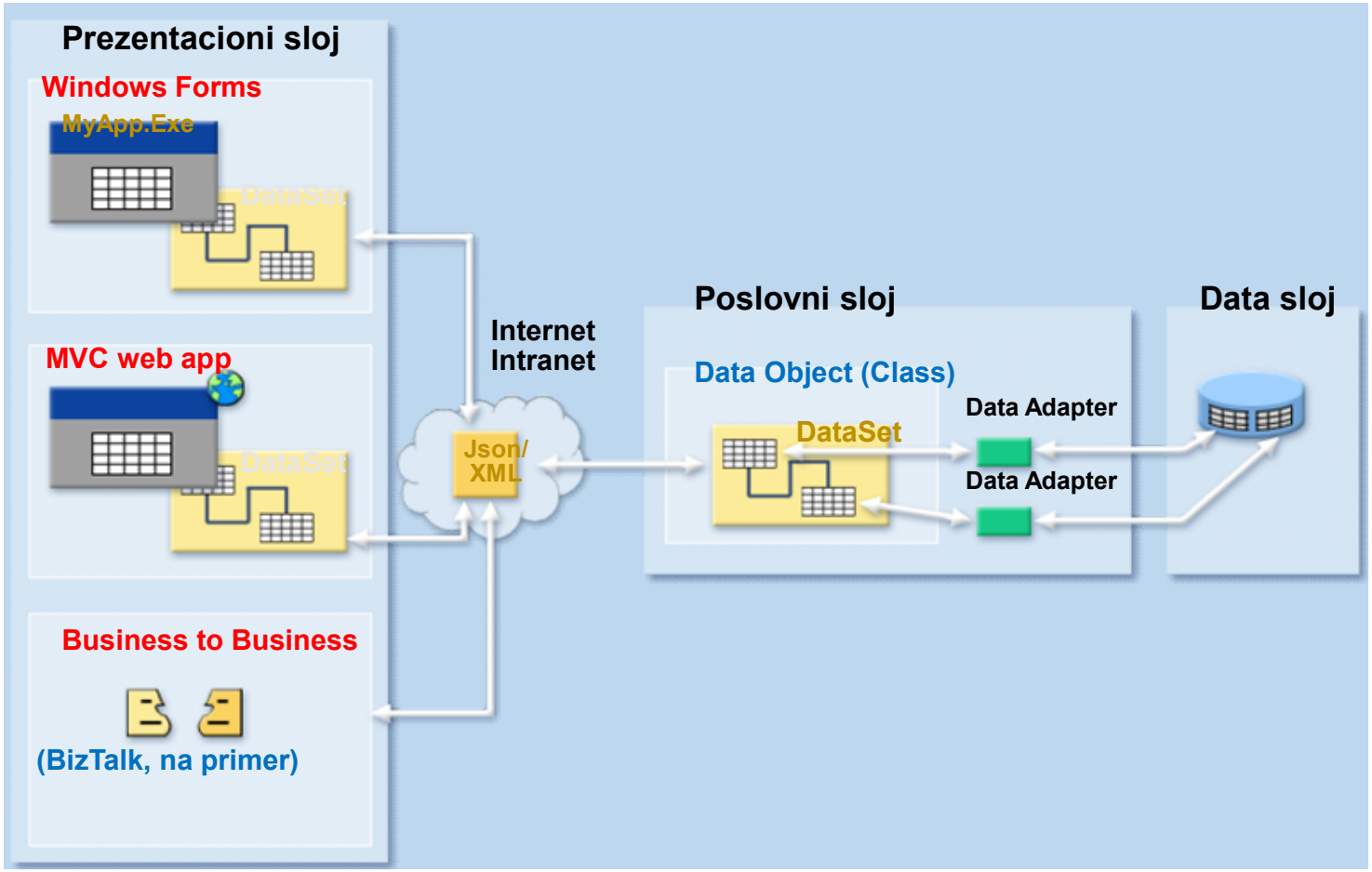
ADO.NET

Architecture

