

# Aplikacija nad bazom podataka

---

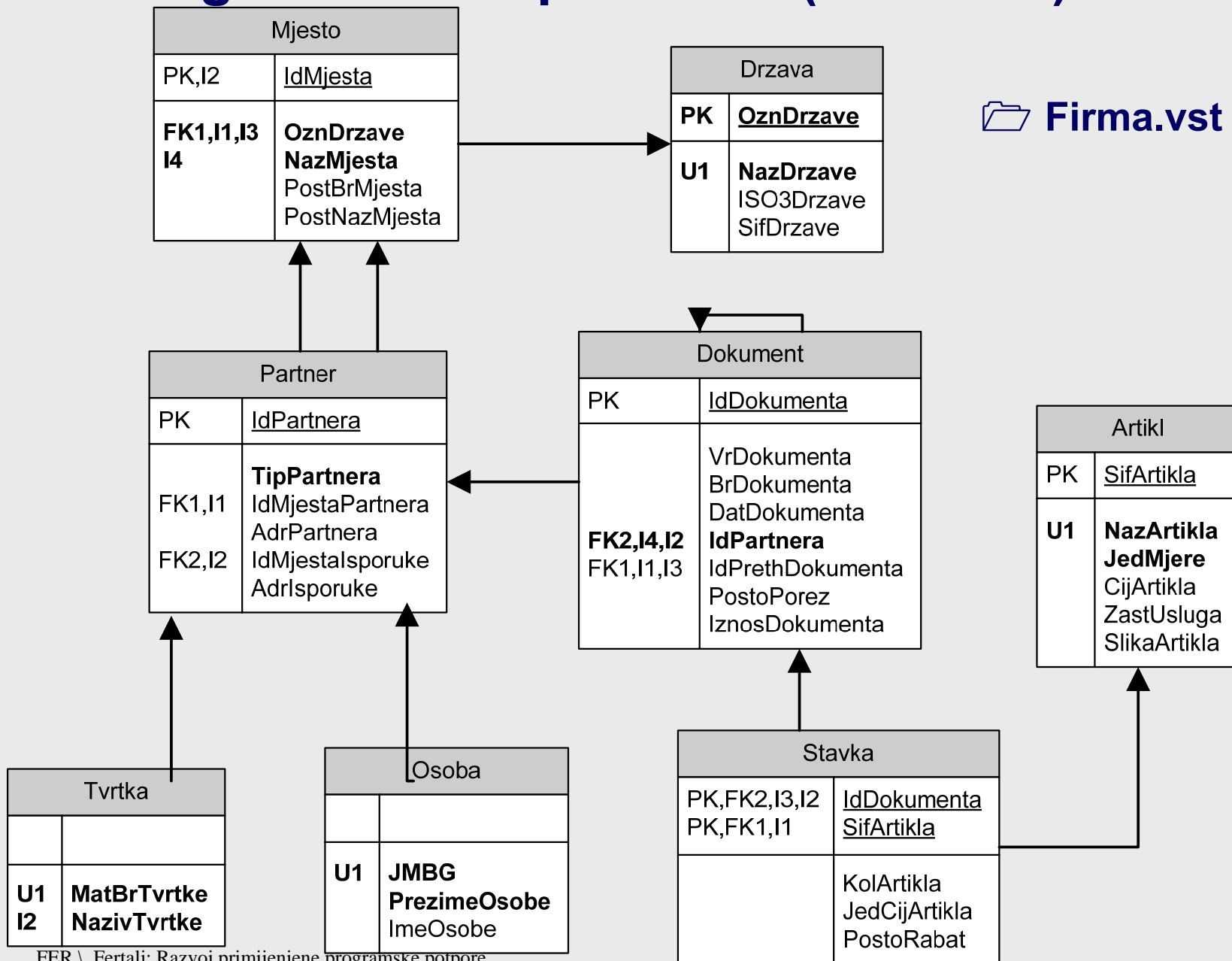
6/13

# Ogledna baza podataka

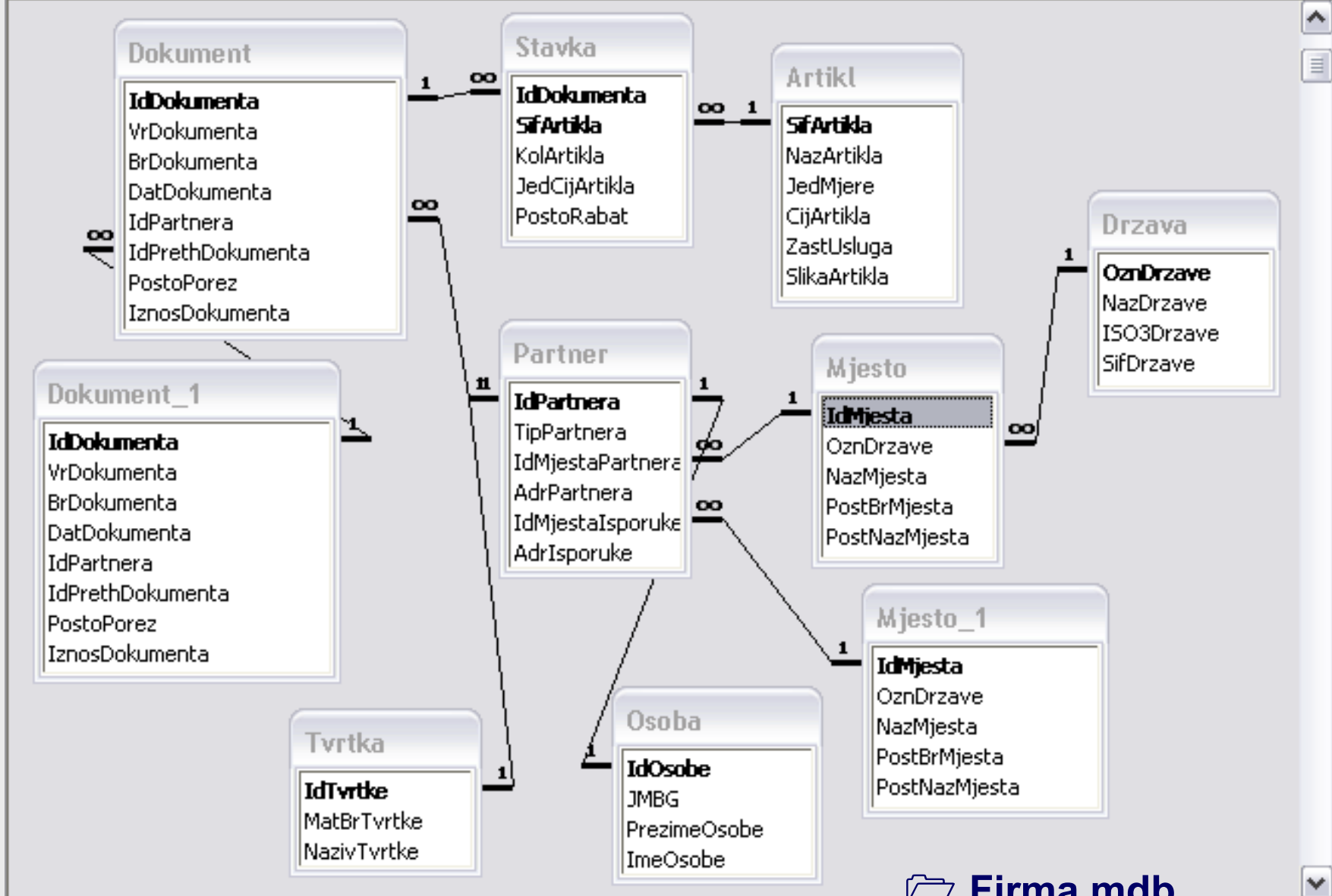
- ❑ **Primjer baze podataka (BazePodataka.zip)**
- ❑ **BazePodataka.txt – kratke upute**
- ❑ **Firma.mdf, Firma.Idf – SQL Server baza podataka**
- ❑ **Firma.mdb – MS Access baza podataka za demonstraciju OleDb**
- ❑ **Firma.vst – MS Visio dijagram baze podataka**

# Ogledna baza podataka (MS Visio)

 **Firma.vst**



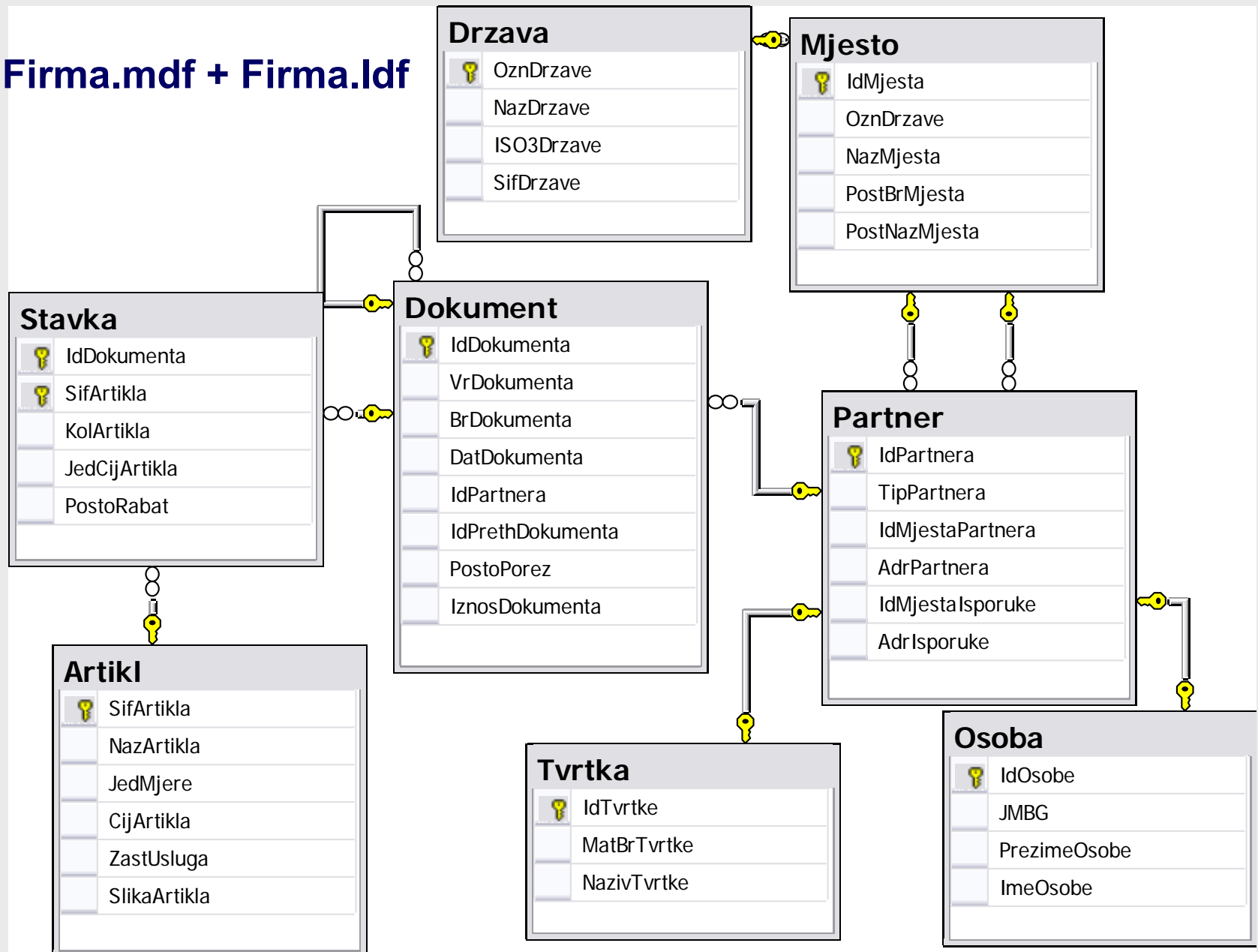
# Ogledna baza podataka (MS Access)



Firma.mdb

# Ogledna baza podataka (MS SQL Server)

Firma.mdf + Firma.ldf



# SQL Server - sustav za upravljanje BP (SUBP)

- ❑ Control Panel \ Administrative Tools \ Services
- ❑ My Computer \ Manage \ Services and Applications

The screenshot shows the Windows XP 'Computer Management' console. The left pane shows the tree view with 'Services and Applications' expanded. The right pane displays a list of services. 'SQL Server (SQL05)' is selected. A properties dialog box for 'SQL Server (SQL05)' is open, showing the 'General' tab. The service name is 'MSSQL\$SQL05', the display name is 'SQL Server (SQL05)', and the description is 'Provides storage, processing and controlled access of data and rapid transaction processing.' The path to the executable is 'C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Binn\sqlservr.'. The startup type is set to 'Manual'. The service status is 'Started'. At the bottom of the dialog are buttons for 'Start', 'Stop', 'Pause', and 'Resume'. Below these buttons is a text box for 'Start parameters'.

Name	Description	Status
Simple Mail Transfer Protocol (SMTP)	Transports...	
Smart Card	Manages a...	
SoundMAX Agent Service		Started
SQL Server (SQL05)	Provides st...	Started
SQL Server (SQLEXPRESS)	Provides st...	
SQL Server Active Directory Helper	Enables int...	
SQL Server Agent (SQL05)	Executes j...	
SQL Server Browser	Provides S...	Started
SQL Server FullText Search (SQL05)	Quickly cre...	
SQL Server VSS Writer	Provides th...	Started
SQLSERVERAGENT		
SSDP Discovery Service	Enables dis...	Started
System Event Notification	Tracks syst...	Started
System Restore Service	Performs s...	Started
Task Scheduler	Enables a ...	Started
TCP/IP NetBIOS Helper	Enables su...	Started
Telephony	Provides T...	Started
Telnet	Enables a r...	
Terminal Services	Allows mul...	Started

**SQL Server (SQL05) Properties (Local Computer)**

General | Log On | Recovery | Dependencies

Service name: MSSQL\$SQL05

Display name: SQL Server (SQL05)

Description: Provides storage, processing and controlled access of data and rapid transaction processing.

Path to executable: "C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Binn\sqlservr."

Startup type: Manual

Service status: Started

Start Stop Pause Resume

You can specify the start parameters that apply when you start the service from here.

