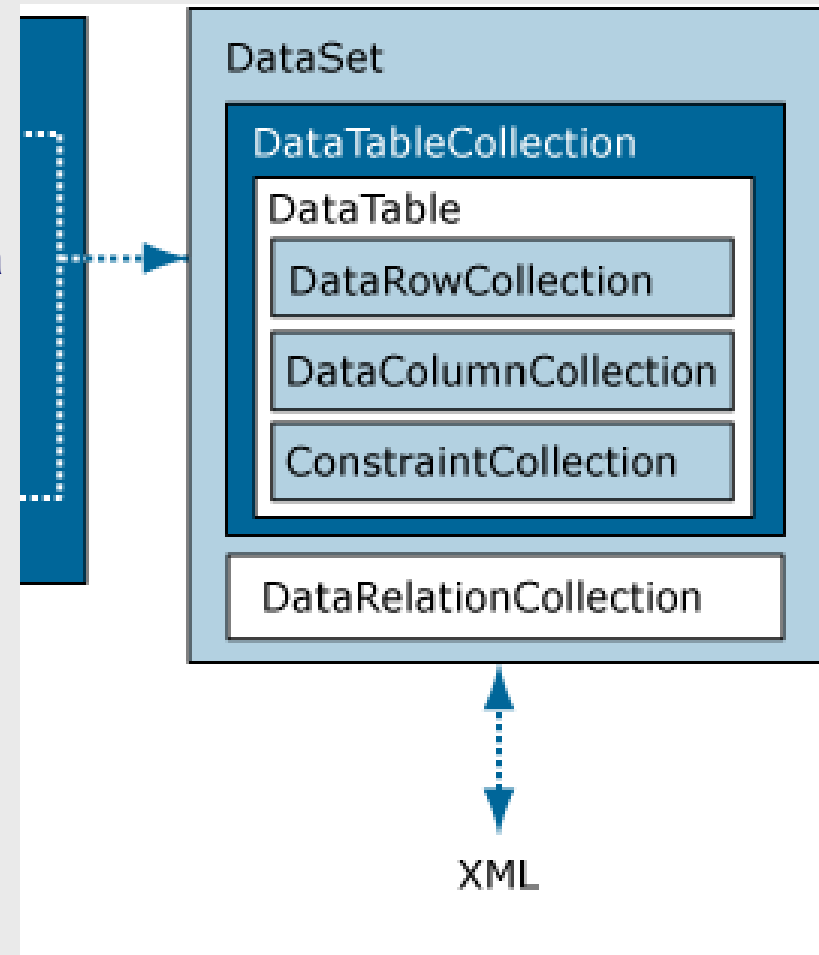
- definira stupce u `DataTable`

## ❑ DataRelationCollection

- kolekcija `DataRelation` objekata
- `DataRelation` - veza između dvije tablice (`DataTable`)

## ❑ DataViewManager

- definira poglede nad skupovima podataka





# Imenik *System.Data.Common*

## Razred

## Opis

`DataAdapter`

Skup objekata koji se koriste za punjenje skupa podataka i ažuriranje izvora podataka.

`DataColumnMapping`

Mapiranje stupaca (izvorni podatak, korišteni podatak) za objekte koji nasljeđuju `DataAdapter`.

`DataColumnMappingCollection`

Kolekcija `DataColumnMapping` objekata.

`DataTableMapping`

Opis mapirane veze između izvorne tablice i `DataTable`.

`DataTableMappingCollection`

Kolekcija `DataTableMapping` objekata.

`DbDataAdapter`

Implementacija `IDbDataAdapter` sučelja, osnova za funkcionalnost `DataAdapter`a.

`DBDataPermission`

Omogućava pružatelju provjeru prava pristupa.

`DBDataPermissionAttribute`

Pridružuje korisnički definiranu sigurnosnu akciju.

`DbDataRecord`

Implementira `IDataRecord` i `ICustomTypeDescriptor`, omogućuje povezivanje podataka za `DbEnumerator`.

`DbEnumerator`

Izlaže postupak `GetEnumerator`, za iteraciju kolekcije preko pružatelja podataka.

`RowUpdatedEventArgs`

Osigurava podatke za `RowUpdated` događaj.

`RowUpdatingEventArgs`

Osigurava podatke za `RowUpdating` događaj.

# Uvodni primjeri baze podataka i rada s podacima

## ❑ Windows

- Baze podataka - Microsoft Access, Microsoft SQL Server
- Izvori podataka - Control Panel \ Data Sources

## ❑ Visual Studio .NET

- Server Explorer - Servers, Data Connections
- Solution - Data Project, Windows Form + Data Set ... Schema
- Toolbox \ Data
  - DataSet, DataView
  - OleDbDataAdapter, OleDbConnection, OleDbCommand
  - SqlDataAdapter, SqlConnection, SqlCommand

## ❑ Čarobnjaci (data adapter desni klik)

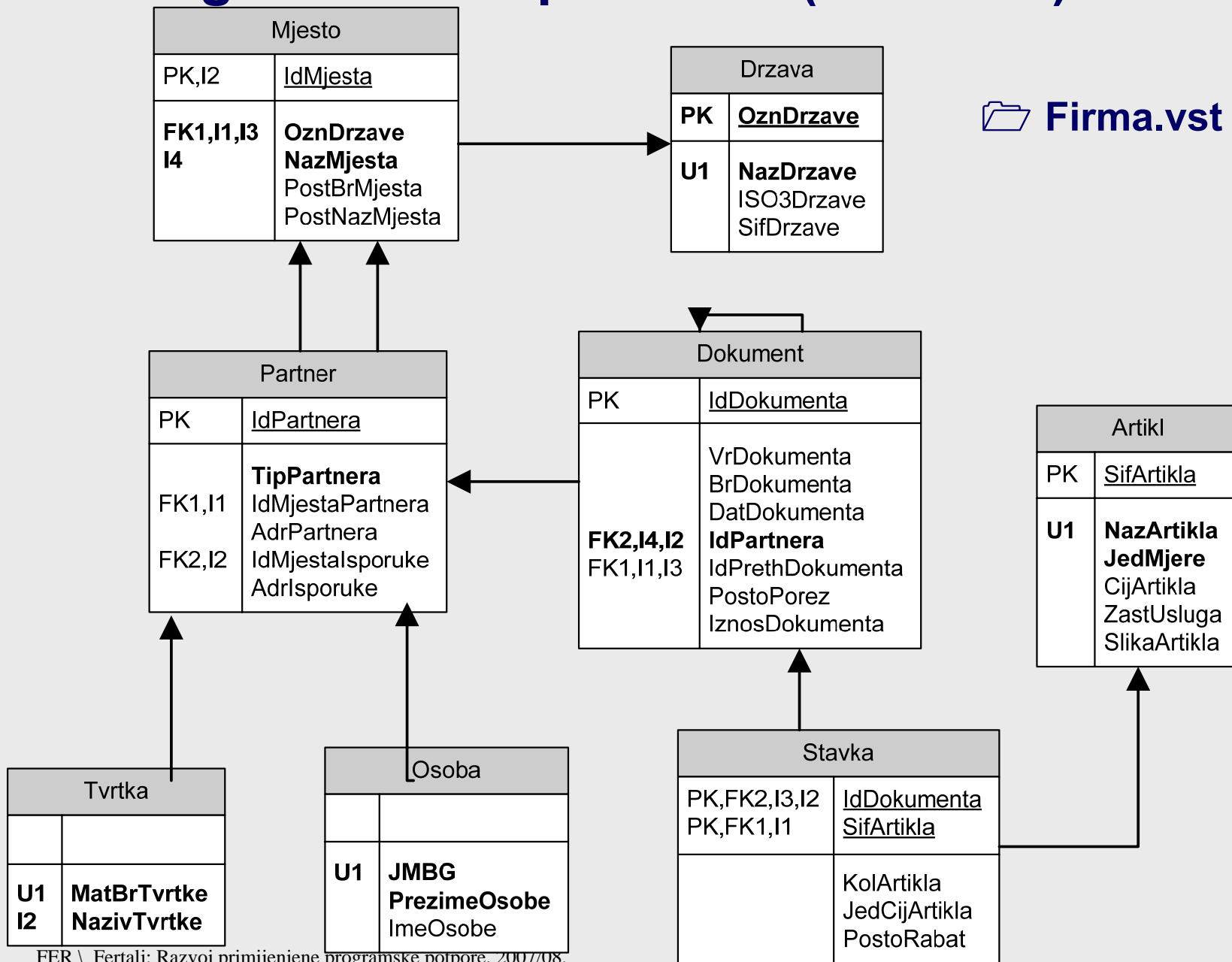
- Configure Data Adapter, Generate DataSet, Preview Data
- generiranje upita, grafička prezentacija struktura podataka

## ❑ Primjer baze podataka

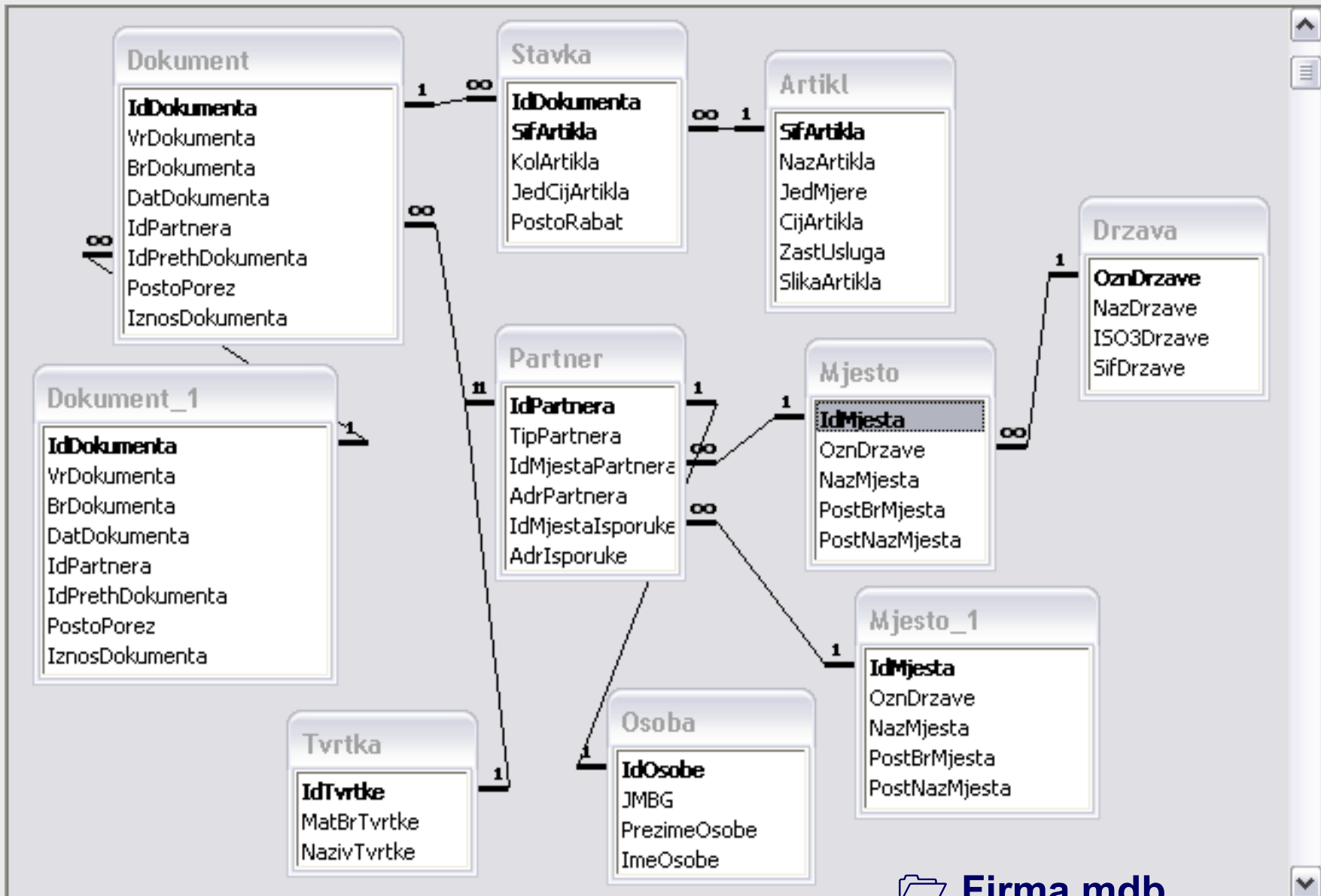
- BazePodataka.zip
- BazePodataka.txt

# Ogledna baza podataka (MS Visio)

 **Firma.vst**



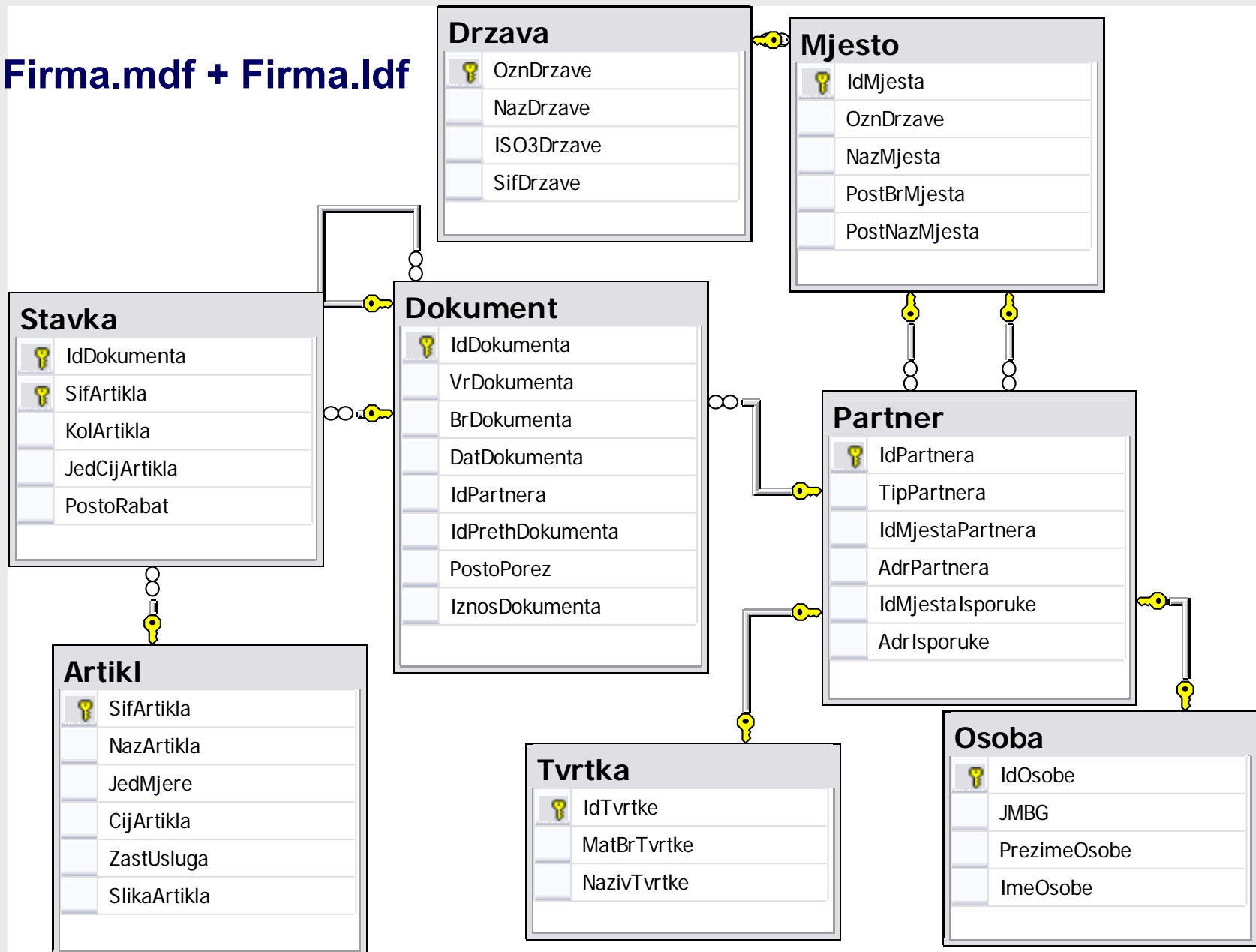
# Ogledna baza podataka (MS Access)



Firma.mdb

# Ogledna baza podataka (MS SQL Server)

Firma.mdf + Firma.Idf





# Zadatak projekta

## ☐ Zadatak projekta

- Ažuriranje rasporeda projekta
- Ugradnja baze podataka

## ☐ Preporuka je ugraditi prvo jednokorisničku bazu podataka

- jednostavnije rukovanje
- sličnost s primjerima na predavanjima

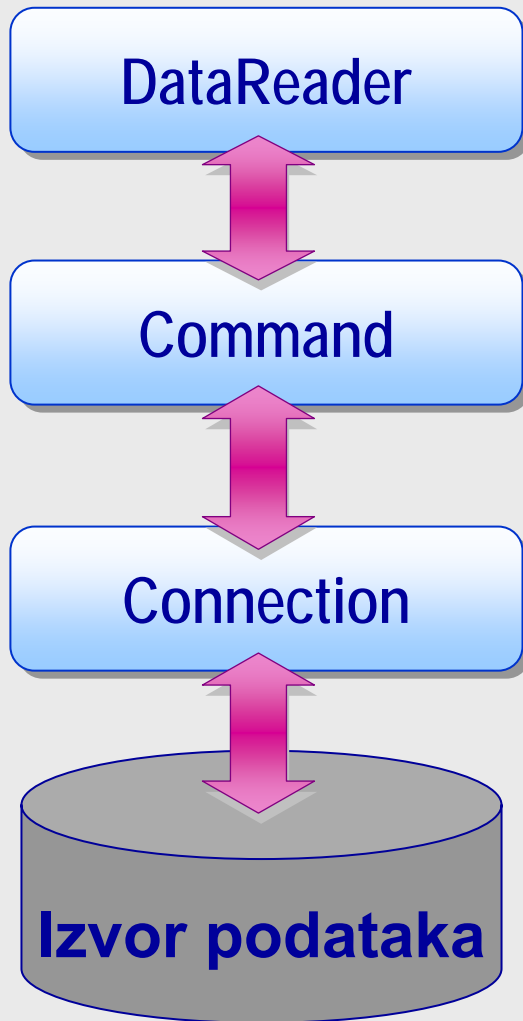
## ☐ Rok za ugradnju baze podataka nije eksplicitno zadan, ali ...

- Baza podataka će poslužiti za rukovanje podacima programa za domaću zadaću u narednom tjednu.