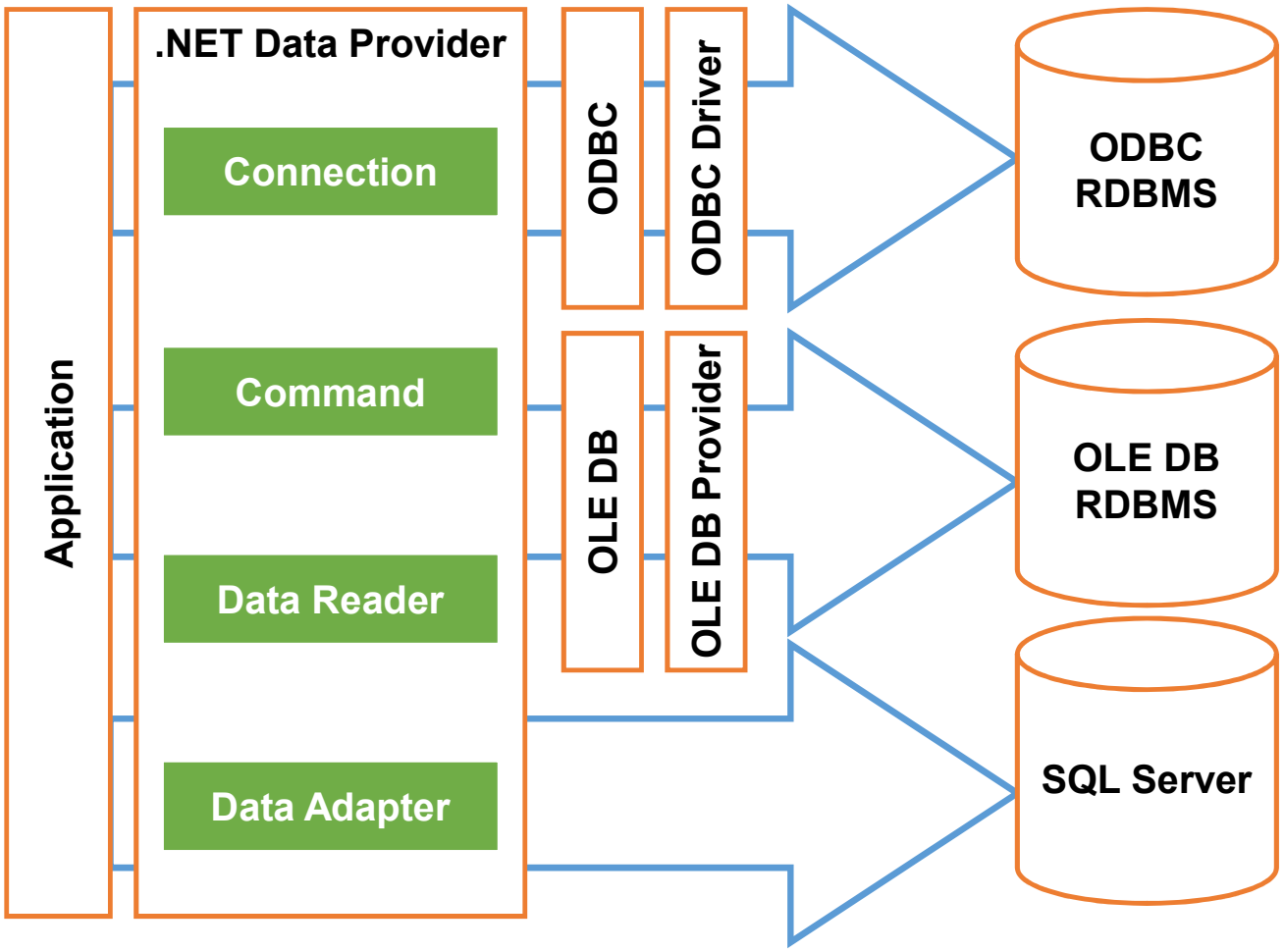


Prostori imena

.NET grupiše klase u prostore imena. To su:

