ADO.NET

Veerle Ongenae

ADO.Net

- Opvolger ADO (Activex Data Objects)
- Toegang vanuit een .NET-programma tot een gegevensbron
  - Ophalen gegevens
  - Manipuleren gegevens
  - Aanpassen gegevens
- Databronnen
  - Gegevensbanken
  - XML-documenten, XML-streams

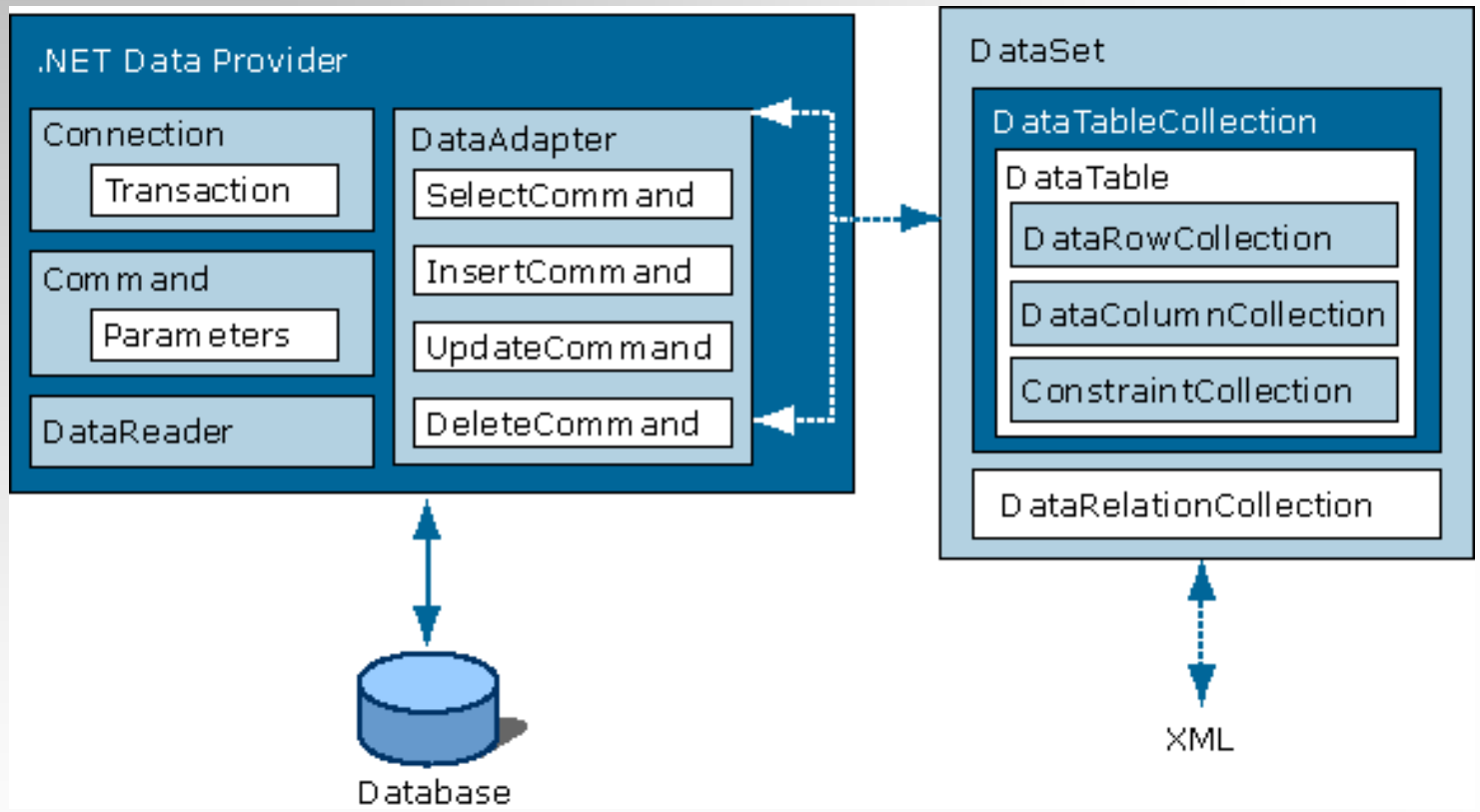
ADO.Net

- Principes
  - Losse koppeling tussen applicatie en databron is mogelijk
    - Stuk van de gegevens in de applicatie bijhouden en manipuleren zonder telkens terug te koppelen naar de gegevensbron
  - Vlotte overgang naar XML

ADO.Net Architectuur

- DataSet
  - Een deel van de gegevens bewaren en manipuleren in applicatie
- .Net Data Provider
