2

Veoma jednostavni primeri povezivanja

O temi povezanosti (connectivity) napisane su cele knjige. U prvih nekoliko knjiga ovog serijala se ova tema namerno potpuno izbegavala u osnovnom verovanju da je to jednostavno jedna ogromna tema — takva koja može pravilno da se obradi jedino u celoj knjizi. To priznajemo, ali imati SQL Server a ne omogućiti programima da se povežu sa njim je isto kao da uopšte nemate SQL Server. Istina, možete da se prijavite u Management Studio i direktno pišete upite, ali realnost jeste da velika većina vaših korisnika u suštini nikada ne vidi bazu podataka direktno.

Zbog samog naslova *Od početka*, ipak ću dotaći ovu temu tek na veoma ograničen način koji će služiti čisto kao kratka referenca pošto pregledate druge izvore, ili možda kao nagoveštaj svega što to obuhvata. Ako ozbiljno nameravate da se odlikujete u korišćenju SQL Servera ili bilo kojeg drugog rešenja podataka zasnovanog na povezanosti, toplo vam preporučujem da nabavite posebnu knjigu o pristupu podacima i povezivanju.

Ovaj dodatak je niz krajnje jednostavnih primera povezivanja u kojima se koriste dva klijentska jezika i dve funkcije u dva različita modela povezanosti. Ubaciću osnovne primere povezivanja u svakoj kombinaciji jezik/model, a dalje istraživanje prepuštam vama.

NAPOMENA *Ne mogu previše da naglasim do koje mere su ovi primeri zaista elementarni. Za povezivanje ima mnogo, mnogo toga da se bira i optimizuje. Svaki model povezanosti ima vlastite smicalice, a različiti jezici ponekad doprinose vlastitu notu. Ako ste očekivali više od osnovnog prikaza kako se to radi (ili možda primedbe „U da, taj deo sam zaboravio"), moraćete da potražite knjigu koja se isključivo bavi povezivanjem za vašu konkretnu kombinaciju jezika i modela povezanosti.*

Ovde su obrađeni modeli povezanosti:

- ADO.NET
- ADO

U okviru primera za ADO.NET, videćete par različitih pristupa korišćenju omogućenih funkcija.

NAPOMENA *Svi primeri koda koje ovde dajem napravljeni su u Visual Studiju 2010. Osnovni principi koji su ovde izloženi trebalo bi da funkcionišu i pod drugim verzijama Visual Studija ili drugim postojećim editorima za jezike. Moguće je da sam ubacio neku dosetku iz .NET Frameworka 4.0 a da nisam ni primetio. (U tom slučaju se izvinjavam onima koji rade pod drugim verzijama .NET Frameworka.)*

NEKI OPŠTI KONCEPTI

Dok se nisam previše udubio u „sam kôd", ovde je nekoliko ključnih konstrukata koje morate da shvatite. Tokom vremena je više različitih modela povezivanja dolazilo i odlazilo; slušaćete, između ostalih, o imenima kao što su OLE-DB, ODBC, i, naravno, ADO. Modeli povezivanja koji sada preovlađuju kod Microsofta su ADO.NET ili LINQ. Mada to važi već nekoliko godina, ipak je verovatnije (kako sa tim stvarima obično biva) da će, dok izađe sledeća verzija SQL Servera, to biti neki drugi model.

I pored toga što je toliko različitih modela došlo i prošlo, neki koncepti izgleda uvek postoje u svakom objektnom modelu – sada ćemo ih sasvim ukratko pogledati:

- **Konekcija (connection):** Objekat konekcije je uglavnom to što biste i pomislili – objekat koji definiše i uspostavlja samu komunikcionu vezu sa vašom bazom podataka. Vrste parametarskih informacija koje će postojati za objekat konekcije obuhvataju pojmove kao što su korisničko ime, lozinka, baza podataka i server sa kojim želite da se povežete. Neki od metoda koji postoje biće povezivanje (connect) i kidanje veze (disconnect).
- **Komanda (command):** Ovo je objekat koji prenosi informaciju o tome šta hoćete da uradite. Neki objektni modeli ne sadrže ovaj objekat, odnosno, ne izlažu ga, ali je koncept uvek tu. (Ponekad je sakriven kao metod objekta konekcije.)
- **Skup podataka (data set):** To je rezultat upita – to jest, ako upit vraća podatke. Neki upiti u kojima izvršavate, na primer, jednostavnu naredbu `INSERT`, ne vraćaju rezultate, ali, ako se rezultati vraćaju, postojaće neka vrsta skupa podataka (ponekad se zove rezultujući skup (result set) ili skup zapisa (recordset)) u koji ih upit vraća. Objekti skupa podataka uglavnom će omogućavati da se krećete po njihovim zapisima (često jedino u rastućem smeru ali je obično omogućeno podešavanje da bi se postiglo robustnije pozicioniranje). Oni takođe uglavnom omogućavaju da se podaci ažuriraju, insertuju i brišu.