- System.Data – ADO.NET arhitektura
- System.Data.Common – Klase koje dele druge klase
- System.Data.OleDb-za OLE DB dobavljače
- System.Data.SqlClient –dobavljači za SQL server
- System.Data.SqlTypes –za izvorne tipove podataka za SQL server
- System.XML – klase za podršku XMLu

Osnovni objekti

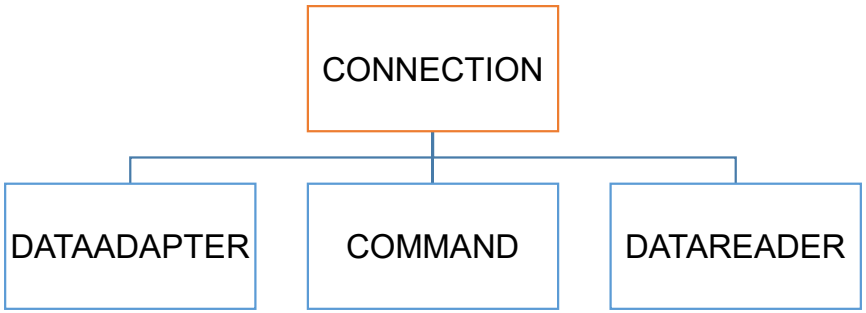


Konekcije

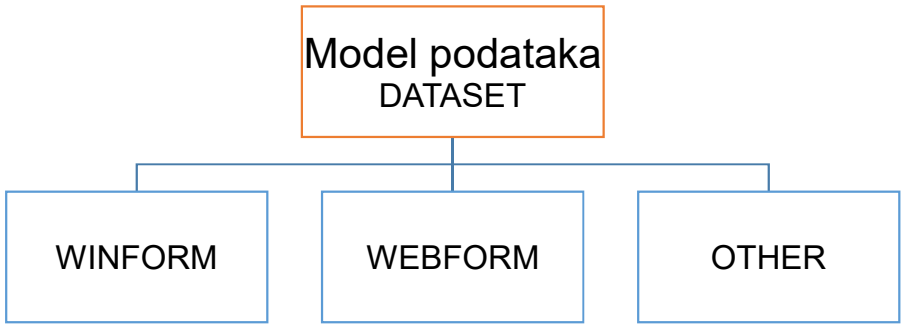
OLE DB

- Implementiraju skup COM interfejsa koji dozvoljavaju pristup podacima u standardnom formatu red/kolona.
- Ovi dobavljači su dizajnirani za svaki izvor podataka posebno – kao i ODBC
- Takodje postoji OLE DB dobavljač za ODBC upravljačke programe (drajvere)

Dobavljači



POTROŠAČI

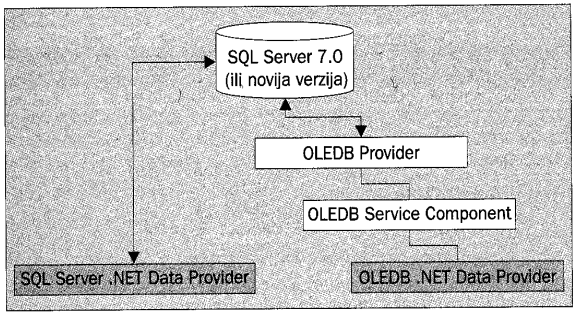


Zadatak .NET dobavljača...

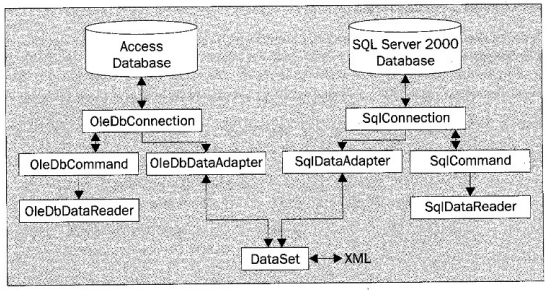
...je da efikasno obavlja dve funkcije:

1. Obezbedi pristup podacima preko aktivne veze
2. Obezbedi prenos podataka nezavisnom objektu za skladištenje tj. modelu podataka i od njega

Više dobavljača kako bi se efikasno koristila snaga određene baze podataka za koju je vaš program projektovan.



