Datenbankzugriff

Ingo Köster

Diplom Informatiker (FH)

Was ist ADO.NET?

- Sammlung von Klassen für den Zugriff auf relationale Datenbanken
- Klassen für Verbindungen, Abfragen und Ergebnisse
- Einheitliche API für den Zugriff auf Daten aus unterschiedlichen Datenquellen
- Stellen Verbindung zu einer Datenbank her

ADO.Net Datenprovider

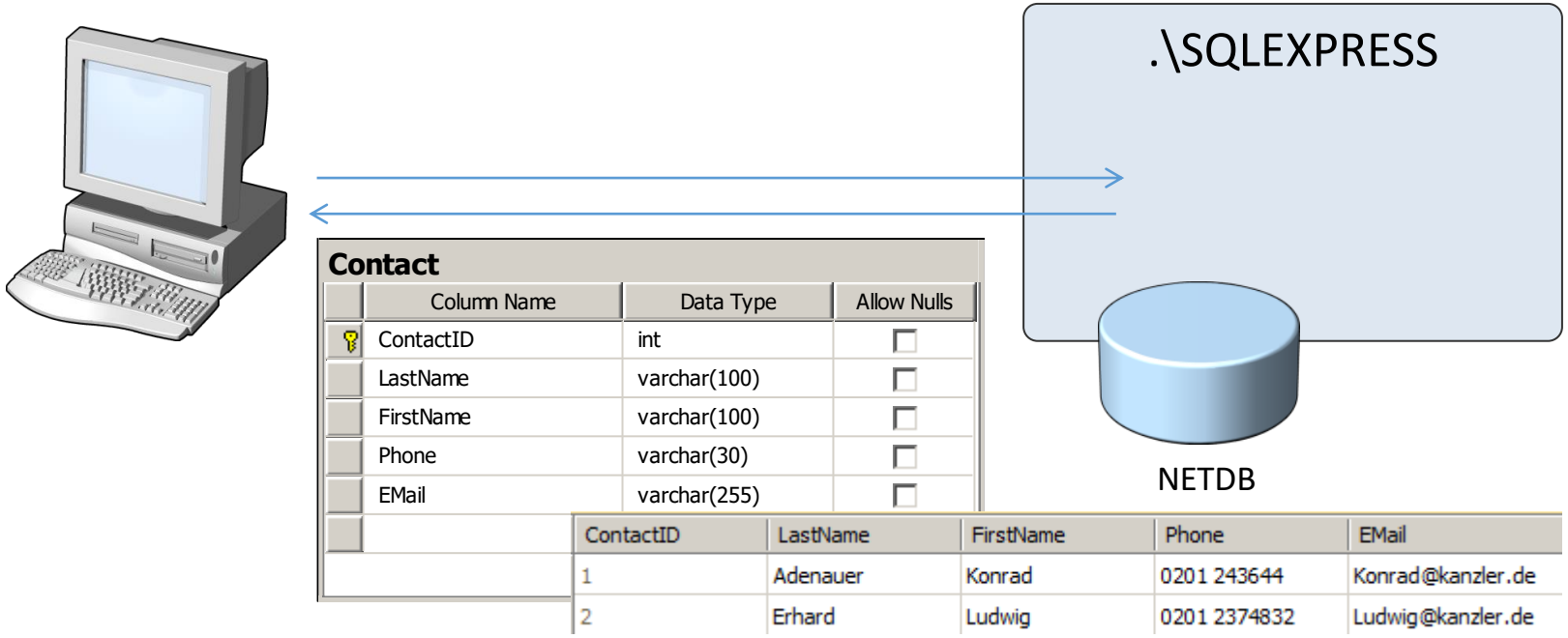
- Der Datenprovider ist die Schnittstelle zu einer Datenbank
- Er muss daher die Datenbank kennen
- D.h. für unterschiedliche Datenbanken existieren individuelle Datenprovider
  - Z.B. Datenprovider für Microsoft SQL Server, MySQL, Oracle, etc.

ADO.Net Datenprovider

- Kernkomponenten der .NET-Datenprovider
  - Connection
    - Stellt eine Verbindung zur Kommunikation mit einer Datenband her
  - Command
    - Benötigt ein Connection-Objekt
    - Führt z.B. Abfragen, Anweisungen und gespeicherte Prozeduren aus
      - Z.B. SELECT-, UPDATE- oder DELETE-Kommandos
  - DataReader
    - Benötigt ein Command-Objekt
    - Zum Lesen von Datensätzen die z.B. durch eine Abfrage oder Prozedur ermittelt wurden