# Izravna obrada podataka

---

# Izravna obrada podataka



## ☐ Resursi na poslužitelju

1. Otvori konekciju
2. Izvrši naredbu
3. Obradi podatke u čitaču
4. Zatvori čitač
5. Zatvori konekciju



# Veza s podacima

## ❑ Konekcija (Connection)

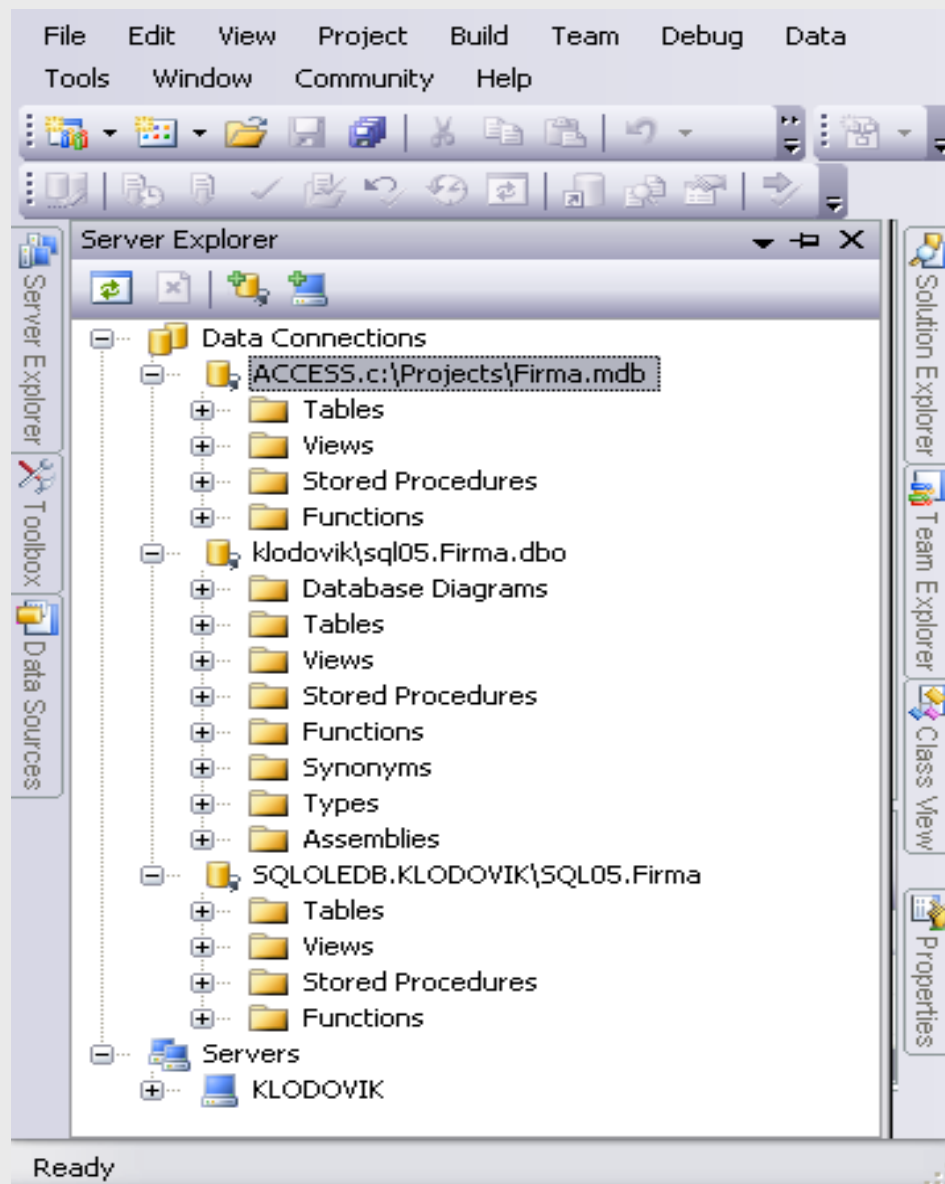
- veza s fizičkim izvorom podataka
- otvara i zatvara vezu s izvorom
- omogućuje transakcije

## ❑ Sučelje

- System.Data.IDbConnection

## ❑ Implementacije

- OleDbConnection i SqlConnection



# Osnovni članovi konekcije

## □ Svojstva

- `ConnectionString` – string parova postavki oblika naziv-vrijednost
  - jedino promjenjivo svojstvo
- `State` – bitovna oznaka stanja konekcije
  - `enum ConnectionState { Broken, Closed, Connecting, Executing, Fetching, Open }`

## □ Postupci

- `Open` – prikapčanje na izvor podatak
- `Close` - otkapčanje s izvora podataka

```
const string connStringOLEDBSQL =  
    "Provider=SQLOLEDB.1;Data Source=SERVER;  
    Initial Catalog=BAZA;User Id=KORISNIK;Password=SIFRA;";  
OleDbConnection oleDbConnection;  
...  
oleDbConnection = new OleDbConnection(connStringOLEDBSQL);  
oleDbConnection.Open();  
...
```

# Ostali članovi konekcije

## □ Svojstva

- `ConnectionTimeout` – vrijeme u kojem se čeka na otvaranje konekcije
  - ako se veza u tom roku ne otvori, nastupa iznimka (`SQLException` ili `OleDbException`)
  - standardno 15 sekundi; 0 – "beskonačno" čekanje
- `Database` – naziv baze podataka kojoj se želi pristupiti
  - može se promijeniti postupkom `ChangeDatabase()`, što ne vrijedi za BP *Oracle*

## □ Postupci

- `ChangeDatabase` – povezivanje s drugom bazom podataka
  - alternativno se za SQL Server može koristiti SQL `USE` naredba
- `CreateCommand` – vraća `IDbCommand` objekt specifičan za davatelja

## □ Događaj

- `StateChange(object sender, StateChangeEventArgs e)` – okida pri promjeni stanja konekcije
- `StateChangeEventArgs` objekt ima `ConnectionState` svojstva `CurrentState` i `OriginalState`

# Elementi svojstva `ConnectionString`

<code>AttachDBFilename</code>	Koristi se kada se želi pristupiti bazi podataka koja nije registrirana u SUBP (npr. .MDF koji nije vezan na SQL Server). Uobičajeno se koristiti parametar <code>Initial Catalog</code> .
<code>ConnectTimeout</code>	Vrijednost svojstva <code>Timeout</code> – čekanje do otvaranja ili iznimke
<code>Data Source</code>	Naziv samostojne baze podataka (npr. MS Access .MDB), naziv poslužitelja ili mrežna adresa poslužitelja baze podataka. Na lokalnom računalu može se koristiti naziv <code>local</code> ili <code>".."</code> .
<code>Initial Catalog / Database</code>	Naziv baze podataka.
<code>Integrated Security</code>	Postavlja se na <code>false</code> (default), <code>true</code> ili SSPI (Security Service Provider Interface – standardizirano sučelje za sigurnost distribuiranih aplikacija). Kada se postavi na SSPI, .NET se povezuje koristeći sustav zaštite OS Windows.
<code>Persist Security Info</code>	<code>True</code> ili <code>false</code> (default). Kad je postavljen na <code>false</code> , sigurnosno osjetljive postavke (npr. lozinka) se automatski uklanjaju iz <code>ConnectionString</code> nakon što je konekcija otvorena.
<code>User ID / UID</code>	Identifikator korisnika (korisničko ime) u bazi podataka.
<code>Password/PWD</code>	Lozinka za korisničko ime.

# Povezivanje s bazom podataka i postavljanje upita

## ❑ Primjer: ADO\Upitnik

**Postavke konekcije**

☐ SQL konekcija  
☐ OLEDB SQL konekcija  
☒ OLEDB Access konekcija

ConnectionString:  
Provider=Microsoft.Jet.OLEDB.4.0;Data Source=c:\Projects\Firma.mdb

Otvori Obavi NonQuery Scalar Zatvori

**SQL upit:**  
SELECT COUNT(\*) AS BrojDokumenata FROM Dokument

**Rezultat upita:**

BrojDokumenata
856

# Primjeri konekcija

## ❑ Primjer konekcija: (VS \ View) Server Explorer

- Za lokalni poslužitelj može se navesti (*local*) ili . točka

## ❑ Primjer: ADO\Upitnik

### ■ OleDb ConnectionString za MS Access

- `Provider=Microsoft.Jet.OLEDB.4.0;Data Source=c:\Projects\Firma.mdb`

### ■ System.Data.OleDb.OleDbConnection na SQL Server

- `Provider=SQLOLEDB;Data Source=KLODOVIK\SQL05;Integrated Security=SSPI;Initial Catalog=Firma`

### ■ System.Data.SqlClient.SqlConnection

- `Data Source=KLODOVIK\SQL05;Initial Catalog=Firma;Integrated Security=True`

## ❑ Administriranje korisnika obavlja se u SUBP

# Uspostavljanje veze s izvorom podataka

## ❑ Primjer: ADO\Upitnik

```
string connString = "";

OleDbConnection oleDbConnection;
SqlConnection sqlConnection;
IDbConnection konekcija;

...

    if ((radioButtonSQLCON.Checked))
    {
        sqlConnection = new SqlConnection(connString);
        konekcija = sqlConnection;
    }
    else
    {
        oleDbConnection = new OleDbConnection(connString);
        konekcija = oleDbConnection;
    }
konekcija.Open(); // višeobličje
```

# Zadatak za vježbu

- ❑ **Doraditi primjer konekcije obradom događaja i informiranjem o stanju konekcije**

```
...  
oleDbConnection.StateChange += new  
StateChangeEventHandler(oleDbConnection_StateChange);  
  
...  
private void myConn_StateChange  
    (object sender, StateEventArgs e) {  
    // e.OriginalState // .ToString()  
    // e.CurrentState // .ToString()  
}
```



# Sučelje IDbCommand

- ❑ **Reprezentira SQL naredbe koje se obavljaju nad izvorom podataka**
  - upit može biti SQL naredba ili pohranjena procedura
- ❑ **Implementacija u .NET pružateljima koji pristupaju relacijskim BP**
  - *OleDbCommand* i *SqlCommand*
- ❑ **Svojstva**
  - `Connection`: konekcija na izvor podataka
  - `CommandText`: SQL naredba, ime pohranjene procedure ili ime tablice
  - `CommandType`: tumačenje teksta naredbe, standardno `Text`
    - `enum CommandType { Text, StoredProcedure, TableDirect }`
- ❑ **Postupci**
  - `ExecuteReader` – izvršava naredbu i vraća `DataReader` cursor
  - `ExecuteNonQuery` – izvršava naredbu koja vraća broj obrađenih zapisa, npr. naredbe `UPDATE`, `DELETE` ili `INSERT`.
  - `ExecuteScalar` – izvršava naredbu koja vraća jednu vrijednost, npr. rezultat agregatne funkcije

# Sučelje *IDataReader*

## ❑ Razredi *OleDbDataReader* i *SqlDataReader* – implementiraju `System.Data.IDataReader`

- Rezultat upita nad podacima (forward-only, read-only connected result set).

## ❑ Svojstva

- `FieldCount` - broj stupaca u rezultatu upita
- `HasRows` – indikator da `DataReader` objekt sadrži zapise
- `IsClosed` – indikator da je `DataReader` objekt zatvoren
- `Item` – vrijednost stupca u izvornom obliku
  - `public virtual object this[int] {get;}`
- `RecordsAffected` - broj zapisa obrađenih naredbom koja mijenja podatke
  - 0 ako nije obrađen ni jedan zapis, -1 za `SELECT` naredbu

## Postupci

- `Read` – čita sljedeći zapis u `DataReader`
  - vraća `true` ako postoji još zapisa
- `Close` – zatvara `DataReader` objekt, ali ne i `Connection` koji čita

# Primjer izvođenja naredbe za dohvat podataka

## ❑ Primjer: ADO\Upitnik

```
// analogno za OleDb
SqlCommand sqlCommand = new SqlCommand(textBoxUpit.Text,
                                       sqlConnection);
SqlDataReader sqlReader = sqlCommand.ExecuteReader();

IDataReader reader = sqlReader;

// obrada čitača
while (reader.Read()) // isto što i sqlReader.Read()
{
    for (int i = 0; i < reader.FieldCount; i++)
        // čini nešto s reader[i] //.ToString()
}
reader.Close(); // isto što i sqlReader.Close()
```

# Primjer izvođenja drugih upita

## ❑ Primjer: ADO\Upitnik

```
// analogno za OleDb

SqlCommand sqlCommand = new SqlCommand(textBoxUpit.Text,
                                         sqlConnection);

IDbCommand command = sqlCommand;

// naredba koja vraća broj obrađenih zapisa
int result = command.ExecuteNonQuery();

// naredba koja vraća jednu vrijednost
object o = command.ExecuteScalar();
```

# Ostali članovi *IDbCommand*

## ❑ Svojstva

- `CommandTimeout`: broj sekundi čekanja na izvršenje (standardno 30s)
- `Parameters` – kolekcija parametara (argumenata) naredbe
  - `Parameter` - parametar parametrizirane SQL naredbe ili pohranjene procedure, sa svojstvima: `DbType`, `IsNullable`, `OleDbType`, `ParameterName`, `Precision`, `Scale`, `Size`, `SourceColumn`
- `Transaction` – transakcija koje je naredba dio (o transakcijama kasnije)
- `UpdatedRowSource` – određuje način ažuriranja izvora podataka, kad se naredba koristi sa skupom podataka i prilagodnikom podataka

## ❑ Postupci

- `Cancel` – pokušaj prekida naredbe koja se izvršava
  - da bi prekid bio moguć, naredba mora biti pokrenuta na drugoj niti
  - u protivnom će kod biti blokiran, jer se naredbe izvršavaju sinhrono
- `CreateParameter` – kreira novi `Parameter` objekt, koji se dodaje u kolekciju `Command.Parameters`
  - primjer: `public DbParameter CreateParameter();`
- `Prepare` – kada je `CommandType` postavljen na `StoredProcedure`, postupak se koristi za pripremu (prekompilaciju) naredbe na izvoru podataka, s namjerom poboljšanja brzine njenog izvođenja

# Ostali članovi *IDataReader*

## ❑ Postupci

- **GetName** – vraća naziv za zadani redni broj stupca
- **GetOrdinal** – vraća redni broj za zadano ime stupca
- **GetValue** – dohvaća vrijednost zadanog stupca za aktualni redak
  - `public virtual object GetValue( int ordinal );`
- **GetValues** – dohvaća aktualni redak kao polje objekata
  - `public virtual int GetValues( object[] values );`
- **GetType** – dohvaća vrijednost zadanog stupca u određenom tipu, na primjer `GetChar` ili npr.
  - `DataReader rdrArtikl = cmdArtikl.ExecuteReader();`
  - `int sifraArtikla = readerArtikl.GetInt32(0);`
- **GetSchemaTable** – dobavlja `DataTable` objekt s opisom podataka
- **NextResult** – pomiče se na sljedeći rezultirajući skup, za naredbe koje vraćaju više skupova
  - vraća `true` ako postoji još rezultata

# Zadaci za vježbu

## ❑ Za svaki pročitani zapis iz tablice *Artikli*

- Provjeriti ima li jedinicu mjere iz skupa { "h", "kom", "kg", "l", "pak" } .
- Ukoliko nema, ažurirati jedinicu mjere vrijednošću "---".
- Ažuriranje provesti kreiranjem odgovarajuće SQL naredbe za svaki zapis koji treba mijenjati.

## ❑ Prebaciti pojedini redak čitača u polje objekata.

- Podatke iz polja objekata prikazati u ListBox kontroli koristeći naredbu *ForEach* prema uzoru indeksiranja iz sljedećeg primjera.

```
while (reader.Read())
{
    Object [] cols = new Object[10] ;
    reader.GetValues( cols );

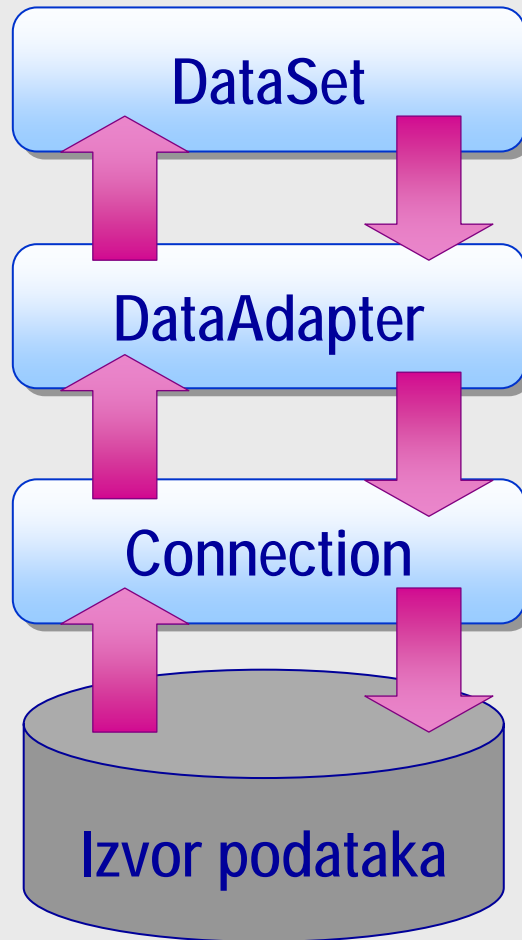
    Console.WriteLine( cols[0].ToString() + " | " + cols[1] );
}
```

# Lokalna obrada podataka

---



# Lokalna obrada podataka



☐ Podaci se obrađuju lokalno

1. Otvori konekciju
2. Napuni *DataSet*
3. Zatvori konekciju
4. Obradi *DataSet*
5. Otvori konekciju
6. Ažuriraj izvor podataka
7. Zatvori konekciju

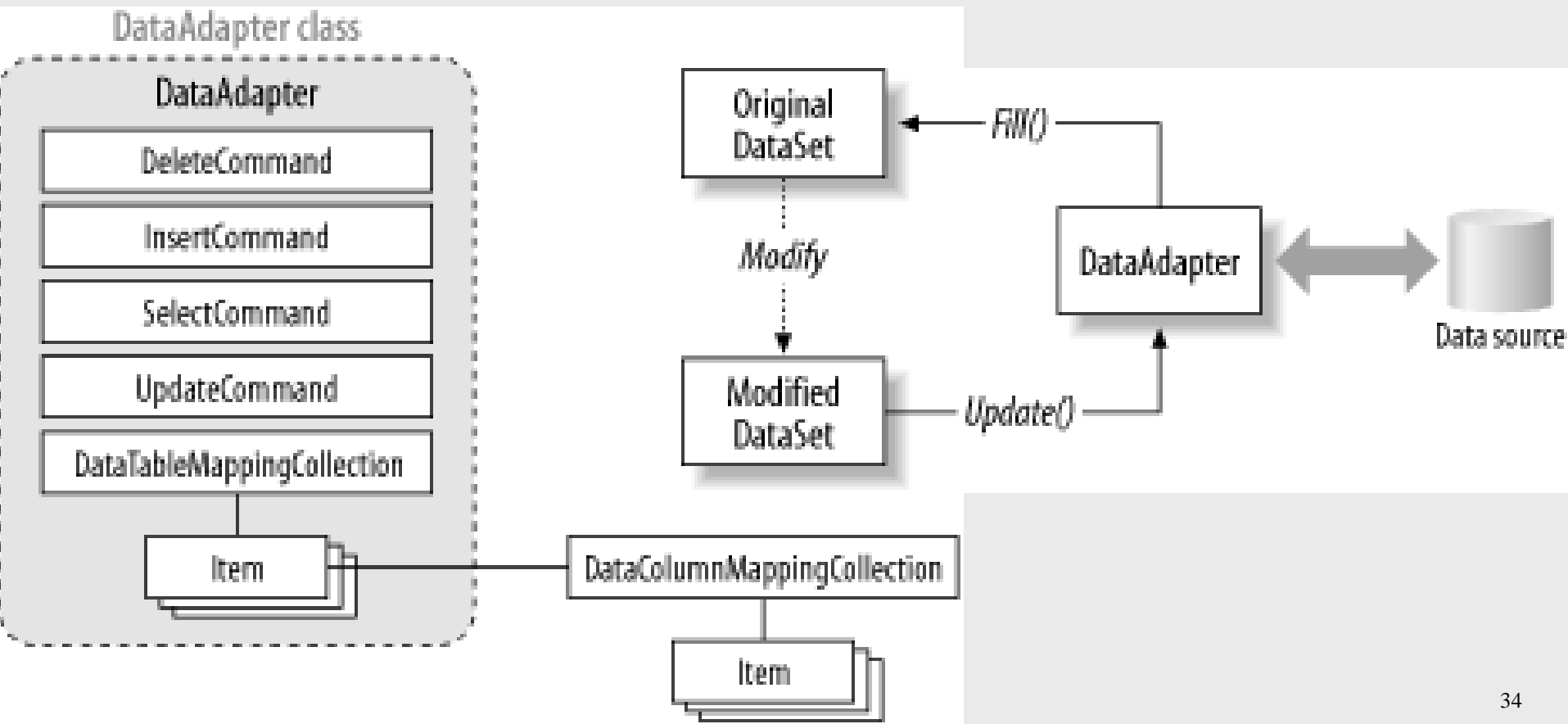
# ***IDataAdapter***

## ❑ Prilagodnik - most između skupa podataka i izvora podataka

- *OleDbDataAdapter* i *SqlDataAdapter* nasljeđuju `System.Data.Common.DataAdapter`

## ❑ Primjer: kreiranje DataAdapter objekta

- Toolbox– Drag&Drop komponente na formu



# ***DataAdapter članovi***

## **❑ Svojstva**

- `DeleteCommand`, `InsertCommand`, `SelectCommand`, `UpdateCommand` – naredbe za rukovanje podacima

## **❑ Postupak `Fill` – dodaje ili osvježava zapise u `DataSet`**

- `Int32 rowCount = DataAdapter.Fill(DataSet ds);`
- `Int32 rowCount = DataAdapter.Fill(DataTable dt);`
- `Int32 rowCount = DataAdapter.Fill(DataSet ds, String tableName);`
  - `rowCount` - broj uspješno stvorenih ili osvježanih zapisa

## **❑ Postupak `Update` – provjerava stanje zapisa (`RowState`) i poziva odgovarajuću SQL naredbu za svaki umetnuti, ažurirani ili obrisani redak te tako ažurira izvorne podatke**