Dva .NET dobavljača mogu da rade istovremeno i da dele jedan model podataka



POVEZIVANJE SA BAZOM PODATAKA

Objekat *Connection*

Namena?

- Konekcije su odgovorne za rukovanje fizičkom konekcijom između skladišta podataka i .NET aplikacije.
- Svaki objekat iz grupe snadbevača implementira sopstvenu verziju ovog objekta.

Objekti Connection u .NET dobavljačima

Objekat	Prostor imena
`OleDbConnection`	`System.Data.OleDb`
`OdbcConnection`	`System.Data.Odbc`
`SqlConnection`	`System.Data.SqlClient`

Povezivanje sa Access bazom koristeći OLEDB

- Uvodjene prostora imena za klase dobavljača
  - `using System.Data.OleDb;`
- Koristi se i prostor imena za klase od opšteg značaja (DataTable, DataSet, . . .)
  - `using System.Data;`

- Kreiranje konekcijskog objekta koristeći konstruktor bez argumenata
  - `OleDbConnection conn = new OleDbConnection();`

Parametri za povezivanje

- Parametri za povezivanje se definišu preko svojstva ConnectionString.
- Kako se primenjuje konekcijski string?
- 1. Način: neposredno preko konstruktora Connection objekta
- 2. Način: naknadnom podešavanjem svojstva

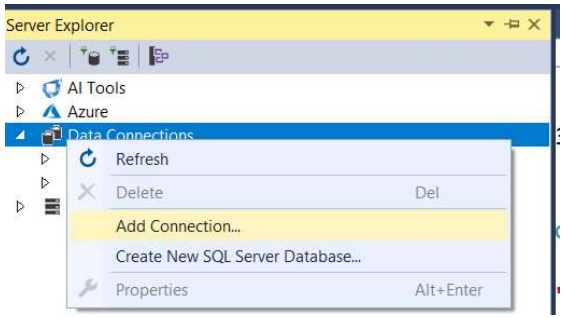
```
// 2. nacin
OleDbConnection con1 = new OleDbConnection();
con1.ConnectionString = connStr;

// 1. nacin
OleDbConnection con2 = new OleDbConnection(connStr);
```

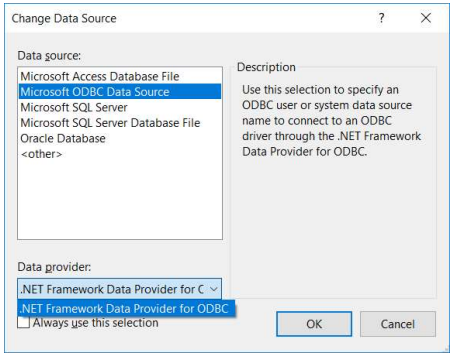
Određivanje konekcijskog stringa preko IDE okruženja

- Otvoriti prozor „Server Explorer“, pogledati prvu sliku.

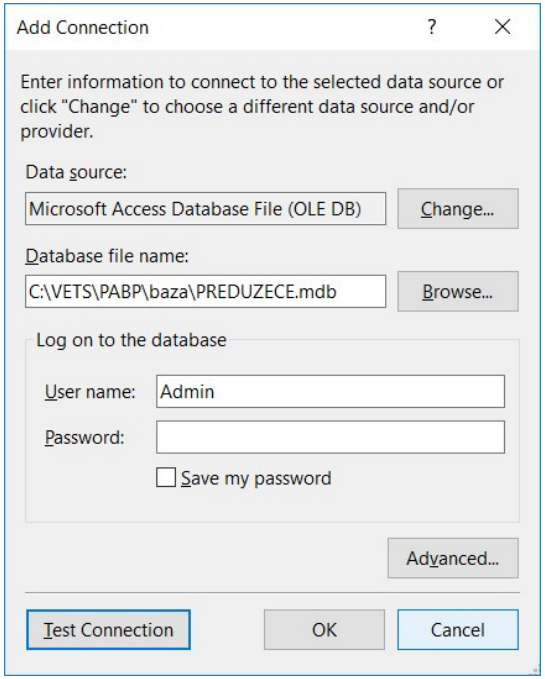
- Desnim klikom na „Data Connection“ odabrati opciju „Add Connection“