  - Brug tussen applicatie en gegevensbank
- Namespace
  - System.Data
  - onderliggende namespaces van System.Data

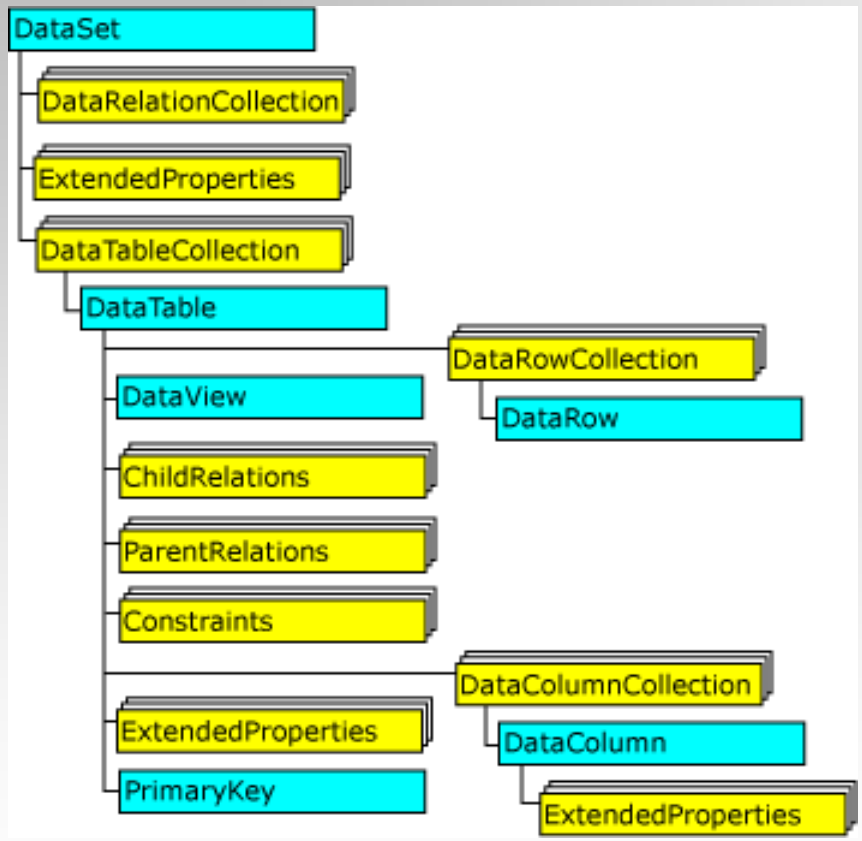
ADO.Net Architectuur



DataSet

- Lokaal bewaren en manipuleren gegevens in de applicatie
- Doorgeven gegevens aan andere modules/delen van de applicatie
- Mini databank in het geheugen

DataSet



Geel

- verzamelingen
- Collections

Tekst

- Klassenaam

DataSet

- Hoe opgebouwd?
  - Verzameling DataTable-objecten
  - Relaties en beperkingen tussen de DataTable-objecten (verwijssleutels, …)
- DataTable
  - Beschrijving tabel
    - Kolommen
    - Beperkingen: sleutels, …
  - Verzameling rijen van één tabel

