ADO.NET
Jelena Nikolic

Uvod i DataProvider klase

Šta je ADO.NET?

- ADO.NET je skup klasa koje omogućavaju pristup i manipulaciju podacima u .NET aplikacijama.
- Omogućava interakciju sa različitim izvorima podataka (npr. SQL Server, Oracle, XML).

ADO.NET

ADO.NET sadrži niz klasa koje se mogu podeliti u dva osnovna skupa :

- one koje pružaju pristup izvoru podataka (bazi)
- klase koje podržavaju DataSet model (lokalni podskup podataka iz baze)

Diskonektovan pristup bazi podataka

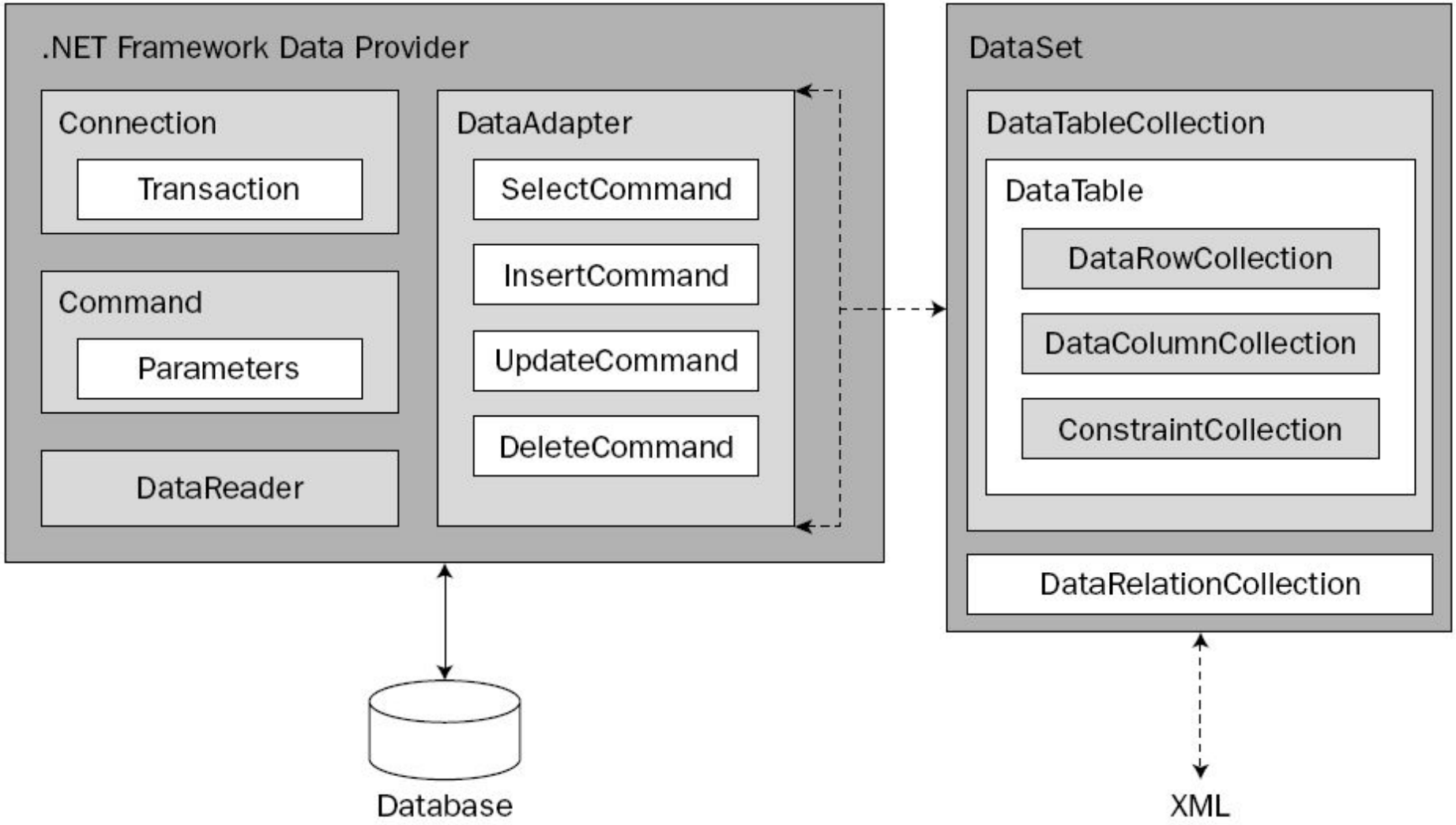
Jedna od prednosti ADO.NET-a je da podržava arhitekturu prekinute veze. Korišćenjem ove arhitekture aplikacije se povezuju sa serverom baze podataka samo da preuzmu ili ažuriraju podatke. Ovo omogućava da se smanji broj otvorenih veza prema raznim serverima baze podataka.

