ADO .NET

Katedra za informatiku
Fakultet tehničkih nauka
Univerzitet u Novom Sadu

Uvod

- ADO.NET predstavlja skup klasa koje obezbeđuju servise za pristup podacima
- Klase su univerzalno dizajnirane, tako da omoguće rad sa različitim izvorima podataka
- Dve grupe servisa
  - Preuzimanje podataka iz izvora podataka
  - Pristup podacima

Preuzimanje podataka

- Data Provider
  - Povezivanje na izvor podataka
  - Izvršavanje upita
  - Preuzimanje podataka
- Specijalizovani Data Provider za svaki izvor podataka
- Ugrađeni Data Provider
  - Za Microsoft SQL Server (Prostor imena `System.Data.Sql`)
  - Za OLE DB pristup
  - Za ODBC pristup

Data Provider komponente

- Connection
- Command
- DataReader
- DataAdapter

Uspostavljanje veze – klasa Connection

- Connection
  - Uspostavljanje i upravljanje vezom sa izvorom podataka
  - Podržava upravljanje transakcijama
  - DbConnection – klasa predak za sve tipove konekcija
  - Različiti tipovi konekcija za različite izvore podataka
    - SqlConnection – za Microsoft SQL Server
    - OleDbConnection – za pristup izvoru podataka preko Microsoft OLE DB API
    - OdbcConnection – za pristup izvoru podataka preko ODBC API
    - OracleConnection – za pristup Oracle DBMS

Uspostavljanje veze - sintaksa

```
using (SqlConnection conn = new
  SqlConnection(connectionString))
{
  conn.Open();
  // rad sa bazom podataka
}
```

- **`Using`** izraz obezbeđuje automatsko zatvaranje konekcije pri uništavanju objekta (poziva se **`Dispose`** metoda, objekat mora da implementira **`IDisposable`** interfejs)

Uspostavljanje veze – Connection String

- Connection String – tekst koji specificira izvor podataka i parametre povezivanja
- Sintaksa (nazivi parametara) je različita za svaki tip izvora podataka
- Generalna sintaksa
  - *parametar1=vrednost1;parametar2=vrednost2;*
- Primer
  - ```
    conn.ConnectionString = @"
      Data Source = .\SQLExpress;
      Integrated Security = true;
      Initial Catalog = ComputerShop";
    ```
    ```

Uspostavljanje veze – Connection pooling

- Uspostavljanje veze troši vreme i resurse
- U praksi se u aplikaciji često vrši otvaranje/zatvaranje iste veze
- *Connection pooling* – mehanizam za optimizaciju korišćenja resursa pri povezivanju sa jednim izvorom podataka
- Automatski se inicijalizuje određen broj veza ka jednom izvoru podataka
- Kada zatraži otvaranje veze, aplikacija dobija na korišćenje jednu od veza iz *pool*-a, a pri zatvaranju konekcije vraća vezu u *pool*
- U ADO.NET *Connection pooling* mehanizam je automatski uključen

Connection pooling - Primer

```
using (SqlConnection conn = new SqlConnection(
   "Data Source = .\SQLExpress;
    Integrated Security = true;"))
{
   conn.Open(); // kreira se pool
}

using (SqlConnection conn = new SqlConnection( "
    Data Source = .\SQLExpress“
    Integrated Security = true;))
{
   conn.Open(); // posto je isti Connection String,
                // uzima se jedna konekcija iz ranije
                // kreiranog pool-a
}
```

Command

- Command
  - Izvršavanje upita nad izvorom podataka
  - Moguće je poslati i preuzeti parametre
  - Može se izvršavati u okviru transakcije
  - DbCommand – klasa predak za sve tipove komandi. Postoje različiti tipovi komandi za različite izvora podataka

Kreiranje komande

- Prva varijanta
  - Instanciranje komande i postavljanje konekcije

```
SqlCommand comm = new SqlCommand();
comm.Connection = conn;
```

- Druga varijanta
  - Odgovarajući tip komande se dobija iz konekcije