- Koristeći Dugme „Change“ za „Data Source:“ izabrati opciju „Microsoft Access Database File“



- Odabrati odgovarajući fajl i obavezno testirati konekciju.

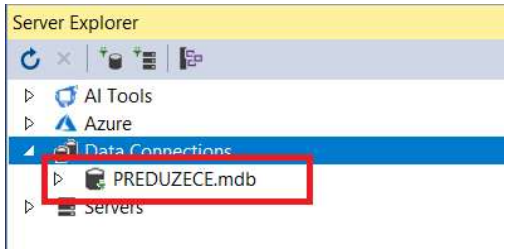


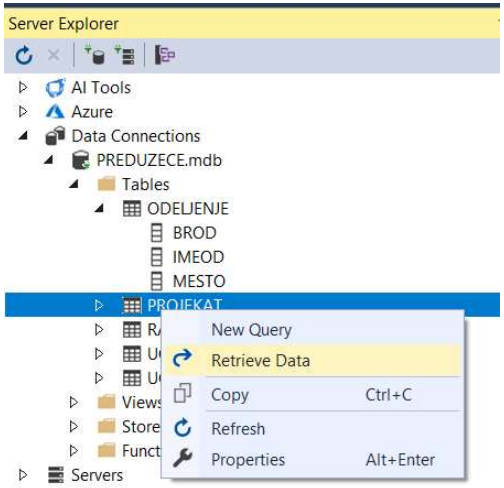
Prozor *Server Explorer*

- Nakon uspostavljene konekcije, ona postaje vidljiva preko prozora Server Explorer, pogledajte prvu sliku

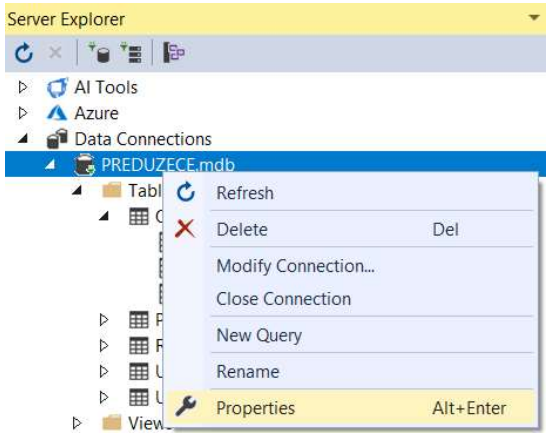
- Koristeći ovaj prozor moguće je:
  - Pogledati tabele, poglede, ugrađene procedure…
  - Testirati neke upite
  - Pogledati sadržaj nekih tabel

- Sve gore pomenuto se obavlja posredstvom već uspostavljene konekcije.

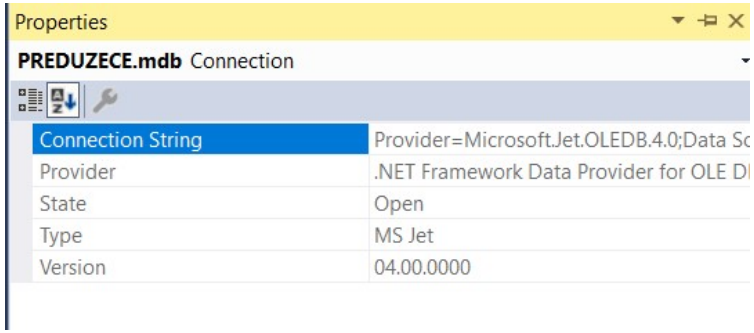




- Pogledajmo svojstva konekcije koju smo napravili:



- Otvara se prozor „Properties“ i u njemu se vide sva svojstva.
- Svojstvo koje nam je od posebnog značaja je „Connection String“
- Kopiranjem vrednosti ovog svojstva dobijamo konkekcijski string koji možemo koristiti u aplikacijama.