ADO.NET dozvoljava da se koriste naredbe za podatke kako bi se izvršili upiti SQL ili uskladištene procedure iz klijentske aplikacije. Ove naredbe za podatke omogućavaju da se izvrše naredbe SQL na serveru baze podataka lako i brzo.

Komponente ADO.NET-a

Pristup podacima u ADO.NET arhitekturi oslanja se na dve osnovne komponente:

- DataSet - komponenta koja smešta podatke na lokalnu memoriju računara
- Data Provider – skup komponenti koje posreduju u interakciji između aplikacije i baze podataka.


.NET Framework Data Provider
Connection
Transaction
Command
Parameters
DataReader
DataAdapter
SelectCommand
InsertCommand
UpdateCommand
DeleteCommand
DataSet
DataTableCollection
DataTable
DataRowCollection
DataColumnCollection
ConstraintCollection
DataRelationCollection
Database
XML

Ključne Komponente

Connection: Klasa za uspostavljanje veze sa bazom podataka.

Command: Klasa za izvršavanje SQL upita i procedura.

DataReader: Klasa za čitanje podataka iz baze u realnom vremenu.

DataSet: In-memory reprezentacija podataka koja podržava višekratne interakcije.

DataAdapter: Klasa koja omogućava popunjavanje DataSet-a i sinhronizaciju sa bazom.

1.Uspostavljanje Veze - SqlConnection

```
using (SqlConnection connection = new
SqlConnection(connectionString))
{
    connection.Open();
    // Operacije sa podacima
}
```

Uspostavljanje Veze - SqlConnection

SqlConnection je klasa koja predstavlja vezu sa SQL Server bazom podataka. Ona se koristi za otvaranje i zatvaranje veze sa bazom, kao i za izvođenje SQL komandi.

Ključne karakteristike SqlConnection:

1. **Connection String**: Da bi se uspostavila veza, potrebno je navesti niz podataka (connection string) koji sadrži informacije kao što su naziv servera, naziv baze, korisničko ime i lozinka.
2. **Otvaranje i zatvaranje veze**: SqlConnection koristi metode Open() i Close() za upravljanje vezama. U praksi, često se koristi blok using, koji automatski zatvara vezu kada se završi sa radom.
3. **Asinhrono povezivanje**: SqlConnection takođe podržava asinhrono povezivanje preko metoda poput OpenAsync().
4. **Transakcije**: Omogućava upravljanje transakcijama, što je korisno kada želite da grupišete više SQL operacija.
5. **Pooling**: SqlConnection koristi pooling, što znači da se veze ponovo koriste umesto da se stalno otvaraju i zatvaraju, što poboljšava performanse.

```
// skolski connection string

        public string SchoolKonekcija =
            "data source=o6LABXX\\SQLEXPRESS;"
+
            "database=db;" +
            "user id = sa;" +
            "password = ikt";
```

```
// Windows auth

public string HomeKonekcija =
      "data source=DESKTOP-JA1FBLC;" +
      "database=FashionFlex;" +
      "integrated security=SSPI;" +
      "persist security info=True;";
```

SqlCommand

```
SqlCommand command = new SqlCommand("SELECT * FROM Users", connection);
```

Izvršavanje SQL Komandi - SqlCommand

SqlCommand je klasa koji se koristi za izvršavanje SQL izraza ili skladištenih procedura. Omogućava izvršavanje različitih tipova SQL komandi, kao što su SELECT, INSERT, UPDATE i DELETE.