```
SqlCommand comm = conn.CreateCommand();
```

- Postavljanje upita koji komanda izvršava

```
comm.CommandText = "SELECT * FROM
                    STUDENT";
```

Izvršavanje komande

- Pozivom različitih metoda komande, zavisno od tipa podatka koji je rezultat izvršavanja:
  - ExecuteReader - Vraća **`DataReader`** objekat; Koristi se kada je rezultat upita kolekcija podataka

    `(SELECT * FROM STUDENT)`
  - ExecuteScalar - Vraća jedan podatak

    `(SELECT IME FROM STUDENT WHERE ID=1)`
  - ExecuteNonQuery – Za upite koji ne vraćaju podatke

    `(UPDATE STUDENT SET IME='PETAR' WHERE ID=1)`
  - ExecuteXMLReader – Vraća **`XMLReader`** objekat; Podržano samo za SqlCommand klasu

Tipovi komandi

- Tip se definiše svojstvom **`CommandType`** u komandi
- Podržani tipovi
  - Text
    - SQL upit koji treba biti izvršen nad izvorom podataka
  - StoredProcedure
    - Izvršavanje uskladištene procedure na serveru. CommandText je u tom slučaju naziv procedure.
  - TableDirect
    - Preuzimanje svih podataka iz jedne tabele. CommandText je u tom slučaju naziv tabele.

Parametrizovane komande [1]

- Komanda može da sadrži parametre upita
- Primer
  - `(SELECT IME FROM STUDENT WHERE ID=@ID_PRM)`
  - `@ID_PRM` je parametar koji se postavlja za komandu
  - Postavljanje parametra