- `Int32 rowCount = DataAdapter.Update(DataSet ds);`
- `Int32 rowCount = DataAdapter.Update(DataRow[] dra);`
- `Int32 rowCount = DataAdapter.Update(DataTable dt);`
  - `rowCount` – broj osvježanih zapisa
  - `dra` - polje `DataRow` objekata koji se usklađuju s izvorom

# DataAdapter primjeri

❑ Primjer:

📁 ADO\Artikli

OleDbAdapter1

❑ Properties

❑ CommandText

❑ QueryBuilder

The screenshot shows a Windows application window titled "Unos artikala". It contains a list of items with the following text:

- RAM, 512 MB, SDRAM, PC-133, brand name
- SODIMM PC-133, 256 MB, 144 pins, brand name (for notebooks)
- Knjiga "PC Škola - Visual Basic 6.0"
- Knjiga "PC Škola - Office XP"
- Knjiga "PC Škola - OpenOffice.org 1.0.1"
- Torba za notebook, MONAS M, cma
- SODIMM SSSR PC 2700, 256 MB, 200 pins, 333 MHz, brand name (for notebooks)
- Mrežna kartica wireless CANYON, CN-7106PC, bežična mreža 11 Mbps, PCMCIA
- USB 2.0 card, CONNEX, 4 port, PCMCIA
- USB 2.0 card, CONNEX, 2 port, PCMCIA
- Security lock TARGUS (PA410E)
- USB 2.0 + FireWire card, CONNEX, PCMCIA

Below the list are input fields for the following fields:

- Šifra: 1234
- Naziv: RAM, 512 MB, SDRAM, PC-133, brand name
- Cijena: 47,00
- Mjera: kom
- Usluga ☐

At the bottom of the window are several buttons and status indicators:

- Navigation buttons: <|, <, >, >|
- Status: 1 od 1548, Unchanged
- Action buttons: Add, Edit, Delete, Command, Fill (Refresh), Update, Save, Cancel

# DataAdapter primjeri

## ❑ Primjer: ADO\Artikl – Artikl.Designer.cs

- svojstva i kreiranje članova *DataAdapter* objekta

```
this.oleDbDataAdapter1 =  
    new System.Data.OleDb.OleDbDataAdapter();  
...  
this.oleDbDataAdapter1.SelectCommand =  
    this.oleDbSelectCommand1;  
  
this.oleDbSelectCommand1.CommandText =  
    "SELECT SifArtikla, NazArtikla, JedMjere, " +  
        CijArtikla, ZastUsluga, SlikaArtikla " +  
    "FROM Artikl";  
this.oleDbSelectCommand1.Connection =  
    this.oleDbConnection1;  
oleDbDataAdapter1.Fill(dataSetArtikli);
```

## ❑ Primjer: ADO\Artikl – Artikl.cs (Form\_Load, buttonFill\_Click)

- oleDbDataAdapter1.Fill(dataSetArtikli);

# *DataAdapter* događaji

## □ Događaji

- `FillError` – pogreška pri `Fill` operaciji
  - `argument` `FillEventArgs`, sa svojstvom
    - `Continue` – indikator da li treba nastaviti s punjenjem
- `RowUpdating` – operacija inicirana s `Update` treba započeti
  - `argument` `RowUpdatingEventArgs`, sa svojstvima
  - `Command` – objekt koji se izvršava pri provedbi `Update` postupka
  - `Errors` – iznimka koja je nastupila pri izvedbi
  - `Row` – redak koji se obrađuje
  - `StatementType` – vrsta naredbe koja se izvršava
    - `enum StatementType { Select, Insert, Update, Delete }`
  - `UpdateStatus` – akcija za preostale retke
    - `enum UpdateStatus { Continue, ErrorsOccurred, SkipAllRemainingRows, SkipCurrentRow }`
- `RowUpdated` – operacija inicirana s `Update` je završila
  - `argument` `RowUpdatedEventArgs`, sa svojstvima
  - kao `RowUpdating` i dodatnim svojstvom
  - `RecordsAffected` – broj obrađenih redaka

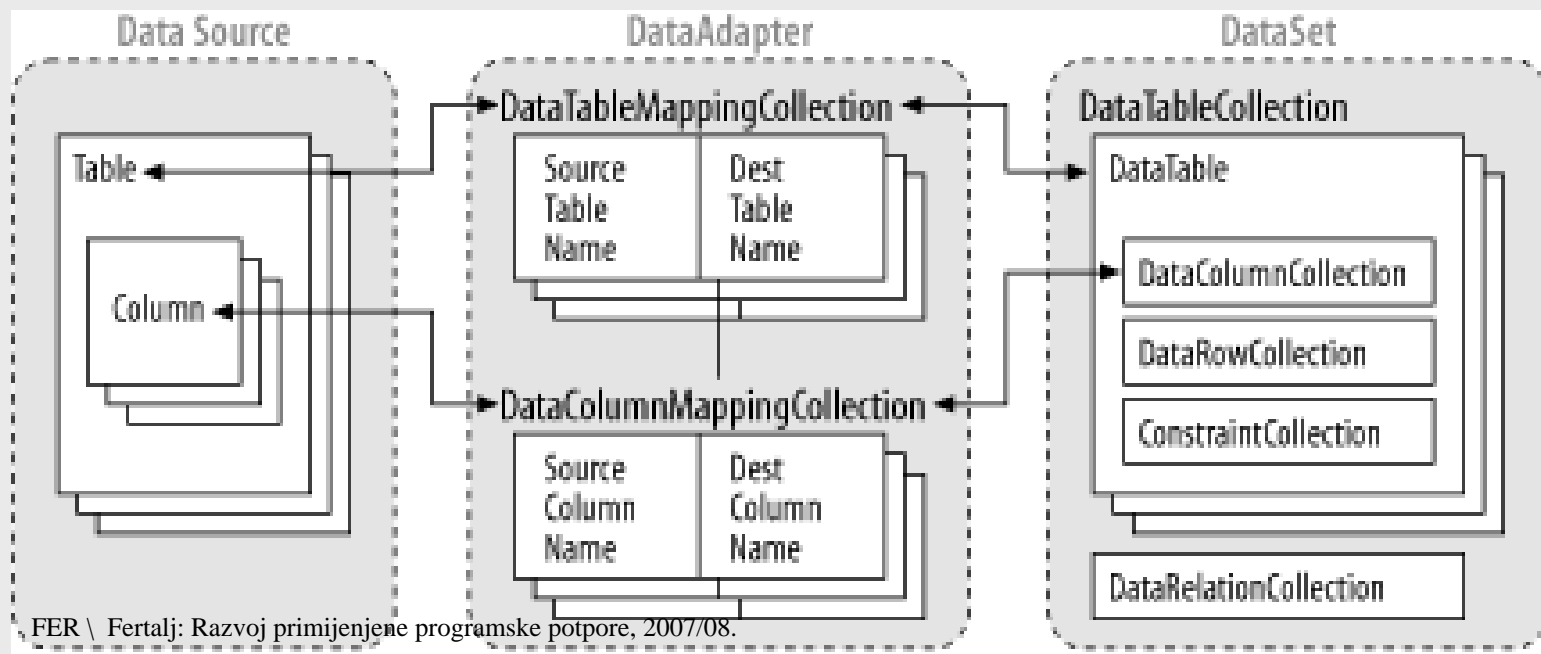
# Ostali *DataAdapter* članovi

## □ Svojstva

- **MissingSchemaAction** – akcija u slučaju dodavanja podataka u DataSet kad tablica ili kolona ne postoji
  - `enum MissingSchemaAction { Add, AddWithKey, Error, Ignore }`
- **TableMappings** – kolekcija mapiranja (preslikavanja naziva stupaca tablice) između izvorne tablice i DataTable

# Mapiranje objekata

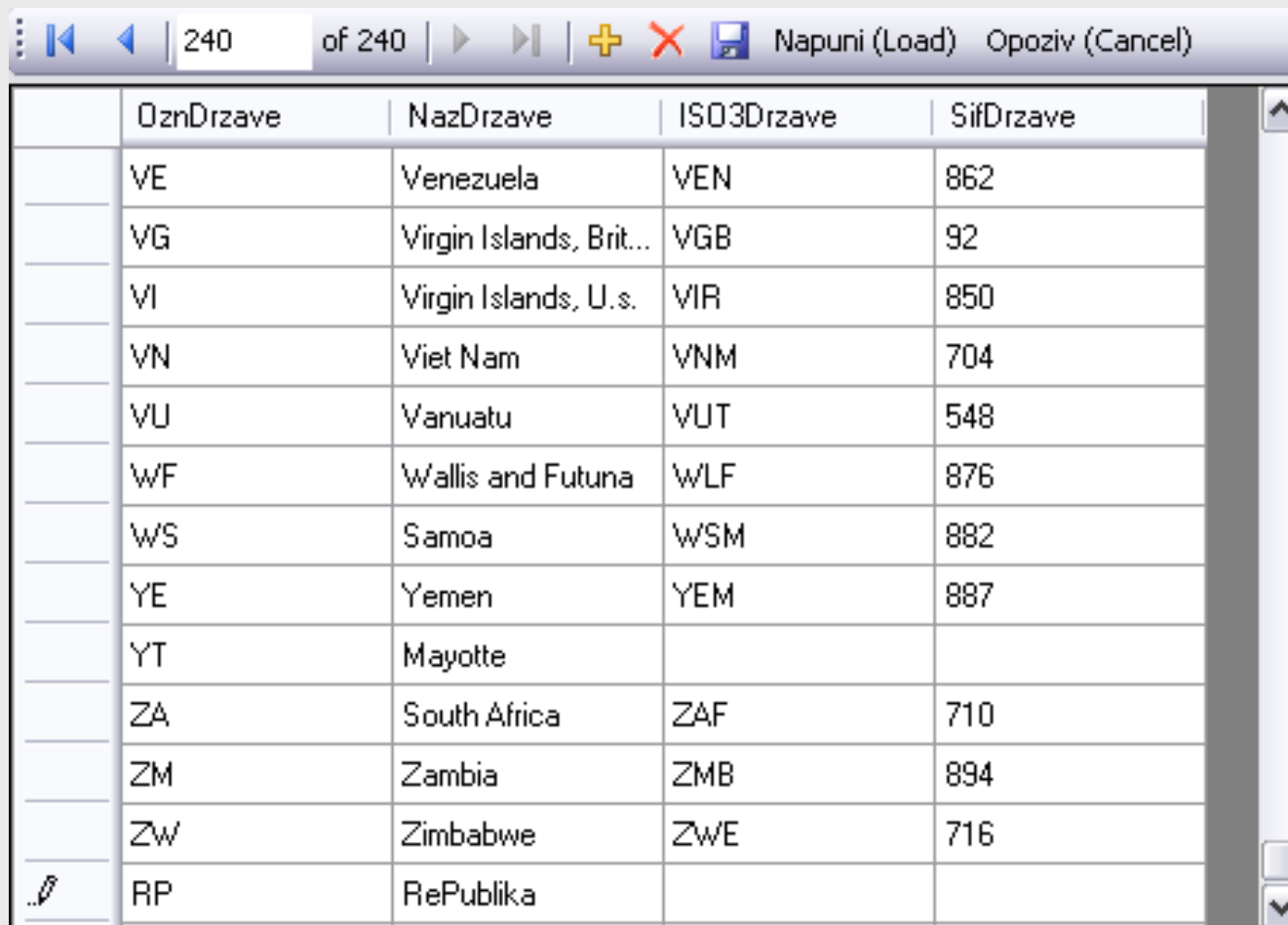
- ❑ Izvorne tablice i stupci mogu se različito zvati u skupu podataka
- ❑ **TableMappings** svojstvo **DataAdaptera**
  - `DataTableMappingCollection` kolekcija pridruživanja izvornih i `DataSet` tablica, instanca razreda
  - Pojedini `DataTableMapping` objekt ima svojstvo `DataColumnMapping` – kolekciju pridruživanja stupaca
- ❑ **Primjer:**  **ADO\Artikl ili ADO\Drzava ... \<idataset>.Designer.cs**
  - `DataAdapter / Properties / TableMapping (collection)`






# TableAdapter

- ❑ **TableAdapter** - Razred / komponenta, koja učahuruje adapter i ostale elemente za pristup podacima
- ❑ Primjer:  **ADO\Drzava**




	OznDrzave	NazDrzave	ISO3Drzave	SifDrzave
	VE	Venezuela	VEN	862
	VG	Virgin Islands, Brit...	VGB	92
	VI	Virgin Islands, U.s.	VIR	850
	VN	Viet Nam	VNM	704
	VU	Vanuatu	VUT	548
	WF	Wallis and Futuna	WLF	876
	WS	Samoa	WSM	882
	YE	Yemen	YEM	887
	YT	Mayotte		
	ZA	South Africa	ZAF	710
	ZM	Zambia	ZMB	894
	ZW	Zimbabwe	ZWE	716
	RP	RePublika		

# TableAdapter

- ❑ **Primjer:**  **ADO\Drzava – FirmaDataSet.Designer (Design)**
- ❑ **Primjer:**  **ADO\Drzava – FirmaDataSet.Designer.cs (Source)**

```
namespace Drzava.FirmaDataSetTableAdapters {  
    ...  
    public partial class DrzavaTableAdapter  
        : System.ComponentModel.Component {  
  
        private System.Data.OleDb.OleDbDataAdapter _adapter;  
        private System.Data.OleDb.OleDbConnection _connection;  
        private System.Data.OleDb.OleDbCommand[] _commandCollection;
```

- ❑ **Primjer:**  **ADO\Drzava – DrzavaForm\_Load (i drugi postupci)**
  - `drzavaTableAdapter.Connection.Open();`
  - `drzavaTableAdapter.Fill(dataSet.Drzava);`

# Zadaci za vježbu

- ☐ **Proučiti opcije razvojne okoline za pristup podacima**
  - *Data \ Data Sources – DataSet – tablica – padajući izbornik – drag&drop*
  - *tablicaTableAdapter*
  - *DataSet \ Table \ TableAdapter*
  
- ☐ **Napraviti zaslonsku masku za odabranu tablicu baze podataka.**
  
- ☐ **Ugraditi korisnički definirane poruke obrađenih iznimki.**

# Skupovi podataka

---

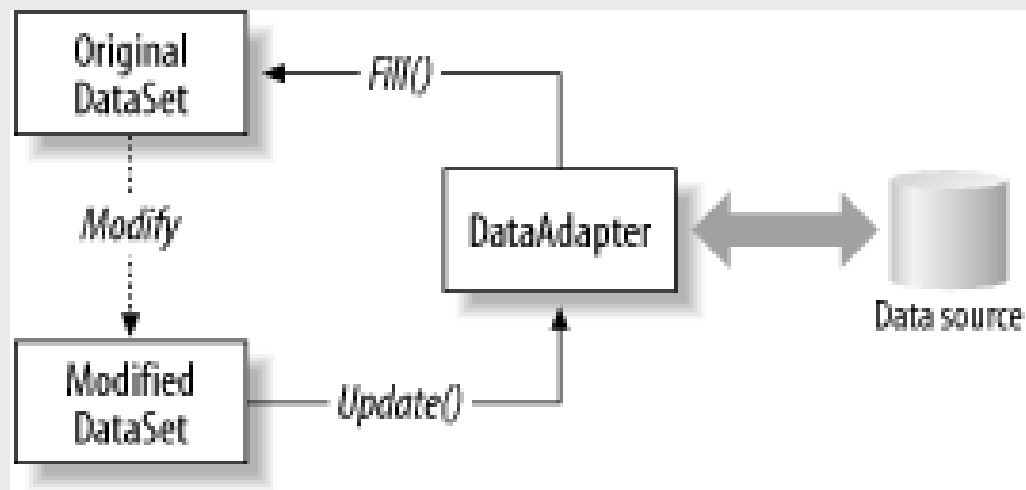
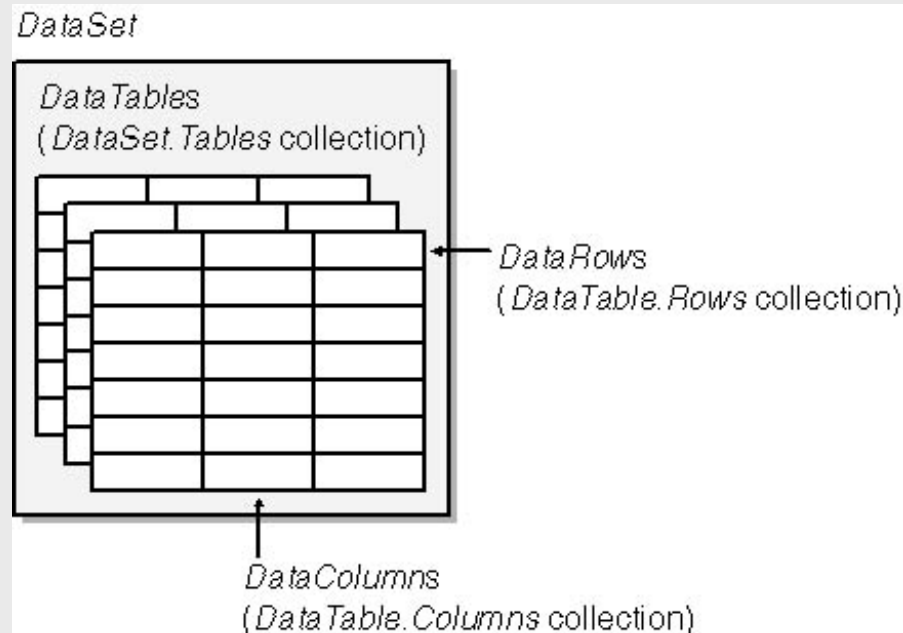
# DataSet

## ❑ System.Data.DataSet

- Skup podataka u memoriji računala
- Univerzalni spremnik podataka, koji se može, ali ne mora nalaziti u bazi podataka
- može sadržavati podatke iz jedne ili više tablica

## ❑ Korištenje XML tehnologije

- za zapisivanje i čitanje podataka
- za pohranu sheme podataka (XMLSchema)



# Vrste *DataSet* skupova

## ❑ Primjeri:

- Dva načina pristupa do vrijednosti atributa `SifArtikla` prvog zapisa tablice `Artikl` u skupu `dsArtikl`

## ❑ Typed - instanca `System.Data.DataSet`

- sadrži shemu podataka, koju prevoditelj koristi za provjeru sintakse i kompatibilnosti tipova podataka u izrazima (`strong typing`)
- `dsArtikl.Artikl[0].SifArtikla`
- pogreška pri prevođenju ukoliko ne postoji `Artikl` ili `SifArtikla`

## ❑ UnTyped - razred naslijeđen iz `System.Data.DataSet`

- struktura podataka ne mora biti unaprijed poznata - program tijekom izvođenja može primiti podatke od vanjske komponente ili servisa (npr. Web servis)
- `dsArtikl.Tables["Artikl"].Rows[0].Item["SifArtikla"]`
- `dsArtikl.Tables["Artikl"].Rows[0]["SifArtikla"]`
- pogreška tek pri izvođenju kada ne postoji `Artikl` ili `SifArtikla`

# *DataSet* članovi

## ☐ Izbornik Data

- Add ...
- Show ...