PRIMERI POVEZANOSTI

U ovom odeljku ću ponuditi neke veoma, veoma jednostavne primere kako da se povežete pomoću dva uobičajena jezika koji sada preovlađuju – C# i VB.NET. Za svaki od ova dva jezika, pokazaću dva primera – po jedan za svaku od dve različite vrste operacija (pribavljanje jednostavnog skupa podataka i izvršavanje upita koji ne vraća skup podataka).

Povezivanje u jeziku C#

C# je prilično jasan jezik i relativno se lako uči, što je bio značajan deo projektne specifikacije za njega, a ima tu prednost da sadrži neke koncepte zasnovane na jeziku C i takođe mu je relativno blizak po sintaksi (pa se lako prelazi sa jednog na drugi).

Listing C-1 je jednostavan ADO.NET primer napisan u jeziku C#, u kojem se pravi konekcija i izvršava jedna komanda kojom se vraća skup podataka. Ovaj primer radi u kontekstu konzolne aplikacije Visual Studija. Imajte na umu da vaša instalacija SQL Server Data Tools nije dovoljna da bi se pravile konzolne aplikacije; morate da imate primerak Visual Studija da biste ovaj primer koristili a ne samo čitali kao referencu.

LISTING C-1: Vraćanje skupa podataka

```
using System;
using System.Data.SqlClient;
class Program
{
    static void Main()
    {
        // Pravljenje osnovnih stringova da biste ih posmatrali
        // odvojeno od komandi u kojima se izvršavaju

        // Integrisana bezbednost – skinuti znak za komentar sa sled. reda
        string strConnect = "Data Source=(local);Initial
Catalog=master;Integrated Security=SSPI";
        // SQL bezbednost – skinuti znak za komentar sa sledećeg reda
        //string strConnect = "Data Source=(local);Initial Catalog=master;User
Id=sa;Password=MyPass";

        string strCommand = "SELECT Name, database_id as ID FROM sys.databases";

        SqlDataReader rsMyRS = null;

        SqlConnection cnMyConn = new SqlConnection(strConnect);

        try
        {
            // "Otvaranje" konekcije (ovo je u stvari prvi put da se
            // kontaktira sa serverom baze podataka)

            cnMyConn.Open();

            // Sada se pravi objekat komande
            SqlCommand sqlMyCommand = new SqlCommand(strCommand, cnMyConn);
```

nastavlja se

LISTING C-1 (*nastavak)*

```
            // Pravi se rezultujući skup
            rsMyRS = sqlMyCommand.ExecuteReader();

            //Pokažite šta ste dobili
            while (rsMyRS.Read())
            {
                // Ispisati prvu kolonu (ovde redni brojevi)
                // Možemo takođe da referišemo kolonu po imenu
                Console.WriteLine(rsMyRS["Name"]);
            }
            Console.WriteLine();
            Console.WriteLine("Press any key to continue...");
            Console.ReadKey();
        }
        finally
        {
            // Čišćenje za sobom
            if (rsMyRS != null)
            {
                rsMyRS.Close();
            }

            if (cnMyConn != null)
            {
                cnMyConn.Close();
            }
        }
    }
}
```

Listing C-2 je sličan listingu C-1, ali je zamišljen da izvrši SQL komandu koja ne vraća podatke (na primer, komandu `INSERT`, `UPDATE`, `DELETE` ,ili snimljenu proceduru koja ne proizvodi izlaz).