Start parameters:

OK Cancel Apply

# SQL Server Management Studio (sučelje SUBP)

- ❑ **Priključak na sustav**
- ❑ **Server name: računalo \ instance**
  - "." = "ovo" računalo
  - SQL05, SQL2008, SQLEXPRESS – instance (ovisno o instalaciji)
- ❑ **Autentifikacija – identificiranje korisnika**
- ❑ **Windows Authentication – provjera korisnika domene, oblika**
  - NazivDomene\Korisnik
- ❑ **SQL Server Authentication – provjera korisnika čije je ime definirano u SUBF**
  - Npr. sa (system administrator), korisnik kreiran prilikom instalacije



# Kreiranje baze podataka

## ❑ Databases \ New Database

- Database name – osnovno ime baze podataka
- Database files – imena datoteke s podacima (Data) i datoteke s dnevničkim zapisima (Log) trebaju biti različiti
- Owner – vlasnik: sa ili domenski korisnik

## ❑ Zadaci za vježbu:

- kreirati bazu podataka na osobnom računalu
- priključiti se na poslužitelj baza podataka student prema uputama na VSS

**New Database**

Select a page: General, Options, Filegroups

Script Help

Database name: Baza

Owner: <default>

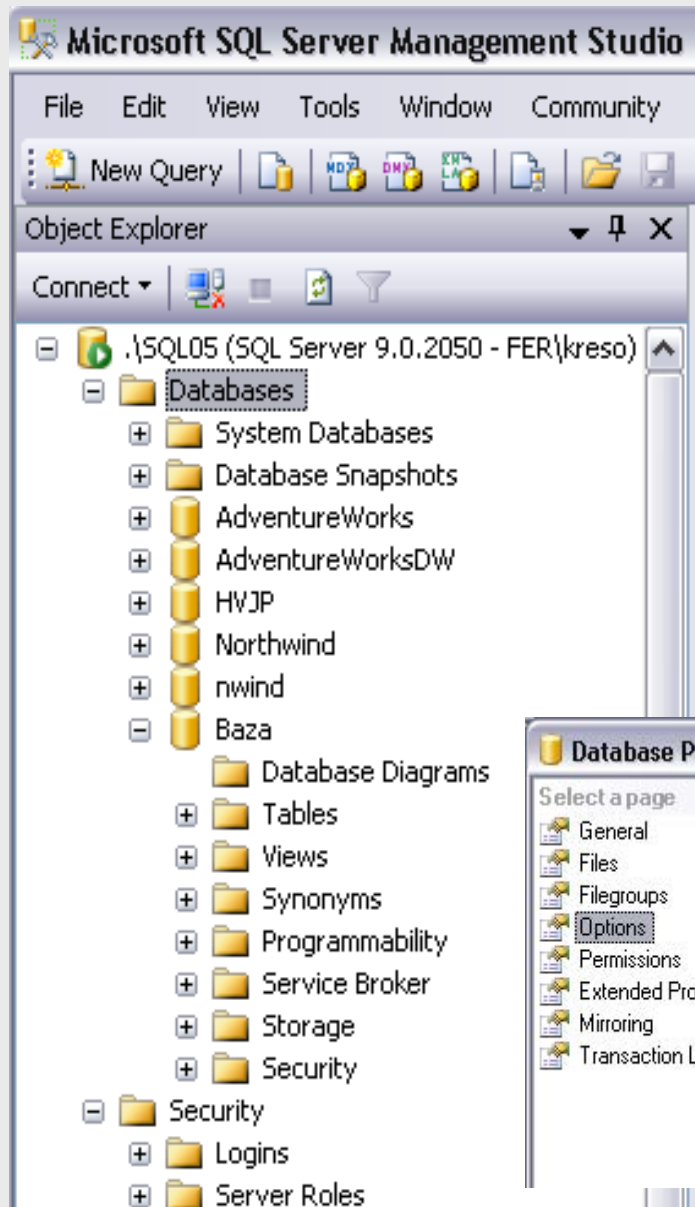
☐ Use full-text indexing

Database files:

Logical Name	File Type	Filegroup	Initial Size (MB)	Autogrowth	Path	File Name
Baza	Data	PRIMARY	3	By 1 MB, unre...	c:\projects	
Baza	Log	Not Applicable	1	By 10 percent, ...	c:\projects	

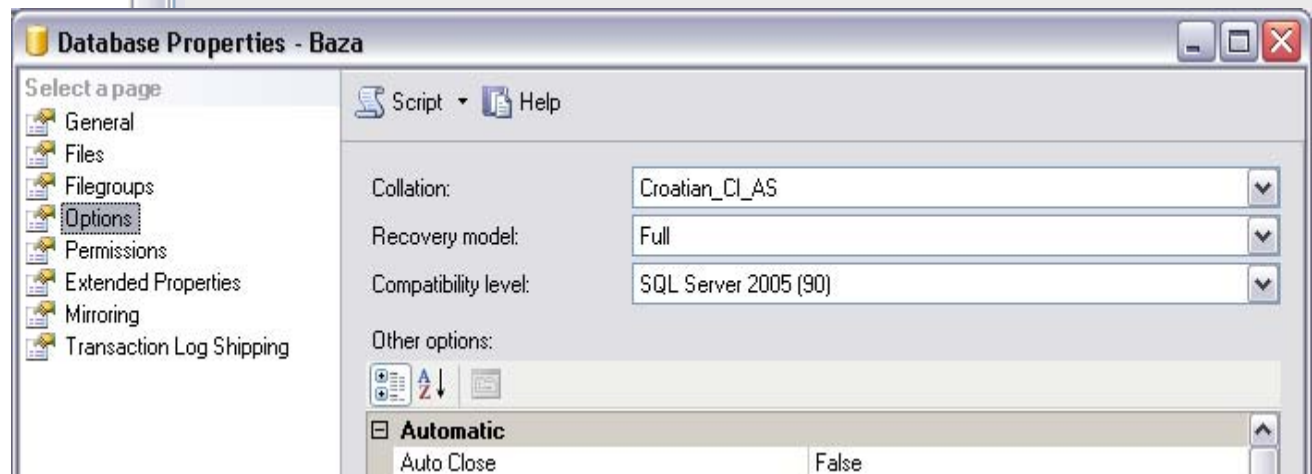


# Objekti i svojstva baze podataka



## ❑ Postavljanje svojstava – desni gumb na bazi \ Properties

- Files \ Owner – vlasnik: sa ili domenski korisnik
  - nakon kopiranja baze podataka s drugog računala, kada se može dogoditi da nije moguć pristup dijagramima
- Options: Collation (Croatian CI AS) , Compatibility Level, ...



# Prikapčanje i otkapčanje baze podataka

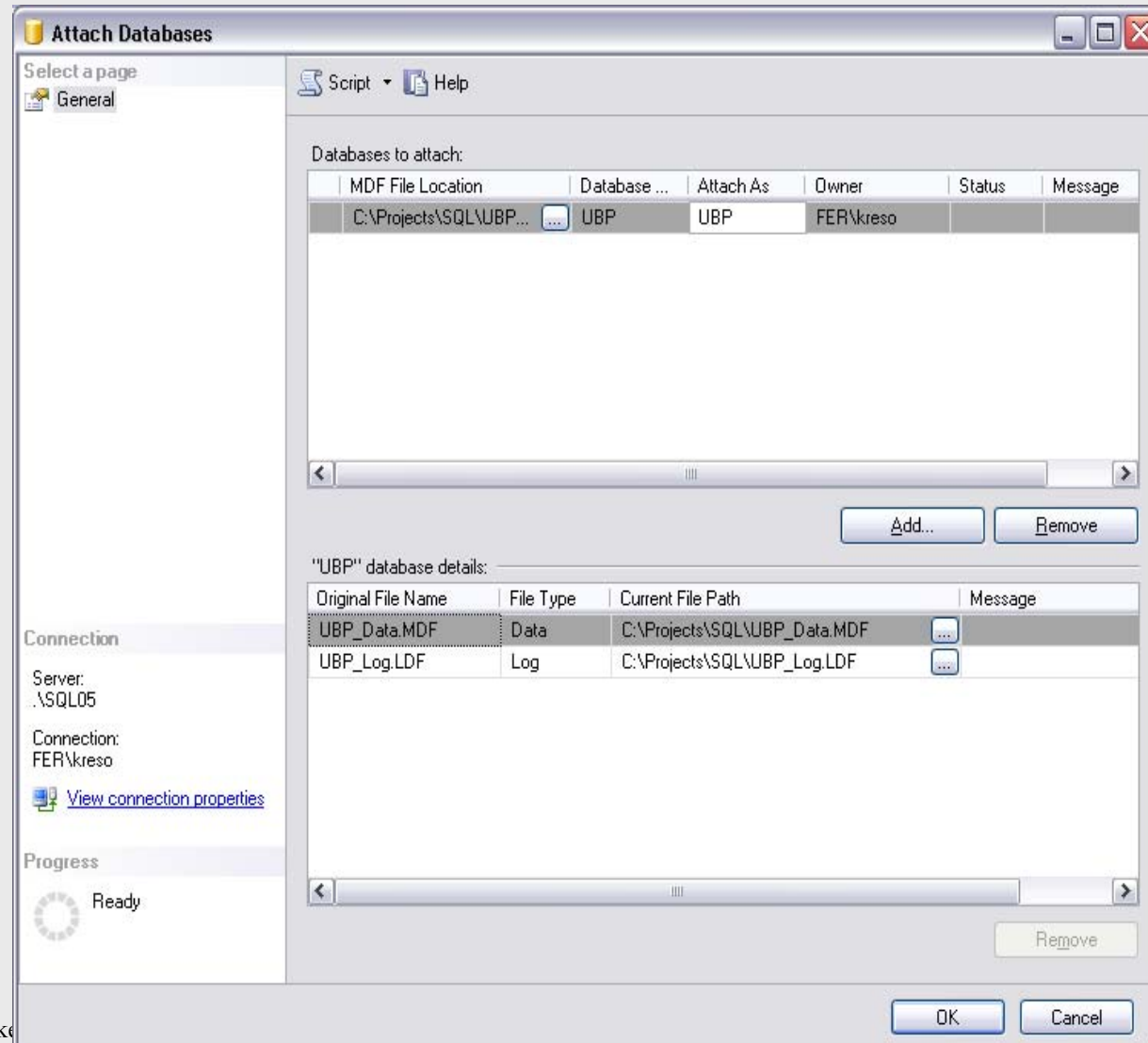
## ❑ Databases \ Attach

- fizičke datoteke stavlja pod kontrolu SUBP
- provjeriti Owner !

## ❑ Baza (desni klik) \ Tasks

## ❑ Detach - otkapčanje baze podataka

- BP prestane biti pod kontrolom sustava, datoteka više ne bude zaključana pa se može kopirati



# Izrada rezervnih kopija (backup)

**Back Up Database - Baza**

Select a page: **General** | Options

Script | Help

Source

Database: Baza

Recovery model: FULL

Backup type: Full

Backup component:

- ☒ Database
- ☐ Files and filegroups:

Backup set

Name: Baza-Full Database

Description:

Backup set will expire:

- ☒ After: 0
- ☐ On: 21. 2. 2009

Destination

Back up to: ☒ Disk ☐ Tape

c:\projects\sql\Baza.bak

Add... Remove Contents

Progress: Ready

OK Cancel

**Back Up Database - Baza**

Select a page: **General** | Options

Script | Help

Overwrite media

- ☒ Back up to the existing media set
- ☐ Append to the existing backup set
- ☒ Overwrite all existing backup sets

# Obnova, vraćanje baze podataka (restore)

**Restore Database - Baza**

Select a page  
General  
Options

Script Help

Destination for restore

Select or type the name of a new or existing database for your restore operation.

To database: Baza

To a point in time: Most recent possible

Source for restore

Specify the source and location of backup sets to restore.

☐ From database: Baza

☒ From device: C:\Projects\SQL\Baza.bak

Select the backup sets to restore:

Restore	Name	Component	Type	Server	Database
<input type="checkbox"/>	Baza-Full Database Backup	Database	Full	KLODOVIK\SQL05	Baza

Connection

Server: .\SQL05

Connection: FER\kreso

[View connection properties](#)

Progress

Ready

OK Cancel

# Pristup bazi podataka iz razvojnog okruženja

## ❑ SQL Server Management Studio

- Tables \ New Table - kreiranje tablice
- tablica \ desni klik – dizajn i rukovanje podacima
- New Query ili File – Open + Execute

## ❑ Administriranje korisnika

- poslužitelj \ Security \ Logins \ New Login ... (rppp, rppp, Firma)
- baza podataka \ Security \ Users \ New User ... (rppp, rppp, db\_owner)
- opcionalno, povezivanje User s Login
  - ALTER USER rppp WITH LOGIN=[rppp]

## ❑ Razvojno okruženje Visual Studio .NET

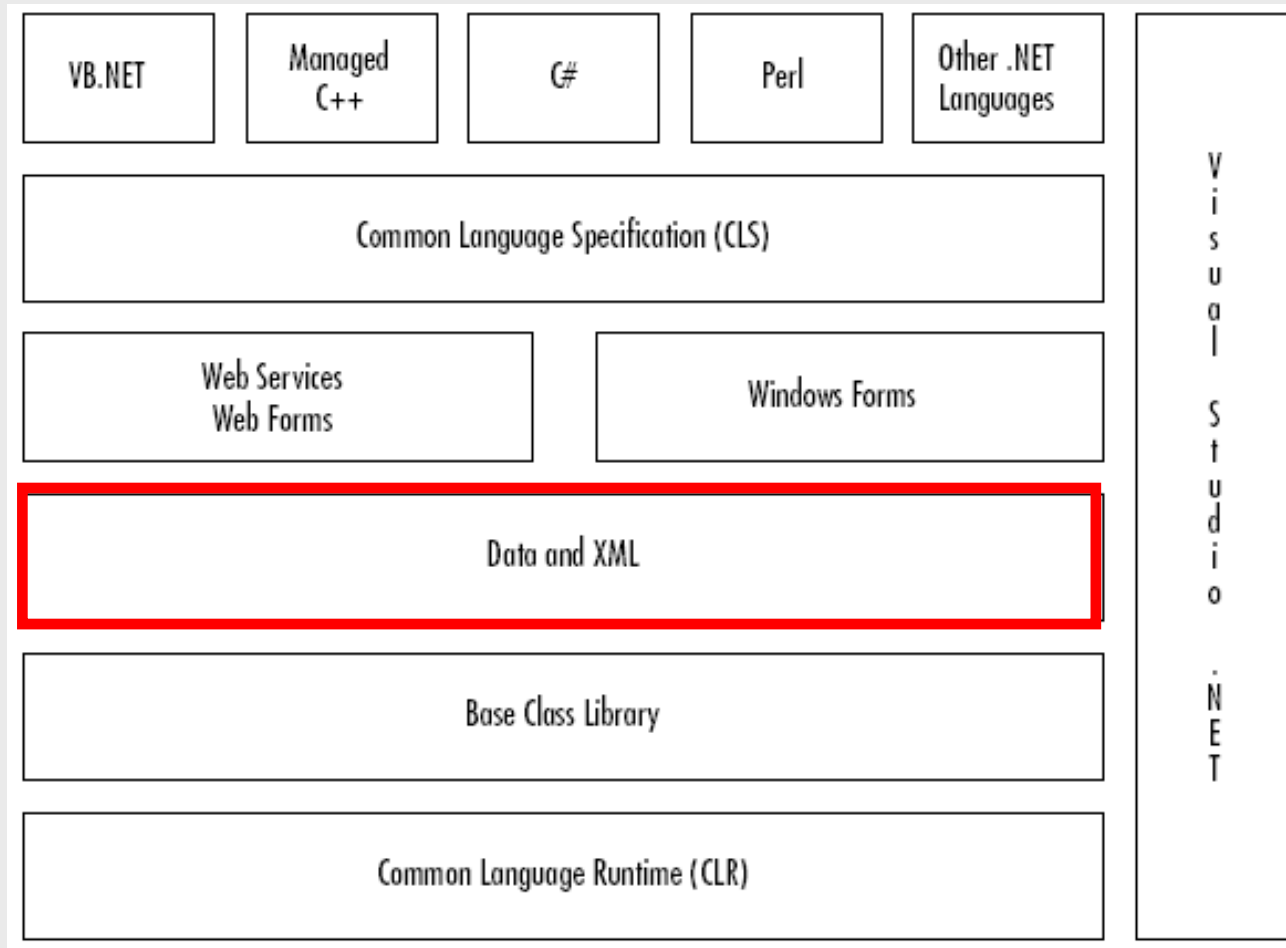
- Server Explorer \ Servers – rukovanje sistemskim servisima
- Server Explorer \ Data Connections – rukovanje bazom podataka i podacima
  - primjer: Add Connection, pa pogledati svojstva, objašnjenje slijedi

## ❑ Projekti sa skriptama za rad s bazom podataka

- New Project – Other project types \ DataBase, ..., Script (run) # Output  
slično:
- New Project – Visual C# \ Database \ SQL Server, ..., test.sql (run) # Output

# ADO.NET i .NET Framework

- ❑ **ActiveX Data Objects .NET (ADO.NET)** je .NET Framework tehnologija za rukovanje podacima.



