

C# Advanced ADO.NET connected

Koen Bloemen



**DE HOGESCHOOL
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt

www.pxl.be





SqlConnection
SqlCommand
SqlDataReader
SqlParameter
SqlDataAdapter
DataView

ADO.NET connected

Connected ADO.NET gebruikt SQL Server als database

- <https://www.microsoft.com/nl-be/sql-server/sql-server-downloads>
- Installeer SQL Server Express editie
- Kies voor Basic of Custom installatie
- Installeer ook Microsoft SSMS
- ConnectionString na installatie:
Server=localhost\SQLEXPRESS;Database=master;Trusted_Connection=True;

ADO.NET connected

- Klassen waarbij applicatie continu verbonden is met een database

DataAdapter	DataSet vullen via query of stored procedure SelectCommand, InsertCommand, UpdateCommand, DeleteCommand
DataReader	Read-only en forward-only snelle gegevenstoegang tot database
Command	SQL commando
Connection	Verbinding met database

ADO.NET Managed Providers

- Klassen die toegang geven tot SQL Server, OLE DB, Oracle 8i en hoger
- Delen dezelfde interface
- Voor SQL Server: `using System.Data.SqlClient;`

SqlConnection	Verbinding naar SQL Server gegevensbron via ConnectionString
SqlCommand	(Transact-)SQL commando of stored procedure voor SQL Server
SqlDataReader	Verzameling SqlCommand's en SqlConnection om DataSet te vullen
SqlParameter	Parameter op veilige manier meegeven aan SqlCommand
SqlError	Verzamelt foutmeldingen in ADO.NET toepassing

SqlConnection

- Regelt communicatie tussen applicatie en SQL Server.
- Verbinding naar SQL Server database via een `ConnectionString`.
- `ConnectionString` bestaat uit

Data Source (of Server)	Servernaam, IP-adres of local voor lokale server.
Initial Catalog (of Database)	Naam van database op server
Integrated Security	SSPI (Security Support Provider Interface)
User ID	Gebruikersnaam
Password	Wachtwoord voor gebruikte User ID
AttachDbFilename	SQL Server Express: pad naar bestand van database

