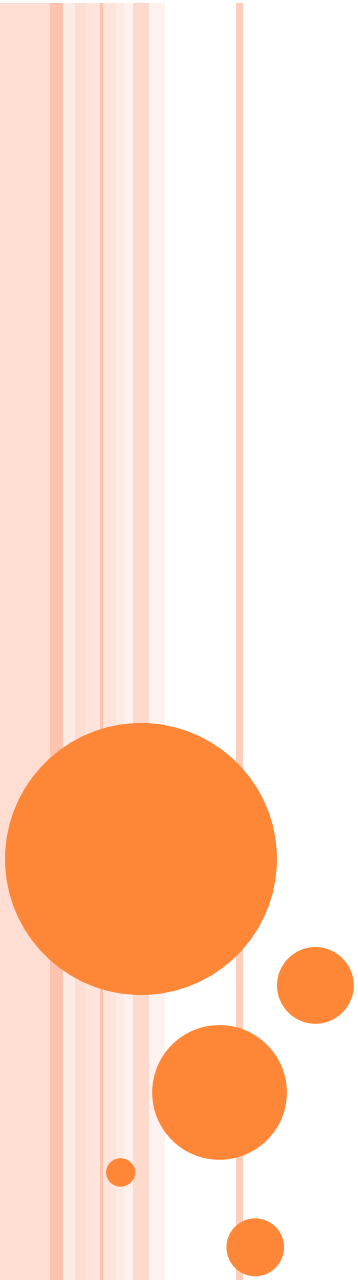


TRANSAKCIJE

.

ŠTA JE I ČEMU SLUŽI?

- Transakcija je serija akcija koja se mora posmatrati kao jedna celina. Akcije moraju sve da uspeju ili da ne uspe ni jedna.

- Primer: Prenos novca sa jednog računa na drugi sastoji se od dve akcije:
- **1)Skidanje novca sa prvog računa**
- **2)Dodavanje istog iznosa na drugi račun**
- **!!!U slučaju da neka od ovih akcija ne uspe automatski treba poništiti obe tj. ove dve akcije se smatraju jednom celinom i predstavljaju jednu TRANSAKCIJU!**

PRAVLJENJE TRANSKACIJA

- Objekat Transaction je realizovan kao deo snadbevača podataka, znači koristimo
  - OleDbTransaction
  - SqlTransaction

NOVA TRANSAKCIJA...

- ...se pravi pozivom metode...
  - ***BeginTransaction***
- ...objekta Connection

- *...SqlTransaction objekat, podržava imenovane transakcije i konstruktor za ovu transakciju sadrži dodatne oblike za postavljanje imena transakcije.*

UGNEŽDENE TRANSAKCIJE

- Prvo pravi transakcija a zatim se na toj transakciji pravi druga transakcija itd.
- Transakcije se dodeljuju komandama koje učestvuju u njima, zatim se komande izvršavaju i na kraju se transakcije zatvaraju ili njihovim izvršavanjem ili vraćanjem na pređašnje stanje.

POTVRĐIVANJE ILI PONIŠTAVANJE TRANSAKCIJA

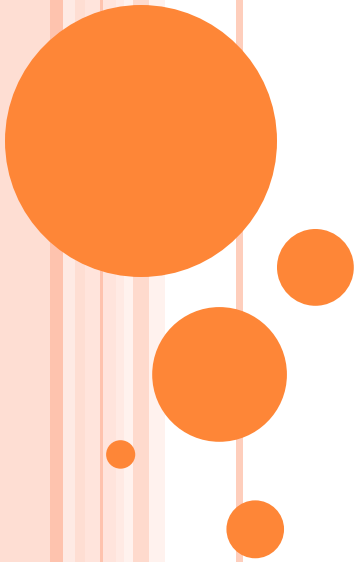
- **Commit – uspešna transakcija**
- **Rollback – vraćanje na prvobitno stanje**

REDOSLED

```
//deklaracija transakcije
OleDbTransaction tr;
//otvaranje konekcije
this.conn.Open();
//kreiranje transakcije
tr = this.conn.BeginTransaction();
//vezivanje komandi za kreiranu transakciju
Cmd1.Transaction = tr;
Cmd2.Transaction = tr;
da.InsertCommand.Transaction = tr;
```

REDOSLED

```
try // pokušaj izvršavanja komani transakcije
{
        Cmd1.ExecuteNonQuery();
        Cmd2.ExecuteScalar();
        da.Update(ds.Porudzbe);
        tr.Commit();
}
catch(Excepton e1) // neuspeh
{
        tr.Rollback();
        MessageBox.Show("Tra la la…");
}
finally // u svakom slučaju uraditi...
{
        conn.Close();
}
```



PRIMERI

- Primer 1.
  - Kreirati preko transakcije prenos dela plate sa jednog radnika na drugog.