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

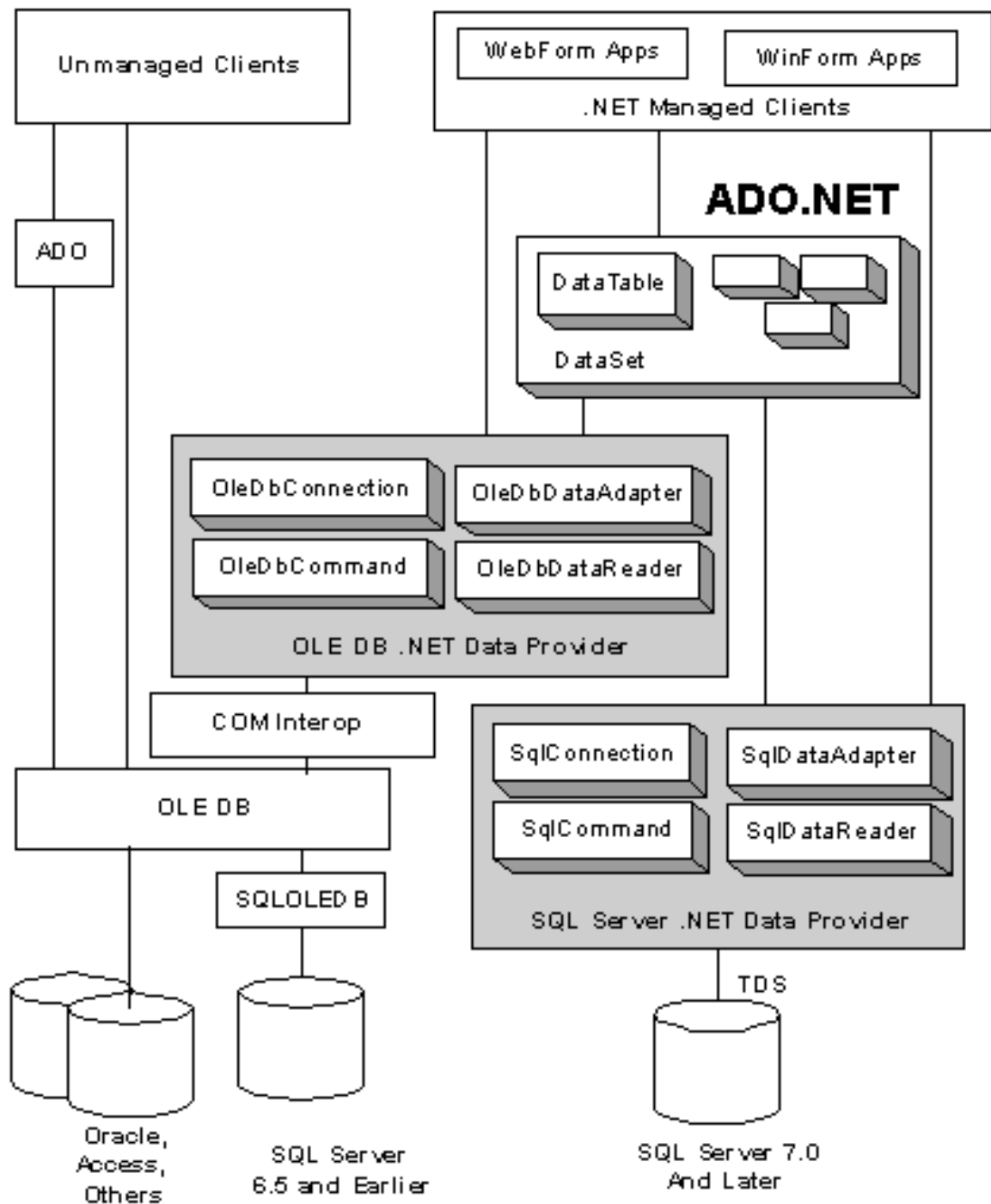
# Pohrana podataka

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

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

# Arhitektura

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





# Opskrbljivači, 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 davatelj za rad s bilo kojim OLE Database (OLE DB) izvorom
    - npr: Oracle, MS JET, SQL OLE DB
  - Razredi: *OleDbCommand*, *OleDbConnection*, *OleDbDataReader*, *OleDbDataAdapter*
- ❑ Navedeni razredi implementiraju zajednička sučelja pa 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

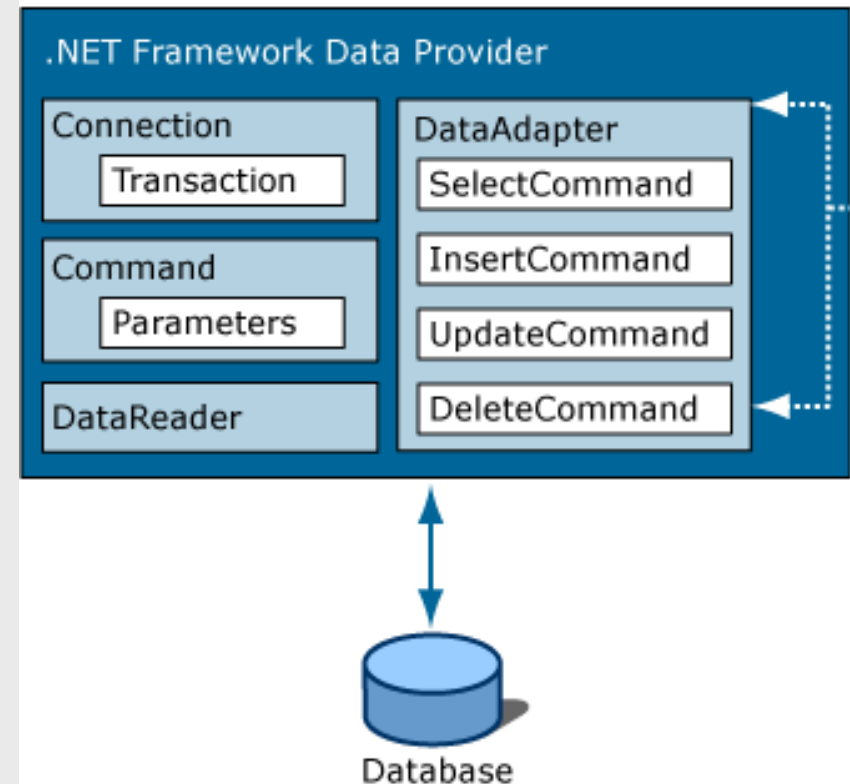
- Parametar parametrizirane SQL naredbe ili pohranjene procedure

## ☐ Transaction

- Nedjeljiva grupa naredbi nad podacima

## ☐ DataAdapter

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



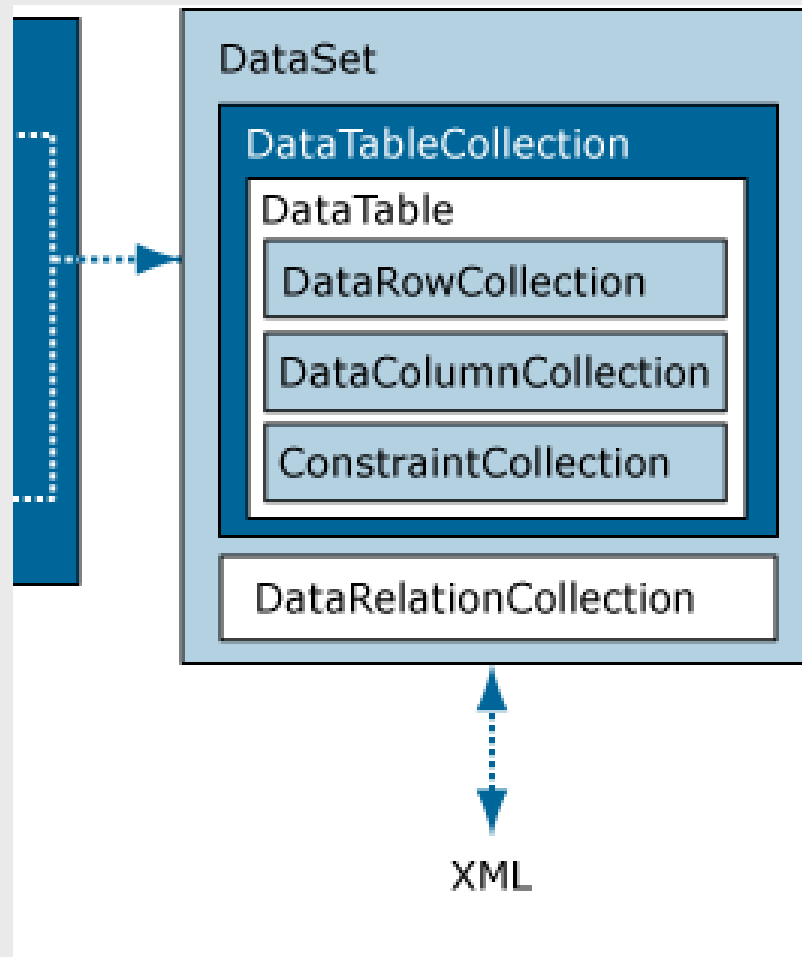
# Struktura i smještaj podataka

## ❑ 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

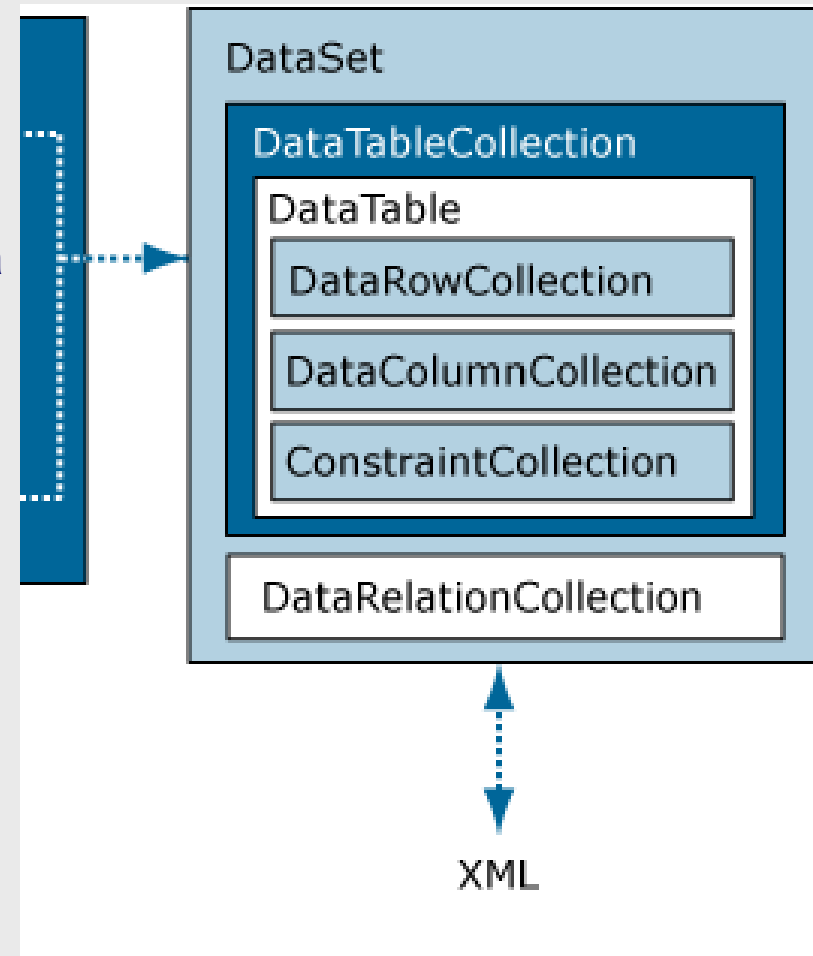
- definira stupce u `DataTable`

## ❑ DataRelationCollection

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

## ❑ DataViewManager

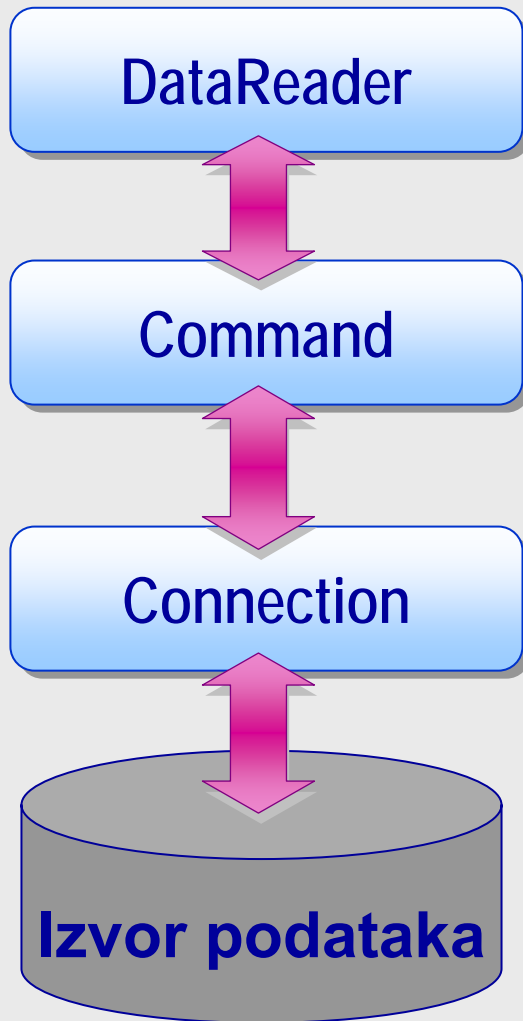
- definira poglede nad skupovima podataka