DataRow

- DataRow
  - Bewaart twee versies van zijn gegevens
    - Huidige (current)
    - Oorspronkelijke (original)
  - Aanpassingen data kunnen geïdentificeerd worden

DataSet – Aanmaken en uitvoer

- Aanmaken en Opvullen
  - “manueel” (in de applicatie)
  - Vanuit een gegevensbank
  - Vanuit een XML-stroom of -bestand
- Uitvoer
  - XML
  - Naar onderliggende gegevensbank

Voorbeeld

- Aanmaak DataSet
- Opvullen DataSet

Voorbeeld DataSet

KlantDS

- Aanmaak
  - Via Visual Studio
    - Genereert code
  - In code

Aanmaak DataSet en DataTable

- Aanmaak DataSet

```
DataSet Gegevens = new DataSet();
DataSet KlantDS = new DataSet("KlantOrders");
```

- Aanmaak Tabel

```
DataTable Bestellingen =
    klantDS.Tables.Add("Bestellingen");
```

Aanmaak Kolommen

- Aanmaak Kolommen

```
DataColumn BestelId =
    Bestellingen.Columns.Add("BestelID",typeof(Int32))
    ;
Bestellingen.Columns.Add("Hoeveelheid",
    typeof(Int32));
Bestellingen.Columns.Add("Firma", typeof(string));
```

- Beperkingen: sleutels

```
Bestellingen.PrimaryKey = new DataColumn[]
    {BestelId};
```

API DataSet en DataTableCollection

- DataSet
  - Eigenschap Tables: type DataTableCollection
- DataTableCollection
  - Methode Add
    - Argument: naam tabel
    - Teruggeefwaarde: DataTable-object
  - Indexer
    - Argument: naam tabel of volgnummer
    - Teruggeefwaarde: DataTable-object

DataTable

- DataTable
  - Eigenschap Columns: type DataColumnCollection
  - Eigenschap PrimaryKey: type DataColumn[]
- DataColumnCollection
  - Methode Add
    - Mogelijke argumenten: naam kolom, type kolom
    - Teruggeefwaarde: DataColumn-object
- typeof
  - Sleutelwoord
  - Bepaalt type van een klasse

Voorbeeld

- Aanmaak DataSet
- Opvullen DataSet

Tabel opvullen

- Tabellen opvullen
  - Rij aanmaken

```
DataRow rijBestelling = Bestellingen.NewRow();
```

  - Rij opvullen

```
rijBestelling[BestelID] = 3145;
rijBestelling["Hoeveelheid"] = 3;
rijBestelling[2] = "Mijn Bedrijf";
```