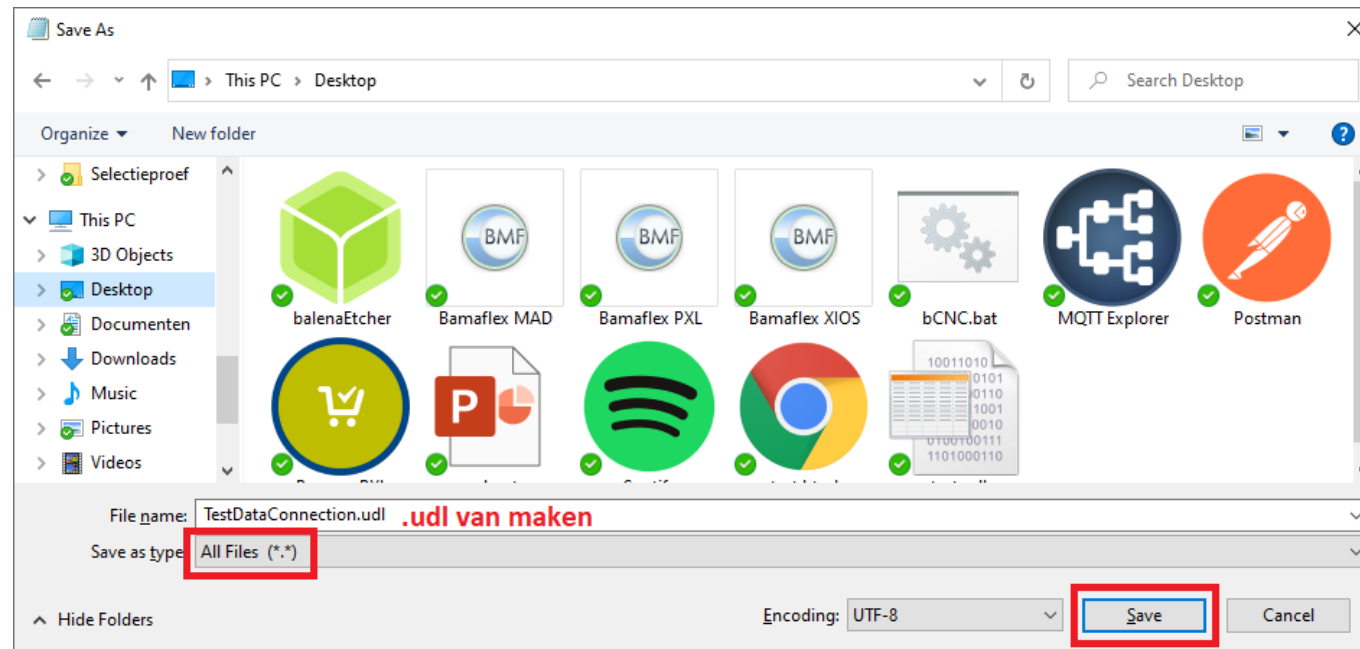
SqlConnection

- Voorbeeld van een SqlConnection string

```
SqlConnection connect = new SqlConnection(@"Integrated Security = SSPI;  
Persist Security Info = False; Initial Catalog = Medewerkers;  
Data Source = localhost\SQLEXPRESS");
```

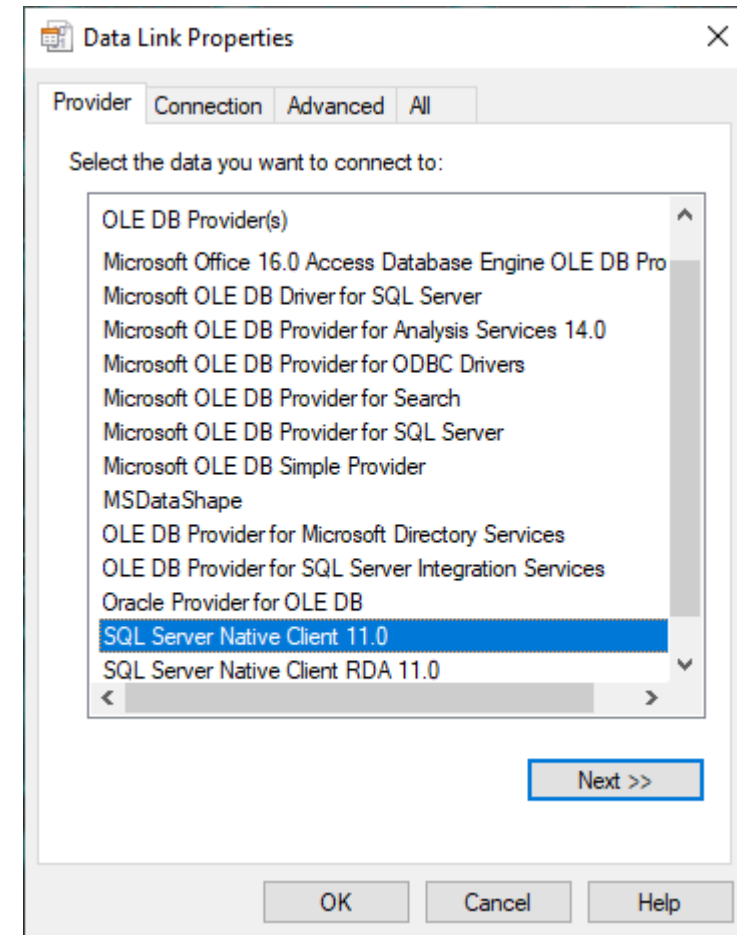
Connectie testen met de database

- Open Notepad
- Maak een nieuw tekstbestand aan en sla op als een *.UDL* file
Bijvoorbeeld **TestDataConnection.udl**