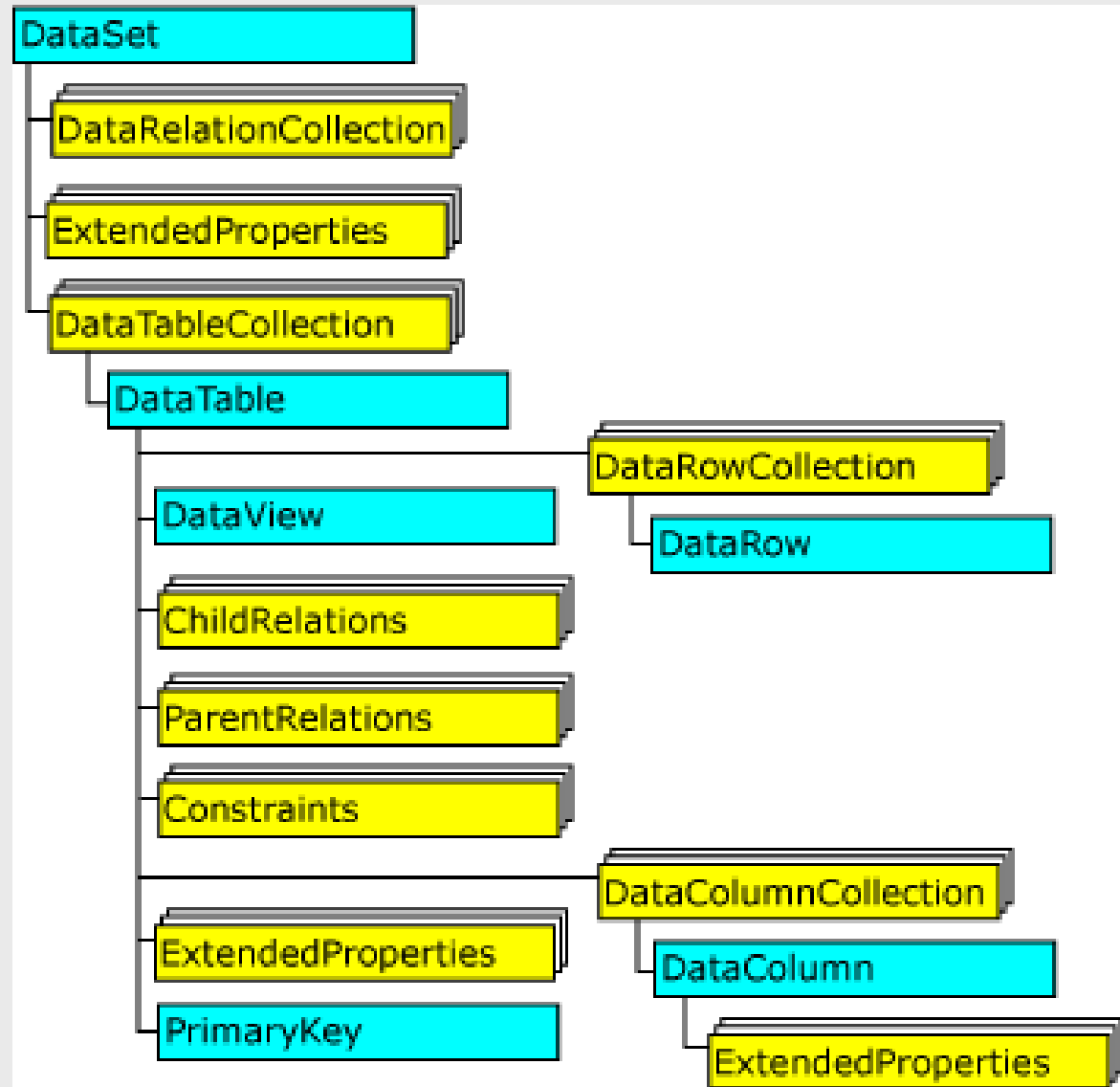
## ☐ Kreiranje skupa dovlačenjem komponente iz Toolbox- a na formu

- DataSet
- DataGridView

## ☐ Generiranje Typed skupa

- DataAdapter: Generate  
DataSet

## ☐ Dovlačenje skupa iz *Data Sources* na formu



# *DataSet* članovi

## ❑ **DataSet konstruktori**

- `DataSet()` ; # vraća `Untyped DataSet` s imenom *NewDataSet*
- `DataSet(DataSetName)` ; # vraća `DataSet` zadanog imena

## ❑ **Svojstva**

- `CaseSensitive` – razlikovanje velikih i malih slova
- `DataSetName` – naziv skupa podataka
- `Tables` – kolekcija `DataTable` objekata # `Dataset / Properties / Tables`
- `Relations` - kolekcija `DataRelations` objekata
- `EnforceConstraints` – nametanje ograničenja na podatke
- `HasChanges` – bilo novih, mijenjanih, brisanih zapisa
- `HasErrors` – oznaka pogreške u nekoj od tablica
  - `DataTable.GetErrors` – daje kolekciju pogrešaka
  - `DataRow.RowError` – pogreška retka

## ❑ **Postupci koji se odnose na skupove podataka**

- `Clear` – čisti podatke u svim tablicama
- `Clone` – kopira strukturu uključujući veze i ograničenja, ali ne i podatke
- `Copy` – kopira strukturu i podatke `DataSet` objekta koji izvodi postupak



# Uređivanje i ažuriranje podataka

## ❑ Uobičajeni redoslijed

- podaci se dohvaćaju s izvora i pohranjuju u memoriji, npr. izvođenjem `DataAdapter.Fill`
- dolazi do promjene podataka, npr. dodavanje, izmjena, brisanje
  - programski ili putem povezanih (*data-bound*) kontrola
- promjene koje treba trajno sačuvati prosljeđuju se na izvor podataka
  - koristi se `DataAdapter.Update` ili `Command` objekt
- po potrebi se ažurira `DataSet` da bi odražavao novo stanje podataka
  - postupcima `DataSet.AcceptChanges` ili `DataTable.AcceptChanges`
  - `DataAdapter.Fill` i `DataAdapter.Update` automatski pozivaju `AcceptChanges`
  - kada se koristi `Command` objekt, `AcceptChanges` treba pozvati eksplicitno

## ❑ Primjeri: **Drzava**, **Artikl # Fill, Update, AcceptChanges**

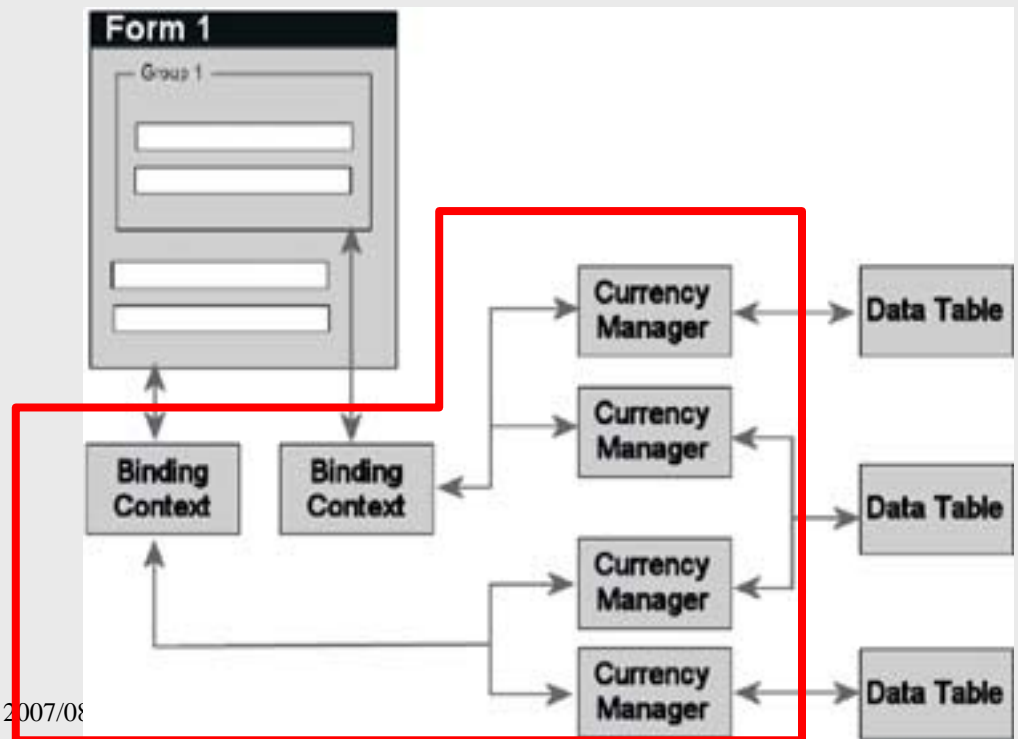
- `this.firmaDataSet = new Drzava.FirmaDataSet();`
- `this.dataSetArtikli = new UnosArtikala.DataSetArtikli();`

# Povezivanje podataka

---

# Povezivanje podataka

- ❑ **Data Binding - Mehanizam vezanja (povezivanja, privezivanja) elemenata grafičkog sučelja na podatke**
  - povezuje podatke (izvor podataka) sa svojstvima kontrola, najčešće s njihovim vrijednostima, a općenito s bilo kojim svojstvom
    - npr. `ListBox.DataMember` ili `TextBox.Text`
    - npr. `ForegroundColor`, `Font`
- ❑ **Povezivanje se ostvaruje putem suradnje nekoliko vrsta objekata**



# Razred *BindingSource*

## ❑ **BindingSource**

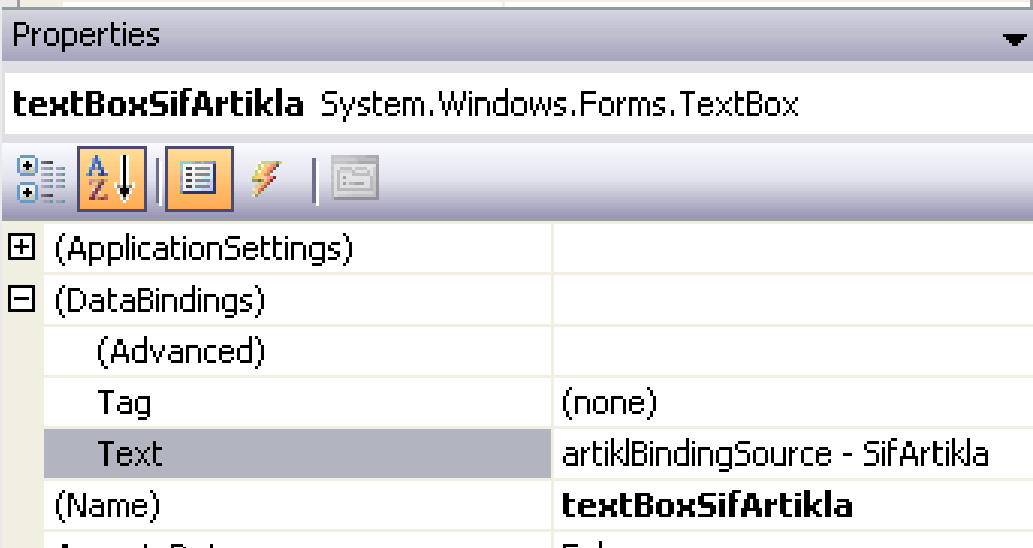
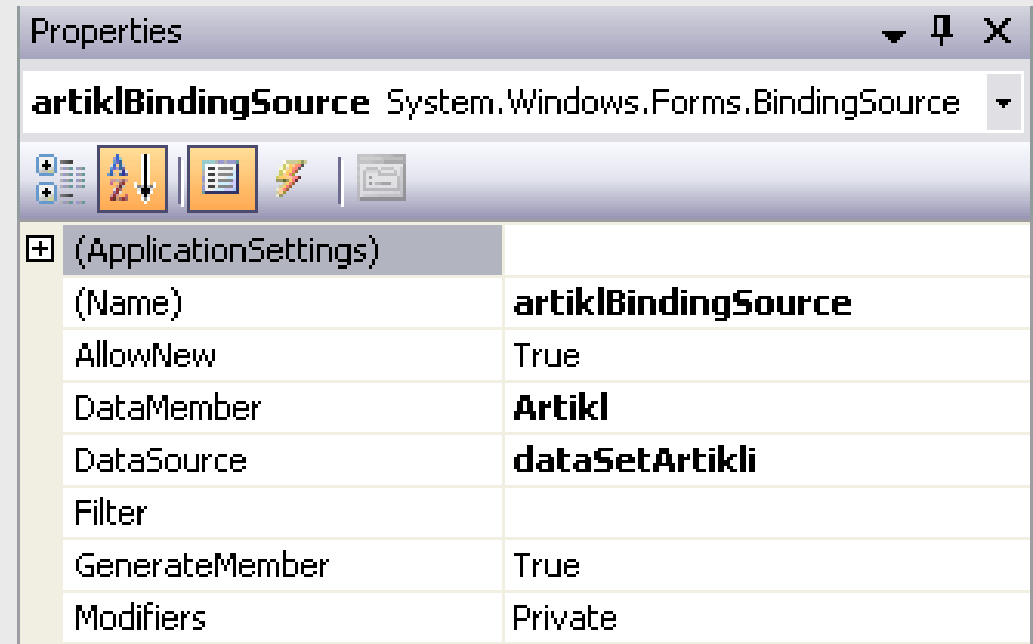
- Razred koji ućahuruje izvor podataka i prati trenutnu poziciju zapisa
- Ima funkcionalnosti *BindingContext* i *CurrencyManager*

## ❑ **Glavni članovi**

- *DataSource* – skup podataka
- *DataMember* – tablica skupa

## ❑ **Primjer**

- Svojstva komponente *BindingSource* i postavljanje komponente kao izvora podataka *TextBox*




# Razredi i članovi koji sudjeluju u povezivanju

## ❑ **BindingContext** – upravlja kolekcijom **BindingManagerBase** objekata

- **svojstvo** `Control.BindingContext`

## ❑ **BindingManagerBase** – upravlja povezanim izvornim podacima

- po jedan objekt za svaki podatkovni objekt povezan s kontrolom
- `BindingContext[ source ]`
- `BindingContext[ source, member ]`
- primjer:  `ADO\Artikl`  

```
System.Windows.Forms.BindingManagerBase bmb;  
bmb = BindingContext[dataSetArtikli.Artikl];  
bmb = BindingContext[dataSetArtikli, "Artikl"];
```
- apstraktni razred, ne može se instancirati – objekti su instance razreda:
  - `PropertyManager` – kad izvor vraća pojedinačni objekt
  - `CurrencyManager` – za povezivanje s kolekcijom objekata na izvoru

## ❑ **ControlBindingsCollection** – kolekcija poveznica kontrola-podaci

- **svojstvo** `Control.DataBindings`

## ❑ **Binding** – pojedinačna poveznica svojstvo-podatak

- `DataBindings[ property ]`

# Binding

## ❑ Konstruktor

- `public Binding(string propertyName, Object dataSource, string dataMember);`

## ❑ Svojstva

- `BindingManagerBase` – osnovica za povezivanje
- `BindingMemberInfo` – definira pojedinačno povezivanje
  - `BindingMember` – puno ime atributa, npr. "Categories.CategoryID"
  - `BindingPath` – staza do atributa, npr. "Categories"
  - `BindingField` – atribut koji se povezuje, npr. "CategoryID"
- `Control` – povezana kontrola, npr. "textBoxNazArtikla" (TextBox)
- `DataSource` – izvor podataka, npr. "dataSetArtikl"
- `IsBinding` – bool oznaka da je poveznica aktivna
- `PropertyName` – naziv svojstva kontrole koje se povezuje, npr. "Text"

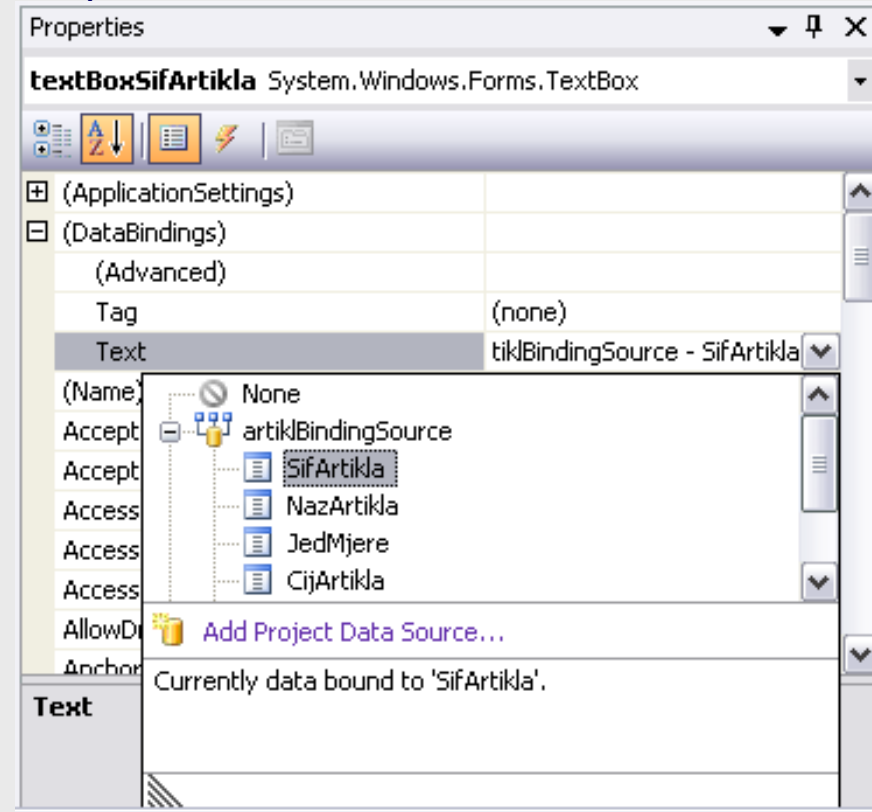
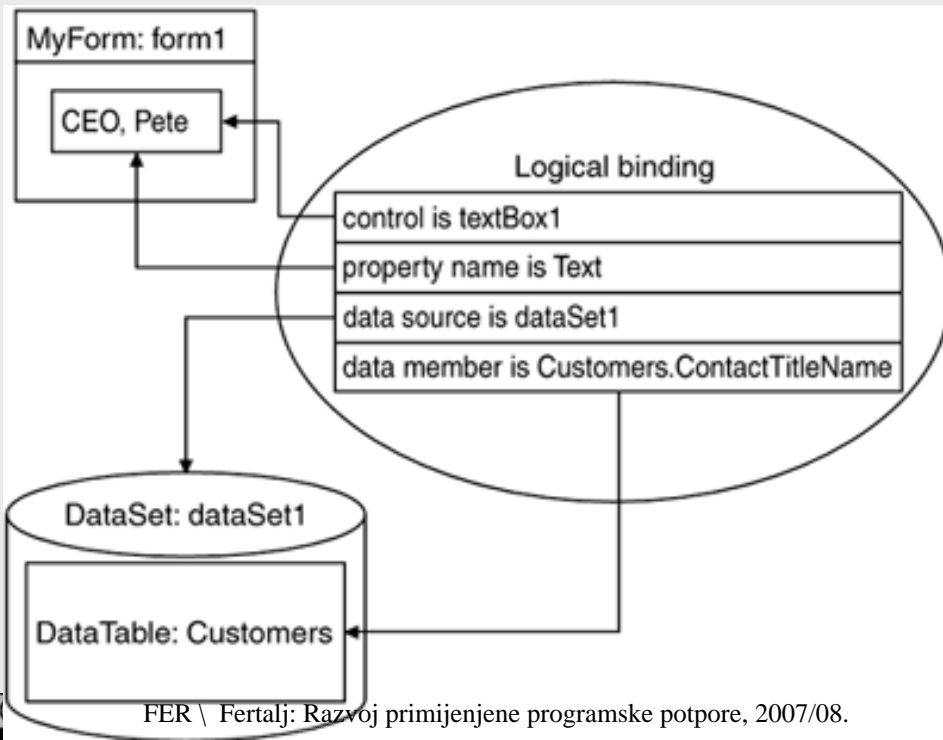
# Jednostavno povezivanje (Simple Binding)

## ❑ Povezivanje kontrole koja prikazuje jednu vrijednost

- svojstvo kontrole, npr. "Text"
- izvor podataka, npr. "dataSet1"
- puna staza do vrijednosti, npr. "Customers.ContactTitleName"

## ❑ Primjer: ADO\Artikli

- Properties – (Bindings): Text ili (Advanced)



# Jednostavno povezivanje dinamički

## ❑ Primjer: ADO\Artikl

- povezati svojstva u dizajnu a neka dinamički

```
textBoxSifArtikla.DataBindings.Add("Text",  
    artiklBindingSource, "SifArtikla");  
textBoxNazArtikla.DataBindings.Add("Text",  
    artiklBindingSource, "NazArtikla");  
textBoxJedMjere.DataBindings.Add("Text",  
    artiklBindingSource, "JedMjere");
```