# Izravna obrada podataka

---

# Izravna obrada podataka na poslužitelju



## ❑ 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)

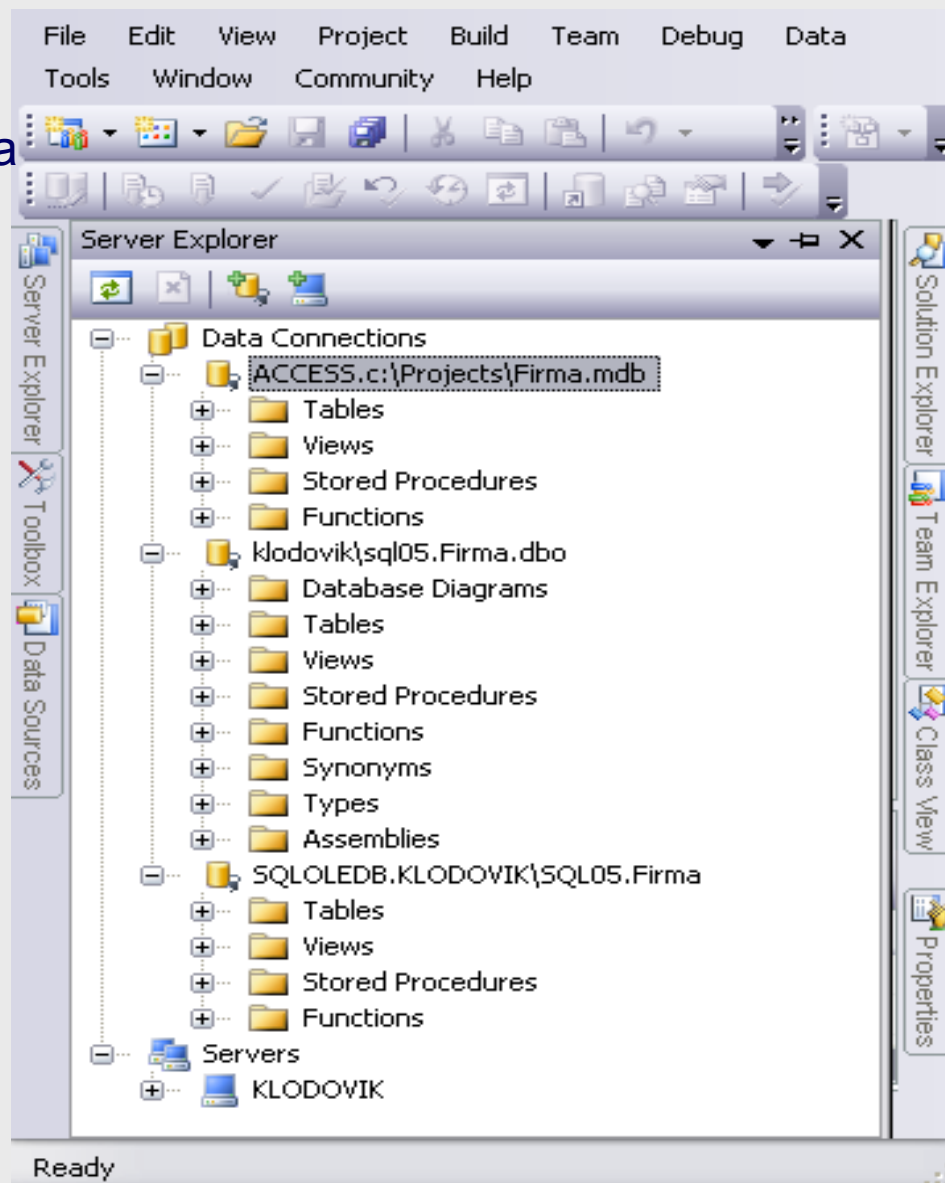
- veza s fizičkim izvorom podataka
- otvara i zatvara vezu s izvorom
- omogućuje transakcije

## ❑ 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` – prikapćanje na izvor podataka
- `Close` - otkapćanje 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`

`AttachDBFilename`

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 `Initial Catalog`.

`ConnectTimeout`

Vrijednost svojstva `Timeout` – čekanje do otvaranja ili iznimke

`Data Source`

Naziv samostojne baze podataka (npr. MS Access .MDB), naziv poslužitelja ili mrežna adresa poslužitelja baze podataka. Na lokalnom računalu može se koristiti naziv `local` ili `".."`.

`Initial Catalog /  
Database`

Naziv baze podataka.

`Integrated Security`

Postavlja se na `false` (default), `true` 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.

`Persist Security Info`

`True` ili `false` (default). Kad je postavljen na `false`, sigurnosno osjetljive postavke (npr. lozinka) se automatski uklanjaju iz `ConnectionString` nakon što je konekcija otvorena.

`User ID / UID`

Identifikator korisnika (korisničko ime) u bazi podataka.

`Password/PWD`

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 – NekeKonekcije.txt

### ■ OleDbConnectionString 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`

# Uspostavljanje veze s izvorom podataka

## ❑ Primjer: ADO\Upitnik

```
string connString = "";

OleDbConnection oleDbConnection;
SqlConnection sqlConnection;
IDbConnection konekcija; // sučelje

...

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

# Zadatak za vježbu

- ❑ Doraditi primjer obradom događaja i prikazom stanja 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. neka od naredbi `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 (gumb Obavi) – NekiUpiti.txt

```
// 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 – NekiUpiti.txt

```
// 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.

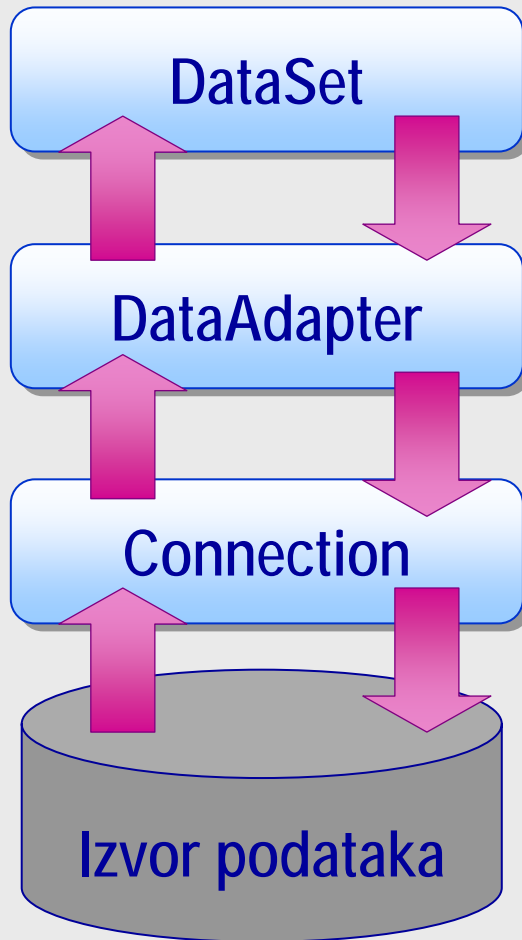
```
while (reader.Read())
{
    Object [] cols = new Object[10] ;
    reader.GetValues( cols );

    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

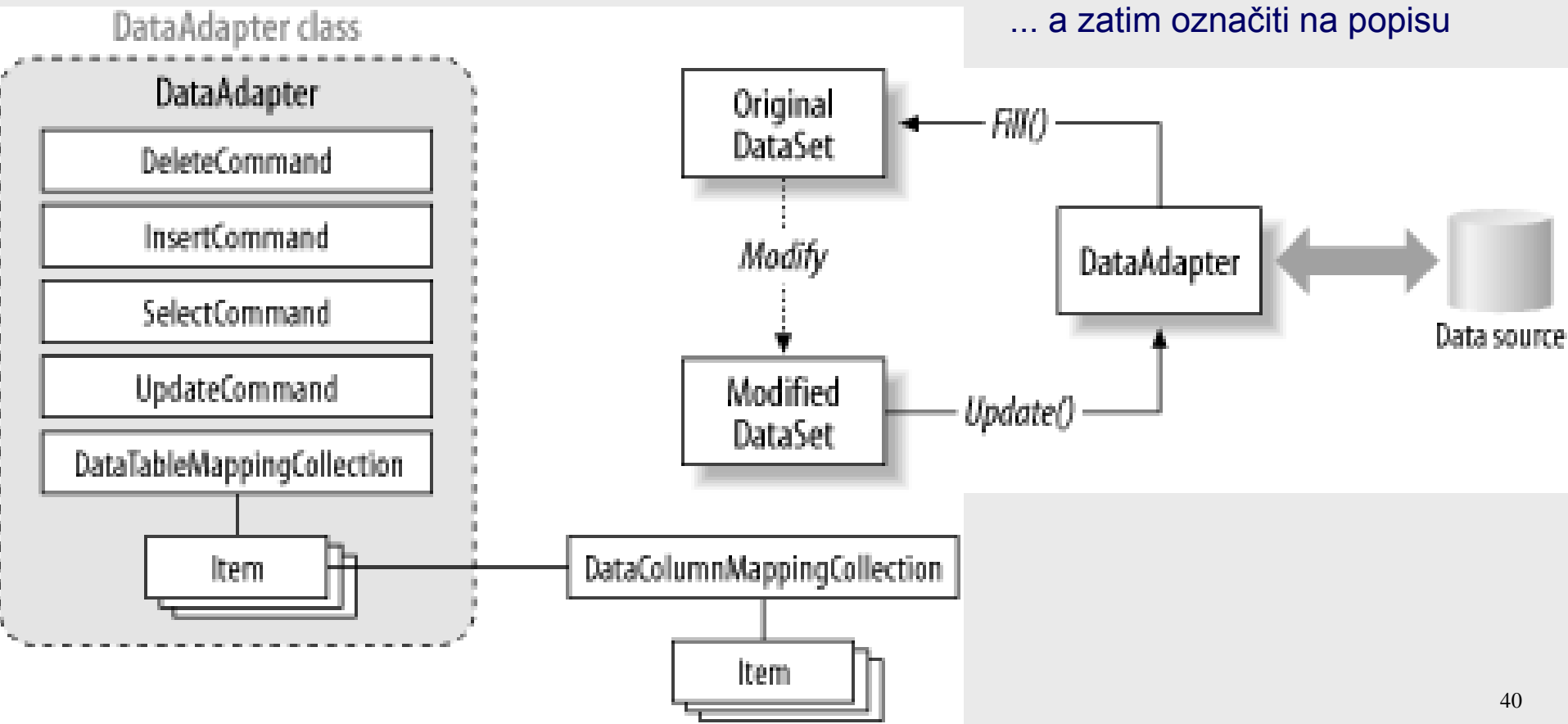
# IDataAdapter

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

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

## ❑ Toolbox \ Data (drag-drop na formu)

- **SqlDataAdapter**, SqlConnection, SqlCommand (analogno OleDb)
  - ukoliko se komponente ne vide treba ih omogućiti desnim klikom na Toolbox \ Data \ Choose Items ... a zatim označiti na popisu





# ***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 osvježениh 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 osvježениh zapisa
  - `dra` - polje `DataRow` objekata koji se usklađuju s izvorom

# DataAdapter primjeri

## ❑ Primjer:

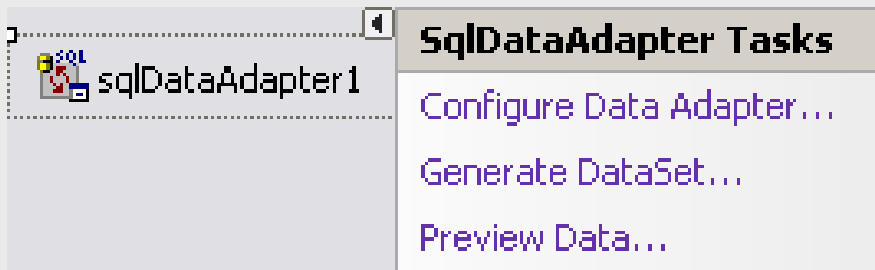
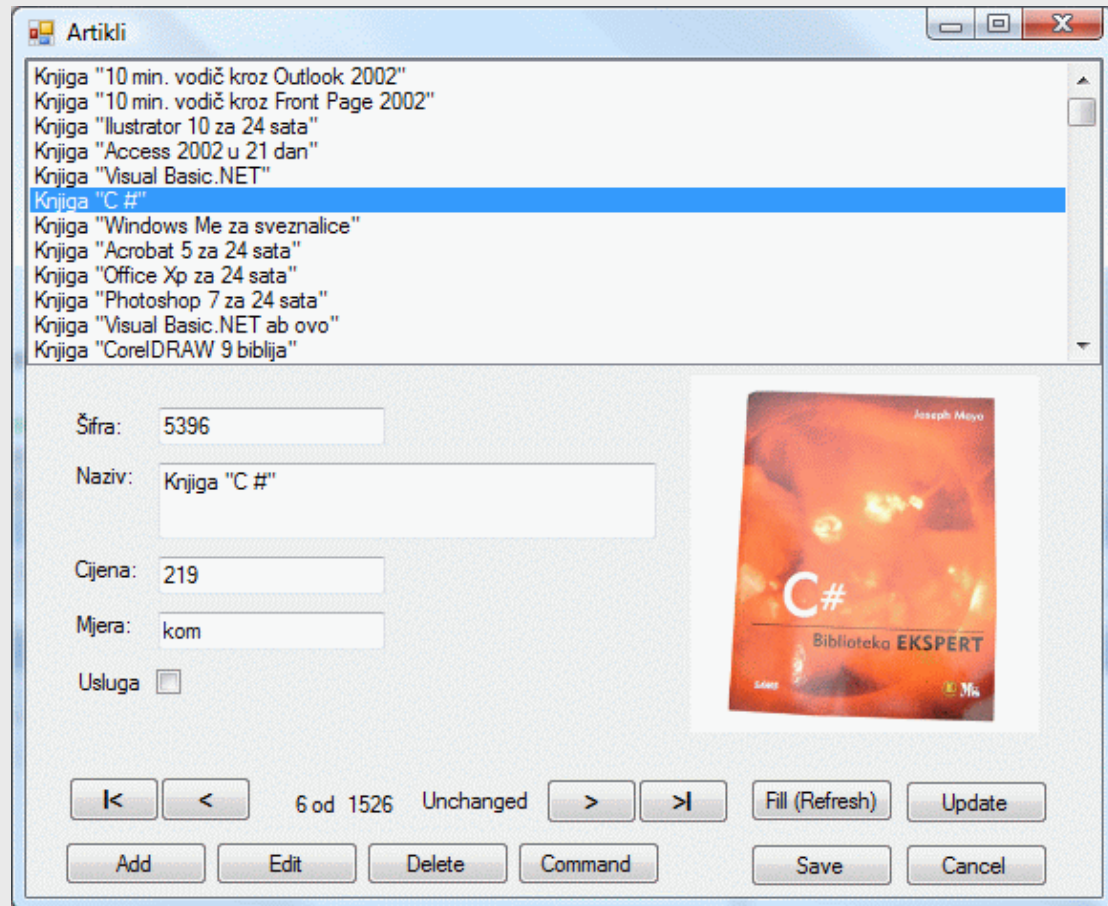
ADO\Artikli

## SqlDataAdapter1

## ❑ Properties

## ❑ Tasks (trokutić gore desno)

- Configure Data Adapter,
- Generate DataSet,
- Preview Data



# DataAdapter primjeri

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

- svojstva i kreiranje članova *DataAdapter* objekta

```
this.sqlDataAdapter1 =  
    new System.Data.SqlClient.SqlDataAdapter();  
this.sqlSelectCommand1 = new  
System.Data.SqlClient.SqlCommand();  
...  
this.sqlDataAdapter1.SelectCommand =  
    this.sqlSelectCommand1;  
...  
this.sqlSelectCommand1.CommandText =  
    "SELECT Artikl.*" +  
    "FROM Artikl";  
this.sqlSelectCommand1.Connection =  
    this.sqlConnection1;  
...
```