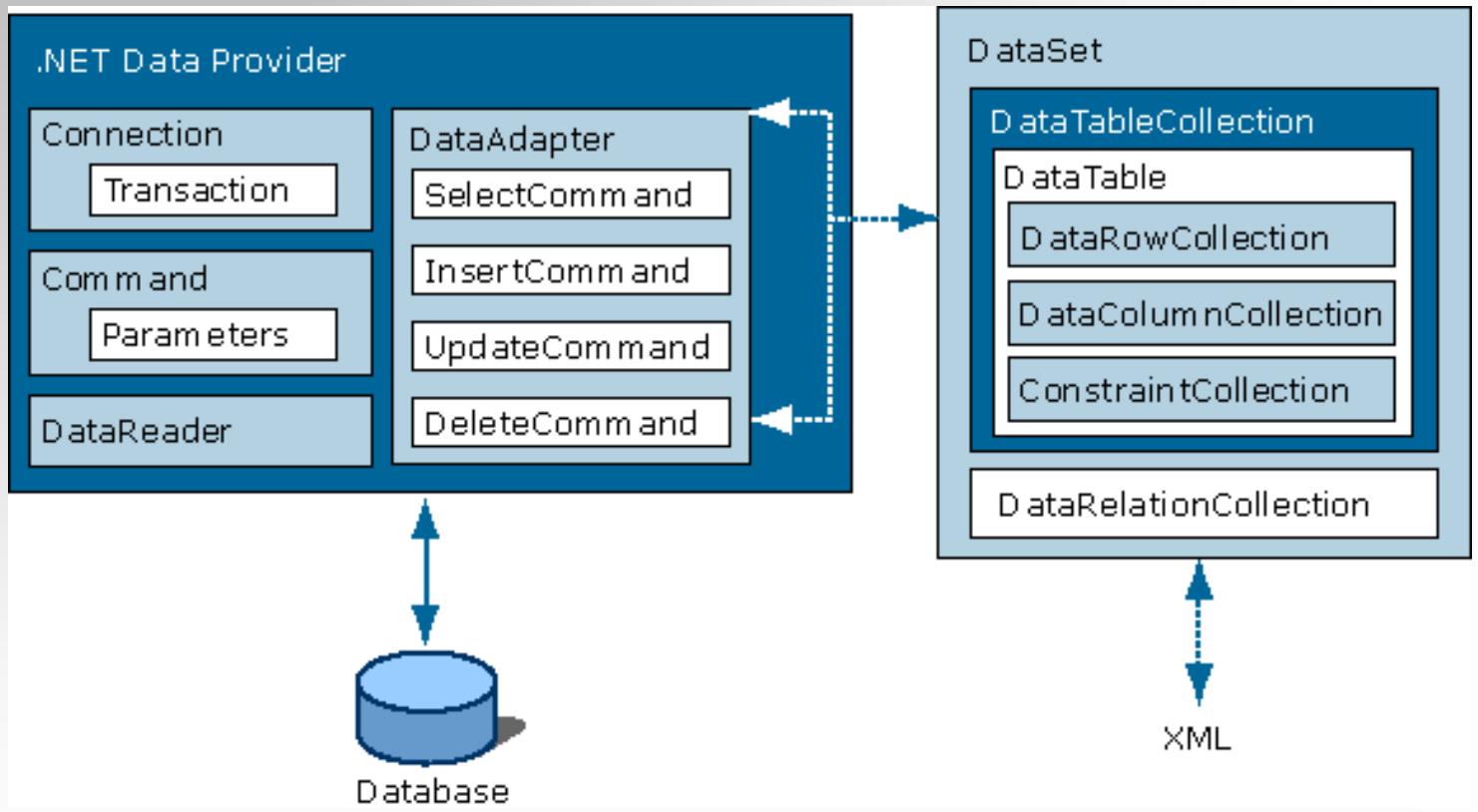
  - Rij toevoegen

```
Bestellingen.Rows.Add(rijBestelling);
```

DataTable

- DataTable
  - Methode NewRow
    - Teruggeefwaarde: DataRow-object met de structuur van de tabel
  - Eigenschap Rows: type DataRowCollection
- DataRowCollection
  - Methode Add
    - argument: DataRow-object
- DataRow
  - Indexer: string, int, DataColumn

ADO.Net Architectuur



.Net Data Providers

- Verschaffen toegang tot gegevensbanken vanuit een .NET-applicatie
- Standaardacties
  - Verbinding maken met een gegevensbank
  - Opdrachten uitvoeren
  - Gegevens ophalen
- Koppeling maken tussen gegevensbanken en DataSet
  - DataSet opvullen
  - Veranderingen aan DataSet doorvoeren in gegevensbank

Sleutelobjecten

- Mogelijke acties → sleutelobjecten
  - Connection (verbinding maken)
  - Command (opdrachten uitvoeren)
  - DataReader (gegevens ophalen)
  - DataAdapter (koppeling DataSet en gegevensbank)

Interfaces sleutelobjecten

- Elke Provider implementeert bijhorende interfaces
  - System.Data.IDbConnection
  - System.Data.IDbCommand
  - System.Data.IDataReader
  - System.Data.IDbDataAdapter
- Klassen behoren tot een subnamespace van System.Data