Povezivanje sa Accessom

- Vrednost konekcijskog stringa je: Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\VETS\PABP\baza\PREDUZECE.mdb
- Sastoji se od članova oblika
- *name = value;*
- U okviru stringa članovi mogu biti u proizvoljnom redosledu.
- ; je separator i ne mora da stoji na kraju poslednjeg člana.

- //ime dobavljača za SQL bi bio SQLOLEDB
- “Provider=Microsoft.Jet.OLEDB.4.0;” +
- //izvor tj. mdb datoteka
- “Data Source = c:\...\northwind.mdb;” +

O konekcijama!

- SKUPE su.
- Zauzimaju memorijski prostor na klijentskoj mašini ali i serveru.
- Previše otvorenih veza usporava rad i može da spreči otvaranje novih.
- Nekada se broj veza posebno doplaćuje.

Otvaranje / zatvaranje konekcije

- Obavezno u bloku za prihvatanje i obradu izuzetaka:

```
try
{
    ….
    conn.Open();
    ….
    conn.Close();
    ….
}
catch(Exception ex)
{

}
```

Svojstva/metode konekcije

- `State` - Trenutno stanje konekcije
- `Open` – Otvaranje konekcije
- `Close` – Zatvaranje konekcije
- `CreateCommand` – Kreiranje komande koristeći konekcijski objekat

- Neki objekti automatski otvaraju i zatvaraju konekciju kao na primer Adapter.
- Komanda (sql naredba) se može izvršiti jedino na otvorenoj konekciji.
- NE ZABORAVITE DA KONEKCIJU ZATVORITE.

RUKOVANJE DOGAĐAJIMA KONEKCIJE

- StateChage
  - Argument tipa StateChangeEventArgs
    - Svojstva:
      - OriginalState
      - CurrentState
    - Moguće vrednosti:
      - Open – konekcija je otvorena
      - Closed – konekcija je zatvorena
      - Executing – izvršava se komanda
      - Fetching – vraća podatke
      - Connecting – uspostavljanje veze
      - Broken – otvorena ali ne funk. Može se ponovo otvoriti.

SQL komande

Komanda

- Komanda je služi da se izvrši SQL naredba ili uskladištena procedura koristeći objekat *Connection*.
- Komanda može kao rezultat imati vraćeni niz zapisa ili pak samo izazvati neke promene u bazi.

Kreiranje

1. Samostalno:
   1. `SqlCommand cmd = new SqlCommand();`
2. Koristeći *Connection* objekat
   1. `SqlCommand cmd = conn.CreateCommand();`

- U prvom slučaju se mora obezbediti naknadno povezivanje sa nekim konekcijskim objektom. Zašto? Jer komanda mora da se izvršava preko neke (otvorene) konekcije.
- `cmd.Connection =conn;`

Vrste konstruktora

- `new xxCommand()`
- `new xxCommand(string komanda)`
- `new xxCommand(string komanda, xxConnection konekcija)`

- Dodeljivanje komandnog teksta tj. sql naredbe:
- `cmd.CommandText = sqlNaredba; //string`
- Treba da bude u ***try-catch*** bloku.