## ❑ 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

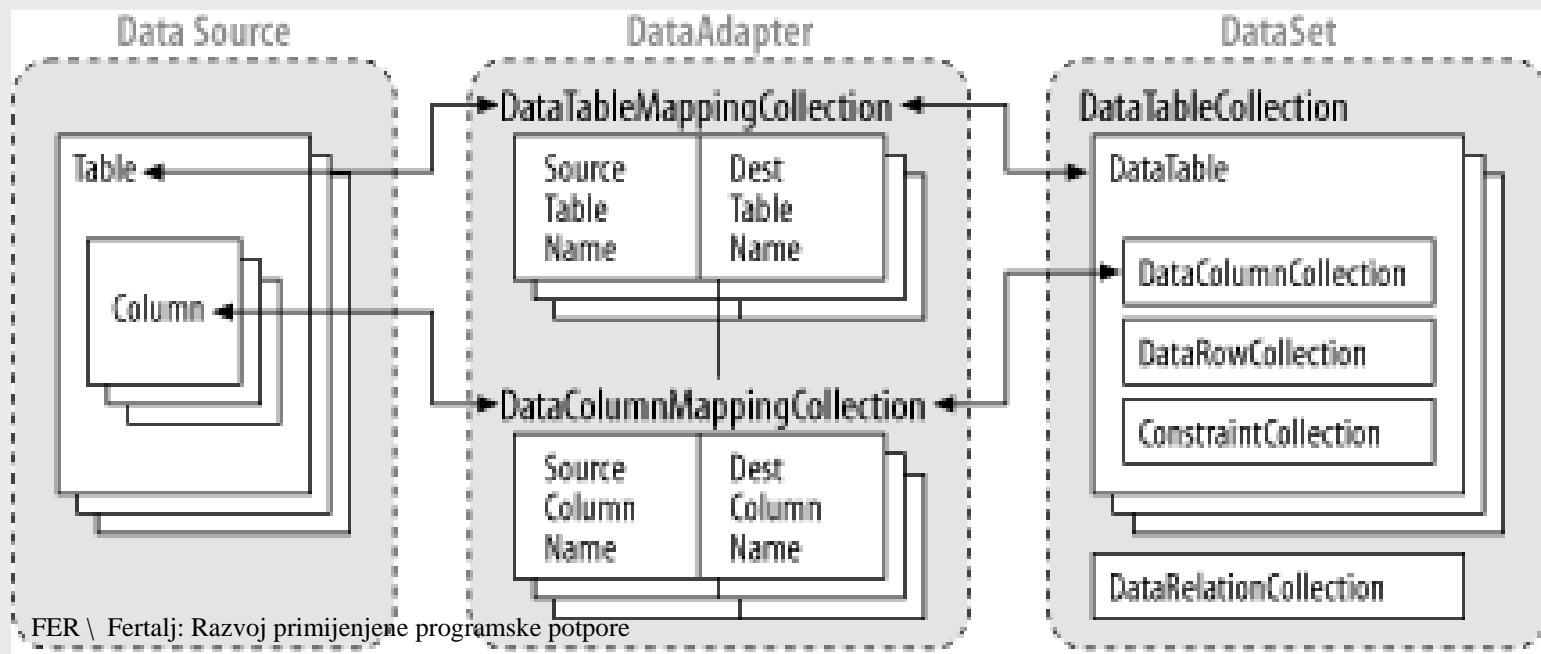
# Ostali *DataAdapter* članovi

## ❑ 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)`



# Konfiguracijska datoteka

- ❑ Konfiguracijska datoteka (app.config) je XML datoteka koja sadrži postavke specifične za aplikaciju koja je koristi
  - appSettings (add-key-value)
  - connectionSettings (add-name-connectionString)
- ❑ Primjer  **ADO\Artikl – App.config**

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key="ApplicationTitle" value="Artikli" />
  </appSettings>
  <connectionStrings>
    <add name="FirmaConnectionString"
      connectionString="Data Source=.\SQLEXPRESS;
      Initial Catalog=Firma;Integrated Security=True"/>
  </connectionStrings>
</configuration>
```

# Čitanje postavki konfiguracije

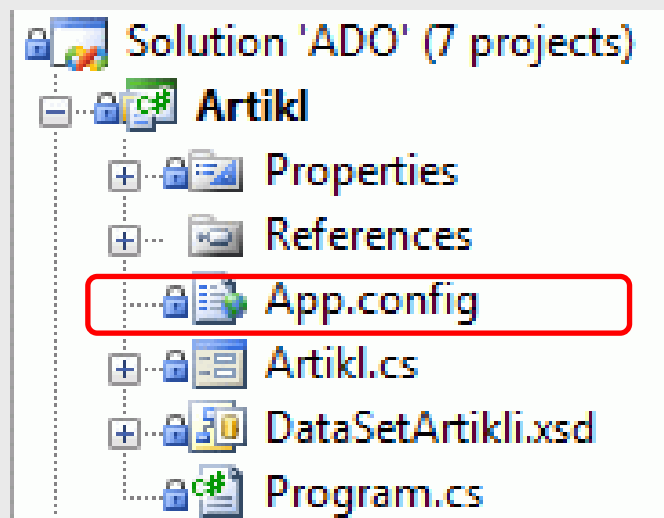
## ❑ Primjer ADO\Artikl – Artikl.cs (Form\_Load)

- Dohvat aplikacijskih postavki

```
this.Text =  
ConfigurationManager.AppSettings["ApplicationTitle"];
```

- Dohvat *connection string-a*

```
sqlConnection1.ConnectionString =  
ConfigurationManager.ConnectionStrings  
["FirmaConnectionString"].ConnectionString;
```

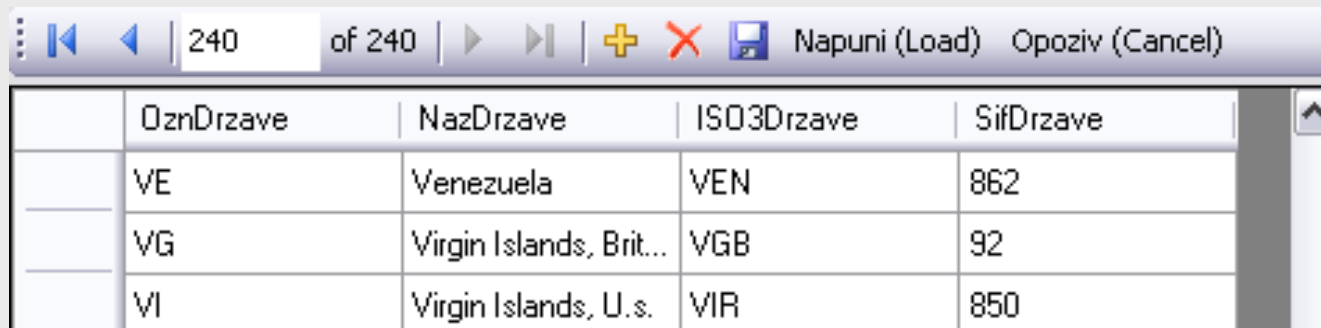




# TableAdapter

- ❑ **TableAdapter** - Razred / komponenta, koja ućahuruje adapter, konekciju i druge 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

- ❑ **Kako nastane ?**

- tool Data Sources \ Add ili izbornik Data \ Add New Data Source
- tool Data Sources \ drag-drop tablice na formu
- Napomena: Dodavanjem *TableAdapter*-a, automatski se aplikaciji dodaje app.config datoteka

- ❑ **Primjeri (u dizajnu)**

- firmaDataSet \ EditInDesigner ... – DrzavaTableAdapter Properties

# TableAdapter

## ❑ Primjer: ADO\Drzava – FirmaDataSet.Designer.cs (Source)

```
namespace Drzava.FirmaDataSetTableAdapters {  
    ...  
    public partial class DrzavaTableAdapter  
        : System.ComponentModel.Component {  
  
        private System.Data.SqlClient.SqlDataAdapter _adapter;  
        private System.Data.SqlClient.SqlConnection _connection;  
        private System.Data.SqlClient.SqlCommand[] _commandCollection;  
  