Abstracte implementatie sleutelobjecten

- Abstracte klassen
  - System.Data.Common.DbConnection (sinds 2.0!)
  - System.Data.Common.DbCommand (sinds 2.0!)
  - System.Data.Common.DataReader (sinds 2.0!)
  - System.Data.Common.DbDataAdapter
- Implementeren corresponderende interfaces
- Klassen typisch voor een .Net Provider zijn hiervan afgeleid
- Kunnen gebruikt worden om Provider onafhankelijk te werken

.Net Data Providers

- Standaard beschikbaar
  - .NET Framework Data Provider for SQL Server
  - .NET Framework Data Provider for OLE DB
  - .NET Framework Data Provider for ODBC
  - .NET Framework Data Provider for Oracle

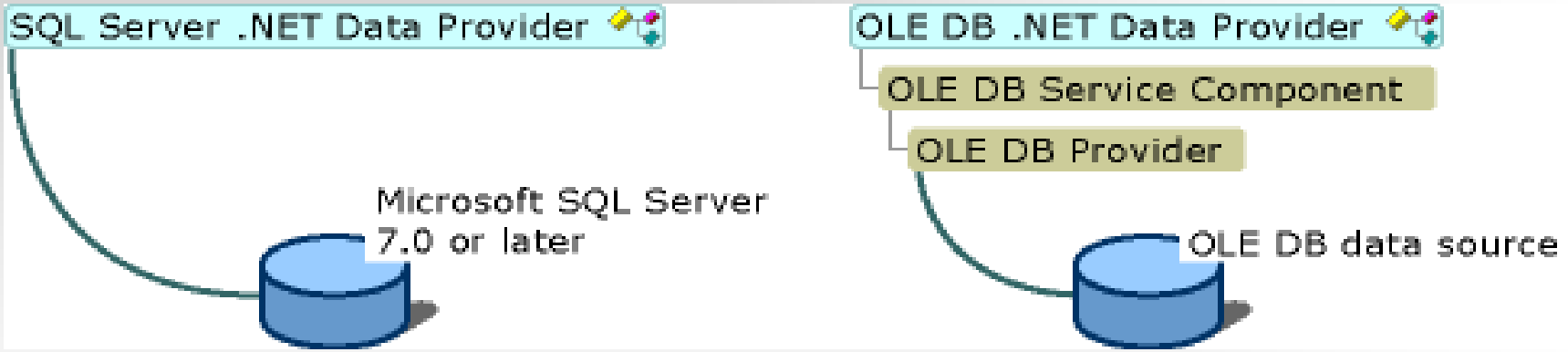
SQL Server

- .NET Framework Data Provider for SQL Server
  - SQL Server 7.0 of later
  - Communiceert direct met data-transfer-protocol van SQL Server
  - System.Data.SqlClient

OleDB

- .NET Framework Data Provider for OLE DB
  - Object Linking and Embedding for Databases
  - OLE DB gegevensbanken
    - SQL Server
    - Oracle
    - Microsoft Jet
  - lets minder efficiënt
  - Uniforme toegang tot verschillende databronnen
  - Gebruikt OLE DB-laag om te communiceren
  - System.Data.OleDb