LISTING C-2: Izvršavanje komandi bez skupa podataka

```
using System;
using System.Data.SqlClient;

class Program
{
    static void Main()
    {
        // Pravljenje osnovnih stringova da biste ih posmatrali
        // odvojeno od komandi u kojima se izvršavaju

        // Integrisana bezbednost - skinuti znak za komentar sa sledećeg reda
```

```
        string strConnect = "Data Source=(local);Initial
Catalog=master;Integrated Security=SSPI";
        // SQL bezbednost - skinuti znak za komentar sa sledećeg reda
        //string strConnect = "Data Source=(local);Initial Catalog=master;User
Id=sa;Password=MyPass";

        string strCommand = "CREATE TABLE Foo(Column1 INT NOT NULL PRIMARY KEY)";
        string strCommand2 = "DROP TABLE Foo";

        SqlConnection cnMyConn = new SqlConnection(strConnect);

        try
        {
            // "Otvaranje" konekcije (ovo je u stvari prvi put da se
            // kontaktira sa serverom baze podataka)
            cnMyConn.Open();

            // Sada se pravi objekat komande
            SqlCommand sqlMyCommand = new SqlCommand(strCommand, cnMyConn);

            // Izvršavanje komande
            sqlMyCommand.ExecuteNonQuery();
            Console.WriteLine("Table Created");
            Console.WriteLine("Press enter to continue (možete najpre da
proverite da li je tamo) ");
            Console.ReadLine();

            // Menjanje komande
            sqlMyCommand.CommandText = strCommand2;

            sqlMyCommand.ExecuteNonQuery();

            Console.WriteLine("It's gone");

            Console.WriteLine();
            Console.WriteLine("Press any key to continue...");
            Console.ReadKey();
        }
        finally
        {
            // Čišćenje za sobom
            if (cnMyConn != null)
            {
                cnMyConn.Close();
            }
        }
    }
}
```

Povezivanje u jeziku VB.NET

VB.NET još uvek zamenjuje starodrevni Visual Basic. Mada ja lično sve više koristim C# , VB još uvek ima dovoljno korisnika zbog kojih ga vredi ovde prikazati.

U Listingu C-3 imamo primer koji po funkcionalnosti odgovara Listingu C-1. Napisan u VB.NET-u kao konzolna aplikacija, ovaj program će izvršiti čitanje baze podataka, pa se očekuje vraćanje podataka.

LISTING C-3: Vraćanje skupa podataka

```
Imports System
Imports System.Data
Imports System.Data.SqlClient

Module Program
    Sub Main()

        'Pravljenje osnovnih stringova da biste ih posmatrali
        'odvojeno od komandi u kojima se izvršavaju
        ' Integrisana bezbednost - skinuti znak za komentar sa sledeća 2 reda
        '     za Windows proveru autentičnosti
        Dim strConnect As String = _
            "Data Source=(local);Initial Catalog=master;Integrated
Security=SSPI"
        ' SQL bezbednost - skinuti znak za komentar sa sledeća 2 reda za
        '   SQL Server proveru autentičnosti

        'Dim strConnect As String = _
        '   "Data Source=(local);Initial Catalog=master;User
Id=sa;Password=MyPass"

        Dim strCommand As String = _
            "SELECT Name, database_id as ID FROM sys.databases"

        Dim rsMyRS As SqlClient.SqlDataReader

        Dim cnMyConn As New SqlClient.SqlConnection(strConnect)

        ' "Otvaranje" konekcije (ovo je u stvari prvi put da se
        ' kontaktira sa serverom baze podataka)
          cnMyConn.Open()

        ' Sada se pravi objekat komande
        Dim sqlMyCommand As New SqlClient.SqlCommand(strCommand, cnMyConn)

        ' Pravi se rezultujući skup
        rsMyRS = sqlMyCommand.ExecuteReader()

        'Pokažite šta ste dobili
        Do While rsMyRS.Read
            ' Ispisati prvu kolonu (ovde redni brojevi)
            ' Možemo takođe da referišemo kolonu po imenu
            Console.WriteLine(rsMyRS("Name"))
        Loop

        Console.WriteLine()
        Console.WriteLine("Press any key to continue...")
```

```
        Console.ReadKey()

        ' Čišćenje za sobom
        rsMyRS.Close()
        cnMyConn.Close()

    End Sub

End Module
```

Ovaj poslednji primer, prikazan kao Listing C-4, na sličan način odgovara prethodnim primerima u jeziku C#. Kao i u Listingu C-2, u ovom primeru se pravi konekcija, zatim se izvršava SQL komanda od koje se ne očekuje da proizvede rezultujući skup.