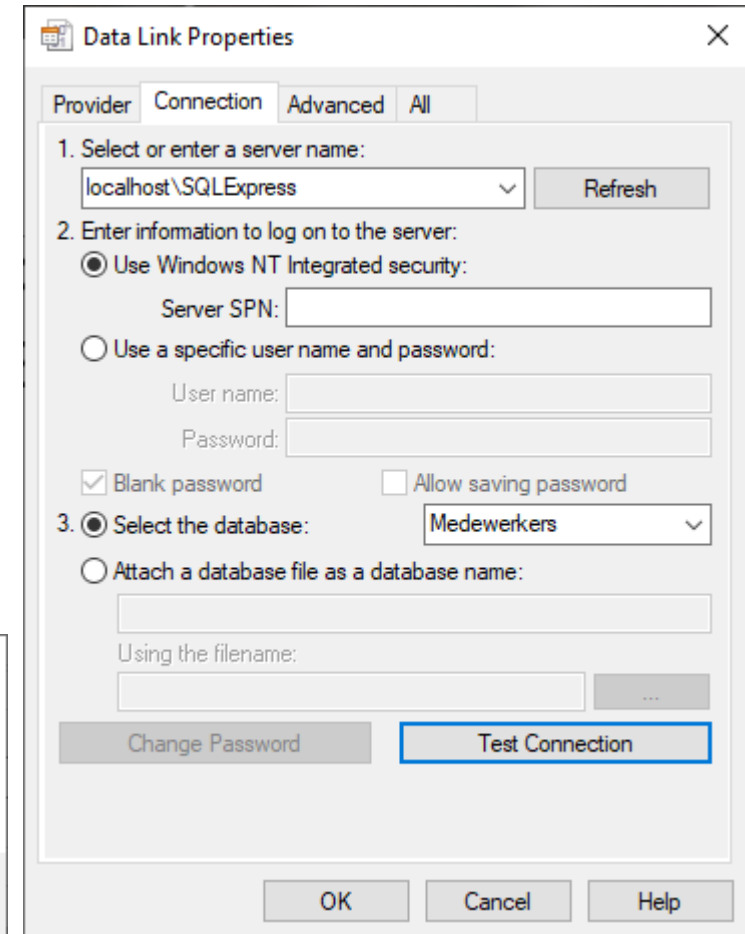
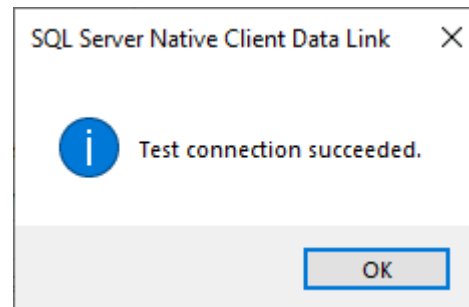
Connectie testen met de database

- Dubbelklik op TestDataConnection.udl
- Ga naar Providers (Voorziening) tabblad
- Kies voor SQL Server Native Client
 - Sneller dan Microsoft OLE DB



Connectie testen met de database

- Ga naar Connection (Verbinding) tabblad
- Server name: localhost\SQLEXPRESS
(Druk NIET op dropdown pijltje of Refresh)
- Vink aan
Windows NT Integrated Security
(indien anders: username + password opgeven)
- Selecteer de database op de server
Medewerkers
- Klik op Test Connection.
- Klik 2x op OK.



ConnectieString uit UDL file halen

- Open de TestDataConnection.udl file in Notepad (Kladblok).
 - Rechtsklik op udl-bestand.
 - Open with... (Openen met...)
 - Kies voor Notepad (Kladblok).
- Hierin staan gegevens voor in de ConnectionString

```
[oledb]  
; Everything after this line is an OLE DB initstring  
Provider=SQLNCLI11.1;Integrated Security=SSPI;Persist SecurityInfo=False;User ID="";Initial  
Catalog=Medewerkers;DataSource=localhost\SQLEXPRESS;Initial File Name="";Server SPN=""
```

Applicatie maken

- Maak een nieuw project aan met template **WPF App (.NET Framework)**
- Selecteer de laatste nieuwe versie van het Framework