        private void InitAdapter() {
```

## ❑ 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 – Properties*
- potražiti "provider", "DataAdapter" i "DataSet" u programskom kodu

# Skupovi podataka

---

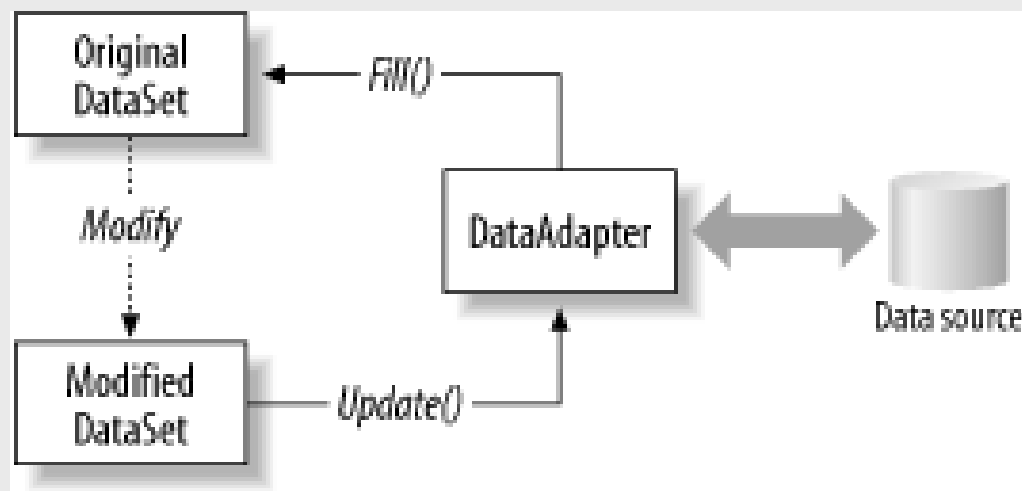
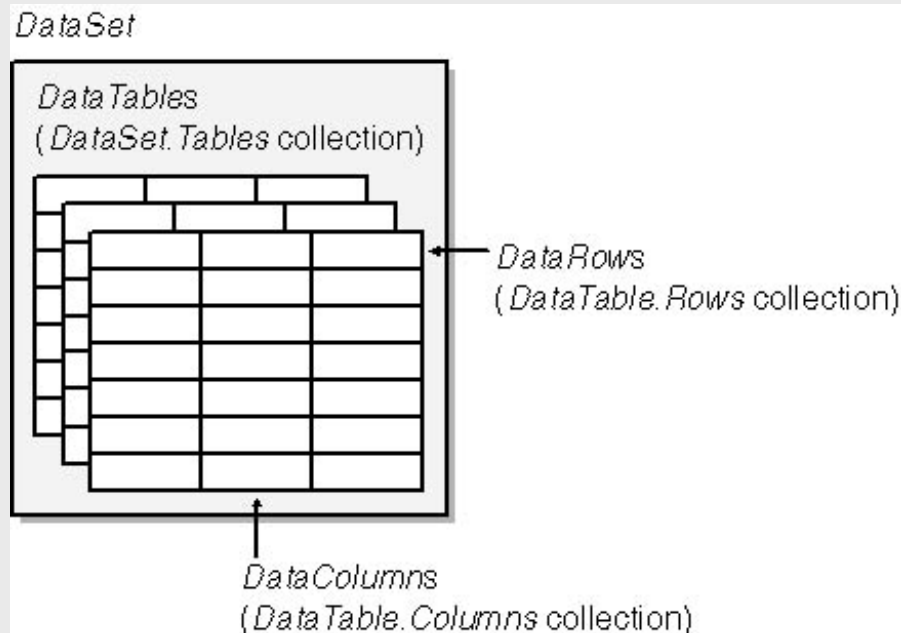
# DataSet

## ❑ 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`

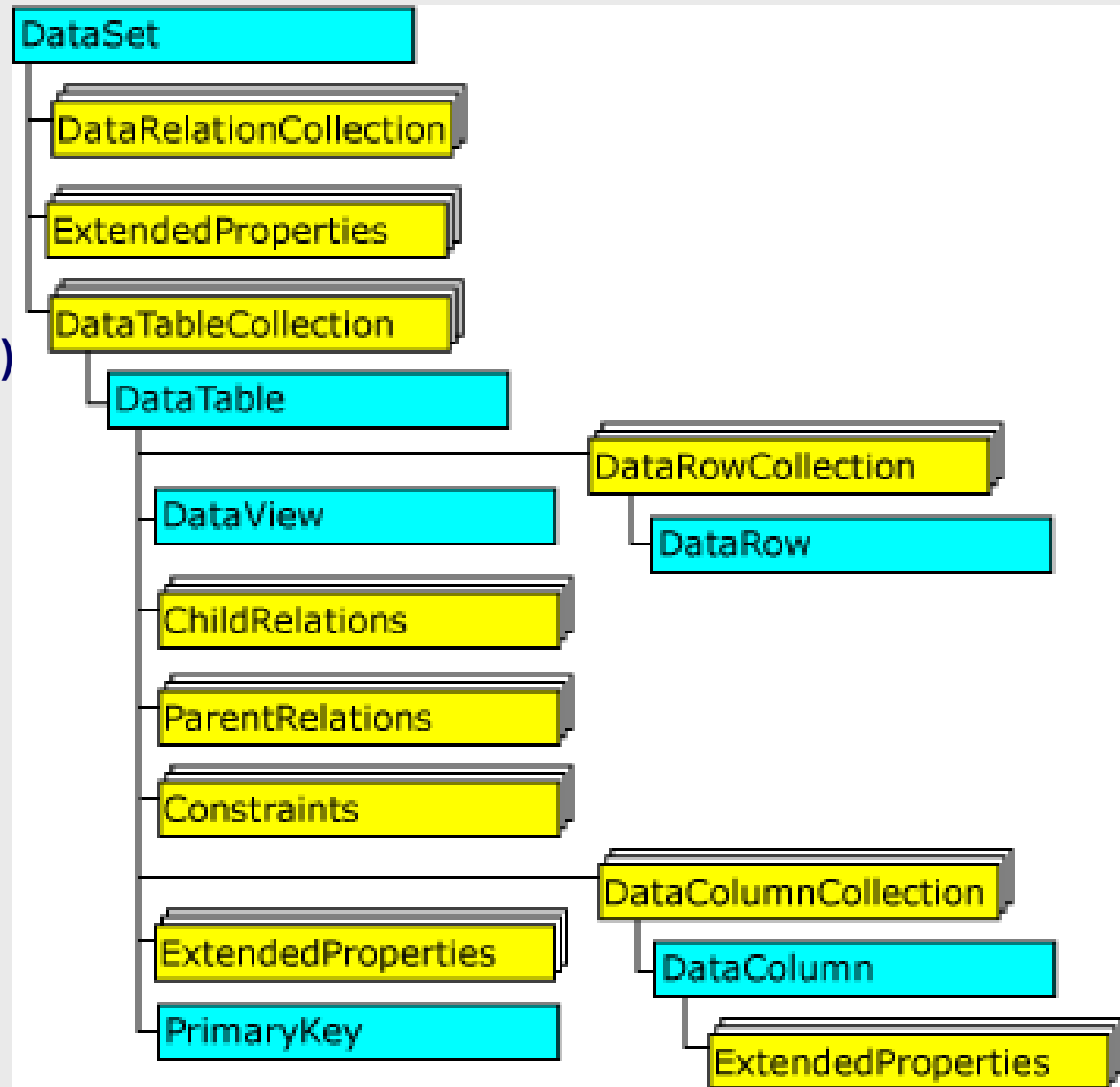
# *DataSet* članovi

## ☐ Izbornik Data

- Add ... – stvara DataSet
- Show Data Sources ...
- Preview Data

## ☐ Dizajn forme (već viđeno)

- Toolbox \ Data  
dovlačenje DataSet na formu
- DataAdapter: Generate DataSet



# *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` (primjer `Drzava`)

## ❑ Primjeri: 📁 `Drzava`

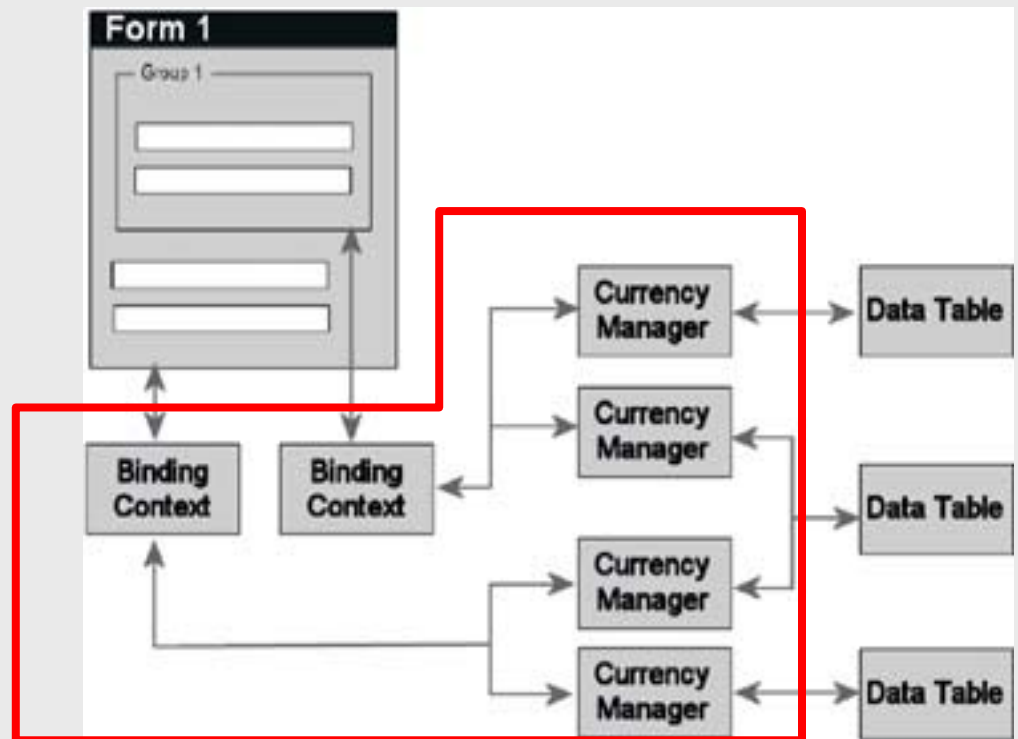
- `this.firmaDataSet = new Drzava.FirmaDataSet();`
- ... `Fill`, `Update`, `RejectChanges`

# 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 (Toolbox\Data)**

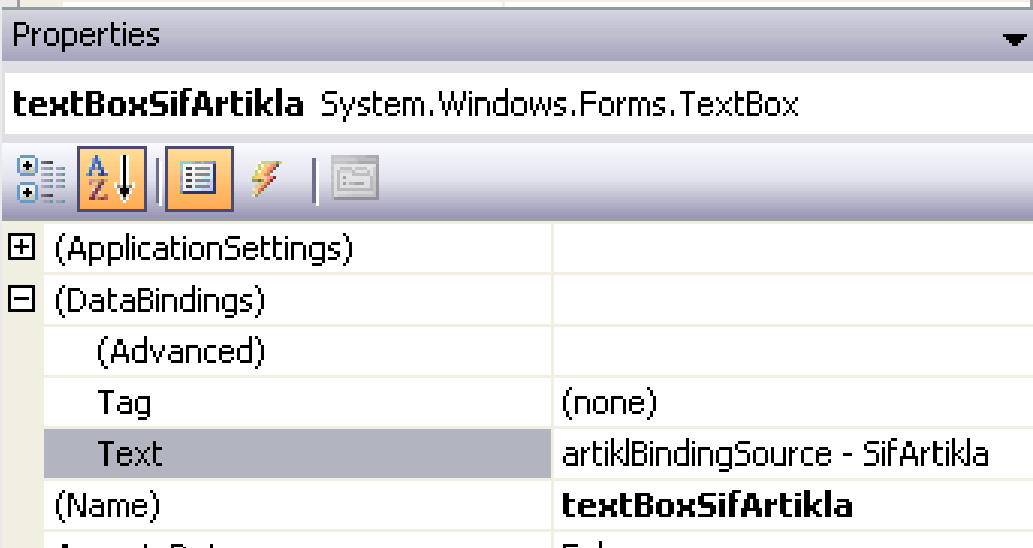
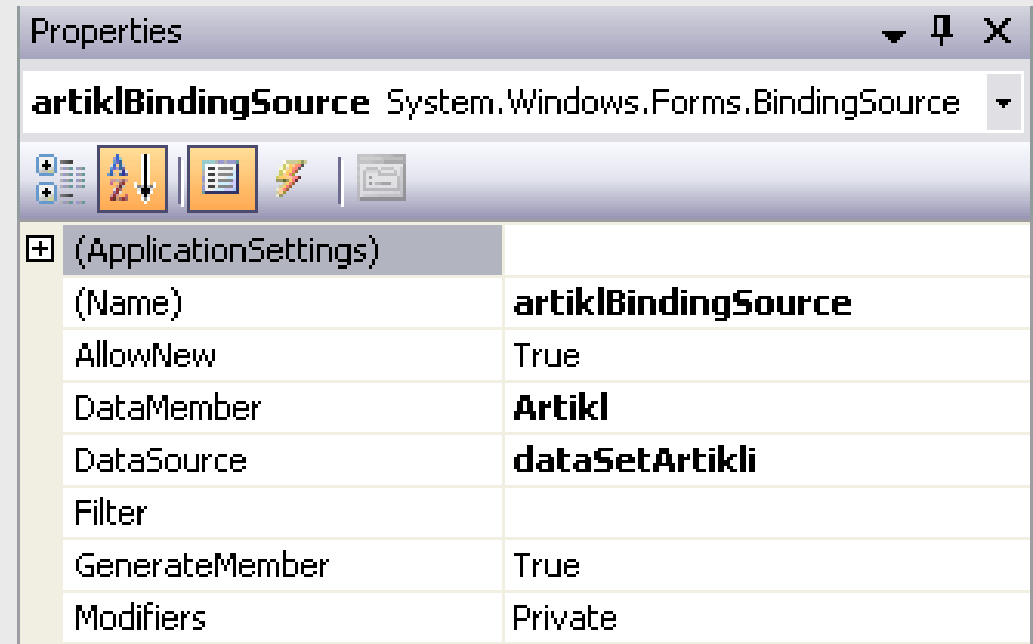
- Učahuruje izvor podataka i prati trenutnu poziciju zapisa
- Ima funkcionalnosti *BindingContext* i *CurrencyManager*

## ❑ **Glavni članovi**

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

## ❑ **Primjer: ADO\Artikl**

- 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"

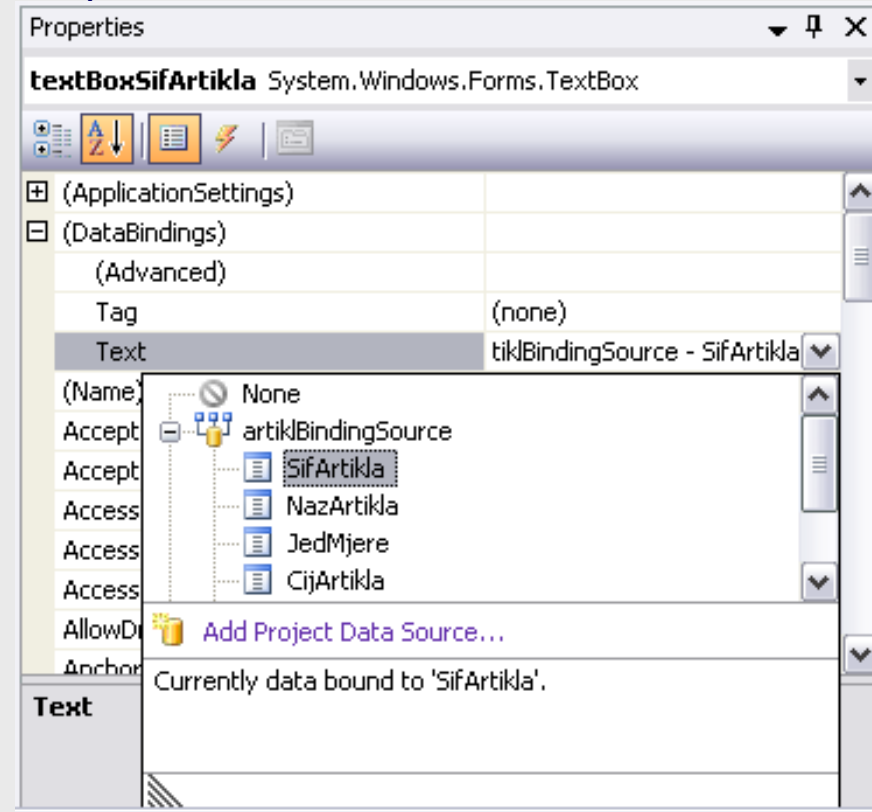
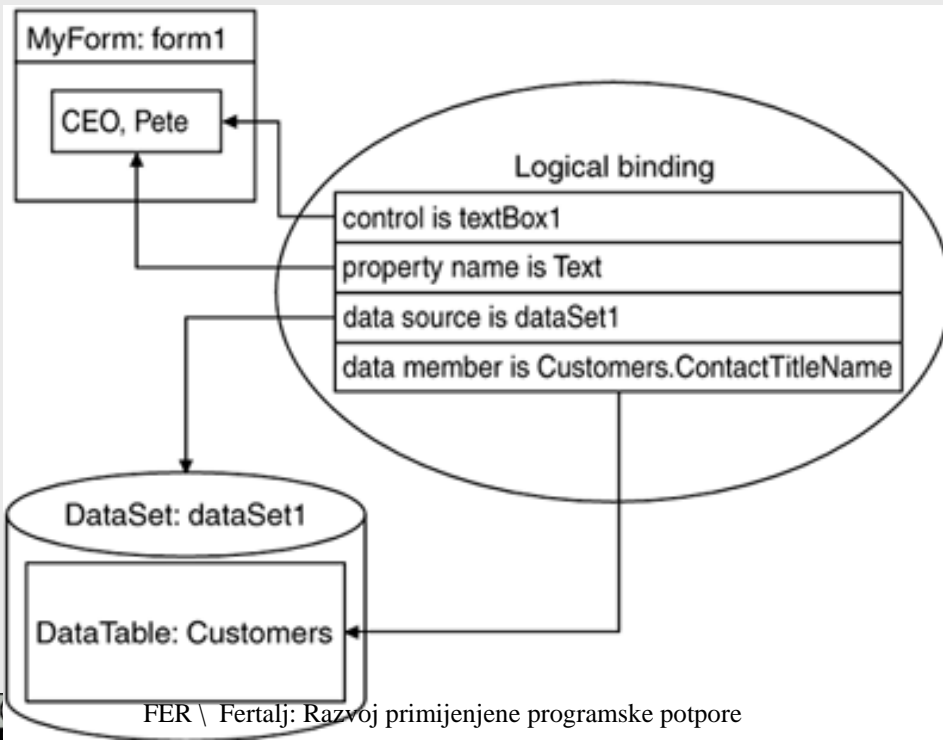
## Jednostavno povezivanje (Simple Binding)

## ☐ Povezivanje kontrole koja prikazuje jednu vrijednost

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

## ❑ Primjer: ADO\Artikl

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



# Jednostavno povezivanje dinamički

## ❑ Primjer: ADO\Artikl

- povezati neka svojstva u dizajnu a neka dinamički

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

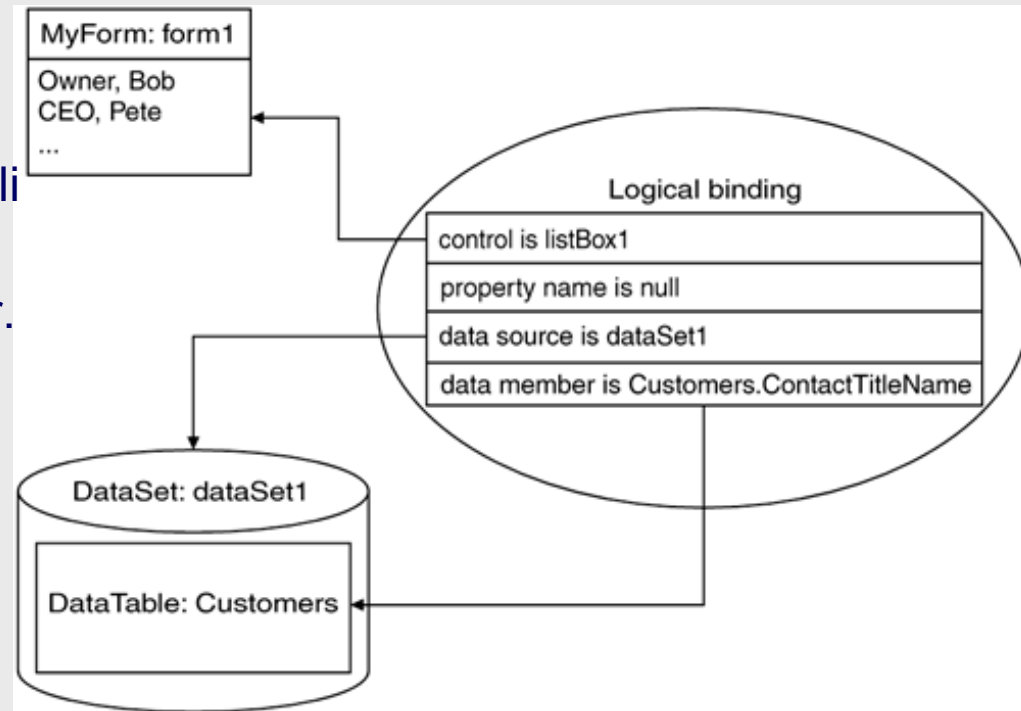
- `control.DataBindings` – poveznice kontrole
- Binding – pojedinačna poveznica svojstvo – izvor - član/podatak
  - `DataBindings[ property ]`
  - `public Binding(string propertyName, Object dataSource, string dataMember);`



# Složeno povezivanje (Complex Binding)

## ❑ Povezivanje složene kontrole s više redaka, postavljanjem

- DataSource – izvor podataka, npr. `artiklBindingSource` ili `dataSetArtikli.Artikl`
- DisplayMember – atribut, npr. `NazivArtikla`
  - kontrole `ListBox` i `ComboBox` – jedna vrijednost za redak
- DataMember – čitava tablica, npr. "Drzava"
  - kontrola `DataGridView` – više vrijednosti za jedan redak (primjer ADO\Drzava)



## ❑ Primjer: ADO\Artikl (BindData)