```
String id_studenta = "1";
SqlParameter paramId = new
  SqlParameter("@ID_PRM", id_studenta);
comm.Parameters.Add(paramId);
```

Parametrizovane komande [2]

- Tipovi parametara
  - Input (ulazni - default), Output (izlazni), InputOutput (ponaša se i kao ulazni i kao izlazni), ReturnValue (sadrži povratnu vrednost komande – npr. kod poziva uskladištene procedure)
- Parametrizovane komande predstavljaju zaštitu od *SQL-Injection* napada
- Primer *SQL-Injection* napada:
  - Pri formiranju upita se ne koriste parametri u komandi:
    - **`Comm.CommandText = "SELECT * FROM STUDENT WHERE brojIndeksa = " + tbBrIndeksa.Text;`**
  - U polje “Broj indeksa” u aplikaciji korisnik unese tekst:

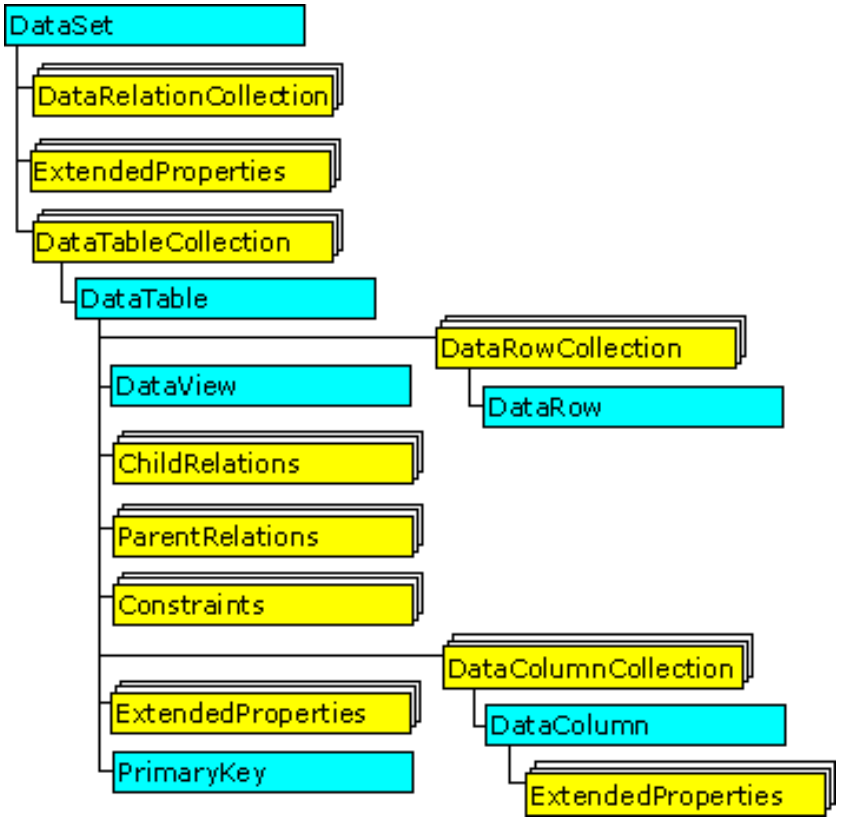
    123; DROP TABLE STUDENT
  - Pošto promenljiva **`tbBrIndeksa.Text`** sadrži uneseni SQL upit, taj SQL upit će biti izvršen i tabela sa spiskom studenata obrisana

DataReader

- Objektna struktura koja sadrži preuzete redove iz izvora podataka
- Više tipova zavisno od izvora podataka
  - SqlDataReader, OleDbDataReader, OdbcDataReader, ...
- Popunjavanje objekta
  - `SqlDataReader reader=comm.ExecuteReader();`
- Preuzimanje podataka
  - `reader.Read()` – čitač se pozicionira na naredni red
  - `reader.GetString(0)` – preuzimanje string vrednosti iz prve kolone reda
- Nakon korišćenja, potrebno je zatvoriti DataReader
  - `reader.Close();`

Pristup podacima

- DataSet - memorijska reprezentacija podataka preuzetih iz izvora podataka



DataSet

- Struktura DataSet objekta odgovara relacionoj bazi podataka
- DataSet sadrži tabele
  - DataTable klasa
  - Svojstvo Tables sadrži kolekciju DataTable objekata
  - Indekser za preuzimanje jedne tabele
    - `ds.Tables[0]`
    - `ds.Tables["Student"]`

DataSet

- Tabela sadrži redove
  - DataRow klasa predstavlja jedan red
  - Svojstvo Rows klase DataTable sadrži kolekciju redova
  - Indekser za preuzimanje jednog reda
    - `ds.Tables[0].Rows[3]`
- Tabela sadrži kolone
  - DataColumn klasa predstavlja jednu kolonu
  - Svojstvo Columns klase DataTable sadrži kolekciju kolona
  - Indekser za preuzimanje jedne kolone
    - `ds.Tables[0].Columns[3]`
    - `ds.Tables[0].Columns["Br_Indeksa"]`

DataSet

- Red sadrži vrednosti (ćelije u redu)
  - Svojstvo ItemArray sadrži kolekciju vrednosti
  - Indekser za preuzimanje vrednosti iz ćelije
    - `Object o = row[2];`
    - `Object o = row[“Br_Indeksa”];`

DataSet relacije

- Između tabela mogu se definisati relacije
  - Relacije omogućuju poštovanje referencijalnog integriteta među podacima u DataSet objektu
  - DataRelation klasa predstavlja relaciju između dve tabele
    - ParentTable – prva tabela u vezi
    - ChildTable – druga tabela u vezi
    - ParentColumns – kolone prve tabele koje formiraju relaciju
    - ChildColumns – kolone prve tabele koje formiraju relaciju
  - Svojstvo Relations klase DataSet sadrži kolekciju relacija
  - Indeksiranje veze
    - `ds.Relations[0]`
    - `ds.Relations["Veza_Student_Predmet"]`

DataSet – pristup podacima preko relacije

- Kada je formirana relacija, mogu se za određeni red jedne tabele preuzeti podaci iz druge tabele koji su povezani sa ovim redom
- DataRow klasa sadrži metodu GetChildRows koja vraća kolekciju redova iz druge tabele
- Primer
  - Preuzimanje predmeta određenog studenta