Configure your new project

WPF App (.NET Framework) C# XAML Windows Desktop

Project name
VoorbeeldADO

Location
C:\OefeningenC#Advanced

Solution
Create new solution

Solution name ⓘ
VoorbeeldADO

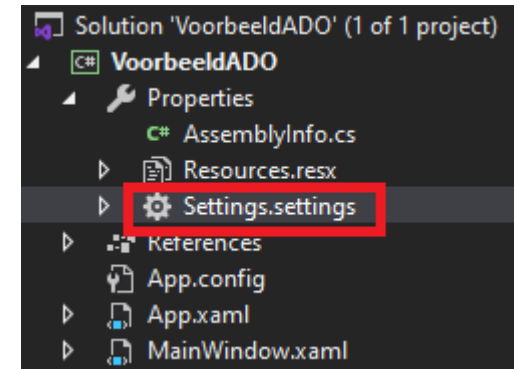
☐ Place solution and project in the same directory

Framework
.NET Framework 4.7.2

Back Create

ConnectionString toevoegen aan applicatie

- Indien je geen App.config zou hebben
 - Rechtsklik op Project. **Add > New Item > Application Configuration File**
 - Geef deze config file de naam: **App.config**
- Dubbelklik onder Properties op de file Settings.settings
- Maak een nieuwe lijn aan
 - Name: CNstr
 - Type: (Connection String)
 - Scope: Application
 - Value: Copy/Paste de geel gearceerde tekst van de UDL file
 - Sla op met CTRL+S!



	Name	Type	Scope	Value
	CNstr	(Connectio...	Application	Integrated Security=SSPI;Persist Security Info=False;User ID="";Initial Catalog=Medewerkers;Data Source=localhost\SQLEXPRESS;
*				

ConnectionString toevoegen aan applicatie

- Als je nu de App.config opent dat zie je dat deze veranderd is
 - De connectionString is hieraan toegevoegd

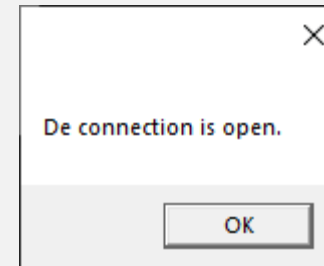
```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
  </configSections>
  <connectionStrings>
    <add name="VoorbeeldADO.Properties.Settings.CNstr"
connectionString="Integrated Security=SSPI;Persist Security Info=False;User
ID=&quot;&quot;;Initial Catalog=Medewerkers;Data Source=localhost\SQLExpress;"
/>
  </connectionStrings>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
  </startup>
</configuration>
```

ConnectionString uittesten

- Maak een nieuwe Button aan op je Window met de naam BtnConnectieTest
- Voeg de juiste namespace en click event toe

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

private void BtnConnectieTest_Click(object sender, RoutedEventArgs e)
{
    // Haal de ConnectionString uit de settings file
    string cn = Properties.Settings.Default.CNstr.ToString();
    SqlConnection sqlcn = new SqlConnection(cn);
    sqlcn.Open();
    if (sqlcn.State == System.Data.ConnectionState.Open)
    {
        MessageBox.Show("De connection is open.");
    }
    else
    {
        MessageBox.Show("De connection is NIET open.");
    }
}
```



SqlCommand

- Maak een nieuwe Button aan op je Window met de naam BtnSqlCommand
- Voeg de juiste namespace en click event toe

```
private void BtnSqlCommand_Click(object sender, RoutedEventArgs e)
{
    SqlConnection sqlcn = new SqlConnection(@" Integrated Security = SSPI;
        Persist Security Info = False; Initial Catalog = Medewerkers; Data Source
        = localhost\SQLEXPRESS");
    string query = "select * from medewerkers";
    SqlCommand cmd = new SqlCommand();
    cmd.Connection = sqlcn; // connection doorgeven aan commando
    cmd.CommandType = CommandType.Text; // we zeggen: commando staat in tekstformaat
    cmd.CommandText = query; // commando tekst is de query
    TxtResultaat.Text = cmd.CommandText; // Geeft: select * from medewerkers
}
```


SqlCommand

- Een kortere manier is
 - Query doorgeven aan constructor van SqlCommand