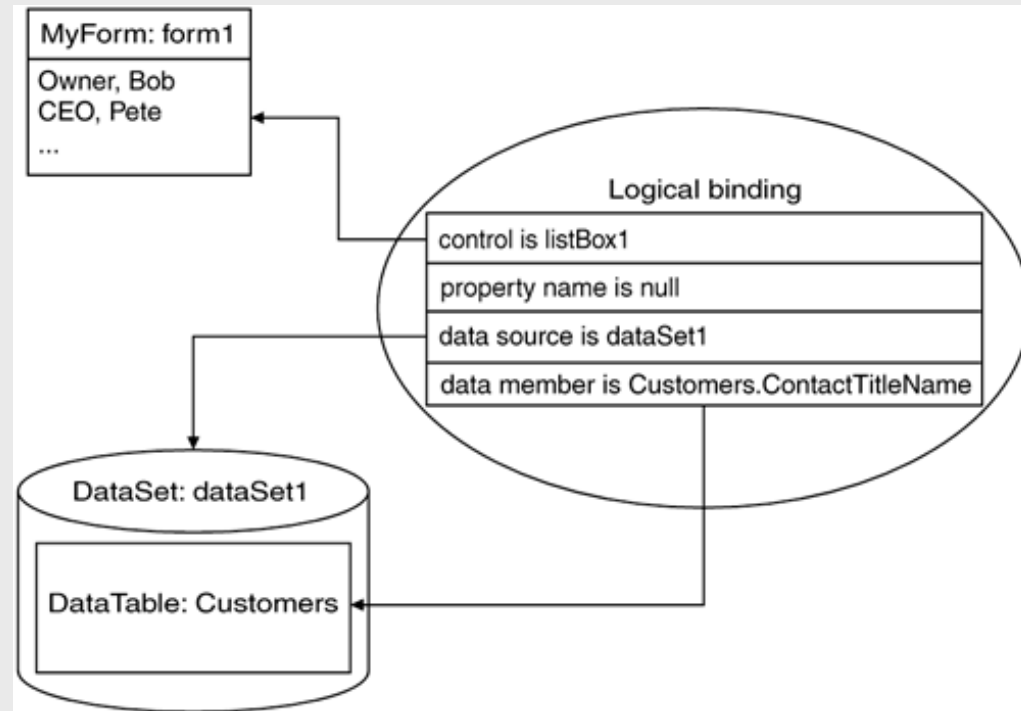
- `control.DataBindings` – poveznice kontrole
- Binding – pojedinačna poveznica svojstvo – izvor - član/podatak
  - `DataBindings[ property ]`
  - `public Binding(string propertyName, Object dataSource, string dataMember);`



# Složeno povezivanje (Complex Binding)

## ❑ Povezivanje složene kontrole s više redaka, postavljanjem

- `DataSource` – izvora podataka, npr. `firmaDataSet`
- `DisplayMember` – atributa, npr. `"Artikl.NazivArtikla"`
  - `ListBox` i `ComboBox` – jedna vrijednost za redak
- `DataMember` – tablice, npr. `"Drzava"`
  - `DataGridView` – više vrijednosti za jedan redak



## ❑ Primjer: ADO\Artikli

```
listBoxArtikli.DataSource = dataSetArtikli.Artikl;  
listBoxArtikli.DisplayMember = "NazArtikla";
```

# ***BindingSource*** članovi

## ❑ **Svojstva**

- `AllowEdit`, `AllowNew`, `AllowRemove` – indikatori da je postupak moguć
- `DataSource` – skup podataka
- `DataMember` – tablica skupa koja se povezuje
- `Count` – broj elemenata u listi podataka
- `object Current` - objekt aktualni element izvora
- `int Position` - indeks aktualnog elementa

## ❑ **Događaji**

- `CurrentChanged` – promjena `Current`
- `ItemChanged` – ažuriran aktualni element `List`
- `PositionChanged` – promjena `Position`

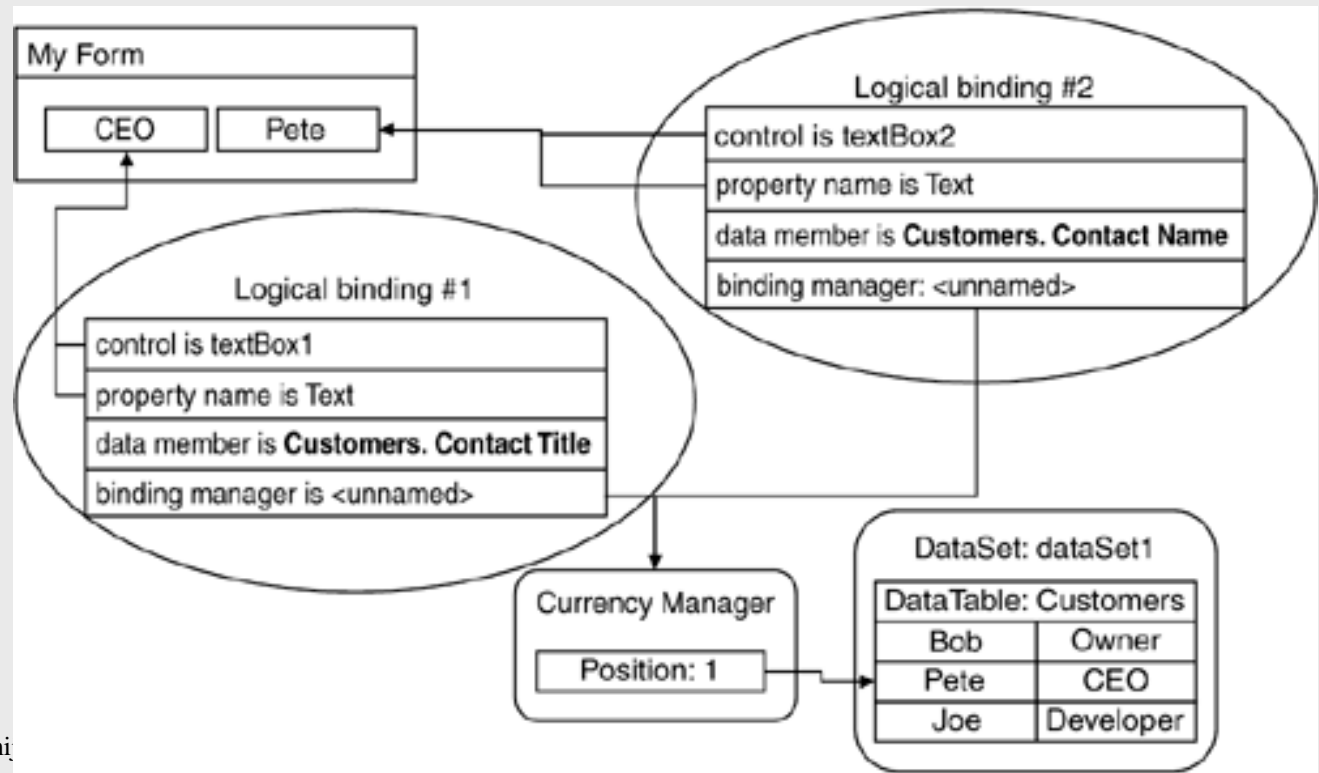
## ❑ **Postupci**

- `AddNew` – dodavanje novog elementa na izvor
- `CancelEdit` – opoziv uređivanja koje je u tijeku
- `EndEdit` - dovršetak uređivanja koje je u tijeku, pohrana na izvoru
- `Remove`, `RemoveAt(int index)` – brisanje elementa s izvora
- `MoveFirst`, `MoveLast`, `MoveNext`, `MovePrevious` – navigacija

# Evidencija i sinkronizacija aktualnih podataka

## ❑ **CurrencyManager** – povezivanje s kolekcijom objekata na izvoru

- evidentira položaj aktualnog objekta u listi s izvora i upravlja poveznicama s tim izvorom, pri čemu izvor ne zna koji se element trenutno prikazuje
- za svaki izvor podataka postoji zasebna instanca razreda `CurrencyManager`
- za više kontrola iste forme koje se povezuju na isti izvor, kreira se samo jedna instanca razreda `CurrencyManager`



# Navigacija podacima

❑ Primjer:

📁 ADO\Artikli

Unos artikala

RAM, 512 MB, SDRAM, PC-133, brand name  
SODIMM PC-133, 256 MB, 144 pins, brand name (for notebooks)  
**Knjiga "PC Škola - Visual Basic 6.0"**  
Knjiga "PC Škola - Office XP"  
Knjiga "PC Škola - OpenOffice.org 1.0.1"  
Torba za notebook, MONAS M, crna  
SODIMM SSSR PC 2700, 256 MB, 200 pins, 333 MHz, brand name (for notebooks)  
Mrežna kartica wireless CANYON, CN-7106PC, bežična mreža 11 Mbps, PCMCIA 28.3.2008 16:06:25  
USB 2.0 card, CONNEX, 4 port, PCMCIA 28.3.2008 16:04:50  
USB 2.0 card, CONNEX, 2 port, PCMCIA  
Security lock TARGUS (PA410E)  
USB 2.0 + FireWire card, CONNEX, PCMCIA

Šifra: 2937

Naziv: Knjiga "PC Škola - Visual Basic 6.0"

Cijena: 100,00

Mjera: kom

Usluga ☐

3 od 1548 Unchanged

Navigation buttons: < > >|

Buttons: Add Edit Delete Command Fill (Refresh) Update Save Cancel

# Navigacija podacima vlastitim metodama

```
artiklBindingSource.PositionChanged +=  
    new EventHandler(artiklPositionChanged) ;
```

```
private void buttonNext_Click(object sender, EventArgs e)  
{  
    artiklBindingSource.MoveNext();  
}
```

```
private void artiklPositionChanged(  
    Object sender, EventArgs e)  
{  
    // ažuriranje stanja o kontekstu  
    UpdateDisplay();  
  
    if (artiklBindingSource.Count == 0) return;  
    // nešto drugo
```

```
void UpdateDisplay() {  
    labelPosition.Text =  
        ((artiklBindingSource.Position + 1).ToString()  
            + " od " +  
            artiklBindingSource.Count.ToString());  
}
```

# Navigacija komponentom *BindingNavigator*

## □ **BindingNavigator** – komponenta grafičkog sučelja za navigaciju i rukovanje povezanim podacima

### ■ Svojstva:

- **BindingSource** – povezljivi izvor s podacima
- **MoveFirstItem, MoveLastItem, MoveNextItem, MovePreviousItem** – **ToolStripItem** pojedine funkcionalnosti navigacije kroz podatke
- **PositionItem** – **ToolStripItem** za prikaz pozicije aktivnog zapisa
- **AddNewItem, DeleteItem, SaveItem** – **ToolStripItem** funkcionalnosti dodavanja, brisanja i spremanja podataka

### ■ Događaji:(zanimljiviji su oni pojedinih elemenata)

```
private void drzavaBindingNavigatorSaveItem_Click(  
    object sender, EventArgs e)  
{  
    this.Validate();  
    this.drzavaBindingSource.EndEdit();  
    this.drzavaTableAdapter.Update(this.firmaDataSet.Drzava);  
}
```

**Primjer:**  **ADO\Drzava**

# Postupci i događaji pri promjeni podataka

## ❑ DataSet postupci

- `AcceptChanges` – potvrđuje promjene napravljene nad podacima od posljednjeg punjenja ili posljednjeg poziva `AcceptChanges`
- `HasChanges` – oznaka novih / obrisanih / izmijenjenih redaka
- `RejectChanges` – odbacuje promjene načinjene otkad je `DataSet` kreiran, odnosno od posljednjeg poziva `AcceptChanges`

## ❑ DataAdapter događaji

- `RowUpdating`, `RowUpdated` – ažuriranje izvora u tijeku / ažuriranje obavljeno

## ❑ DataTable postupci

- `AcceptChanges`, `RejectChanges`
- `GetChanges` – vraća kopiju `DataTable` koja sadrži sve promjene nastale nakon punjenja, odnosno nakon posljednjeg poziva `AcceptChanges`
- `NewRow` – dodavanje novog retka
- `Remove(DataRow)`, `RemoveAt(Index)` – uklanjanje retka iz kolekcije

## ❑ DataTable događaji

- `ColumnChanging`, `ColumnChanged` – prije/poslije promjene elementa
- `RowChanging`, `RowChanged` – prije/poslije promjene sadržaja retka
- `RowDeleting`, `RowDeleted` – prije/poslije brisanja retka

# ***DataRow***

## ❑ **System.Data.DataRow** – predstavlja redak u DataTable

- **Item** – vrijednost stupca u izvornom obliku
  - `public virtual object this[int] {get;}`
- **ItemArray** – polje s vrijednostima svih vrijednosti retka
- **RowError** – korisnički definirano objašnjenje pogreške
- **RowState** – stanje DataRow objekta
  - `enum DataRowState { Added, Deleted, Detached, Modified, Unchanged }`
- **GetType()** – vraća tip zadanog stupca
- **IsNull()** – vraća `bool` indikator null vrijednosti stupca
- **Table** – DataTable kojoj redak pripada

```
dataReader rd = commandObject.ExecuteReader();  
DataRow dr = dataSet.Artikl.NewRow();  
dr[0] = rd.GetInt32(0);  
dr["NazArtikla"] = rd.GetString(1);  
dataSet.Artikl.Rows.Add(dr);
```



# ***DataRowView***

## ☐ **DataRowView**

- redak pogleda – zapravo referenca na `DataRow`

## ☐ **Svojstva**

- `DataRowView` – nadređeni pogled
- `IsEdit` – oznaka da je u tijeku uređivanje izmjena
- `IsNew` – oznaka da se radi o novom retku
- `Item` – vrijednost elementa u aktualnom retku
- `Row` – **aktualni** `DataRow`, izvorni zapis
- `RowVersion` – verzija aktualnog `DataRowView`

## ☐ **Koristi se za adresiranje retka u primjeru `GetCurrentRow`**

```
return ( (DataRowView) artiklBindingSource.Current ).Row;
```

# Rukovanje zapisima u skupu podataka

□ Primjer:  
📁 ADO\Artikli

The screenshot shows a window titled "Unos artikala" with a list of mobile phones and input fields for adding a new record. The list contains the following items:

- Mobitel Sony Ericsson P11
- Mobitel Sony Ericsson S500i
- Mobitel Sony Ericsson W580I, bijeli
- Mobitel Sony Ericsson W580I, sivi
- Mobitel Sony Ericsson W610i, crni
- Mobitel Sony Ericsson W660I, crveni
- Mobitel Sony Ericsson W810i, crni
- Mobitel Sony Ericsson W910I, crni
- Mobitel Sony Ericsson Z320I, crveni 29.3.2008 20:35:09
- Mobitel Sony Ericsson Z320I, plavi
- Punjač za sony ericsson CST-75

The input fields are:

- Šifra:
- Naziv:
- Cijena:  (with a red warning icon)
- Mjera:
- Usluga ☐

The bottom of the window features a set of buttons: "<K", "<", "1549 od 1549", "Added", ">", ">I", "Fill (Refresh)", "Update", "Add", "Edit", "Delete", "Command", "Save", and "Cancel". The "Add" button is highlighted with a blue oval.

# Dodavanje retka na kraj skupa podataka

## ❏ Primjer: ADO\Artikli


```
private void buttonAdd_Click(object sender, EventArgs e)
{
    artiklBindingSource.EndEdit();
    DataRow newRow= dataSetArtikli.Artikl.NewRow();

    // defaults
    newRow["SifArtikla"] = DBNull.Value;
    newRow["NazArtikla"] = DBNull.Value;
    newRow["JedMjere"] = DBNull.Value;
    newRow["CijArtikla"] = 0;
    newRow["ZastUsluga"] = false;
    newRow["SlikaArtikla"] = DBNull.Value;

    // add the new row to the DataTable
    dataSetArtikli.Artikl.Rows.Add(newRow);

    //UpdateDisplay();
    artiklBindingSource.Position = artiklBindingSource.Count-1;
}
```

# Izmjena retka u skupu podataka

- ❑ **Primjer:**  **ADO\Artikli**
- ❑ **Promjena se, naravno, može obaviti uređivanjem sadržaja u kontrolama grafičkog sučelja povezanim na podatke.**
- ❑ **Programski, promjena se obavlja promjenom vrijednosti atributa**

```
private void buttonEdit_Click(object sender, EventArgs e)
{
    System.Data.DataRow currRow;
    currRow = GetCurrentRow();
    currRow["NazArtikla"] = "@#$!";
    //currRow[1] = "@#$!"; // manje jasna alternativa
    currRow["SifArtikla"] = "-1"; // izazove validaciju
    UpdateDisplay();
}
```

- ❑ **Podaci time nisu izmijenjeni na izvoru!**

# Stanja i verzije retka (zapisa)

## ❑ Svojstvo `DataRow.RowState` poprima neku od vrijednosti, u trenutku promjene u odnosu na zadnje obavljene `AcceptChanges`

- `enum DataRowState { Added, Deleted, Detached, Modified, Unchanged }`
  - `Detached` – `DataRow` još nije dodan u tablicu

## ❑ `DataRow` postupci

- `AcceptChanges` – potvrđuje promjene napravljene nad podacima
- `BeginEdit`, `CancelEdit`, `EndEdit` – započinjanje / opoziv / završetak promjena
- `Delete` – briše redak
- `bool HasVersion(DataRowVersion version)` – provjerava postojanje verzije retka, koje postoje ovisno o akcijama od zadnjeg `AcceptChanges`
  - `Current` – aktualne vrijednosti pojedinih stupaca
  - `Default` – standardne vrijednosti koje se koriste za nove zapise
  - `Original` – vrijednosti postavljene u trenutku stvaranja zapisa
  - `Proposed` – vrijednosti pridružene nakon `BeginEdit` postupka
- `RejectChanges` – poništava još nepotvrđene promjene nad podacima

# Spremanje u DataSet i prikaz verzija retka

- ❑ Stanje se mijenja prelaskom na neki drugi zapis ili postupkom:

```
private void buttonSave_Click(object sender, EventArgs e)
{
    DataRowView drv = (DataRowView) artiklBindingSource.Current;
    drv.EndEdit();
}
```

- ❑ Stanje se odražava u svojstvu RowState

```
void UpdateDisplay()
{
    labelRowState.Text = GetCurrentRow().RowState.ToString();
}
```

- ❑ Stanje se prikazuje za trenutni redak

```
private System.Data.DataRow GetCurrentRow()
{
    if (artiklBindingSource.Count == 0) return null;

    return ((DataRowView)artiklBindingSource.Current).Row;
}
```

# Opoziv izmjena

- ❑ **Poništavaju se promjene koje nisu ažurirane u bazi podataka i osvježava zaslon**

```
private void buttonCancel_Click(object sender, EventArgs e)
{
    artiklBindingSource.EndEdit();
    dataSetArtikli.RejectChanges();
    UpdateDisplay();
}
```

# Brisanje retka

- ❑ **DataTable** postupci **Remove (DataRow) , RemoveAt (Index)**
  - redak se uklanja iz kolekcije `DataRow` objekata, ali ne i s izvora podataka
  
- ❑ **Za "pravo" brisanje koristi se postupak `DataRow.Delete`**
  - redak dobiva oznaku `RowState = Deleted` a fizički bude obrisani s `DataSet.AcceptChanges` ili `DataAdapter.Update`
  
- ❑ **Primjer: Da li je zapis stvarno obrisani na izvoru?**

```
private void buttonDelete_Click(object sender, EventArgs e)
{
    try
    {
        // trenutni
        DataRow row = GetCurrentRow();
        // obriši
        row.Delete();
    }
    ...
}
```



# Ažuriranje izvora podataka

## ❑ Promjene načinjene u memoriji prosljeđuju se izvoru

- izvođenjem `Command` objekata nad konekcijom ili
- pozivanjem postupka `DataAdapter.Update`, npr.

```
oleDbDataAdapter1.Update(dataSetArtikli.Artikl);
```

- ovaj postupak zapravo izvršava odgovarajući `Command` objekt adaptera, npr. `DeleteCommand`

## ❑ Obavljaju se sljedeće akcije

- `DataAdapter` provjerava `RowState` redaka i izvodi odgovarajuću naredbu
- podiže se događaj `DataAdapter.RowUpdating`
- izvršava se naredba koja mijenja stanje podataka
- ovisno o `UpdatedRowSource` svojstvu `Command` objekta, `DataAdapter` ažurira `DataSet` (standardna vrijednost `None`)
- podiže se događaj `DataAdapter.RowUpdated`
- poziva se postupak `AcceptChanges` nad `DataSet` ili `DataTable` objektom

# Primjer pohrane promjena na izvoru

```
private void buttonUpdate_Click(
    object sender, EventArgs e)
{
    // prekida prethodno započeto uređivanje podataka
    artiklBindingSource.EndEdit();

    try
    {
        //DataRow zapis = GetCurrentRow();
        //if (zapis.RowState == DataRowState.Added
        //    || zapis.RowState == DataRowState.Modified)

        if (dataSetArtikli.HasChanges())
        {
            // ima i kompliciranija varijanta s .GetChanges());
            oleDbDataAdapter1.Update(dataSetArtikli.Artikl);
            // prikaz promijenjenog stanja retka
            UpdateDisplay();
        }
    }
}
```

