Računarstvo i informatika

Katedra za računarstvo
Elektronski fakultet u Nišu

Baze podataka (Računske vežbe)
ADO.NET (2)

Letnji semestar 2016/2017

Sadržaj

- DataSet
- DataAdapter
- Parametrizovane komande

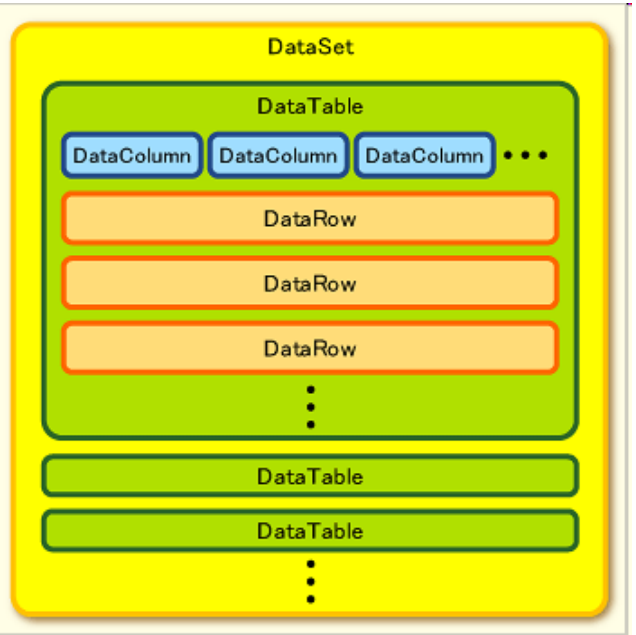
DataSet

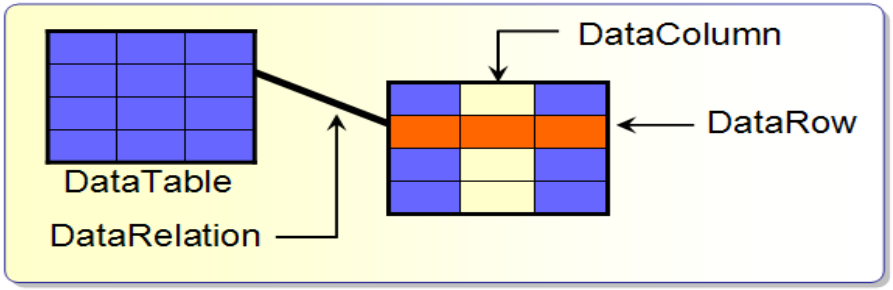
- Direktan pristup podacima (korišćenje Connection, Command i DataReader objekata) podrazumeva da je veza ka izvoru informacija (bazi podataka) otvorena za sve vreme trajanja obrade podataka.
- Osim direktnog pristupa podacima, ADO.NET nudi mogućnost pristupanja podacima i kada je veza ka izvoru informacija (bazi podataka) raskinuta.
- Ovakav način pristupanja podacima zasnovan je na kreiranju lokalne kopije podataka koji se čuvaju u lokalnoj memoriji računara.
- Korišćenjem klasa iz System.Data biblioteke moguće je kreirati lokalni model podataka koji će pored tabela, posedovati poglede, ograničenja primarnog ključa, ograničenja stranog ključa i sve druge karakteristike izvora podataka koji se nalazi u pozadini.
- Lokalni model podataka dozvoljava korisnicima kreiranje i izvršavanje upita nad lokalnim podacima, njihovo filtriranje, sortiranje i snimanje izmena natrag u izvor podataka.

DataSet

- Objekat **DataSet** predstavlja memorijsku reprezentaciju podataka iz jednog ili više izvora podataka.
- Objekat DataSet je posebno projektovan tako da obezbedi rad sa podacima koji su **keširani u memoriji (in memory data)** i izvršavanje operacija nad podacima pri čemu **nije neophodno postojanje otvorene konekcije sa izvorom podataka (disconnected operations on data)**.
- Objekat DataSet je potpuno **nezavistan od izvora podataka** i obezbeđuje **uniformne interfejse** za rad sa podacima bez obzira na njihovo poreklo.

DataSet


DataSet
DataTable
DataColumn
DataColumn
DataColumn
DataRow
DataRow
DataRow
DataTable
DataTable


DataColumn
DataRow
DataTable
DataRelation

DataSet

- DataSet se sastoji od dve kolekcije objekata:
  - **DataTableCollection** – kolekcija DataTable objekata
  - **DataRelationCollection** – kolekcija DataRelation objekata
- Objekat **DataTable** predstavlja jednu tabelu (relaciju) u bazi podataka. Sastoji se od kolekcije DataColumn i DataRow objekata.
- Objekat **DataColumn** predstavlja kolonu relacije i sadrži informacije o imenu i tipu.
- Objekat **DataRow** predstavlja vrstu relacije i omogućava čitanje i ažuriranje podataka.
- Objekat **DataRelation** definiše veze (foreign key – primary key) između tabela.