REALIZACIJA PREKO COMMAND OBJEKATA

```
if (connection.State != ConnectionState.Open)
    connection.Open();
tr = connection.BeginTransaction();

command1.Transaction = tr;
command2.Transaction = tr;
try
{
    command1.Parameters["IDBR"].Value = textBox1.Text;
    command1.ExecuteNonQuery();
    command2.Parameters["IDBR"].Value = textBox2.Text;
    command2.ExecuteNonQuery();
    tr.Commit();
}
catch (Exception ex)
{
    tr.Rollback();
    MessageBox.Show(ex.Message);
}
```

REALIZACIJA PREKO ADAPTER OBJEKATA

```
DSNorthwind.OrdersRow novi1 = dSNorthwind.Orders.NewOrdersRow();
novi1.CustomerID = textBox1.Text;
novi1.EmployeeID = int.Parse(textBox2.Text);
novi1.OrderDate = DateTime.Parse(textBox3.Text);
novi1.RequiredDate = DateTime.Parse(textBox4.Text);
novi1.ShippedDate = DateTime.Parse(textBox5.Text);
dSNorthwind.Orders.AddOrdersRow(novi1);
ordersTableAdapter.Update(novi1);

DSNorthwind.Order_DetailsRow novi2 = dSNorthwind.Order_Details.NewOrder_DetailsRow();
novi2.Discount = float.Parse(textBox7.Text);
novi2.OrderID = novi1.OrderID;
novi2.ProductID = int.Parse(textBox10.Text);
novi2.Quantity = short.Parse(textBox8.Text);
novi2.UnitPrice = decimal.Parse(textBox9.Text);
// throw new Exception();
dSNorthwind.Order_Details.AddOrder_DetailsRow(novi2);
order_DetailsTableAdapter.Update(novi2);
```

```
con.Open();
tr = con.BeginTransaction();
ordersTableAdapter.Transaction = tr;
order_DetailsTableAdapter.Transaction = tr;
try
{
 // prethodni kod
    tr.Commit();
}
catch
{
    tr.Rollback();
}

this.order_DetailsTableAdapter.Fill(this.dSNorthwind.Order_Details);
this.ordersTableAdapter.Fill(this.dSNorthwind.Orders);
```

Formatiranje i parsiranje podataka

- Omogućava pisanje koda u vidu događaja koji se vezuju za DataBinidings objekte a preko kojih se može izvršiti izmena prikaza u odnosu na povezane podatke, odnosno promena podataka u izvoru podataka u odnosu na one koje su prikazane preko povezanih kontrola.
- += Parse – promena podataka u DS u odnosu na prikaz
- += Format – promena prikaza u kontroli u odnosu na DS

Primer

- Postaviti DataGridView kontrolu i vezati je za tabelu Radnik.
- Postaviti textbox kontrolu i povezati je sa kolonom imena u tabeli radnik.
- Napraviti rukovaoce dogadjaja za Format i Parse dogadjaje napraviti rukovaoce dogadjaja .....

```
textBox1.DataBindings[0].Parse += new ConvertEventHandler(Form1_Parse);
textBox1.DataBindings[0].Format += new ConvertEventHandler(Form1_Format);
```

Testirati promenu tekućeg zapisa i upis novog preko textbox kontrole.

U rukovaocima događaja postaviti MessageBox prozore sa porukom.

DataView

- Sličan objektu BindingSource, ali se tiče samo tabele.
- Može biti zaseban objekat.
- Deo DataTable objekta (DefaultView).
- Metod Select objekta DataTable obezbeđuje mehanizme filtriranja i sortiranja objekata DataRow.
- Objekat DataView je zaseban objekat i nalazi se neposredno iznad objekta DataTable u hijerarhiji klasa.

ČEMU SLUŽI?

- Obezbeđuje filtriran i sortiran pogled na pojedinačan objekat DataTable.
- Za razliku od niza objekata DataRow koje vraća *Select*, objekat DataView se može koristiti kao izvor podataka za povezane kontrole!
- Možete napraviti više objekata DataView za bilo koji objekat DataTable.
- Svaki objekat DataTable sadrži bar jedan objekat DataView u svom svojstvu DefaultDataView.
  - Svojstva DefaultDataView objekta se podešavaju u toku izvršavanja!

REDOVI DATAVIEW OBJEKTA

- Nisu DataRow tipa već
- **DataViewRow** tipa, mada su slični.