Ključne karakteristike SqlCommand:

1. **CommandText**: Svojstvo koje sadrži SQL izraz ili naziv skladištene procedure koja će se izvršiti.
2. **Parameters**: SqlCommand omogućava korišćenje parametara u SQL upitima, što pomaže u zaštiti od SQL injekcija i poboljšava performanse.
3. **Execute metodi**: SqlCommand ima više metoda za izvršavanje, uključujući ExecuteReader(), ExecuteScalar(), i ExecuteNonQuery(), u zavisnosti od tipa rezultata koji očekujete.
4. **Connection**: SqlCommand mora biti povezan sa SqlConnection da bi mogao da izvrši komandu.

Ključne metode

1. **ExecuteReader()**: Koristi se za izvršavanje SQL upita koji vraća više redova, obično se koristi za SELECT upite.
2. **ExecuteScalar()**: Izvršava upit koji vraća jedan vrednost (npr. rezultat agregatne funkcije kao što je COUNT).
3. **ExecuteNonQuery()**: Izvršava SQL komande koje ne vraćaju podatke (npr. INSERT, UPDATE, DELETE) i vraća broj redova koji su promenjeni.
4. **Add()**: Metoda koja se koristi za dodavanje parametara komandi.

Čitanje Podataka - SqlDataReader

DataReader je objekat koji omogućava brzo i efikasno čitanje podataka iz SQL Server baze podataka. On se koristi kada je potrebna samo čitajuća operacija, bez potrebe za ažuriranjem podataka.

Ključne karakteristike DataReader-a:

1. **Brzina**: DataReader je veoma brz jer koristi minimalnu količinu resursa i ne učitava sve podatke u memoriju odjednom. Umesto toga, čita redove jedan po jedan.
2. **Forward-only**: Podaci se mogu kretati samo unapred. To znači da ne možete ići unazad kroz rezultate.
3. **Nezavisnost**: Kada koristite DataReader, nemate potrebu da zadržavate vezu sa bazom podataka otvorenom dok čitate podatke, ali je potrebno da se veza drži otvorenom dok se podaci čitaju.
4. **Podrška za različite tipove podataka**: Može čitati različite tipove podataka, kao što su stringovi, brojevi, datumi itd.

Ključne metode i svojstva

- **Read()**: Ova metoda se koristi za pomeranje na sledeći red u skupu podataka. Vraća true ako je uspešno učitan red, a false ako su podaci završeni.
- **GetXXX()**: Ove metode (npr. GetString(int i), GetInt32(int i), GetDateTime(int i), itd.) se koriste za pristup vrednostima u trenutnom redu, gde je i indeks kolone.
- **FieldCount**: Svojstvo koje vraća broj kolona u trenutnom DataReader.

```
SqlCommand command = new SqlCommand("SELECT * FROM YourTable", connection);

// Izvršite komandu i dobijte DataReader
using (SqlDataReader reader = command.ExecuteReader())
{
    // Prolazite kroz podatke
    while (reader.Read())
    {
        // Čitanje podataka iz redova
        int id = reader.GetInt32(0); // Prva kolona kao int
        string name = reader.GetString(1); // Druga kolona kao string
        DateTime date = reader.GetDateTime(2); // Treća kolona kao DateTime
```

Prednosti ADO.NET

Brz pristup podacima.

Podrška za različite izvore podataka.

Mogućnost rada sa offline podacima preko DataSet-a.

Nastavak DataSet...