```
private void BtnSqlCommand_Click(object sender, RoutedEventArgs e)
{
    SqlConnection sqlcn = new SqlConnection(@" Integrated Security = SSPI;
        Persist Security Info = False; Initial Catalog = Medewerkers; Data Source
        = localhost\SQLEXPRESS");
    string query = "select * from medewerkers";
    SqlCommand cmd = new SqlCommand(query); // query doorgeven aan constructor van
    SqlCommand
    cmd.Connection = sqlcn; // connection doorgeven aan commando
    TxtResultaat.Text = cmd.CommandText; // Geeft: select * from medewerkers
}
```

SqlCommand

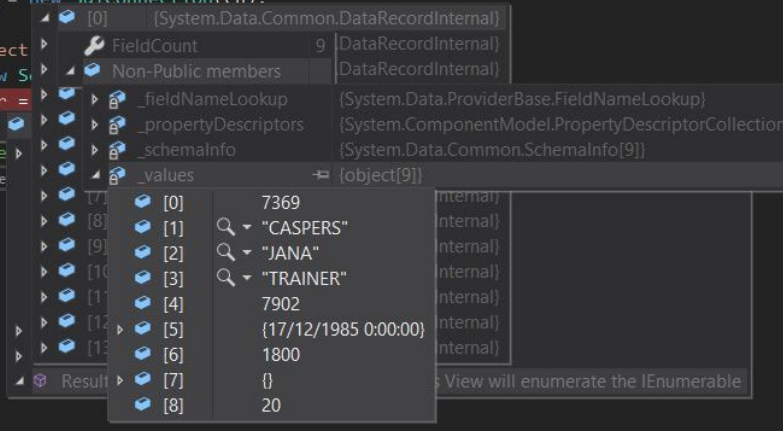
- De kortste manier is
 - Zowel query als SqlConnection meegeven aan constructor van SqlCommand

```
private void BtnSqlCommand_Click(object sender, RoutedEventArgs e)
{
    string cn = Properties.Settings.Default.CNstr.ToString();
    SqlConnection sqlcn = new SqlConnection(cn);
    string query = "select * from medewerkers";
    SqlCommand cmd = new SqlCommand(query, sqlcn); // query en sqlcn doorgeven aan
    constructor van SqlCommand
    TxtResultaat.Text = cmd.CommandText; // Geeft: select * from medewerkers
}
```

DataReader

- Command object heeft method `ExecuteReader()` en die retournt een `DataReader` object
 - ⇒ voor als je meerdere rijen retournt
- Read-only en forward-only snelle gegevenstoegang tot database (forward-only is enkel van voor naar achter lezen)
- Efficiënt: houdt enkel de huidige rij in het geheugen.

```
// === Data (meerdere rijen en/of kolommen) opvragen ===
string cn = Properties.Settings.Default.CNstr.ToString();
SqlConnection sqlcn = new SqlConnection(cn);
sqlcn.Open();
string query = "select";
SqlCommand cmd = new SqlCommand(query, sqlcn);
SqlDataReader reader = cmd.ExecuteReader();
// SqlDataReader object is nu beschikbaar
// Gebruik zie volgende code
sqlcn.Close();
```



Index	Value
[0]	7369
[1]	"CASPERS"
[2]	"JANA"
[3]	"TRAINER"
[4]	7902
[5]	{17/12/1985 0:00:00}
[6]	1800
[7]	{}
[8]	20

DataReader

Properties

FieldCount	Aantal kolommen van de query die je uitvoert
HasRows	true als de query minstens 1 rij teruggeeft, anders false
reader["kolomnaam"]	Returnt waarde van kolom in huidige gelezen rij
reader[kolomnummer]	Returnt waarde van kolom in huidige gelezen rij via het kolomnummer (vb/ Item(0) geeft de eerste kolomwaarde

DataReader

Methods

Read()	Lees rij uit database. Returnt true indien gelukt, anders false
GetString(kolomnr) GetInt32(kolomnr) GetDecimal(kolomnr) ...	Returnt waarde van kolom in huidige gelezen rij. Kolom wordt bepaald via kolomnummer. ZIJN PERFORMANTER DAN Item()!!!
GetOrdinal("kolomnaam")	Returnt kolomnummer voor gegeven kolomnaam. 0 voor de 1ste kolom van query 1 voor de 2de kolom van query,...
GetName(kolomnummer)	Returnt de kolomnaam voor kolom met dit kolomnummer
Close()	Sluit de DataReader. Altijd doen, want deze houdt de connectie voor zichzelf... Anders kan je geen andere queries meer doen. Close kan je vermijden ⇒ using (SqlDataReader reader = ...) De using zorgt voor automatisch sluiten van DataReader

ExecuteReader(): om alle rijen uit te lezen