Svojstva

- **CommandText**, je znakovni niz, sadrži ili stvarni tekst komande koja treba da se izvrsi na konekciji ili ime uskladištene procedure iz izvora podataka.
- **CommandTimeout** određuje vreme koje će komanda čekati na odgovor od servera pre nego sto generiše grešku. Uzmite u obzir da je ovo vreme koje protekne pre nego što objekat Command počne da prima rezultate, a ne vreme koje je potrebno komandi da se izvrši. Izvoru podataka moze biti potrebno puno vremena da vrati sve redove neke ogromne tabele, ali pod uslovom da je prvi red vraćen u toku zadatog perioda CommandTimeout nece se generisati nikakva greška.
- **CommandType** govori objektu Command na koji nacin treba da protumači sadržaj svojstva CommandText.
  - Vrednost TableDirect podržana je za objekat OleDbCommand, ali ne i za objekat SqlCommand, a ekvivalentna je naredbi SELECT * FROM <ime_abele>, pri cemu je <ime_tabele> ime tabele specifikovano u svojstvu CommandText

- Svojstvo **Parameters** objekta Command sadrži kolekciju parametara za SQL naredbu ili uskladistenu proceduru navedenu u svojstvu ConunandText.
- Ovu kolekciju cemo detaljno ispitati u nastavku.

- Svojstvo **Transaction** sadrži referencu na objekat Transaction i služi pri obradi transakcija. Ovo svojstvo cemo detaljno ispitati kasnije

Izvršavanje

1. Iskaz nije upit – ***ExecuteNonQuery***
2. Jedna vrednost – ***ExecuteScalar***
3. Jedan ili više redova – ***ExecuteReader***
4. XML – *ExecuteXMLReader*

Slede primeri:

ExecuteNonQuery

Za upite koji ne vraćaju redove.

- `cmd.CommandText = "insert into Employees (FirstName,LastName) values ('Perica','P')";`
- `cmd.Connection.Open();`
- `int cnt = (int)cmd.ExecuteNonQuery();`

ExecuteScalar

Za upite koji vraćaju jedan podatak.

- `cmd.CommandText = "select count(*) from Employees";`
- `cmd.Connection.Open();`
- `int cnt = (int)cmd.ExecuteScalar();`

ExecuteReader

Za upite koji vraćaju redove podataka.

```
cmd.CommandText = "select EmployeeID, FirstName, LastName from Employees where
LastName like '" + textBox1.Text + "%'";
cmd.Connection.Open();
OleDbDataReader reader = cmd.ExecuteReader();

while(reader.Read())
{
    string prezime = (string)reader["LastName"];
    int id = (int)reader["EmployeeID"];
    string ime = (string)reader["FirstName"];

    listBox1.Items.Add("" + id + "; " + ime + " " + prezime);
}
```

Parameters

❖Šta je to kolekcija?

1. Jedna vrsta listi definise u .NET
2. Veličina nije ograničena (samo memorijom na računaru)
3. Podrška različitih metoda za rad sa kolekcijom.

❖ Kako se koristi?

1. Specificirati parametre kroz SQL komandu u upitima ili uskladištenim procedurama
2. Definisati odgovarajuće parametre u kolekciji `Parameters`
3. Dodeliti vrednosti

1. Specifikacija parametara kolekcije *Parameters* kroz SQL komande

- Postoje dve različite vrste oznaka u zavisnosti od vrste dobavljača
- OleDbCommand
  - SELECT * FROM Customer WHERE CustomerID = **?**
- SqlCommand
  - SELECT Count(*) AS OrderCount FROM OrderTotals WHERE (EmployeeID = **@emplD**) AND (CustomerlD = **@custlD**)

Obratite pažnju: ***Objekat OleDbCommand ne podržava imenovane parametre!***
Redosled je od presudnog značaja!

2. Definisanje kolekcije

- 1. Ukoliko koristite IDE za krerinje komandi, ova kolekcija će biti automatski formirana.

- 2. Ukoliko sami u kodu kreirate kolekciju, nove parametre dodajete pomoću metode **Add*(noviParametar)***

- Napomena: Za sve kolekcije važe iste metode. Na primer, isto ime metode za dodavanje novog elementa - Add.