DataSet

- Postoje dve vrste DataSet objekata:
  - Neimenovani (Untyped) DataSet
  - Imenovani (Typed DataSet)
- **Neimenovani DataSet** predstavlja generički DataSet objekat koji se koristi u situacijama kada šema podataka nije unapred poznata.
- **Imenovani DataSet** je klasa koja je izvedene iz osnovne DataSet klase na bazi poznate šeme podataka (xsd opisa). Za kreiranje se koristi designer koji je sastavni deo Visual Studio 2008 okruženja ili alat **xsd.exe**.
- Imenovani DataSet obezbeđuje veću brzinu i otpornost na greške u odnosu na neimenovani DataSet.

DataSet

- Postoji više načina za kreiranje DataSet objekta:
  - Korišćenje objekta DataAdapter
  - Programskim kreiranjem objekata tipa DataTable, DataRow, DataColumn
  - Učitavanjem podataka iz XML dokumanata
  - Na bazi postojećih DataSet objekata
- Objekat DataSet obezbeđuje mehanizme za čuvanje i obradu podataka. DataSet niej zadužen za interakciju sa izvorom podataka.
- Za interakciju sa izvorom podataka su zadužene druge ADO.NET komponente, odnosno komponente koje mplementiraju direktan pristup podacima.

DataSet

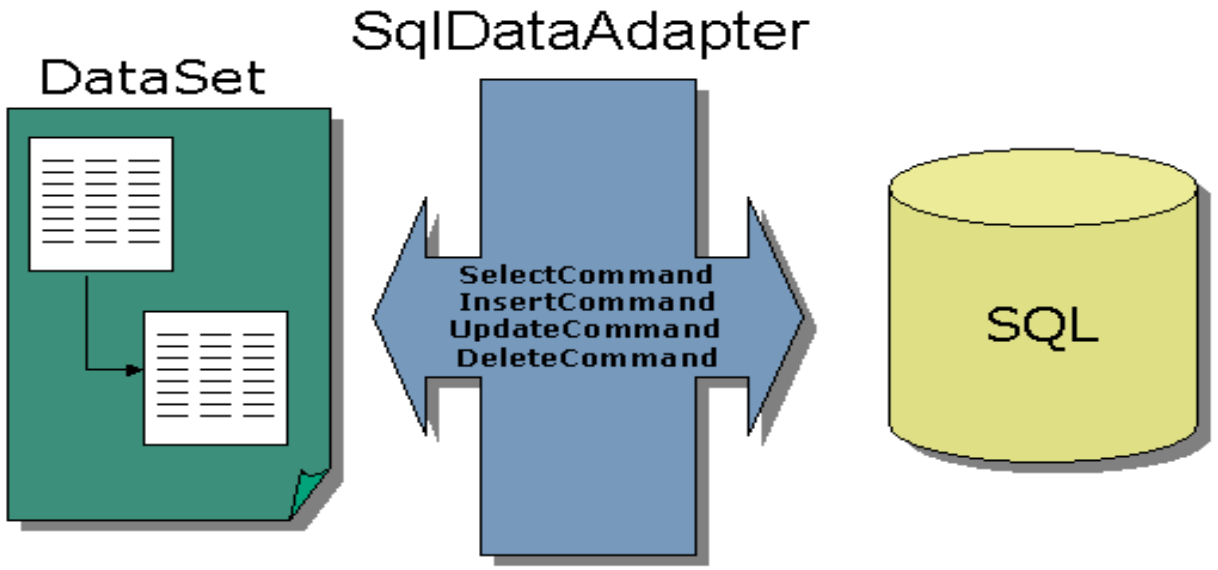
- Tipičan način korišćenja objekta DataSet:
  1. Obavezno uključiti biblioteku System.Data

```
using System.Data;
```

  2. Otvaranje konekcije sa izvorom podataka
  3. Učitavanje podataka u DataSet
  4. Zatvaranje konekcije
  5. Iako je konekcija sa izvorom podataka zatvorena, podaci se mogu koristiti bez ikakvih ograničenja. Ukoliko se pribavljeni podaci izmene, izmene je potrebno zapamtiti i u izvoru podataka. U tu svrhu se ponovo otvara konekcija sa izvorom podataka.
  6. Otvaranje konekcije sa izvorom podataka
  7. Izmenjeni podaci iz DataSet-a se upisuju u izvor podataka
  8. Zatvaranje konekcije

DataAdapter

- Objekat **DataAdapter** funkcioniše kao spona između DataSet objekta i objekta Connection odnosno izvora podataka.