```
string cn = Properties.Settings.Default.CNstr.ToString(); // ConnectionString uitlezen
// using zorgt voor automatische Close(), maar we moeten wel nog zelf Open() doen!!!
using (SqlConnection sqlcn = new SqlConnection(cn))
{
    sqlcn.Open();
    string queryString = "select * from medewerkers";
    SqlCommand cmd = new SqlCommand(queryString, sqlcn);
    // Dankzij using wordt de SqlDataReader automatisch gesloten!
    using (SqlDataReader reader = cmd.ExecuteReader())
    {
        // SqlDataReader gebruiken we om elke teruggegeven rij (record) te inspecteren
        while (reader.Read())
        {
            // Snelst, let op dat je de juiste types neemt die overeenkomen met SQL server:
            TxtResultaat.AppendText($"{reader.GetInt16(0)} {reader.GetString(1)}\r\n");
            // Of trager:
            TxtResultaat.AppendText($"{reader[0]} {reader[1]}\r\n");
        }
    }
}
```

ExecuteScalar(): om 1 waarde op te vragen

```
string cn = Properties.Settings.Default.CNstr.ToString();
using (SqlConnection sqlcn = new SqlConnection(cn))
{
    sqlcn.Open();
    string query = "select max(mnr) from medewerkers";
    SqlCommand cmd = new SqlCommand(query, sqlcn);
    // short van C# komt overeen met smallint in SQL Server
    short maxMed = (short)cmd.ExecuteScalar();
    TxtResultaat.Text = $"{maxMed}"; // Geeft grootste medewerkersnummer
}
```

ExecuteNonQuery(): INSERT command

```
string cn = Properties.Settings.Default.CNstr.ToString();
using (SqlConnection sqlcn = new SqlConnection(cn))
{
    sqlcn.Open();
    string insertString = @"insert into medewerkers(mnr, naam, voorn, gbdatum,
maandsal, functie)
values (7001, 'QUAREME', 'TOM', '1992-01-12', 2500, 'TRAINER')";
    SqlCommand cmd = new SqlCommand(insertString, sqlcn);
    cmd.ExecuteNonQuery(); // geen query maar commando
}
```


ExecuteNonQuery(): UPDATE command

```
string cn = Properties.Settings.Default.CNstr.ToString();
using (SqlConnection sqlcn = new SqlConnection(cn))
{
    sqlcn.Open();
    string updateString = @"update medewerkers
        set naam = 'TESLA', voorn = 'NIKOLA'
        where mnr = 7001";
    SqlCommand cmd = new SqlCommand(updateString, sqlcn);
    cmd.ExecuteNonQuery(); // geen query maar commando
}
```

ExecuteNonQuery(): DELETE command