Metode kolekcije

- **Add(Value)** Na kraj kolekcije dodaje novi parametar cija je vrednost *Value.*
- **Add(Parameter)** Na kraj kolekcije dodaje objekat Parameter.
  - *.... Još par varijanti metode Add() ...*
- **Clear** Uklanja sve parametre iz kolekcije.
- **Insert(Index, Val)** Umece novi parametar cija vrednost je *Value* na mesto odredeno indeksom *Index* koji se racuna pocevsi od nule.
- **Remove(Value)** Iz kolekcije uklanja parametar cija vrednost je *Value.*
- **RemoveAt(Index)** Iz kolekcije uklanja parametar koji se nalazi na mestu odredenom indeksom *Index* koji se racuna pocevsi od nule.
- **RemoveAt (Name)** Iz kolekcije uklanja parametar cije ime je specifikovano u znakovnom nizu *Name.*

Konfigurisanje parametara

```
• cmd.CommandText = "select ime, posao, plata from Radnik
  where brod = ?";

• OleDbParameter p1 = new OleDbParameter();
• p1.Value = int.Parse(textBox1.Text);
• p1.ParameterName = "prm1";
• cmd.Parameters.Add(p1);
```

Zaštita od umetnutih sql naredbi

- Neka je potrebno proveriti broj radnika koji imaju odgovarajuće ime i posao u tabeli Radnik. Forma izgleda nalik jednoj formi za prijavu (login) na neki sistem.

- Standardan upit:
- ```
  string sqlUpit = "select count(*) from Radnik where ime='" + txtIme.Text
  + "' and posao='" + txtPosao.Text + "'";
  ```

Provera:

```
OleDbCommand cmd = new OleDbCommand(sqlUpit, conOle);
cmd.Parameters.Add(parIme); cmd.Parameters.Add(parPosao);
conOle.Open();
int br = (int)cmd.ExecuteScalar();
if (br > 0) MessageBox.Show("Postoji");
else MessageBox.Show("Ne postoji");
```
```

- Međutim, ako se pri unosu vrednosti u drugo polje unese:

  `vozač' or '1'='1`
- nastupa greška, odnosno za svaki red vretiće se tačan uslov.
- Ovako umetanje sql narebi u polja namenjena vrednostima naziva se „sql injection“.
- Problem se rešava primenom parametarizovanih komandi. Dakle:

```
string sqlUpit = "select count(*) from Radnik where ime=? and posao=?";

OleDbParameter parIme = new OleDbParameter("i", txtIme.Text);
OleDbParameter parPosao = new OleDbParameter("p", txtPosao.Text);

OleDbCommand cmd = new OleDbCommand(sqlUpit, conOle);
cmd.Parameters.Add(parIme);
cmd.Parameters.Add(parPosao);
conOle.Open();
int br = (int)cmd.ExecuteScalar();
if (br > 0) MessageBox.Show("Postoji");
else MessageBox.Show("Ne postoji");
```

Metode objekta Command

- **Cancel** - Otkazuje izvrsenje komande za podatke.
- **CreateParameter -** Pravi novi parametar.
- **ExecuteNonQuery -** Izvršava komandu na objektu Connection i vraća broj izmenjenih redova. Koristi se u slučajevima kada SQL naredba ili SP ne vraća ni jedan red (INSERT, DELETE, UPDATE)
- **Prepare -** Pravi pripremljenu (kompaj]iranu) verziju komande na izvoru podataka.
- **ResetCommandTimeout -** Svojstvo CommandTimeout ponovo postavlja na podrazumevanu vrednost.

A DataReader?

- DataReader se koristi kada komanda vraća skup zapisa.
- Tok podataka koji vraća DataReader je takav da se može **samo čitati** i kretati **samo unapred.**
- U memoriji se istovremeno nalazi samo jedan red podataka!

Zadatak

- Napisati aplikaciju sa jednom formom koja obezbedjuje prikaz radnika za odbarano odeljenje.
- Odeljenje se bira preko ComboBox kontrole