Zugriff auf eine SQL Server Datenbank

- Beispiel:
  - Lokaler SQL Server (z.B. Express-Edition)
  - Datenbank heißt NETDB, Tabelle `Contact`
  - Windows- und SQL Server Authentifizierung ist aktiv



Contact

	Column Name	Data Type	Allow Nulls
🔑	ContactID	int	☐
	LastName	varchar(100)	☐
	FirstName	varchar(100)	☐
	Phone	varchar(30)	☐
	EMail	varchar(255)	☐

ContactID	LastName	FirstName	Phone	EMail
1	Adenauer	Konrad	0201 243644	Konrad@kanzler.de
2	Erhard	Ludwig	0201 2374832	Ludwig@kanzler.de

Grundsätzlicher Ablauf

- Verbindung zur Datenbank aufnehmen
  - Connection-String definieren
  - `SqlConnection`-Objekt erstellen
  - Verbindung öffnen
- `SqlCommand`-Objekte erstellen
  - `CommandText`-Eigenschaft festlegen
  - `CommandType`-Eigenschaft festlegen
- Kommando zur Datenbank senden und Ergebnisse verarbeiten
  - `SqlDataReader` auslesen
  - Änderungen an Daten senden

Zur Datenbank verbinden

› Einen Connection-String für die Datenbank angeben

› Ein SqlConnection-Objekt erzeugen

› Die Methode Open( ) des SqlConnection-Objekts aufrufen

```
string connString = @"Data Source=.\SQLEXPRESS;" +
                    "Initial Catalog=NETDB;" +
                    "Integrated Security=true;";
SqlConnection connection = new SqlConnection(connString);
connection.Open();
```

Daten lesen

- Ein SqlCommand-Objekt erzeugen und den Text des SQL-Kommandos angeben
  - `SqlCommand sql = connection.CreateCommand();`
  - `sql.CommandText = "SELECT LastName, FirstName, EMail FROM Contact";`
- Methode `ExecuteReader()` des SqlCommand-Objekts aufrufen
- Ergebnisse verarbeiten, z.B. in einer while-Schleife

Daten lesen – Beispiel

```
SqlCommand sql = connection.CreateCommand();
sql.CommandText = "SELECT LastName, FirstName, EMail FROM Contact";

SqlDataReader reader = sql.ExecuteReader();

while (reader.Read())
{
      Console.WriteLine("{0} {1} - {2}",
            reader["FirstName"],
            reader["LastName"],
            reader["EMail"]);
}
reader.Close();
```

Daten einfügen

- Ein SqlCommand-Objekt erzeugen und den Text des SQL-Kommandos angeben
  - `SqlCommand sql = connection.CreateCommand();`
  - `sql.CommandText = "INSERT INTO Contact (LastName,FirstName,Phone,EMail) VALUES ('Kiesinger','Kurt-Georg','0201 74832','Kurt@kanzler.de')";`
- Methode `ExecuteNonQuery()` des SqlCommand-Objekts aufrufen
- `ExecuteNonQuery()` wird für alles außer SELECT-Kommandos verwendet

Daten einfügen - Beispiel

```
SqlCommand sql = connection.CreateCommand();
sql.CommandText = "INSERT INTO Contact
(LastName,FirstName,Phone,EMail) VALUES ('Kiesinger','Kurt-
Georg','0201 74832','Kurt@kanzler.de')";


int anzahl = sql.ExecuteNonQuery();


if (anzahl == 1)
{
     Console.WriteLine("Ein Datensatz erfolgreich eingefügt!");
}
```

Daten Ändern, Skalare Ergebnisse und Parameter

- Für UPDATE und DELETE Anweisungen werden ebenfalls `CommandText` Objekte erzeugt
- Für Skalare Ergebnisse (AVG, SUM, etc.) gibt es die Methode `ExecuteScalar()` des SqlCommand-Objekts
  - `ExecuteScalar()` liefert genau einen Wert zurück

- Bei variablen Eingaben werden SQL Parameter verwendet, um vor SQL-Injection zu schützen

```
var sql = @"SELECT * FROM Product WHERE Price BETWEEN @min AND @max";
…
command.Parameters.AddWithValue("min", 100);
command.Parameters.AddWithValue("max", 200);
```

Gespeicherte Prozeduren, langlaufende Abfragen und Provider unabhängiger Code

- Gespeicherte Prozeduren werden ebenfalls mit einem `SqlCommand` aufgerufen
  - Das SQL-Kommando besteht nur aus dem Namen der Prozedur
  - Parameter und Werte für die Prozedur werden übergeben

- Für lang laufenden Abfragen können asynchrone Methoden verwendet werden

- Provider unabhängiger Code erlaubt es, eine andere Datenbank zu verwenden, ohne den Code ändern zu müssen
  - Es gibt Einschränkungen