# Ažuriranje izvora podataka korištenjem Command

## ❑ Primjer: ADO\Artikl

- Budući da se naredba izvodi izravno, DataSet ne odražava promjene
- Treba napuniti DataSet da se osvježe podaci i status

```
private void buttonCommand_Click(object sender, EventArgs e)
{
    System.Data.OleDb.OleDbCommand cmdUpdate;
    System.Data.DataRow currRow = GetCurrentRow();

    cmdUpdate = new OleDbCommand("UPDATE Artikl "
        + " SET NazArtikla = '" + currRow["NazArtikla"]
        + " " + DateTime.Now.ToString()
        + "' WHERE SifArtikla=" + currRow["SifArtikla"]
        , oleDbConnection1);

    this.oleDbConnection1.Open();
    cmdUpdate.ExecuteNonQuery();
    this.oleDbConnection1.Close();

    this.dataSetArtikli.AcceptChanges();
}
```

# Potvrda/opoziv promjena nad skupom podataka

- ❑ **DataAdapter.Update** automatski poziva **AcceptChanges**
- ❑ Nakon izravnog izvođenja naredbe (**Command**), postupak **AcceptChanges** treba pozvati programski, radi ažuriranja statusa i verzija redaka
  - `dataSetArtikli.AcceptChanges();`
- ❑ Opoziv promjena u oba slučaja aktivira se ručno – pozivom **RejectChanges**
  - `dataSetArtikli.RejectChanges();`

# Primjer: povezivanje različitih svojstava

## ❑ Bilo koje javno svojstvo može poslužiti kao *DataMember*

```
// Data sources - varijable
string stringDataSource = "Vidi font";
Font fontDataSource = new Font("Lucida Console", 18);

// Bind the control properties to the data sources

// Control: textBox1
// PropertyName: Text
// DataSource: stringDataSource
// DataMember: null
textBox1.DataBindings.Add("Text", stringDataSource, null);

// Control: textBox1
// PropertyName: Font
// DataSource: fontDataSource
// DataMember: null
textBox1.DataBindings.Add("Font", fontDataSource, null);
```

# ***BindingMemberInfo***

## ❑ **Svojstva**

- `BindingMember` – puno ime atributa, npr. "Categories.CategoryID"
- `BindingPath` – staza do atributa, npr. "Categories"
- `BindingField` – atribut koji se povezuje, npr. "CategoryID"

## ❑ **Primjer:**

```
string strMsg;  
System.Windows.Forms.BindingMemberInfo bmo;  
  
bmo = this.textBoxSifArtikla.DataBindings[0].BindingMemberInfo;  
  
strMsg = "BindingMember:" + bmo.BindingMember.ToString();  
strMsg += "\nBindingPath:" + bmo.BindingPath.ToString();  
strMsg += "\nBindingField:" + bmo.BindingField.ToString();  
MessageBox.Show(strMsg);
```

# Privremeno obustavljanje validacije podataka

## ❑ Postupak **DataRow.BeginEdit**

- obustavlja događaje koji se odnose na promjenu podataka, sve dok se ne izvede `DataRow.EndEdit` ili `DataRow.CancelEdit`
- promjene se obavljaju nad `Proposed` verzijom i to je jedini slučaj kada `Proposed` postoji

## ❑ Primjer

```
private void buttonEdit_Click(object sender, EventArgs e)
{
    System.Data.DataRow currRow;
    currRow = GetCurrentRow();

    // Privremeno obustavljanje validacije podataka
    currRow.BeginEdit();
    currRow["SifArtikla"] = "-1";
    MessageBox.Show(currRow["SifArtikla",
        System.Data.DataRowVersion.Proposed].ToString());
    currRow.CancelEdit(); }
}
```



# Domaća zadaća

## □ Svaki član oblikuje i nad bazom podataka ugrađuje

- jednu zaslonsku masku obrasca za pojedinačni zapis (Form) za matične podatke
  - Dokument, Partner, Projekt, Posao, Račun, Osoba, Transakcija, Zadatak,
  - ...
- jednu zaslonsku masku za tabličnu obradu zapisa (Grid) za šifrnike
  - Vrsta<nečega>
  - Status<nečega>
  - Zvanje
  - Zanimanje
  - ...



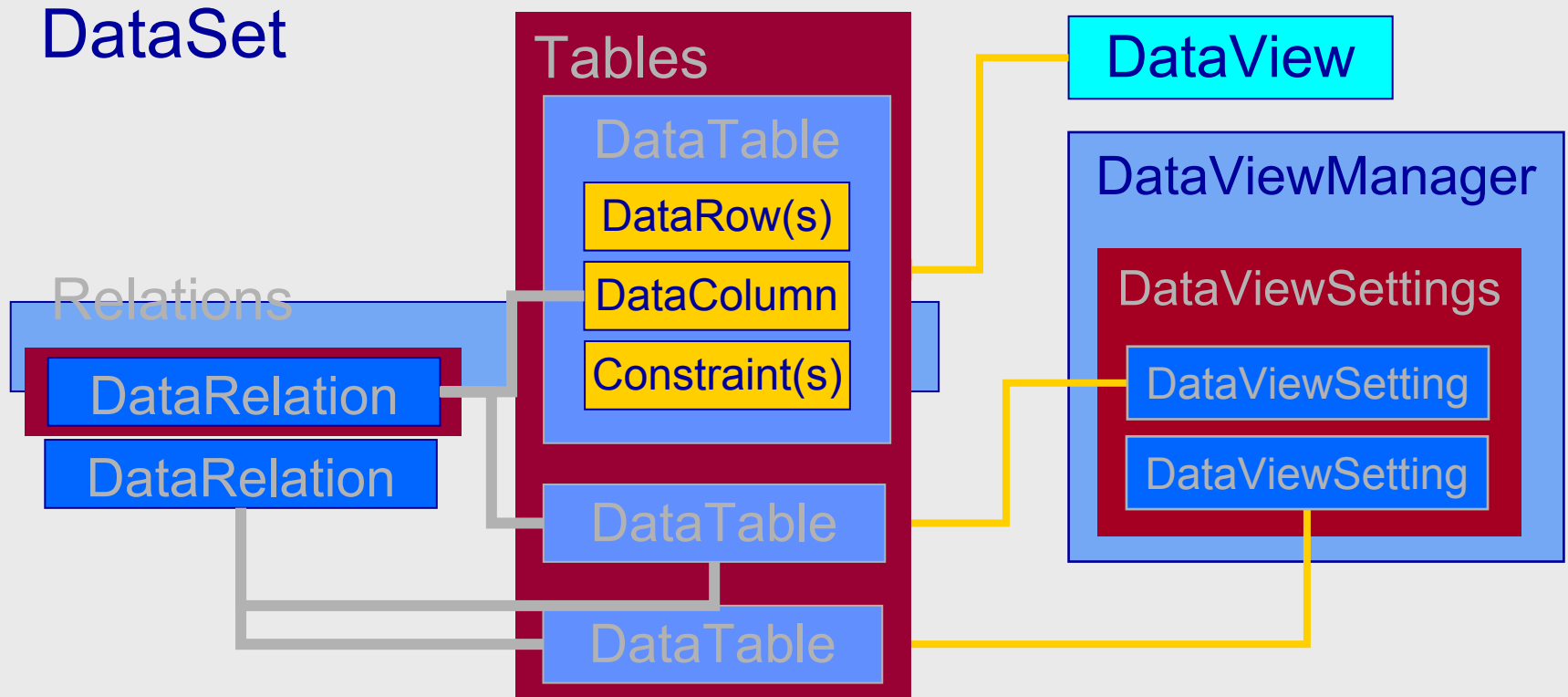
# Pogledi

---

# Pogled na podatke

## □ Glavni razredi

- `DataRow` – pogled na podatke
- `DataManager` – spremnik pogleda, kojima se pristupa preko `DataManagerSettings` kolekcije (analogija s `DataSet` spremnikom)



# Primjer ugradnje pogleda na podatke

## ❑ Primjer: ADO\SortFilter +oleDbDataAdapter (INNER JOIN)

**Sort & Filter**

**Filter**

Filter text input: |      \*     

Find      Select      Filter

**Sort**

☒ NazMjesta      ☐ NazDrzave      ☒ Uzlazno  
☐ PostBrMjesta      ☐ OznDrzave      ☐ Silazno

Sort

	IdMjesta	NazMjesta	PostBrMjesta	PostNazMjesta	NazDrzave	
▶	1	Ada	31214	Laslovo	Croatia	H
	2	Adamovec	10363	Belovar	Croatia	H
	3	Adžamovci	35422	Zapolje	Croatia	H
	4	Alaginci	34000	Požega	Croatia	H
	5	Alan	53271	Krivi Put	Croatia	H

# Razred *DataView*

## ❑ **DataView** - definira poglede na **DataTable** objekte

- Omogućuje sortiranje i filtriranje podataka
- Omogućuje povezivanje (binding) na kontrole korisničkog sučelja
- Nad istom `DataTable` tablicom može se kreirati više pogleda
- Svaka `DataTable` zapravo sadrži `DefaultView`, kojem se može pristupiti samo tijekom pogona (ne i u dizajnu)

## ❑ **Primjer:** **ADO\SortFilter**

- `dataView.Table = dataSetMjesta.MjestoDrzava`
- `drzavaBindingSource.DataSource = dataView`
- `dataGridView.DataSource = drzavaBindingSource` ili `dataView`

# DataViewManager

## ❑ Skup pogleda

- definira različite poglede nad skupom podataka
- omogućuje automatsku primjenu prethodno pohranjenih uvjeta za sortiranje i filtriranje, npr. prilikom izvođenja `GetChildRows`

## ❑ Svojstva

- `DataViewSettings` – `DataViewSettingCollection` kolekcija `DataViewSetting` objekata za sve tablice u skupu podataka
  - Svojstva: `RowFilter`, `RowStateFilter`, `Sort`, `Table`
- `DataSet` – postavlja/vraća skup podataka na koji se odnosi

## ❑ Postupak

- `CreateDataView(DataTable table);`
- kreira pogled nad zadanom tablicom

```
DataViewManager dvMgr = new DataViewManager( mojDataSet );  
  
dvMgr.CreateDataView(mojDataSet.Tables["DokumentStavke"]);  
  
dvMgr.DataViewSettings["Dokument"].Sort = "DatDokumenta DESC";
```

# DataView članovi

## □ Svojstva

- AllowDelete, AllowEdit, AllowNew – omogućeno brisanje/izmjena/dodavanje kroz pogled (promjene putem reference na izvorni redak su uvijek moguće)
- ApplyDefaultSort – određuje da li će se primijeniti predviđeni redoslijed podataka, određen izvorom podataka
- Count – broj zapisa, to jest pripadnih DataRowView objekata
- DataViewManager – kojem DataView pripada
- Item(Index) – DataRowView objekt na zadanom indeksu
- RowFilter – izraz na temelju kojeg se obavlja selekcija podataka
- RowStateFilter – filter za selekciju na temelju stanja podataka
- Table – DataTable objekt koji predstavlja izvor podataka
- Sort – izraz (stupci) po kojima se sortira

## □ Postupci

- AddNew – dodaje novi DataRowView
- Delete – briše redak na zadanom indeksu
- Find – pronalazi DataRowView objekte koji sadrže zadane vrijednosti ključa

# Primjeri *Sort, RowFilter, Find*

## ❑ Primjer: ADO\SortFilter

- `DataGridView dvMjesta = dataSetMjesta.MjestoDrzava.DefaultView`

## ❑ Sort

- `dvMjesta.Sort = "IdMjesta";`
- `dvMjesta.Sort = "IdDrzave, NazMjesta DESC";`

## ❑ RowFilter

- `dvMjesta.RowFilter = "IdMjesta = 13";`
- `dvMjesta.RowFilter = "NazMjesta LIKE '%Ada%' AND PostBrMjesta > 31000"`

## ❑ Find – upit na jednakost vrijednosti ključa

- Kada se Find izvodi nad pogledom na iste podatke, indeks pronađenog retka odgovara onom u `BindingContext` objektu

```
dvMjesta.Sort = "IdMjesta"
int idxFound = dvMjesta.Find(13);

// nadjen je this.dvMjesta[idxFound]["NazMjesta"];
```

# ***DataColumn.Expression***

- ❑ **Izrazi za postavljanje uvjeta na podatke odgovaraju onima koji se koriste (→ prethodni primjeri, primjeri u nastavku)**
  - za kreiranje izračunatih polja
  - za selekciju postupkom `DataTable.Select`
- ❑ **Elementi izraza**
  - operatori usporedbe: `AND`, `OR`, `NOT`, `<`, `>`, `<=`, `>=`, `<>`, `IN`, `LIKE`
  - aritmetički operatori: `+` `-` `*` `/` `%`
  - agregatne funkcije: `Sum`, `Avg`, `Min`, `Max`, `Count`, `StDev`, `Var`
- ❑ **Primjeri izraza**
  - obični string: `"CustomerName = '" + strName + "'"`
  - posebni znakovi: `"[UkPrihod/UkSati] > 100"`
  - datumi: `"OrderDate > #12/31/2003#"`
- ❑ **Izrazi mogu referencirati "roditelja" i "dijete"**
  - pr: `"Child.OrderTotal > 3000"`
  - pr: `"Parent.CustomerID= 'AFLKI'"`



# Selekcija redaka

## ❑ Postupak `DataTable.Select`

- `public DataRow[] Select();`
- `public DataRow[] Select(filterExpression);`
- `public DataRow[] Select(filterExpression, sort);`

## ❑ Postupak `DataRowView.FindRows(object[])`;

- vraća polje `DataRowView` objekata po vrijednostima stupaca Sort ključa

## ❑ Primjer: `ADO\SortFilter`

```
private DataRow[] drFound;

private void buttonSelect_Click(object sender, EventArgs e)
{
    drFound = this.dataSetMjesta.MjestoDrzava.Select(
        textBoxFilter.Text);

    //foreach (System.Data.DataRow dr in drFound)
        dataGridView.DataSource = drFound;
```

# Filtriranje po uvjetu na stanje podataka

## ❑ Svojstvo pogleda RowStateFilter

- None – prazan skup
- Added – samo novo dodani zapisi
- OriginalRows – originalni zapisi
- CurrentRows – aktualni zapisi
- Deleted – obrisani zapisi (ali još nepotvrđeni na izvoru)
- ModifiedOriginal – originalne vrijednosti naknadno promijenjenih zapisa
- ModifiedCurrent – aktualne vrijednosti promijenjenih zapisa

## ❑ Primjer:

```
System.Data.DataRowView drNew;  
drNew = this.dvDokument.AddNew();  
drNew["TipDokumenta"] = "R";  
drNew["BrojDokumenta"] = 1;  
  
//Set the RowStateFilter  
this.dvOrders.RowStateFilter = DataRowState.Added;
```

# Zadaci za vježbu

- ☐ Preraditi zadatak s predavanja *SortFilter*, tako da se podaci iznova filtriraju prilikom svakog utipkanog slova.
- ☐ Ugraditi sortiranje i filtriranje podataka u *Drzava*
- ☐ Ugraditi sortiranje i filtriranje podataka u *Artikl*

# Složene zaslonske maske

# Zaslonska maska oblika zaglavlje-stavke

## ❑ Primjer: ADO\DokumentStavka

- Prikaz i obrada detalja sinkronizirani s promjenama zaglavlja

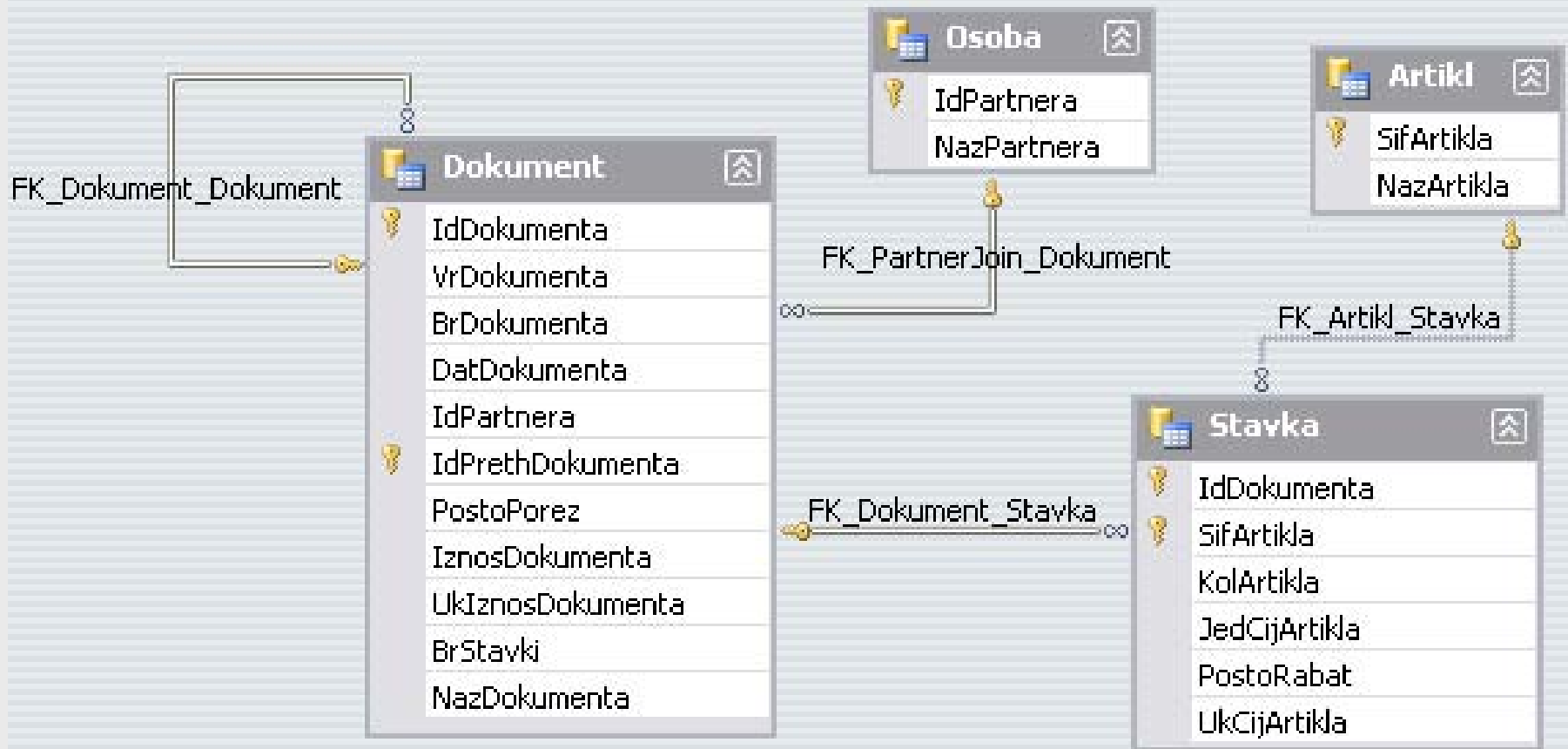
Id dokumenta:	<input type="text" value="9"/>	Vrsta:	<input type="text" value="0"/>	Broj:	<input type="text" value="81"/>	Datum:	<input type="text" value="5. 1 .2007"/>
Partner:	<input type="text" value="Polenus Mario (1705946330105)"/>					Porez:	<input type="text" value="0.22"/> [0 -1]
Prethodni:	<input type="text"/>					Iznos:	<input type="text" value="246.499,29 kn"/>

<input type="button" value="K"/>	<input type="button" value="←"/>	1 od 856	<input type="button" value="→"/>	<input type="button" value="&gt; "/>	Stavki: <input type="text" value="21"/>	<input type="button" value="Spremi"/>	<input type="button" value="Odustani"/>
----------------------------------	----------------------------------	----------	----------------------------------	--------------------------------------	---	---------------------------------------	---


	Naziv artikla	Količina	Jedinična cijena	Rabat	Ukupna cijena
▶	SODIMM SSSR PC 2700, 256 MB... ▼	3.00	366,00 kn	50.00%	549,00 kn
	KPD Athlon 64 3800+, 2400 MHz... ▼	3.00	719,00 kn	0.00%	2.157,00 kn
	Stalak za A/V komponente OMNI... ▼	15.00	199,00 kn	0.00%	2.985,00 kn
	MBO XBIT, s. 775, IL8, i945P, BU... ▼	2.00	846,00 kn	0.00%	1.692,00 kn
	Glazbena linija, GROUNDIG Vertig... ▼	2.00	649,00 kn	0.00%	1.298,00 kn
	TV LCD 66cm, SUNNY KDL-26U... ▼	1.00	3.999,00 kn	0.00%	3.999,00 kn
	Knjiga "Lightwave 3D 8" ▼	2.00	106,00 kn	0.00%	212,00 kn