DataAdapter

- Objekat DataAdapter definiše komande kojima se podaci učitavaju u objekat DataSet kao i komande kojima se podaci ažuriraju u izvoru podataka.
- Objekat DataAdapter ima četiri atributa, koji predstavljaju četiri objekata tipa Command, koji definišu četiri osnovne operacije za rad sa podacima.
  - SelectCommand – definiše komandu za učitavanje podataka
  - UpdateCommand – definiše komandu za modifikaciju podataka
  - InsertCommand – definiše komandu za kreiranje novih podataka
  - DeleteCommand – definiše komandu za brisanje podataka
- Samo atribut SelectCommand se mora eksplicitno postaviti.
- Ostale komande se krieraju po potrebi a u jednostavnijim slučajevima mogu se generisati automatski na osnovu šeme izvora podataka.

DataAdapter

```
//uspostavljanje konekcije sa bazom podataka
con = new OracleConnection(conString);
con.Open();

//kreiranje DataSet objekta
DataSet dsPreduzece = new DataSet();

//kreiranje objekat DataAdapter i definisanje SELECT komande
OracleDataAdapter daRadnici =
    new OracleDataAdapter("SELECT * FROM RADNIK", con);

//u DataSet-u se kreira tabela sa imenom RADNICI
//izvrsava se SELECT komanda DataAdapter objekta
//podaci se ucitavaju u DataSet u tabelu RADNICI
daRadnici.Fill(dsPreduzece, "RADNICI");

//zatvaranje konekcije sa bazom
con.Close();
```

DataAdapter

```
//uspostavljanje konekcije sa bazom podataka
con = new OracleConnection(conString);
con.Open();

//kreiranje DataSet objekta
DataSet dsPreduzece = new DataSet();

//kreiranje komande koja ce posluziti kao SELECT komanda DataAdapter-a
OracleCommand cmdSelect = new OracleCommand("SELECT * FROM RADNIK", con);

//kreiranje objekat DataAdapter i definisanje SELECT komande
OracleDataAdapter daRadnici = new OracleDataAdapter();
daRadnici.SelectCommand = cmdSelect;


//u DataSet-u se kreira tabela sa imenom RADNICI
//izvrsava se SELECT komanda DataAdapter objekta
//podaci se ucitavaju u DataSet u tabelu RADNICI
daRadnici.Fill(dsPreduzece, "RADNICI");

//zatvaranje konekcije sa bazom
con.Close();
```

DataAdapter

```
//zatvaranje konekcije sa bazom
con.Close();

//pristupanje podacima koji se nalaze u DataSet objektu
//pristupanje podacima u tabeli RADNICI
DataTable dtRadnici = dsPreduzece.Tables["Radnici"];

//1. način: koriscenjem foreach petlje
foreach (DataRow r in dtRadnici.Rows)
{
    String strIme = (String)r["LIME"];
    String strPrezime = (String)r["PREZIME"];
    String strInic = (String)r["SSLOVO"];

    Console.WriteLine(strIme + " " + strInic + " " + strPrezime);
}

//2. način: koriscenjem for petlje
for (int iCount = 0; iCount < dtRadnici.Rows.Count; iCount++)
{
    DataRow r = dtRadnici.Rows[iCount];
    String strIme = (String)dtRadnici.Rows[iCount][0];
    String strPrezime = (String)r[2];
    String strInic = (String)r[1];

    Console.WriteLine(strIme + " " + strInic + " " + strPrezime);

}
```

DataAdapter

```
//uspostavljanje konekcije sa bazom podataka
con = new OracleConnection(conString);
con.Open();

//kreiranje DataSet objekta
DataSet dsPreduzece = new DataSet();

//učitavanje podataka o radnicima i sektorima
OracleDataAdapter daRadnici =
            new OracleDataAdapter("SELECT * FROM RADNIK", con);
daRadnici.Fill(dsPreduzece, "RADNICI");

OracleDataAdapter daSektor =
             new OracleDataAdapter("SELECT * FROM SEKTOR", con);
daSektor.Fill(dsPreduzece, "SEKTORI");

//dodavanje veze između tabela RADNIK i SEKTOR
dsPreduzece.Relations.Add("Sef-Sektor",
                          dsPreduzece.Tables["RADNICI"].Columns["MATBR"],
                          dsPreduzece.Tables["SEKTORI"].Columns["MATBRR"]);


//zatvaranje konekcije sa bazom
con.Close();
```