SVOJSTVA DATAROWVIEW

- *DataView* – referenca na objekat DataView kome pripada
- *IsEdit* – označava da li se red trenutno edituje
- *IsNew* – da li je DataRowView potpuno nov
- *Item* – vrednost kolone u objektu DataRowView (slično istoimenom svojstvu kod DataRow)
- *Row* – Objekat DataRow koji se “posmatra”

KREIRANJE DATAVIEW OBJEKTA

```
System.Data.DataRowView dr;
System.Data.DataView dvNew;
dvNew = new System.Data.DataView();

// konfigurisanje objekta DataView
dvNew.Table = this.dsMaster.OrdersTotals;
dvNew.RowFilter = “CustomerID = xxxxxx”;
// povezivanje
this.dgOrders.DataSource = dvNew;
```

SVOJSTVA DATAVIEW...

- // Određuju da li se podaci iz izvora podataka koji su prikazani i objektu DataView mogu menjati korišćenjem objekata DataView

- AllowDelete
- AllowEdit
- AllowNew

SVOJSTVA DATAVIEW...

- Count
- DataViewManager – kolekcija kojoj pripada DataView
- *Item(Index)* – Objekat DataRowView koji se u objektu DataView nalazi na specificiranom Inexu
- RowFilter – izraz koji se koristi za filtriranje

SVOJSTVA DATAVIEW...

- **RowStateFilter**
- *Sort* – Izraz koji se koristi za sortiranje.
- Table – Objekat DataTable koji je izvor podataka.

RowStateFilter

Može se koristiti za ograničavanje objekata DataRowView koji se nalaze u DataView na one koji su u odgovarajućem stanju ili se može koristiti za vraćanje vrednosti stanja.

- Added
- CurrentRow
- Deleted
- ModifiedCurrent
- ModifiedOriginal
- None
- OriginalRows
- Unchanged

METODE DATAVIEW

- **AddNew**
- **Delete**
- **Find** – radi na osnovu primarnog ključa! Vraća jedan red ili više redova ako je prosleđen niz primarnih ključeva.
- Ako se pretraga radi po nekoj drugoj koloni koristi se **RowFilter**

IZRAZI OBJEKTA DATACOLUMN

- Izrazi objekta *DataColumn* (DataColumn Expressions) koriste svojstva RowFilter i Sort objekta DataView.

- Izraz je znakovni niz i za njegovo stvaranje mogu se koristiti sve funkcije za obradu znakovnih nizova.

IZRAZI OBJEKTA DATACOLUMN

- Izraz je **string!**
- Na primer:
  - myExpression = "**CustomerID = '**" + **strCustID** + "**'**";
- Ako imate kolonu koja sadrži specijalne znake onda i se imena takvih kolona postavljaju u uglaste zagrade.
  - myExpression = [asdf/1234] > 10;
- Numeričke vrednosti u izrazu DataColumn ne zahtevaju nikakvo posebno rukovanje, ali datumi se moraju uokviriti znakovima #
  - myExpression = "OrderDate > #01/01/2001#";
- Kolonu iz dečijeg/roditeljskog objekta DataRow možete referencirati tako što ispred imena kolone dodate *Child/Parent*
  - myExpression = "Child.OrderTotal > 3000;

CHILD I PARENT

- Parent.Price referiše kolonu Price iz roditeljske tabele.
- Kada “dete” ima više od jednog “parent” reda, koristi se *Parent(**RelationName**).ColumnName.*
- “Dete” relacije mogu da vrate više redova, mora se uključiti referenca na “dete” kolone koristeći agregatne funkcije.Na primer *Sum(Child.Price)*
- Ako tabela ima više od jedne “dete” relacije, sintaksa je: *Child(RelationName).*
- *Avg(Child(Customers2Orders).Quantity)*
  - *Sum*
  - *Avg*
  - *Min*
  - *Max*
  - *Count*
  - *StDev*

POREĐENJA I LOG. OPERATORI U IZRAZIMA

- AND
- OR
- NOT
- ;
- >
- <
- <=
- >=
- <>
- IN (da li se specifikovana vrednost nalazi u skupu)

myEx = “col IN (`A`,`V`,`C`)”

- LIKE

ARITMETIČKI OPERATORI

- +
- -
- *
- /
- %

SPECIJALNE FUNKCIJE

- **Convert**(Expression, Type)
- **Len**(String)
- **ISNULL**(Expression, ReplacementValue)
- **IF**(Expression, ValueIfTrue, ValueIfElse)
- **SUBSTRING**(Expression, Start,Length)

IZRAZI ZA SORTIRANJE

- myDataView.Sort = “CustomerID, OrderID [ASC/DESC]”;
- myDataView.Refresh();

PRIMERI