# Model pripadajućeg skupa podataka

- ❑ **Pojedina tablica puni se vlastitim prilagodnikom pri inicijalizaciji zaslona (metoda *DokumentStavka\_Load*)**



# Izračunata polja

- ❑ Postavljena u dizajnu, istovjetno učinku sljedećih naredbi
- ❑ Primjer:  ADO\DokumentStavka - DokumentStavka.cs (2 načina)

```
dataSetDokumentStavka.Stavka.Columns.Add( // 1.NAČIN
    "UkCijArtikla", typeof(float),
    "JedCijArtikla * KolArtikla * (1 - PostoRabat)");

dataSetDokumentStavka.Dokument.Columns.Add( // 2.NAČIN
    new DataColumn(
        "UkIznosDokumenta", typeof(float),
        "Sum(Child(FK_Dokument_Stavka).UkCijArtikla)
            * (1 + PostoPorez) " )
    );

dataSetDokumentStavka.Dokument.Columns.Add(
    new DataColumn(
        "BrStavki", typeof(int),
        "Count(Child(FK_Dokument_Stavka).IdDokumenta) " )
    );
```

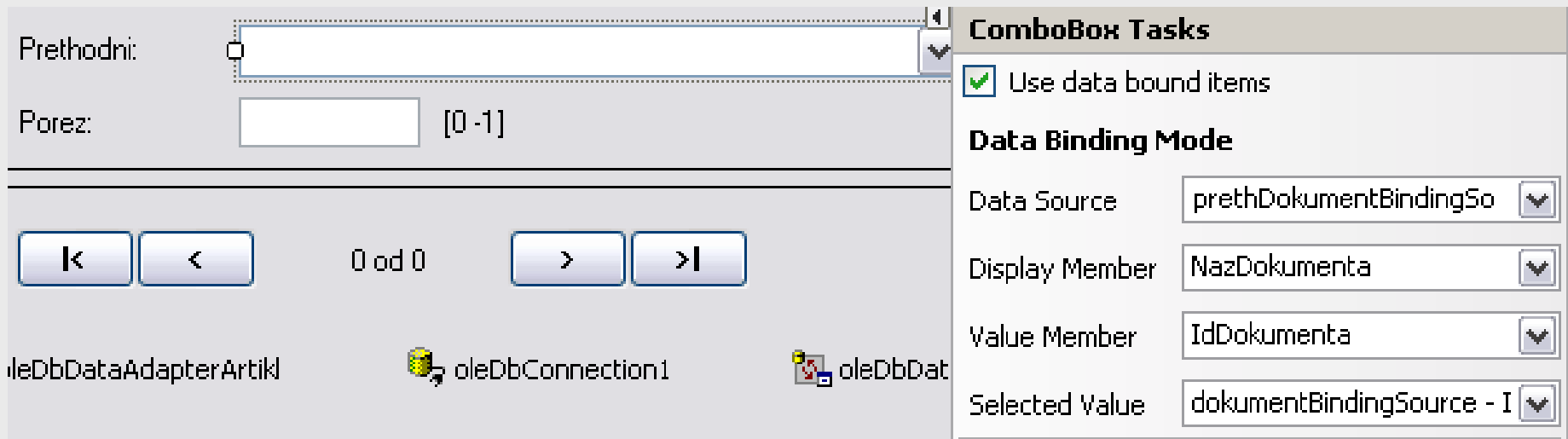
# Odabir vrijednosti stranog ključa

## ❑ Odabir vrijednosti stranog ključa padajućom listom (**ComboBox**)

## ❑ Primjer: ADO\DokumentStavka – definiranje programski

```
comboBoxPartner.DataSource = dataSetDokumentStavka.Osoba;  
comboBoxPartner.DisplayMember = "NazPartnera";  
comboBoxPartner.ValueMember = "IdPartnera";  
comboBoxPartner.DataBindings.Add(  
    "SelectedValue", dokumentBindingSource, "IdPartnera");
```

## ❑ Primjer: ADO\DokumentStavka – definiranje u dizajnu



The screenshot shows a Windows Forms application with a ComboBox control. The text "Prethodni:" is to the left of the ComboBox. Below it, the text "Porez:" is followed by a text box containing "0" and a range "[0 -1]". At the bottom of the form, there are four buttons: "K", "<", ">", and ">I". Below the buttons, there are three labels: "leDbDataAdapterArtikl", "oleDbConnection1", and "oleDbDat". The Visual Studio Properties Window is open on the right, showing the "ComboBox Tasks" section with "Use data bound items" checked. Below that, the "Data Binding Mode" section shows the following settings:

Data Binding Mode	
Data Source	prethDokumentBindingSo
Display Member	NazDokumenta
Value Member	IdDokumenta
Selected Value	dokumentBindingSource - I



# Sinkronizacija stavki

## ❑ Mreža stavki s podacima - *dataGridViewStavke*

## ❑ Primjer: ADO\DokumentStavka – definiranje u dizajnu

### ■ *dataGridViewStavke – Tasks*

```
this.fKDokumentStavkaBindingSource.DataMember =  
    "FK_Dokument_Stavka";  
this.fKDokumentStavkaBindingSource.DataSource =  
    this.dokumentBindingSource;  
...  
this.dataGridViewStavke.DataSource =  
    this.fKDokumentStavkaBindingSource;
```

## ❑ Povezivanje se obavlja koristeći vezu ostvarenu stranim ključem

## ❑ Za odabir prethodnog dokumenta nismo koristili istu mogućnost

### ■ jer bi ograničila mogućnost samo na povezane prethodnike

# Referentne vrijednosti stavke

## ❑ *dataGridViewStavke – Tasks – Edit Columns*

- *NazivArtikla.DisplayStyle = ComboBox*

Selected Columns:

- abl IdDokumenta
- abl Naziv artikla**
- abl SifArtikla
- abl Količina
- abl Jedinična cijena
- abl Rabat
- abl Ukupna cijena

Add... Remove

Bound Column Properties

Data	
DataPropertyName	SifArtikla
DataSource	dataSetDokumentStavka
DisplayMember	Artikl.NazArtikla
Items	(Collection)
ValueMember	Artikl.SifArtikla

Design	
(Name)	sifArtiklaDataGridViewTextBoxColumn1

**(Name)**  
Indicates the name used in code to identify the object.

OK Cancel

# Ažuriranje jedinične cijene artikla

## ❑ Primjer: ADO\DokumentStavka

### ■ Ažuriranje jedinične cijene artikla po odabiru artikla

```
private void dataGridViewStavke_CellEndEdit(
    object sender, DataGridViewCellEventArgs e)
{
    if (e.ColumnIndex == 1) // odabran artikl
    {
        dataSetDokumentStavka.Artikl.DefaultView.Sort = "SifArtikla";
        DataRowView drv =
            (DataRowView) fKDokumentStavkaBindingSource.Current;
        DataRowView row =
            ((dataSetDokumentStavka.Artikl.DefaultView.FindRows(
                ((DataRowView) sender) ["SifArtikla",
                fKDokumentStavkaBindingSource.Position]).Value))) [0];

        drv["JedCijArtikla"] = row["CijArtikla"];

        drv.EndEdit();

        ...
    }
}
```

# Spremanje podataka

## ❏ Primjer: ADO\DokumentStavka

```
// ažuriranje izvedene vrijednosti
```

```
DataRow row = ((DataRowView)dokumentBindingSource.Current).Row;  
if (row["IznosDokumenta"] != row["ukIznosDokumenta"])  
{  
    row["IznosDokumenta"] = row["ukIznosDokumenta"];  
}  
dokumentBindingSource.EndEdit();  
fKDokumentStavkaBindingSource.EndEdit();
```

```
UpdateData();
```

```
...
```

```
if (dataSetDokumentStavka.HasChanges()) // pohrana  
{  
    OleDbDataAdapterDokument.Update(  
        dataSetDokumentStavka.Dokument);  
    OleDbDataAdapterStavka.Update(  
        dataSetDokumentStavka.Stavka);  
}
```

# Opoziv izmjena

## ❑ **Primjer:** **ADO\DokumentStavka**

- Prekida se izmjena na pojedinoj poveznici
- Odbacivanje promjena zajedničko je i provodi se nad skupom podataka

```
private void buttonOdustani_Click(object sender, EventArgs e)
{
    dokumentBindingSource.CancelEdit();
    fKDokumentStavkaBindingSource.CancelEdit();
    dataSetDokumentStavka.RejectChanges();
}
```


# Command s parametrima

## ❑ Primjer: ADO\DokumentStavka – broj stavki dokumenta

```
void CommandBroji()  
{  
    cmdCount = new OleDbCommand(  
        "SELECT COUNT(*) AS Broj FROM Stavka WHERE "  
        + " (IdDokumenta = @IdDokumenta) "  
        , oleDbConnection1);  
    cmdCount.Parameters.Add(  
        new OleDbParameter("@IdDokumenta",  
            OleDbType.Integer, 4, "IdDokumenta"));
```

```
void PrebrojiStavke()  
{  
    oleDbConnection1.Open();  
    cmdCount.Parameters["@IdDokumenta"].Value =  
        textBoxIdDokumenta.Text;  
    int cnt = (int) cmdCount.ExecuteScalar();  
    oleDbConnection1.Close();  
    textBoxBrStavki.Text = cnt.ToString();  
}
```

# Zadaci za vježbu

- ☐ Doraditi *Artikl* dodavanjem tablice *JedinicaMjere* i veze s artiklom
- ☐ Proučiti i prilagoditi složenije primjere punjenja skupa podataka i ažuriranja podataka na izvoru,  ADO\Drzava
  - FillDataSet(FirmaDataSet dataSet) i UpdateDataSet()
- ☐ Napisati program koji će ažurirati tablicu *Artikl* tako da obradi sadržaj mape u kojoj se nalaze slike naziva oblika <IdArtikla>.jpg

# **Formatiranje i validacija podataka**

---



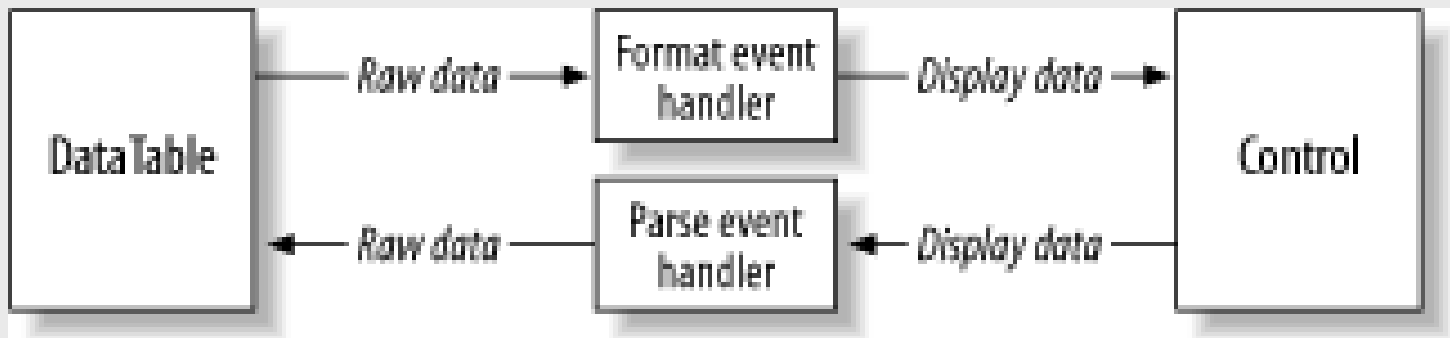
# Formatiranje objekata

## ❑ Događaji razreda `Binding`

- `ConvertEventHandler Format;` - vrijednost s izvora ide na kontrolu
  - zbiva se i nakon što je obavljen `Parse`
- `ConvertEventHandler Parse;` - vrijednost s kontrole ide na izvor
  - pojavljuje se samo ako je vrijednost promijenjena

## ❑ `ConvertEventHandler` je delegat funkcije za obradu događaja

- `public delegate void ConvertEventHandler (object sender, ConvertEventArgs e)`
- postavlja se u konstruktoru forme, nakon `InitializeComponent`



# Delegat ConvertEventHandler

## ❑ Primjer: ADO\DokumentStavka

### ■ Postavljanje ConvertEventHandler

```
Binding b = new Binding(  
    "Text", dokumentBindingSource, "UkIznosDokumenta");  
b.Format +=  
    new ConvertEventHandler(DecimalToCurrencyString);  
b.Parse +=  
    new ConvertEventHandler(CurrencyStringToDecimal);  
textBoxIznos.DataBindings.Add(b);
```

## ❑ ConvertEventArgs sadrži podatke za Format i Parse – svojstva

- DesiredType – tip podatka željene vrijednosti
- Value – vrijednost podatka

# Primjer događaja *Format* i *Parse*

## ❑ Funkcija za obradu **Format**

```
private void DecimalToCurrencyString(  
    object sender, ConvertEventArgs cevent) {  
    if (cevent.DesiredType != typeof(string)) return;  
    cevent.Value = ((decimal)cevent.Value).ToString("c");  
}
```

- "c" označava currency format – format za prikaz valute

## Funkcija za obradu **Parse**

```
private void CurrencyStringToDecimal(  
    object sender, ConvertEventArgs cevent)  
{  
    if (cevent.DesiredType != typeof(decimal)) return;  
    cevent.Value = Decimal.Parse(cevent.Value.ToString(),  
        System.Globalization.NumberStyles.Currency, null);  
}
```

# Validacija na skupu podataka

## □ DataTable događaji

- **ColumnChanging, ColumnChanged** – prije/poslije promjene elementa
- **RowChanging, RowChanged** – prije/poslije promjene sadržaja retka
- **RowDeleting, RowDeleted** – prije/poslije brisanja retka
  - Typed skupovi - mogu se kreirati zasebni rukovateli za pojedine stupce
  - Untyped skupovi - jedan rukovatelj obrađuje sve elemente retka

## □ Primjer: ADO\Artikl - DataSetArtikli.cs

- parcijalni razred DataSetArtikl, odnosno ArtiklDataTable
- preopteretimo EndInit i definiramo rukovatelj

```
public partial class DataSetArtikli {  
    partial class ArtiklDataTable    {  
        public override void EndInit()        {  
            base.EndInit();  
            ColumnChanging += ArtiklColumnChangingEvent;  
        }  
        public void ArtiklColumnChangingEvent(object sender,  
        ...
```

# Validacija pojedinačne vrijednosti

## ❑ Svojstva DataColumnChangeEventArgs argumenta rukovatelja ColumnChanging, ColumnChanged

- Column – stupac koji se ažurira
- ProposedValue – vrijednost koja ažurira postojeće stanje
- Row – referenca nadređenog retka

## ❑ Primjer: ADO\Artikl – implementacija rukovatelja

```
public void ArtiklColumnChangingEvent(object sender,
    System.Data.DataColumnChangeEventArgs e)
{
    if (e.Column.ColumnName == CijArtiklaColumn.ColumnName)
    {
        if (Decimal.Parse(e.ProposedValue.ToString()) <= 0)
            e.Row.SetColumnError("CijArtikla",
                "Cijena mora biti veća od 0");
        else
            e.Row.SetColumnError("CijArtikla", "");
    }
    ...
}
```

# Opis pogreške retka

- ❑ **DataRow ima postupke za postavljanje/dobavljanje opisa pogreške**
  - `public void SetColumnError(DataColumn, string);`
  - `public string GetColumnError(DataColumn);`
- ❑ **Automatizacija dojava pogreške retka obavlja se povezivanjem s `ErrorProvider` kontrolom koja prikazuje opis pogreške**
- ❑ **Primjer**
  - `ErrorProvider` objektu postavljaju se svojstva `DataSource` – postavlja se na `BindingSource` na koji je povezana forma

# Validacija retka

## ❑ Svojstva DataRowChangeEventArgs argumenta rukovatelja RowChanging, RowChanged, RowDeleting, RowDeleted

- Action – akcija koja se dogodila nad retkom, enum DataRowAction
  - Add, Change, Commit, Delete, Nothing, Rollback
- Row – referenca nadređenog retka

## ❑ Primjer:

```
ArtiklRowChanging += new  
ArtiklRowChangeEventHandler(ArtiklDataTable_ArtiklRowChanging);  
...  
void ArtiklDataTable_ArtiklRowChanging(  
    object sender, ArtiklRowChangeEvent e)  
{  
    if (e.Row.CijArtikla <= 0)  
        e.Row.SetColumnError("CijArtikla",  
                                "Cijena mora biti veća od 0");  
    else  
        e.Row.SetColumnError("CijArtikla", "");  
}
```

# Zadaci za vježbu

- ☐ Napisati aplikaciju za prikaz, izmjenu i dodavanje novog partnera (tablica Partner, Osoba i Tvrtka u bazi Firma.mdb).
- ☐ Kreirati poglede i pohranjene procedure u MS SQL Server bazi podataka, po uzoru na one koje postoje u MS Access bazi Firma.mdb
- ☐ Prilagoditi primjere napravljene za rad s bazom Firma.mdb tako da rade s bazom Firma.mdf



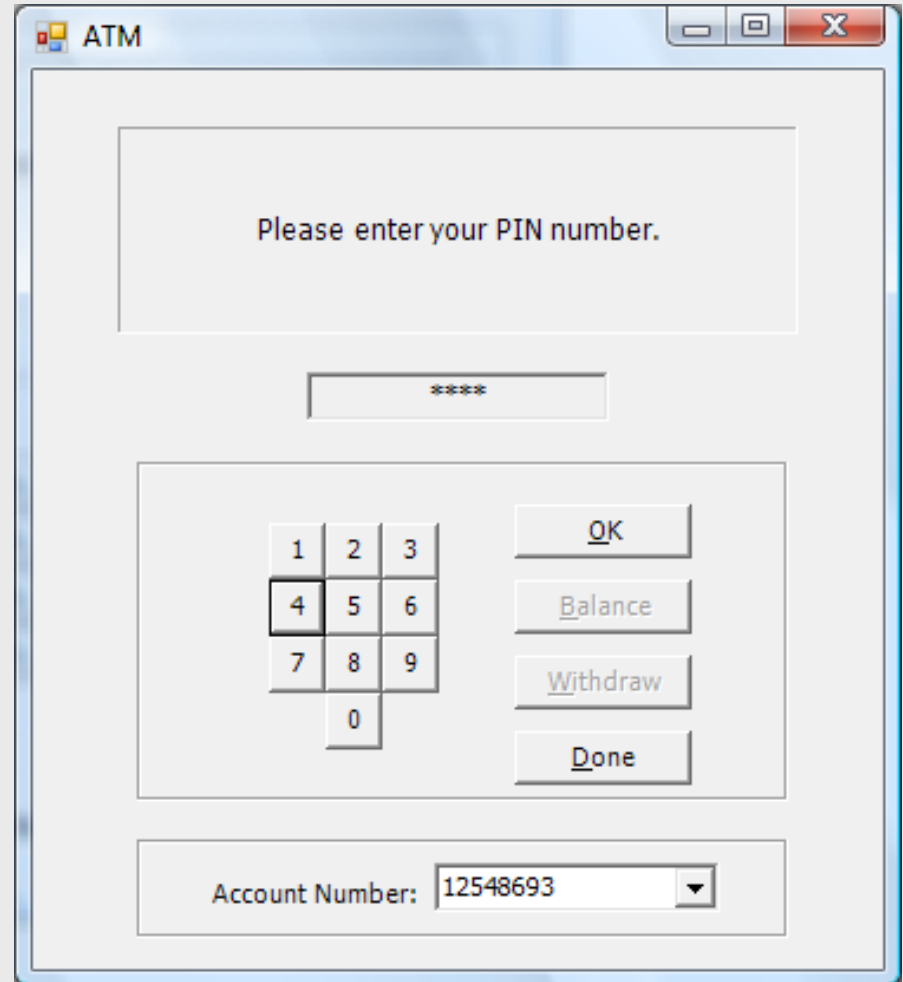
# Zadaci za vježbu

## ❑ Napisati aplikaciju za bankomat.

- U bazi podataka pohranjeni su brojevi računa, PIN-ovi, imena korisnika, te trenutni iznos na računu.
- Unosom broja računa i ispravnog PIN-a korisnik može podići iznos koji ne prekoračuje trenutno stanje računa.

## ❑ Primjer: ADO\ATM

## ❑ Doraditi primjer tako da omogući dozvoljeni minus



ATM

Please enter your PIN number.