LISTING C-4: Izvršavanje komandi bez skupa podataka

```
Imports System
Imports System.Data
Imports System.Data.SqlClient

Module Program

    Sub Main()

        'Pravljenje osnovnih stringova da biste ih posmatrali
        'odvojeno od komandi u kojima se izvršavaju

        ' Integrisana bezbednost - skinuti znak za komentar sa sledeća 2 reda
        Dim strConnect As String = _
            "Data Source=(local);Initial Catalog=master;Integrated
Security=SSPI"
        ' SQL bezbednost - skinuti znak za komentar sa sledeća 2 reda
        ' Dim strConnect As String = _
        ' "Data Source=(local);Initial Catalog=master;User Id=sa;Password=MyPass"

        Dim strCommand As String = _
            "CREATE TABLE Foo(Column1   INT NOT NULL PRIMARY KEY)"
        Dim strCommand2 As String = "DROP TABLE Foo"

        Dim cnMyConn As New SqlClient.SqlConnection(strConnect)

        ' "Otvaranje" konekcije (ovo je u stvari prvi put da se
        ' kontaktira sa serverom baze podataka)
        cnMyConn.Open()

        ' Sada se pravi objekat komande
        Dim sqlMyCommand As New SqlClient.SqlCommand(strCommand, cnMyConn)

        ' Izvršavanje komande
        sqlMyCommand.ExecuteNonQuery()

        Console.WriteLine("Table Created")
```

nastavlja se

LISTING C-4 (*nastavak)*

```
        Console.WriteLine("Press enter to continue (možete najpre da proverite
da li je tamo)")
        Console.ReadLine()

        ' Menjanje komande
        sqlMyCommand.CommandText = strCommand2

        sqlMyCommand.ExecuteNonQuery()

        Console.WriteLine("It's gone")

        Console.WriteLine()
        Console.WriteLine("Press any key to continue...")
        Console.ReadKey()

        ' Čišćenje za sobom
        cnMyConn.Close()

    End Sub

End Module
```

NEKOLIKO REČI O LINQ

LINQ je, dok ovo pišem, kako bi se reklo „nova moda". Za retke od vas koji niste čuli za LINQ, to je metod pristupa zasnovan na pojmu homogenih izvora podataka. LINQ ima vlastiti jezik upita – veoma sličan SQL-u, a ipak dovoljno različit da ima nešto posla ako pokušavate da konvertujete jedno u drugo.

Mada LINQ svakako ima priličan impuls kao moćan način za pristupanje podacima, ja ostajem donekle skeptičan. Nemojte me pogrešno shvatiti. LINQ nudi neke zaista elegantne stvari, ali tokom godina sam često viđao kako dolaze i prolaze pojmovi tipa LINQ-a, pa sam naučio da se mnogo ne uzdam u dugovečnost takvih modela.

Potpun opis LINQ-a daleko prevazilazi namenu ove knjige, a sintaksa upita se toliko razlikuje da ne bih hteo ovde da zbunjujem početnika novim zaokretima. Pošto smo to razjasnili, vredi naglasiti nešto o tome šta LINQ jeste a šta nije.

LINQ koristi model čvrstih pravila za tipove da bi se omogućili upiti nad širokom paletom podataka. Dok je za ovu knjigu najbitniji pristup SQL Serveru, važno je da se naglasi da je LINQ veoma sposoban za povezivanje sa izvorima podataka koji nisu tabelarni. LINQ može da se koristi za pristupanje XML-u (bolji izbor je uglavnom XPath, ali možda hoćete da koristite LINQ zbog konzistentnosti sa pristupanjem ostalim podacima), a može da se koristi i za pristupanje kolekcijama objekata. Ukratko, LINQ je veoma fleksibilan što se tiče vrste podataka koje vraća.

Osim fleksibilnosti, osnovna prednost koju vidim za LINQ je to što koristi čvrsta pravila za tipove (strongly typed). Prethodni modeli pristupa su koristili model vezivanja u kojem vaš program nije mogao stvarno da se pouzda u tip podatka koji će mu se vratiti sve do vreme-

na izvršavanja. LINQ primenjuje daleko ranije vezivanje za podatke, pa prema tome može ranije u procesu kodiranja da primeti mnoge vrste neusklađenosti tipova ili slične greške. To, naravno, nije potpuno sigurno, ali predstavlja napredak u tom konkretnom području.

Ako razmišljate o tome da za sva pristupanja podacima koristite LINQ, mogu jedino da vam savetujem da pre odluke dobro razmislite *zašto* LINQ predstavlja vaš izbor. Nemojte da to bude zato što je to najnovija „silna" stvar, već radije zato što bolje od ostalih raspoloživih opcija odgovara vašim konkretnim potrebama.

BIRANJE PRISTUPNOG METODA

Doživeo sam dolazak i odlazak mnogih pristupnih metoda, a većinu programera sa kojima sam radio uvek je nešto privlačilo najnovijim metodima. Od svih postojećih mogućnosti, birajte metod pristupa po tome koliko odgovara onome što radite, a ne po tome što je najnoviji, najelegantniji, ili što se o njemu najviše priča u štampi. Primena pogrešnog metoda može da napravi ogroman tehnički problem ako kasnije uvidite da morate da ga izbacite.