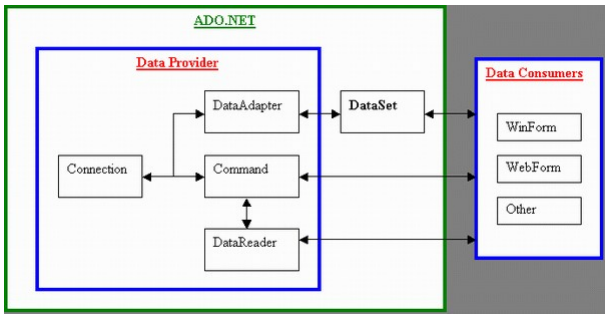
Izvršavanje SQL komandi

ČAS 2.

Komanda

- Komanda je služi da se izvrši SQL naredba ili uskladištena procedura (*store procedure*) 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),*
- *new xx*Command*(string komanda, xxConnection konekcija, Transakcija).*

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

Svojstva

CommandText	SQL naredba ili uskladistena procedura koja treba da se izvršdi.
CommandTimeout	Vreme (u sekundama) za čekanje na odgovor od izvora podataka.
CommandType	Označava kako bi svojstvo CommandText trebalo da se protumači. Podrazumeva se vrednost Text
Connection	Objekat Connection na kojem komanda za podatke treba da se izvrsi.
Parameters	Kolekcija parametara.
Transaction	Transakcija u kojoj ce se komanda izvrsavati.

Detaljnije…

- **CommandText**, je znakovni niz, sadrži ili stvarni tekst komande koja treba da se izvrsi na konekciji ili ime uskladištene procedure iz izvora podataka.
- **CommandTimeout** odreduje vreme koje ce komanda cekati na odgovor od servera pre nego sto generise gresku. Uzmite u obzir da je ova vreme koje protekne pre nego sto objekat Command pocne da prima rezultate, a ne vreme koje je potrebno komandi da se izvrši. Izvoru podataka moze biti potrebno 10 ili 15 minuta da vrati sve redove neke ogromne tabele, ali pod uslovom da je prvi red vracen u toku zadatog perioda CommandTimeout nece se generisati nikakva greSka.
- **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 komande

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

Slede primeri:

ExecuteNonQuery

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

ExecuteScalar

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

Primer: ExecuteReader

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

Primer

- Testiranje komandi sa više vraćenih redova – **ExecuteReader**
- Vratiti imena i prezimena iz tabele *Employees.*

Izvršavanje komandi koje nisu upit: **ExecuteNonQuery**

- Pogledati tabelu Employes.
- Dodati 1 red:

1. ***insert into*** *Employees (FirstName, LastName) values ('Perica', 'P');*
2. ***delete*** *from Employees where FirstName='Perica' and LastName = 'P'*

Kreiranje baze i tabele

1. Otvaranje konekcije
2. "create database *imeNoveBaze*"
3. Promena baze u objektu konekcije
4. "create table *imeNoveTabele* (*imekol tip*)"

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 = **@empID**) AND (CustomerID = **@custID**)

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

2. Definisanje kolekcije

- 1. Ukoliko koristite IDE, 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 EmployeeID, FirstName,
LastName from Employees where LastName like @prm1";

SqlParameter p1 = new SqlParameter();
p1.Value = textBox1.Text + "%";
p1.ParameterName = "@prm1";
cmd.Parameters.Add(p1);
```

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.

Testiranje parametara:

- 1. Kreirajte tabelu:
- *“create table mytable (myname varchar(30), mynumber integer)”*
- 2. Insert komanda*:*
- *“insert into mytabel values(@myname, @mynumber)”*
- 3. Dodavanje parametara
- *cmd.Parameters.Add(“@myname”, SqlDbType.VarChar, 30);*
- *cmd.Parameters.Add(“@mynumber”, SqlDbType.Int);*
- 4. Priprema komande
- cmd.Prepare();
- 5. Dodeljivanje vrednosti i izvršavanje
- string[] names={“pera”, “aca”, “caca”, “mica”};
- for(int i=0; i<names.length; i++){
  - cmd.Parameters[“@myname”].Value = names[i];
  - cmd.Parameters[“@mynumber”].Value = i;
  - cmd.ExecuteNonQuery();
- }

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!

DataSet

Čas 3.

Šta je i čemu služi?

- DataSet je jedan od najbolje osmišljenih i najviše korišćenih objekata, ali i je i jedna od važnijih karakteristika Microsoft-ove .NET platforme.
- Sličan je tradicionalnom ADO recordset-u, ali i sa bitnim razlikama.
- **1)DataSet može da čuva rezultate više različitih SQL upita.**
- **2)Možete koristiti ovaj objekat nezavino od konekcije.**
- **3)Može se kreirati iz XML fajla .**
- **4)Možete kreirati XML fajl iz DataSeta.**

Od čega se sastoji?

- Kolekcija tabela
  - Tabela
    - Kolekcija redova
      - Red
    - Kolekcija kolona
      - Kolona
    - Ograničenja
      - Ograničenje
- Veze
  - Veza

Kako se koristi?

- Pretpostavimo da imamo konekciju do SQL servera, kreirajmo data adapter na sledeći način:

```
sqlDataAdapter1.SelectCommand.CommandText="SELECT * FROM radnik“
DataSet ds= new DataSet();
sqlDataAdapter1.Fill( ds ) ;
DataTable dt = ds.Tables[0];
DataColumn dc       = dt.Columns[2];
                    = dt.Columns[“naziv”];
DataRow dr = dt.Rows[2];
```

- Rezultujući DataSet se sastoji od Tablela koje su označene indeksima počev od 0.
- *listBox1.DataSource=ds.Tables[0] ;*

CommandBuilder

- SqlDataAdapter da = new SqlDataAdapter();
- SqlCommandBuilder bu = new SqlCommandBuilder(da);
- da.DeleteCommand = bu.GetDeleteCommand();
- da.InsertCommand = bu.GetInsertCommand();
- .....
- da.Update();

- Napomena: Samo za tabele sa primarnim ključom i već pripremljenim svojstvom SelectCommand

Vrste DataSet objekata

- **Tipizirani**
- **Netipizirani**

Neka svojstva DataSet-a

- ***Tables***
- *Relations*
- *HasErrors* – označava da li neki od objekata *DataRow* sadrži grešku
- *EnforceConstraints* – određuje da li se prilikom izmena poštuju pravila za ograničenja
- *DataSetName*
- *CaseSensitive* – da li su poređenja osetljiva na velika slova

Primer 1

- Deklarisanje

```
public partial class Form2 : Form
{
    SqlDataAdapter da;
    DataSet ds;
```

- Inicijalizacija

```
public Form2()
{
    InitializeComponent();

    da = new SqlDataAdapter();
    ds = new DataSet();
    da.SelectCommand = new SqlCommand();
    da.SelectCommand.Connection = new SqlConnection(@". . .");
```

- Povezivanje

```
private void button2_Click(object sender, EventArgs e)
{
    da.SelectCommand.CommandText = "select FirstName,LastName
from Employees";
    da.Fill(ds);
    dataGridView1.DataSource = ds.Tables[0];
```