```
listBoxArtikli.DataSource = artiklBindingSource;  
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`

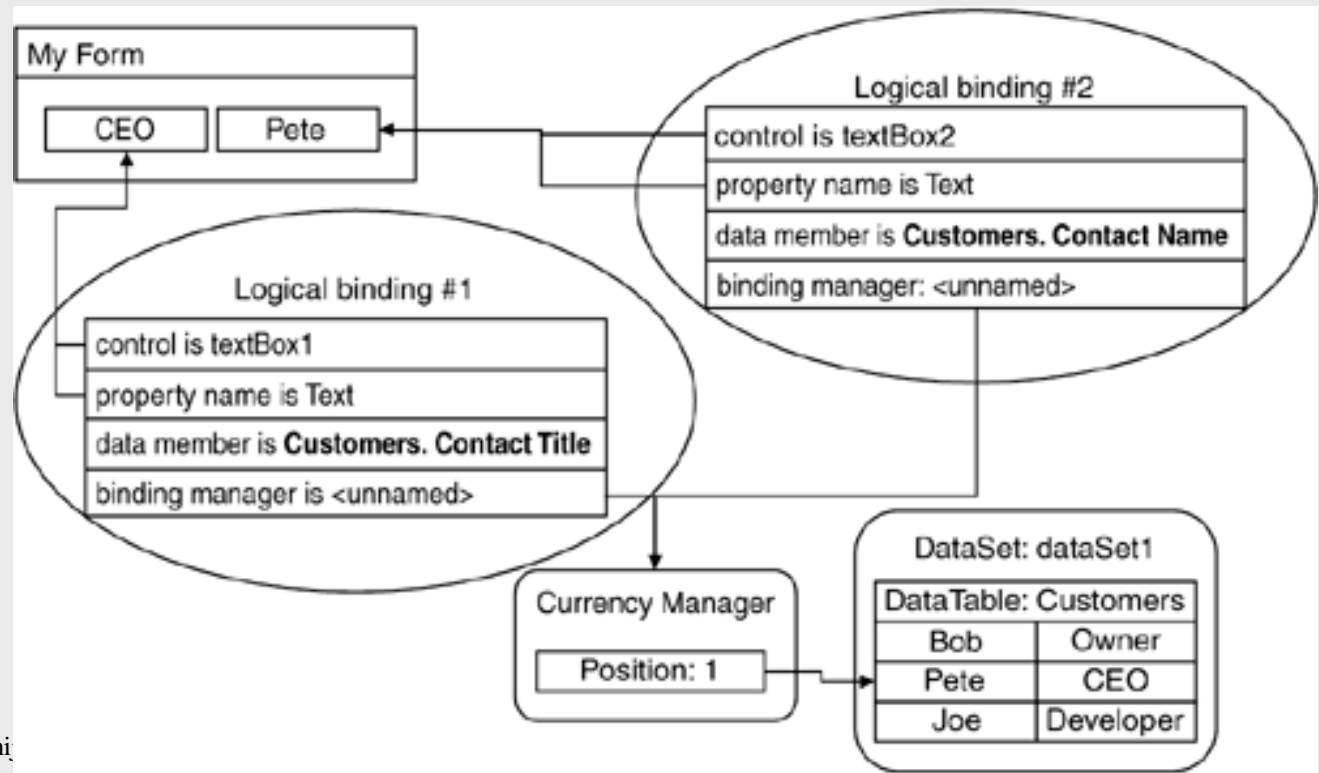
## ❑ Postupci

- `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

The screenshot shows a window titled "Artikli" with a list of books. The selected item is "Knjiga 'C #'". Below the list is a form with fields for "Šifra" (5396), "Naziv" (Knjiga "C #"), "Cijena" (219), "Mjera" (kom), and "Usluga" (unchecked). To the right is a book cover image. At the bottom are navigation buttons: "K" (first), "<" (previous), ">" (next), and ">" (last), along with status text "6 od 1526 Unchanged". Other buttons include "Add", "Edit", "Delete", "Command", "Fill (Refresh)", "Update", "Save", and "Cancel".

Šifra	Naziv	Cijena	Mjera	Usluga
5396	Knjiga "C #"	219	kom	<input type="checkbox"/>

# 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();  
...  
}
```

```
void UpdateDisplay() { // pozicija, status, lista, slika  
    labelPosition.Text =  
        ((artiklBindingSource.Position + 1).ToString() ...  
    labelRowState.Text = GetCurrentRow().RowState.ToString();  
    listBoxArtikli.SelectedIndex=artiklBindingSource.Position;  
    ShowPicture();  
}
```

# Navigacija komponentom *BindingNavigator*

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

### ■ Svojstva:

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

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

```
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 (repetitio est mater studiorum)

## ❑ 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

# Rukovanje zapisima u skupu podataka

□ **Primjer:**  
📁 **ADO\Artikli**

Unos artikala

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

Šifra:

Naziv:

Cijena:

Mjera:

Usluga ☐

<K < 1549 od 1549 Added > >I Fill (Refresh) Update

Add Edit Delete Command Save Cancel



# Rukovanje zapisom (*DataRow*, *DataRowView*)

## ❑ **DataRow** – redak, zapis u DataTable

- `HasError` – zastavica pogreške
- `RowError` – korisnički definirano objašnjenje pogreške
- `RowState` – stanje zapisa u odnosu na zadnji `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` – postupci ažuriranja
- `Delete` – briše redak
- `RejectChanges` – poništava još nepotvrđene promjene nad podacima

## ❑ **DataRowView** – redak pogleda, referenca na DataRow

- `Row` – aktualni `DataRow`, izvorni zapis

## ❑ **Primjer: adresiranje retka u metodi GetCurrentRow**

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

# Dodavanje retka na kraj skupa podataka

## ❏ Primjer: ADO\Artikl

```
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);

    // podesimo poziciju i izazovemo UpdateDisplay();
    artiklBindingSource.Position = artiklBindingSource.Count-1;
}
```

# Izmjena retka u skupu podataka

❑ **Primjer:**  **ADO\Artikl**

❑ **Uređivanje zapisa se provodi promjenom sadržaja vezanih kontrola**

- podaci se spremaju u DataSet klikom na drugu kontrolu, prelaskom na neki drugi zapis ili posebnom metodom (naredna folija)

❑ **Alternativno, promjena se obavlja promjenom vrijednosti atributa**

```
private void buttonEdit_Click(object sender, EventArgs e)
{
    System.Data.DataRow currRow;
    currRow = GetCurrentRow();
    currRow["NazArtikla"] += DateTime.Now.ToString();
    UpdateDisplay();
}
```

❑ **U oba slučaja podaci nisu izmijenjeni na izvoru!**

- status retka u skupu podataka je `Modified`

# 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)
{
    DataRow row = ((DataRowView)artiklBindingSource.Current).Row;
    row.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 uklonjen 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.  

```
sqlDataAdapter1.Update(dataSetArtikli.Artikl);
```

  - postupak izvršava `DeleteCommand` objekt adaptera

## ❑ Redoslijed akcija

- `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)
{
    // završava prethodno započeto uređivanje podataka
    artiklBindingSource.EndEdit();

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

        if (dataSetArtikli.HasChanges())
        {
            sqlDataAdapter1.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.SqlClient.SqlCommand cmdUpdate;
    System.Data.DataRow currRow = GetCurrentRow();

    cmdUpdate = new SqlCommand("UPDATE Artikl "
        + " SET NazArtikla = '" + currRow["NazArtikla"]
        + " " + DateTime.Now.ToString()
        + "' WHERE SifArtikla=" + currRow["SifArtikla"]
        , sqlConnection1);

    this.sqlConnection1.Open();
    cmdUpdate.ExecuteNonQuery();
    this.sqlConnection1.Close();

    this.sqlDataAdapter1.Fill(dataSetArtikli.Artikl);
}
```

# 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 – ugradnja rukovatelja u DataSetArtikli.cs

```
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 `errorProviderArtikl` koja prikazuje pogreške s izvora `artiklBindingSource`**
- ❑ **Primjer**
  - `errorProviderArtikl.DataSource = artiklBindingSource`

# 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", "");  
}
```

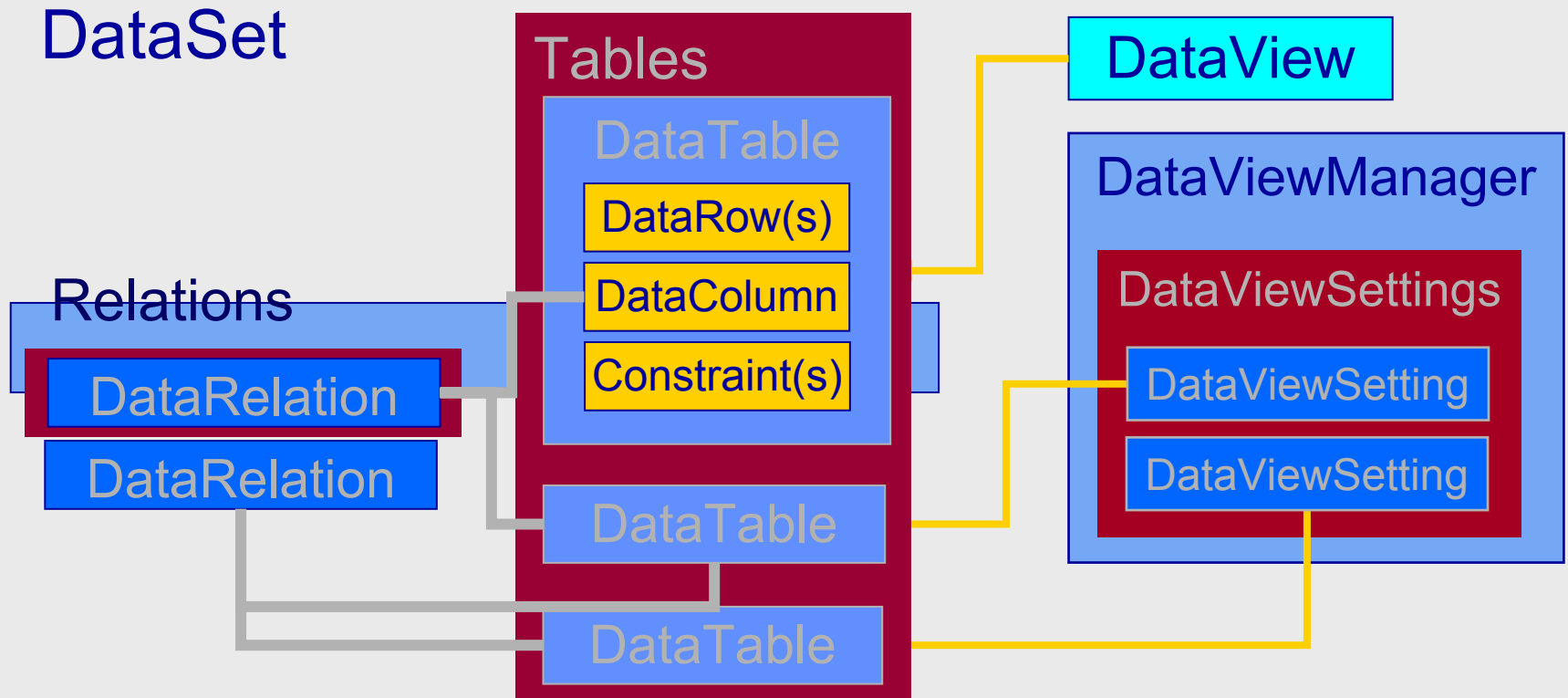
# Pogledi

---

# Pogled na podatke

## □ Glavni razredi

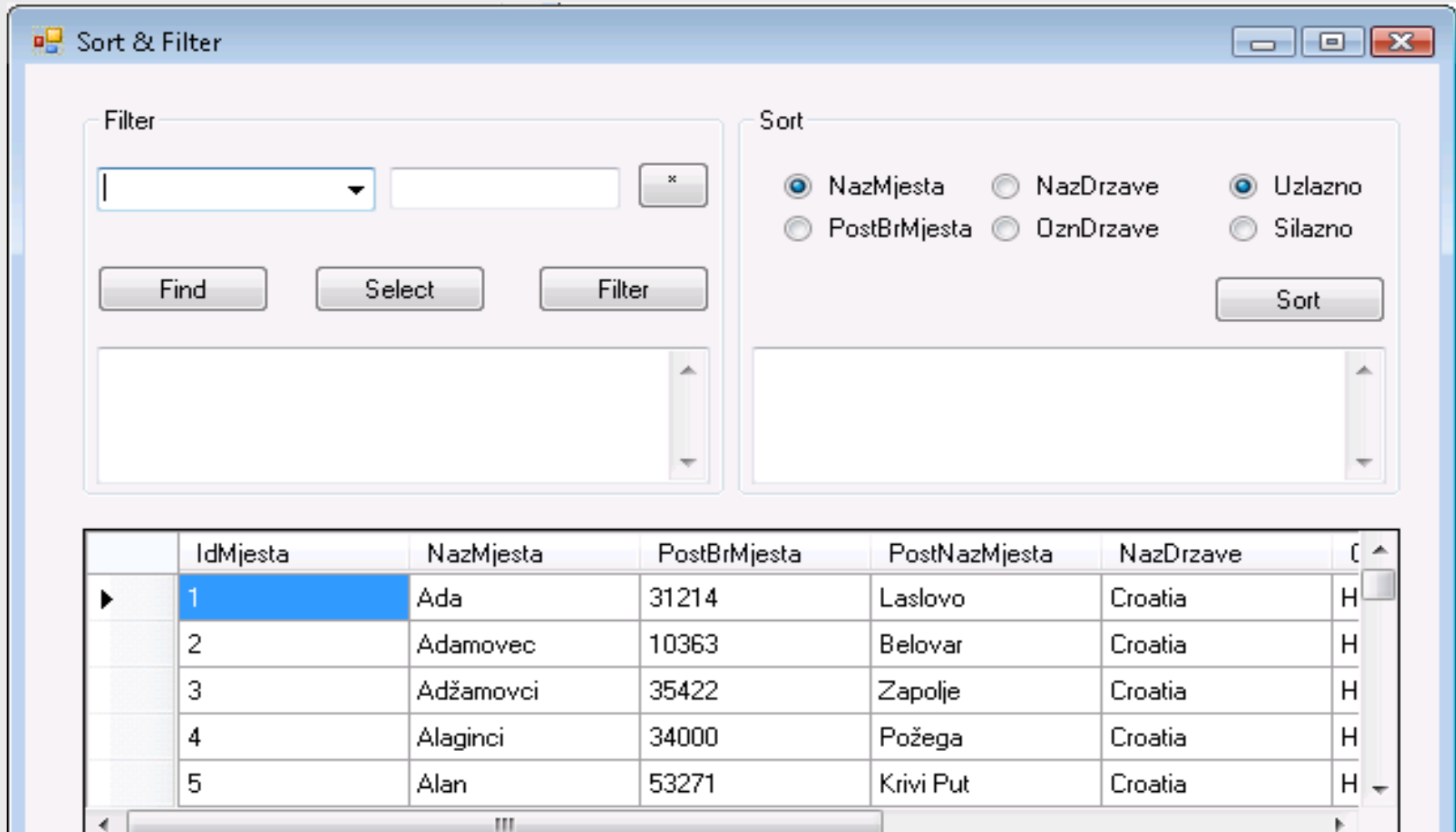
- `DataRow` – pogled na podatke
- `DataManager` – spremnik pogleda, kojima se pristupa preko `DataManagerSettings` kolekcije (analogija s `DataSet` spremnikom)



# Primjer ugradnje pogleda na podatke

## ❑ Primjer: ADO\SortFilter

- SqlDataAdapter.SelectCommand (INNER JOIN) i pripadni DataSet



	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 u dizajnu)

## ❑ **Primjer:** **ADO\SortFilter (dizajn)**

- `dataView.Table = dataSetMjesta.MjestoDrzava`
- `drzavaBindingSource.DataSource = dataViewMjesta`
- `dataGridView.DataSource = drzavaBindingSource` (isto kao da smo postavili na `dataViewMjesta`, što ćemo i činiti dinamički)

# 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
- `DataManager` – kojem `DataView` pripada
- `Item(Index)` – `DataRowView` objekt na zadanom indeksu
- `RowFilter` – izraz na temelju kojeg se obavlja filtriranje 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 – filtrira podatke napunjene u pogledu**

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

## ❑ **Find – traži redak prema zadanoj vrijednosti sort ključa**

- Rezultat/indeks pronađenog retka odgovara onom u `BindingContext` objektu preko kojeg su zapisi povezani