```
string cn = Properties.Settings.Default.CNstr.ToString();
using (SqlConnection sqlcn = new SqlConnection(cn))
{
    sqlcn.Open();
    string deleteString = @"delete from medewerkers where mnr = 7001";
    SqlCommand cmd = new SqlCommand(deleteString, sqlcn);
    cmd.ExecuteNonQuery(); // geen query maar commando
}
```

SqlParameter

1ste manier om door te geven

- C#-variabelen of waarden veilig doorgeven aan SqlCommand
- Voorbeeld: selecteer alle medewerkers met maandsalaris ≥ 3500

```
string cn = Properties.Settings.Default.CNstr.ToString();
using (SqlConnection sqlcn = new SqlConnection(cn))
{
    sqlcn.Open();
    string query = @"select voorn, naam, maandsal from medewerkers where maandsal >= @salaris";
    // Parameter instellen.
    SqlParameter param = new SqlParameter();
    param.ParameterName = "@salaris";
    param.Value = 3500.0f;
    SqlCommand cmd = new SqlCommand(query, sqlcn);
    cmd.Parameters.Add(param);
    using (SqlDataReader reader = cmd.ExecuteReader())
    {
        while (reader.Read())
        {
            // We gebruiken GetDouble() omdat GetFloat() een InvalidCastException heeft...
            // float in SQL server is dus double in C# !!!
            TxtResultaat.AppendText($"{reader.GetString(0)} {reader.GetString(1)} {reader.GetDouble(2)}\r\n");
        }
    }
}
```

SqlParameter

2de manier om door te geven

- C#-variabelen of waardes veilig doorgeven aan SqlCommand
- Voorbeeld: selecteer alle cursussen van het type DSG

```
string cn = Properties.Settings.Default.CNstr.ToString();
using (SqlConnection sqlcn = new SqlConnection(cn))
{
    sqlcn.Open();
    string query = @"select code, omschrijving, type from cursussen where type = @type";
    SqlCommand cmd = new SqlCommand(query, sqlcn);
    cmd.Parameters.AddWithValue("@type", "DSG"); // meteen parameter doorgeven: naam samen met waarde
    using (SqlDataReader reader = cmd.ExecuteReader())
    {
        // SqlDataReader gebruiken we om elke teruggegeven rij (record) te inspecteren
        while (reader.Read())
        {
            // Let op dat je de juiste types neemt die overeenkomen met SQL server:
            TxtResultaat.AppendText($"{reader.GetString(0)} {reader.GetString(1)} {reader.GetString(2)}\r\n");
        }
    }
}
```

Datatypes: SQL Server vs C#

SQL Server	C#
bit	bool \Rightarrow GetBoolean(kolomnr)
bigint	long \Rightarrow GetInt64(kolomnr)
int	int \Rightarrow GetInt32(kolomnr)
smallint	short \Rightarrow GetInt16(kolomnr)
tinyint	byte \Rightarrow GetByte(kolomnr)
float(n) n is standaard 53 bits	float (als $n \leq 32$) \Rightarrow GetFloat(kolomnr) double (als $n > 32$ en $n \leq 64$) \Rightarrow GetDouble(kolomnr)
varchar(n) of char(n)	string \Rightarrow GetString(kolomnr)
date	DateTime \Rightarrow GetDateTime(kolomnr)

SqlDataAdapter

- Gebruik Fill() method van SqlDataAdapter om DataTable te vullen met alle data van database na uitvoeren van commando

```
string cn = @" Integrated Security = SSPI; Persist Security Info = False;  
    Initial Catalog = Medewerkers; Data Source = localhost\SQLEXPRESS";  
DataSet ds = new DataSet();  
DataTable dt = new DataTable();  
dt.TableName = "Medewerkers";  
ds.Tables.Add(dt); // voeg DataTable dt toe aan DataSet ds (indien nog niet gedaan)  
using (SqlConnection sqlcn = new SqlConnection(cn))  
{  
    sqlcn.Open();  
    SqlCommand cmd = new SqlCommand();  
    cmd.Connection = sqlcn;  
    cmd.CommandText = "select * from medewerkers";  
    SqlDataAdapter da = new SqlDataAdapter(cmd);  
    da.Fill(dt); // vul DataTable dt met de resultaten in de SqlDataAdapter da  
}
```