DataAdapter

```
//za jednostavnije slucajeve
//CommandBuilder automatski generiše
//komande INSERT, DELETE i UPDATE
OracleCommandBuilder builder =
                new OracleCommandBuilder(daRadnici);

//kreiramo novu vrstu
DataRow rNew = dsPreduzece.Tables["RADNICI"].NewRow();

//upisujemo podatke i dodajemo novu vrstu
rNew["LIME"] = (Object)"Milan";
rNew["SSLOVO"] = (Object)"V";
rNew["PREZIME"] = (Object)"Petrović";
rNew["MATBR"] = (Object)1000;
rNew["MATBRS"] = (Object)333445555;
rNew["BRSEK"] = (Object)5;

//novu vrstu dodajemo u kolkeciju vrsta tabele RADNICI
dsPreduzece.Tables["RADNICI"].Rows.Add(rNew);

//da nema CommandBuilder-a došlo bi do greške
//jer Insert komanda nije definisana
daRadnici.Update(dsPreduzece, "RADNICI");
```

DataAdapter

```
//za jednostavnije slucajeve
//CommandBuilder automatski generiše
//komande INSERT, DELETE i UPDATE
OracleCommandBuilder builder =
                new OracleCommandBuilder(daRadnici);


//ažuriraju se informacije o plati
foreach (DataRow r in dsPreduzece.Tables["RADNICI"].Rows)
{
    //proverava se da li kolona ima vrednost NULL
    //kako bi se izbegle greske prilikom obrade
    if (r.IsNull("PLATA"))
        continue;

    double plata = (double)r["PLATA"];
    plata = plata + 1000;

    r["PLATA"] = plata;
}


//izmene se prosleđuju u bazu
daRadnici.Update(dsPreduzece, "RADNICI");
```

DataAdapter

```
con = new OracleConnection(conString);
con.Open();

//kreiranje DataSet objekta
DataSet dsPreduzece = new DataSet();

//učitavanje podataka o radnicima i sektorima
OracleDataAdapter daRadnici =
            new OracleDataAdapter("SELECT * FROM RADNIK", con);
//postavlja se da bi se učitale informacije o primarnom klucu
daRadnici.MissingSchemaAction = MissingSchemaAction.AddWithKey;

daRadnici.Fill(dsPreduzece, "RADNICI");


//za jednostavnije slucajeve
//CommandBuilder automatski generiše
//komande INSERT, DELETE i UPDATE
OracleCommandBuilder builder =
               new OracleCommandBuilder(daRadnici);

//na osnovu ključa pronalazi se vrsta koju brišemo
DataRow rDelete = dsPreduzece.Tables["RADNICI"].Rows.Find((Object)1000);

//brisanje vrste
rDelete.Delete();

//izmene se prosleđuju u bazu
daRadnici.Update(dsPreduzece, "RADNICI");
```

Parametrizovane komande

- Često je neophodno **parametrizovati upite** koji se prosleđuju bazi podataka na osnovu podataka koje su korisnici uneli u aplikaciju.
- Kako bi prosleđivanje podataka koje su korisnici uneli bilo što sigurnije, ADO.NET poseduje mogućnost kreiranja parametrizovanih komandi.
- Svaka ADO.NET komanda može da poseduje kolekciju parametara.
- Parametri zauzimaju određeno mesto u naredbi koju komanda treba da izvrši.
- U tekstu naredbe parametar se referencir akorišćenjem karaktera : navođenjem imena koje je parametru dodeljeno.

Parametrizovane komande

```
//kreiranje parametrizovanog upita
//SQL upit koji vraca sve radnike koji rade u zadatom sektoru
//i imaju platu vecu od neke specificirane granice
String strSQL = "SELECT * FROM RADNIK WHERE BRSEK=:sektor AND PLATA > :plata";
OracleCommand cmdParam = new OracleCommand(strSQL, con);

//kreiranje parametra
OracleParameter param1 = new OracleParameter("sektor", OracleDbType.Int32);

OracleParameter param2 = new OracleParameter();
param2.ParameterName = "plata";
param2.OracleDbType = OracleDbType.Double;

//dodavanje parametra u kolekciju parametara komande
cmdParam.Parameters.Add(param1);
cmdParam.Parameters.Add(param2);

//ucitavanje vrednosti parametara
Console.WriteLine("Uneti informaciju o sektoru: ");
param1.Value = (object)Console.ReadLine();

Console.WriteLine("Uneti informaciju o plati: ");
param2.Value = (object)Console.ReadLine();

OracleDataReader dr = cmdParam.ExecuteReader();
```