```
DataRow student = ds.Tables[0].Rows[0];
DataRow[] studentoviPredmeti =
  student.GetChildRows("Veza_Student_Predmet");
```

DataAdapter

- Kopiranje podataka iz izvora podataka u DataSet i obrnuto
- Popunjavanje DataSet objekta
  - Metoda **`Fill`** izvršava komandu predstavljenu svojstvom **`SelectCommand`**

```
DataSet ds = new DataSet();
SqlDataAdapter da = new SqlDataAdapter();
comm.CommandText = @"SELECT * FROM
                     STUDENT";
da.SelectCommand = comm;
da.Fill(ds,"Student");
```

DataAdapter – primary-key constraint

- Pri popunjavanju DataSet objekta automatski se kreira DataTable objekat, ali se ne kreiraju ograničenja (*constraints*) definisana nad tabelom u izvoru podataka
- Za kreiranje *primary-key constraint* (u kojoj koloni se skladište primarni ključevi), dve varijante:
  - Metoda **`FillSchema`**

```
da.FillSchema(ds, SchemaType.Source, "Student");
```

  - Svojstvo MissingSchemaActions

```
da.MissingSchemaAction =
   MissingSchemaAction.AddWithKey;
```

DataAdapter – popunjavanje iz više tabela

- DataSet je moguće popuniti putem različitih DataAdapter objekata od kojih svaki preuzima podatke iz jedne tabele

```
daStudenti.Fill(ds, "Student");
daPredmeti.Fill(ds, "Predmet“);
```

- Između dve tabele u DataSet-u je tada moguće definisati relaciju koja predstavlja spoljni ključ u izvoru podataka

```
ds.Relations.Add(“VezaStudentPredmet",
 ds.Tables[“Student"].Columns[“StudentID"],
 ds.Tables[“Predmet"].Columns[“PredmetID"]);
```

DataAdapter – ažuriranje izvora podataka

- Slanje izmenjenih podataka iz memorijske reprezentacije (DataSet objekta) u izvor podataka
- Metoda Update klase DataAdapter
  - Vrši slanje izmenjenih podataka
  - Izmenjeni podaci mogu biti poslani kao
    - Ceo DataSet
    - Jedna tabela (DataTable objekat)
    - Niz izmenjenih redova (DataRow objekata)

DataAdapter – ažuriranje izvora podataka

- Najpre se analiziraju izmene izvršene nad DataSet objektom
- Svaki red ima podatak o tome da li je izmenjen
  - U klasi DataRow svojstvo RowState (enumeracija DataRowState)
  - Mogući statusi:
    - Added – novi red dodan
    - Deleted – red obrisan
    - Detached – red kreiran, ali ne pripada nijednoj tabeli
    - Modified – red izmenjen
    - Unchanged – red neizmenjen

DataAdapter – ažuriranje izvora podataka

- Zavisno od statusa reda, izvršava se jedna od komandi
- U DataAdapter objektu se definišu komande za svaku akciju:
  - Svojstvo InsertCommand – ova komanda se izvršava za nove redove
  - Svojstvo DeleteCommand – ova komanda se izvršava za obrisane redove
  - Svojstvo UpdateCommand – ova komanda se izvršava za izmenjene redove

DataAdapter – ažuriranje izvora podataka

- Primer

```
SqlCommand delCom = conn.CreateCommand();
delCom.CommandText = @"DELETE FROM STUDENT
    WHERE ID=@ID";
SqlParameter idParam =
 delCom.Parameters.Add("@ID",SqlDbType.Int);
idParam.SourceColumn = "ID";

da.DeleteCommand = delCom;

ds.Tables[0].Rows[0].Delete();
da.Update(ds,"Student");
```

Upravljanje transakcijama

- Klasa SqlTransaction
  - BeginTransaction – pokretanje transakcije
  - Commit – potvrda transakcije
  - RollBack – poništavanje transakcije
- Kreiranje transakcije

```
SqlTransaction t = connection.BeginTransaction();
```

- Izvršavanje komande u transakciji

```
command.Transaction = t;
```