\*\*\*\*

1	2	3
4	5	6
7	8	9
0		

OK

Balance

Withdraw

Done

Account Number: 12548693

# Zadaci za vježbu

- ☐ Implementirati postupke **Format** i **Parse** za ispravan prikaz/spremanje polja **PostoRabat** (u obliku postotka) na formi **DokumentStavke**, po uzoru na postupke za prikaz/spremanje **IznosDokumenta**
- ☐ U formi **DokumentStavke** u **DataGridView** za prikaz stavki dodati još i stupac za šifru artikla te je povezati na isti izvor kao i naziv artikla.
- ☐ Implementirati validaciju dinamičkog skupa za primjer **DokumentStavke**



# Domaća zadaća

- ❑ **Svaki član ugrađuje nad bazom podataka prethodno oblikovanu masku podataka za unos oblika zaglavlje-stavke (Master-Detail).**
  - Ugraditi mogućnost odabira vrijednosti stranog ključa putem padajuće liste u obje varijante, formular i mrežu.
  - U masku treba ugraditi mogućnost jednostavnog pretraživanja i sortiranja podataka, minimalno po najvažnijim atributima (primarni ključ i prvo polje zavisnog dijela).
  
- ❑ **Svaki član treba ugraditi poziv svih svojih maski napravljenih u 2. ciklusu u vlastiti podizbornik unutar zajedničkog izbornika ekipe (jedan Solution, jedan glavni izbornik, više podizbornika).**

# Transakcije

# Transakcije

## ❑ Transakcija - slijed naredbi koji se mora obaviti u cjelini ili se ne obavlja.

### ■ "Ručne" transakcije

- Transakcije definirane eksplicitnim naredbama za početak i kraj transakcije

### ■ Automatske transakcije

- Usluga COM+ tehnologije koja omogućuje dizajn razreda koji će u pogonu sudjelovati u transakcijama  
(`System.EnterpriseServices.ServicedComponent`)
- Jedina opcija za transakcije nad podacima iz raznorodnih izvora.

### ■ Transakcije SUBP

- Logika ovih transakcija sadržana je u pohranjenim procedurama
- Pružaju najbolje performanse, ali ih je nešto teže ugraditi
- Ograničenje - Kad se iz neke transakcije poziva dvije ili više procedura, koristi se samo jedan model transakcija (ručne ili automatske).

## ❑ Ručne transakcije započinju postupkom konekcije

### ■ `BeginTransaction` — programsko započinjanje transakcije

- `npr: conn.BeginTransaction();` ili  
`conn.BeginTransaction(isolationLevel, transName);`

# OleDbTransaction i SqlConnection

## □ Svojstva:

- `Connection` – `OleDbConnection`, odnosno `SqlConnection` objekt pridružen transakciji ili `null` referenca ako transakcija nije više validna
- `IsolationLevel` – razina izolacije zaključavanja za transakciju, vrijednost odgovarajućeg enumeratora izvedenog iz `IsolationLevel`
  - `Chaos` – slično `ReadUncommitted`, s tim da nepotvrđene promjene načinjene na višoj razini izolacije ne mogu biti nadvladane ("pregažene")
  - `ReadCommitted` (standardno) – čitaju se samo potvrđeno pohranjeni podaci (izbjegava prljavo čitanje) uz dijeljeno zaključavanje, koji ipak mogu biti promijenjeni prije kraja transakcije → non-repeatable reads or phantom data
  - `ReadUncommitted` – prljavo čitanje, bez zaključavanja
  - `RepeatableRead` – pročitani zapisi zaključavaju se do kraja transakcije, onemogućujući promjenu drugim korisnicima. Eliminira neponovljiva čitanja, ali ne i fantomske retke.
  - `Serializable` – zaključava se čitav skup pročitanih podataka, koji ostaju zaključani do završetka transakcije
  - `Unspecified` – koristi se neko drugo zaključavanje, čija razina se trenutno ne može odrediti

# *OleDbTransaction i SQLTransaction*

## ❑ Postupci

- `BeginTransaction` — programsko započinjanje transakcije
- `Begin` — programsko započinjanje ugniježdene transakcije
  - osnovna transakcija ne može biti potvrđena tako dugo dok se ne potvrde sve ugniježdene transakcije)
  - samo `OleDbTransaction` ima postupak `Begin`
- `Commit` — spremanje/potvrđivanje promjena načinjenih tijekom transakcije te završetak transakcije
- `RollbackTransaction` — odbacivanje promjena uz završetak transakcije
- `CommitTransaction` i `RollbackTransaction` mogu započeti novu transakciju

## ❑ Nakon što transakcija započne na konekciji, sve naredbe obavljene na toj konekciji moraju sudjelovati u transakciji

- Da bi se to stvarno i dogodilo, naredbe moraju referencirati konekciju, što se ne događa automatski.
- Ipak, za naredbe koje su sudjelovale u transakcijama koje su završile, referenca na konekciju se automatski postavlja na `null`

# Naredbe s transakcijom

## ❑ Otvaranje konekcije i kreiranje transakcije

```
SqlConnection conn = new SqlConnection(...);  
conn.Open();  
SqlTransaction trans =  
    conn.BeginTransaction(IsolationLevel.ReadCommitted);
```

## ❑ Kreiranje naredbe i njeno priključivanje transakciji

```
SqlCommand cmd = new SqlCommand("DELETE FROM ...", conn);  
cmd.Transaction = trans;
```

## ❑ Izvođenje naredbe i završetak transakcije

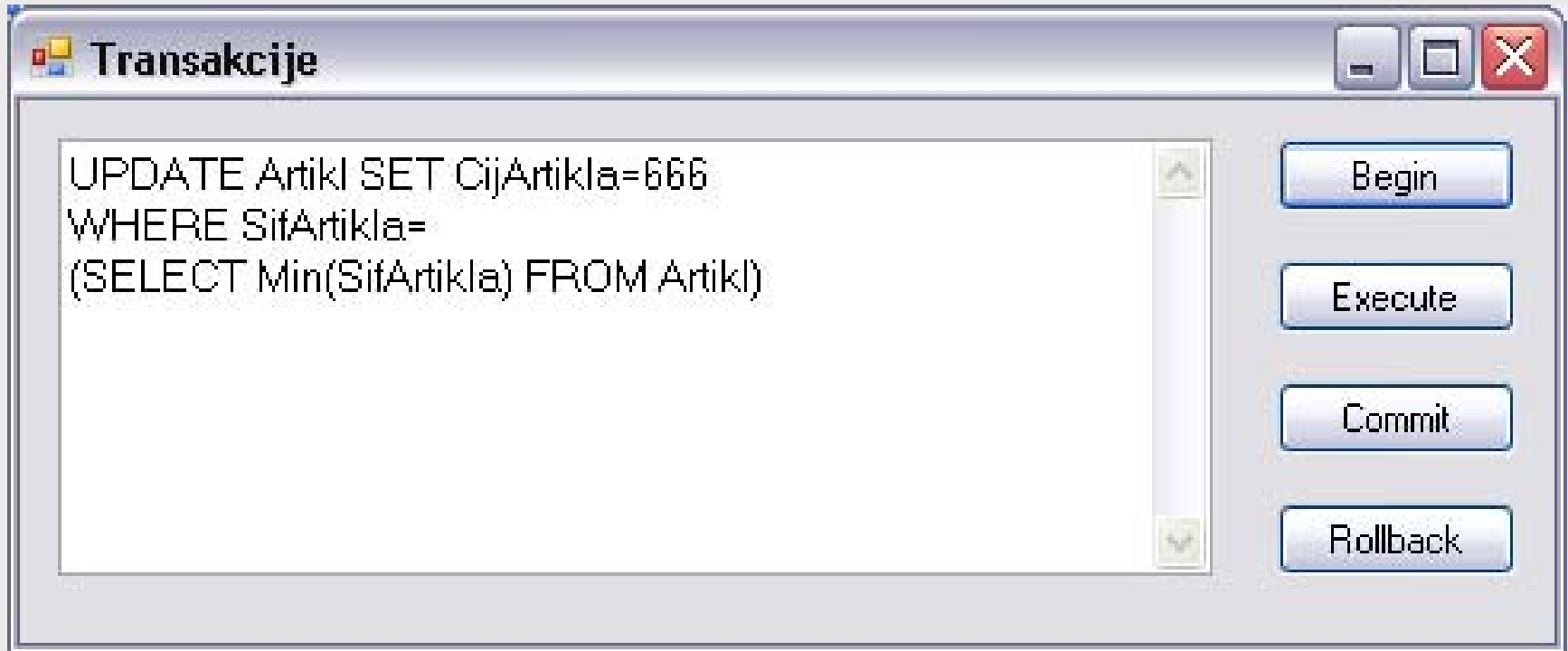
```
try {  
    int cnt = cmd.ExecuteNonQuery();  
    trans.Commit();  
} catch {  
    trans.Rollback();  
} finally {  
    conn.Close();  
}
```

## ❑ Primjer: ADO\Transakcije



# Primjer s transakcijom

## ❑ Primjer: ADO\Transakcije



# **Dinamičko stvaranje skupova podataka**

# Razred *DataTable*

## ❑ Reprezentacija tablice s podacima u memoriji – definira strukturu podataka

- `DataTable()` ;
- `DataTable(string)` ;

## ❑ Svojstva

- `Columns` – kolekcija atributa
- `Rows` - kolekcija `DataRow` objekata
- `ChildRelations` - kolekcija veza u kojima tablica referencira druge tablice
- `ParentRelations` – kolekcija veza kojima je tablica referencirana iz drugih
- `Constraints` – kolekcija ograničenja
- `PrimaryKey` – kolekcija  `DataColumn`  objekata koji čine primarni ključ
- `TableName` – naziv tablice

## ❑ Neki postupci

- `Clear` – čisti podatke
- `Clone` – kopira kompletnu strukturu uključujući ograničenja, ali ne i podatke
- `Copy` – kopira strukturu i podatke objekta koji izvodi postupak
- `NewRow` – kreira novi `DataRow`
- `Select` – vraća kolekciju `DataRow` objekata

# Razred *DataColumn*

## □ DataColumn – reprezentacija sheme DataTable stupca

- `DataColumn()`
- `DataColumn(columnName, dataType)`

## □ Svojstva

- `AllowDBNull` – neobvezne vrijednosti
- `AutoIncrement`, `AutoIncrementSeed`, `AutoIncrementStep` – polje sa samopovećavajućim vrijednostima, početna vrijednost, korak
- `Caption` - labela
- `ColumnName` - naziv
- `DataType` – tip podatka (`Boolean`, `Byte`, `Char`, `DateTime`, `Decimal`, `Double`, `Int16`, `Int32`, `Int64`, `SByte`, `Single`, `String`, `TimeSpan`, `UInt16`, `UInt32`, `UInt64`)
- `DefaultValue` – standardna, predviđena vrijednost
- `Expression` – izraz za izračunatu vrijednost (npr. formula, agregatna f-ja)
- `MaxLength` – najveća duljina tekstovnog podatka
- `ReadOnly` – oznaka da se vrijednost ne mijenja nakon što je redak dodan
- `Unique` – bool oznaka jedinstvene vrijednost u tablici

# Dinamičko kreiranje tablica

## ❑ Primjer: ADO\DataSetUnTyped

Untyped skupovi podataka

Dokumenti:

	IdPartnera	Naziv	Broj	BrojDokumenata
▶	8	Lišnjić Romina	0608956339306	1
	9	LončarFrane	0710951330113	7
	10	LukačevićStipo	0810960330171	9
	11	LjubešićBiljana	1001971330031	1
	12	LjubešićMatija	1111953335166	2
	13	Marohnić Emir	1201960330179	2

Stavke:

	IdDokumenta	VrDokumenta	BrDokumenta	DatDokumenta	IdPartnera
▶	1935	0	983	30.1.2007	8
*					

# Dinamičko kreiranje tablica

## ❑ Primjer: ADO\DataSetUnTyped

- System.Type.GetType(string); # dobavlja tip za zadani naziv

```
DataTable dt = new DataTable("Dokument");

dt.Columns.Add("IdDokumenta", typeof(int));
dt.Columns.Add("VrDokumenta", typeof(string));
dt.Columns.Add("BrDokumenta", typeof(int));
dt.Columns.Add("DatDokumenta", typeof(System.DateTime));
dt.Columns.Add("IdPartnera", typeof(int));
dt.Columns.Add("IznosDokumenta", typeof(decimal));

dsUntyped = new DataSet("DataSetUntyped");
dsUntyped.Tables.Add(dt);
```

# Dinamičko dodavanje stupaca tablice

## ❑ Primjer: ADO\DokumentStavka – izračunata polja

```
dataSetDokumentStavka.Stavka.Columns.Add( // 1.NAČIN
    "UkCijArtikla", typeof(float),
    "JedCijArtikla * KolArtikla * (1 - PostoRabat)");

dataSetDokumentStavka.Dokument.Columns.Add( // 2.NAČIN
    new DataColumn(
        "UkIznosDokumenta", typeof(float),
        "Sum(Child(FK_Dokument_Stavka).UkCijArtikla)
            * (1 + PostoPorez) " )
    );

dataSetDokumentStavka.Dokument.Columns.Add(
    new DataColumn(
        "BrStavki", typeof(int),
        "Count(Child(FK_Dokument_Stavka).IdDokumenta) " )
    );
```

# Razred *DataRelation*

## ❑ Logička veza između dva `DataTable` objekta

- uspostavlja se parovima  `DataColumn`  objekata iz pojedinih  `DataTable` 
  - `DataType`  odgovarajućih  `DataColumn`  mora biti jednak
  - npr. nije dozvoljeno povezivanje  `Int32`  i  `String`  stupaca

## ❑ Konstruktori

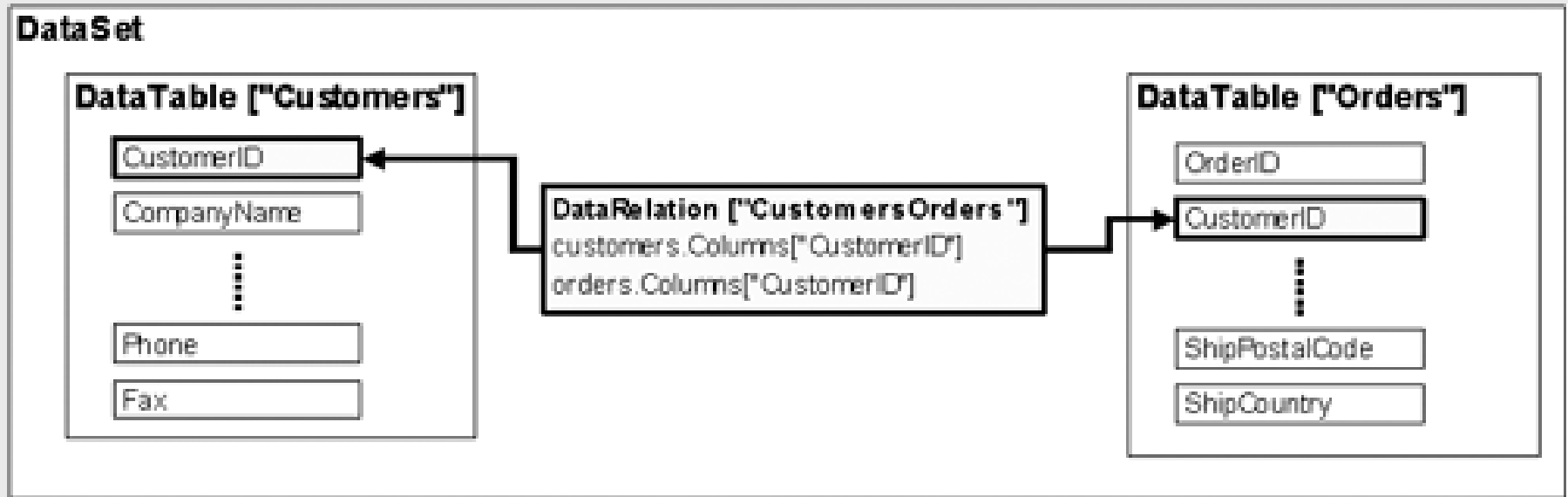
- `public DataRelation(string, DataColumn, DataColumn);`
- `public DataRelation(string, DataColumn[], DataColumn[]);`

## ❑ Svojstva

- `ChildTable, ParentTable`  – referencirajuća/zavisna tablica, odnosno referencirana/nadređena tablica
- `ChildColumns, ParentColumns`  – kolekcije atributa povezanih tablica
- `ChildKeyConstraint, ParentKeyConstraint`  – definiraju  `ForeignKeyConstraint`  ograničenje na dijete, odnosno roditelja
- `DataSet`  – skup podataka kojem veza pripada
- `RelationName`  – naziv veze



# Dinamičko kreiranje veza



```
DataRelation relOrdDet;  
DataColumn colMaster;  
DataColumn colDetail;
```

```
colMaster = dsUntyped.Tables["Partner"].Columns["IdPartnera"];  
colDetail = dsUntyped.Tables["Dokument"].Columns["IdPartnera"];  
relOrdDet = new DataRelation("FK_Partner_Dokument",  
                             colMaster, colDetail);
```

```
dsUntyped.Relations.Add(relOrdDet) ;
```

# Ograničenje veze – stranog ključa

## ❑ ForeignKeyConstraint razred sa svojstvima

- AcceptRejectRule – akcija koja se obavlja prigodom AcceptChanges
- Table, Columns – podređena tablica i njeni atributi ograničenja
- RelatedTable, RelatedColumns – nadređena tablica i njeni atributi ograničenja
- DeleteRule, UpdateRule - ograničenje dodavanja, odnosno izmjene
  - enum Rule = { Cascade, None, SetDefault, SetNull }

## ❑ Primjer: ADO\DatasetUntyped

```
ForeignKeyConstraint fk;  
fk = new System.Data.ForeignKeyConstraint  
    ("FK_Partner_Dokument",  
        dsUntyped.Tables["Partner"].Columns["IdPartnera"],  
        dsUntyped.Tables["Dokument"].Columns["Idpartnera"]  
    );  
dsUntyped.Tables["Dokument"].Constraints.Add(fk);
```

# Ograničenje jedinstvene vrijednosti

## ❑ UniqueConstraint razred sa svojstvima

- Columns – polje atributa na koje se odnosi ograničenje
- ConstraintName – naziv ograničenja
- IsPrimaryKey – bool oznaka da je ograničenje primarni ključ

## ❑ Primjer: ADO\DataSetUntyped

```
UniqueConstraint uc;  
uc = new System.Data.UniqueConstraint(  
    "NewUnique",  
    new DataColumn[] {  
        dt.Columns["VrDokumenta"],  
        dt.Columns["BrDokumenta"] }  
    false );  
// dt.Columns["IdDokumenta"]); // bez new  
  
dt.Constraints.Add(uc) ;
```

# Dohvaćanje zavisnih podataka

## ❑ DataRow postupci

- `GetChildRows` – dohvaća sve zavisne retke
- `GetParentRows` – dohvaća nadređeni redak

## ❑ DataSet postupak Merge – udružuje druge objekte u DataSet

- `Merge(DataRow[]);` // udružuje polje DataRow objekata
- `Merge(DataSet);` // udružuje skup i njegovu shemu
- `Merge(DataTable);` // udružuje tablicu i njezinu shemu

## ❑ DataSet događaj

- `MergeFailed` – narušeni integritet, uz postavljeno `EnforceConstraints`

## ❑ Primjer:

```
DataRowView drvCurr;  
DataSet dsPartnerDoc;  
dsPartnerDoc = new System.Data.DataSet();  
// kreirati vezu PartnerDokument po potrebi  
...  
dsPartnerDoc.Merge(  
    drvCurr.Row.GetChildRows("PartnerDokument"));
```

# Zadaci za vježbu

- ❑ Primjer *DataSetUntyped* prilagoditi tako da se iznos dokumenta računa kao agregatna funkcija u upitu izvora podataka.
- ❑ U *DataSetUntyped* dodati tablicu *VrstaDokumenta* te ju povezati s tablicom *Dokument* stranim ključem. Ograničenje na podatke podržati prethodnim dodavanjem svih do sada korištenih vrsta dokumenata iz tablice *Dokument* u tablicu *VrstaDokumenta*.

# Reference

## ☐ Overview of ADO.NET

- [http://msdn2.microsoft.com/en-us/library/h43ks021\(VS.71\).aspx](http://msdn2.microsoft.com/en-us/library/h43ks021(VS.71).aspx)

## ☐ Quickstarts

- <http://samples.gotdotnet.com/quickstart/howto/doc/adoplus/ADOPlusOverview.aspx>

## ☐ Articles

- <http://www.devarticles.com/c/b/ADO.NET/>