```
dvMjesta.Sort = "IdMjesta"  
int idxFound = dvMjesta.Find(13);  
  
// nadjen je this.dvMjesta[idxFound]["NazMjesta"];
```

# Selekcija redaka u skupu podataka

## ❑ Postupak `DataTable.Select` – vraća polje redaka

- `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
- koristit ćemo kasnije pri traženju artikla u formi `DokumentStavka`

## ❑ Primjer: **ADO\SortFilter**

```
private DataRow[] drFound;

private void buttonSelect_Click(object sender, EventArgs e)
{
    drFound = this.dataSetMjesta.MjestoDrzava.Select(
        textBoxFilter.Text);
    dataGridView1.DataSource = drFound;

    // podatke smo mogli obraditi i "ručno"
    foreach (System.Data.DataRow dr in drFound)
```

# ***DataColumn.Expression***

## **❑ Izrazi za postavljanje uvjeta na podatke koriste se**

- za selekciju i filtriranje (prethodni primjeri)
- za kreiranje izračunatih polja (budući primjeri)

## **❑ 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" u vezi (slijedi)**

- pr: "Child.UkupanIznos > 3000"
- pr: "Parent.IdPartnera = 9999"

# 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 otipkanog slova
  - događaj kontrole KeyPress
  
- ☐ 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

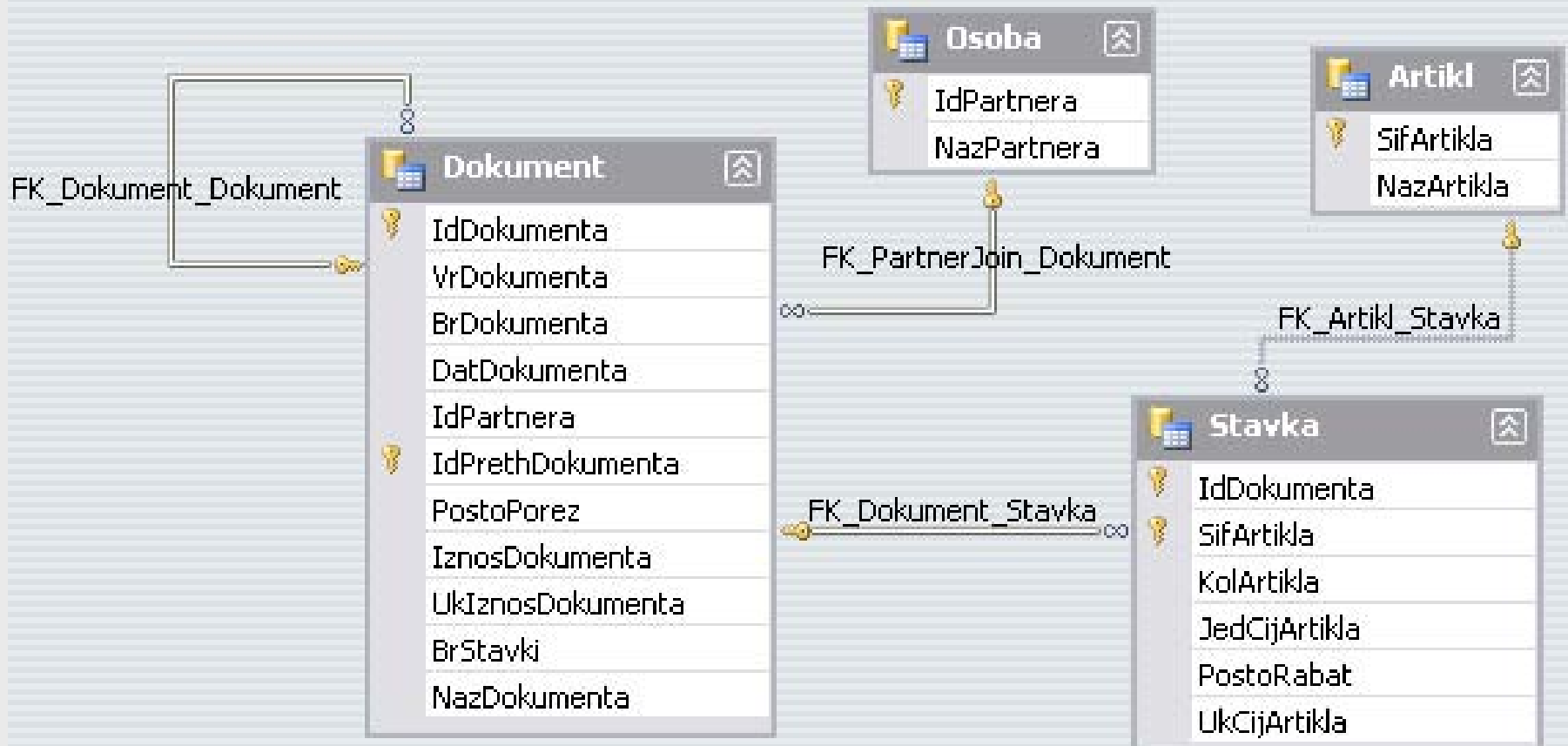
Id dokumenta:	<input type="text" value="9"/>	Vrsta:	<input type="text" value="0"/>	Broj:	<input type="text" value="81"/>	Datum:	<input type="text" value="5. 1 .2007"/>
Partner:	<input type="text" value="Polenus Mario (1705946330105)"/>					Porez:	<input type="text" value="0.22"/> [0 -1]
Prethodni:	<input 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 (adapteri Properties, metoda *DokumentStavka\_Load*)**
  - pogledati svojstva adaptera: npr. Dokument, PartnerJoin, po potrebi osvježiti




# 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)");

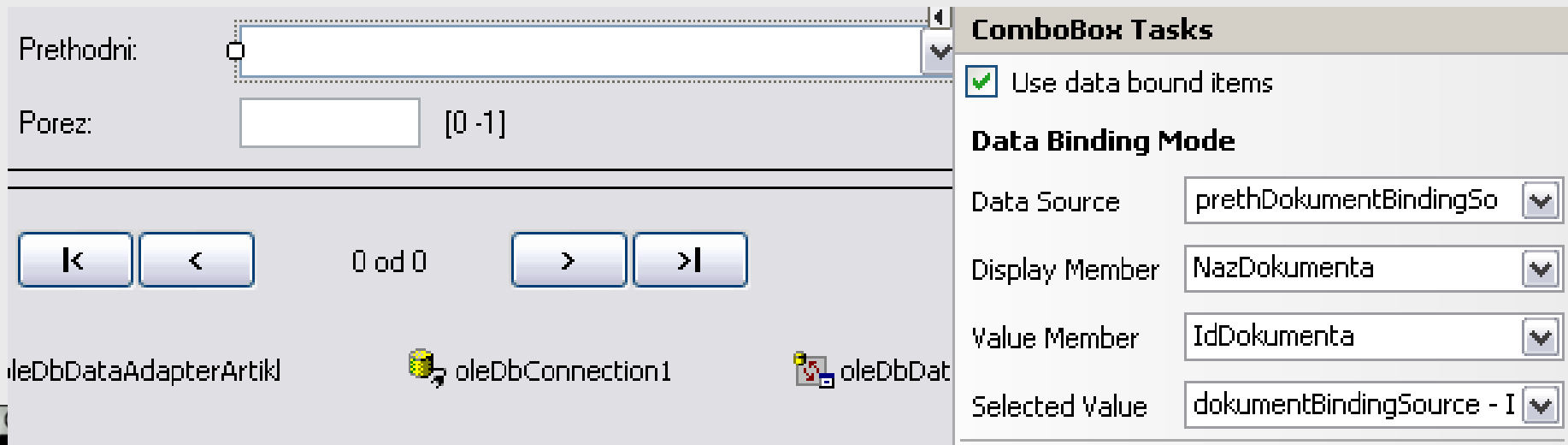
dataSetDokumentStavka.Dokument.Columns.Add(
    new DataColumn(                                   // 2.NAČIN
        "UkIznosDokumenta", typeof(float),
        "Sum(Child(FK_Dokument_Stavka).UkCijArtikla)
            * (1 + PostoPorez)");
    );
dataSetDokumentStavka.Dokument.Columns.Add(
    new DataColumn(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

# Odabir vrijednosti stranog ključa

- ❑ Odabir vrijednosti stranog ključa padajućom listom (**ComboBox**)
- ❑ Primjer:  ADO\DokumentStavka – partner - programski

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



- ❑ Primjer:  ADO\DokumentStavka – prethodni dokument - u dizajnu
  - fkDokumentDokumentBindingSource <>i prethDokumentBindingSource !!!



Prethodni:

Porez:  [0 -1]

Id < 0 od 0 > >I

oleDbDataAdapterArtikl  oleDbConnection1  oleDbData

**ComboBox Tasks**

- ☒ Use data bound items

**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* – *DataGridView 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 izbor samo na povezane prethodnike

# Referentne vrijednosti stavke

## ❑ *dataGridViewStavke – Tasks – Edit Columns*

- *NazivArtikla.DisplayStyle = ComboBox*
- postavljamo *DataSource*, *DisplayMember* i *ValueMember*
- za *DataSource* mogli smo stvoriti i zaseban *BindingSource* nad *Artikl*

Selected Columns:

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

Buttons: 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.

Buttons: OK, Cancel

# Ažuriranje jedinične cijene artikla

## ❑ Primjer: ADO\DokumentStavka

- Ažuriranje jedinične cijene artikla po odabiru artikla

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

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

```
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(
                ((DataGridView) sender) ["SifArtikla",
                fKDokumentStavkaBindingSource.Position]).Value))) [0];

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



# Spremanje podataka

## ❑ Primjer: ADO\DokumentStavka (buttonSpremi\_Click, UpdateData)

```
// ažuriranje redundantne 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
```

```
{
```

```
    sqlDataAdapterDokument.Update(  
        dataSetDokumentStavka.Dokument);
```

```
    sqlDataAdapterDokument.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

- da stavke nismo brojali izračunatim poljem u skupu podataka

```
void CommandBroji()  
{  
    cmdCount = new SqlCommand(  
        "SELECT COUNT(*) AS Broj FROM Stavka WHERE "  
        + " (IdDokumenta = @IdDokumenta) "  
        , sqlConnection1);  
    cmdCount.Parameters.Add(  
        new SqlParameter("@IdDokumenta",  
            SqlDbType.Integer, 4, "IdDokumenta"));
```

```
void PrebrojiStavke()  
{  
    sqlConnection1.Open();  
    cmdCount.Parameters["@IdDokumenta"].Value =  
        textBoxIdDokumenta.Text;  
    int cnt = (int) cmdCount.ExecuteScalar();  
    sqlConnection1.Close();  
    textBoxBrStavki.Text = cnt.ToString();
```

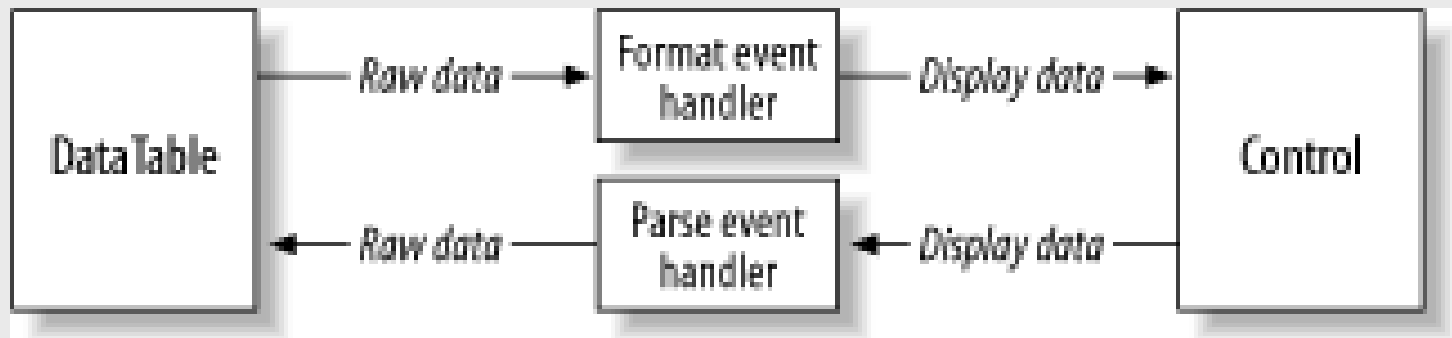
# Formatiranje objekata

## ❑ 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 (Bind\_Data)

### ■ 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 – željeni tip podatka vrijednosti
- Value – vrijednost podatka

# Primjer događaja *Format* i *Parse*

## ❑ Funkcija za obradu **Format**

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

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

## 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);  
}
```



# Zadaci za vježbu

- ☐ Doraditi *Artikl* dodavanjem tablice *JedinicaMjere* i veze s artiklom
- ☐ Napisati program koji će ažurirati tablicu *Artikl* tako da obradi sadržaj mape u kojoj se nalaze slike naziva oblika <IdArtikla>.jpg
- ☐ 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*

# 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/>