.Net Data Providers



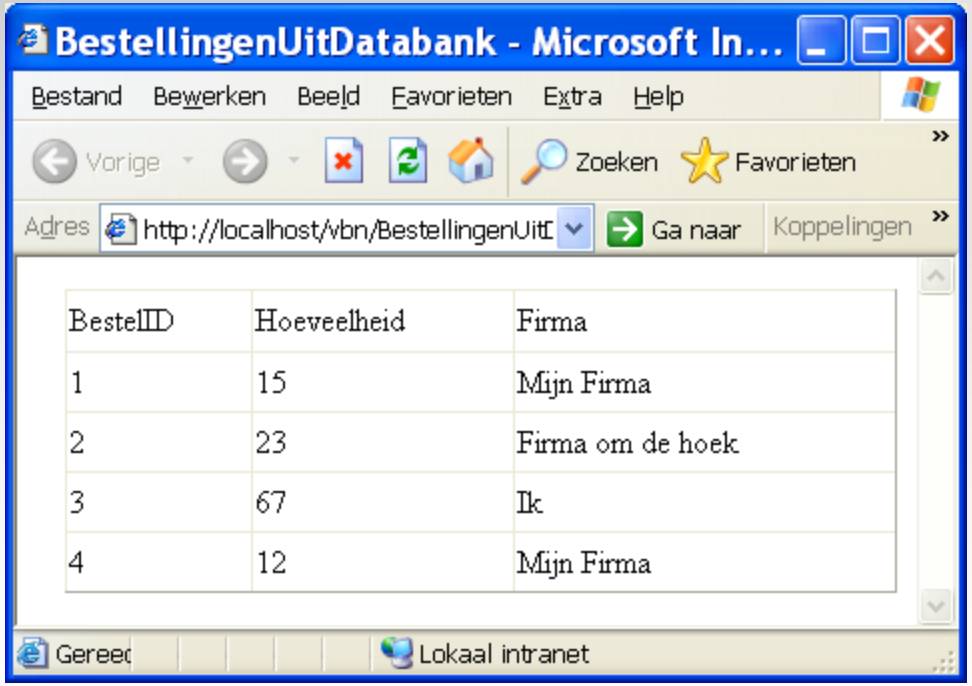
ODBC

- .NET Framework Data Provider for ODBC
  - Open Database Connectivity
  - ODBC-drivers
    - SQL Server
    - Microsoft ODBC for Oracle
    - Microsoft Access Driver (*.mdb)
  - lets minder efficiënt
  - Uniforme toegang tot verschillende databronnen
  - Gebruikt ODBC-laag om te communiceren
  - System.Data.Odbc

Oracle

- .NET Framework Data Provider for Oracle
  - Oracle Client Software 8.1.7 of hoger
  - Performanter
  - System.Data.OracleClient

Voorbeeld gegevensbank



- Gegevens uit de gevensbank inlezen in DataSet

DataSet vullen met gegevens uit databank

- BestellingBeheer.cs
  - Constructor
    - Configuratie
    - Factory
      - Aanmaken verbinding-objecten , commando-objecten, …
  - Methode GetConnection
  - Methode GetBestellingenMetDataReader
    - DataSet opvullen
- App.config
  - Connectiestring
  - SQL-opdrachten
- Program.cs
  - Uittesten

Verschillende stappen

- Configuratiegegevens inlezen
- Factory voor de provider aanmaken
- Connectie-object aanmaken
- Commando-object aanmaken
- Commando-object uitvoeren → eventueel DataReader overlopen

Configuratiegegevens inlezen

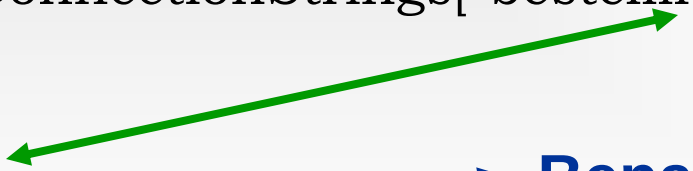
- Configuratiebestand
  - Applicaties
    - App.config
  - Webapplicaties
    - Web.config
- Klasse ConfiguratieManager
  - Ophalen configuratie

Klasse ConfiguratieManager

- Eigenschap ConnectionStrings
  - Indexer
    - Argument: waarde name-attribuut
    - Resultaat: ConnectionStringSettings-object
      - Provider-naam
      - Connectiestring

```
ConnectionStringSettings connStringSet =
    ConfigurationManager.ConnectionStrings["bestellingDB"];


<connectionStrings>
    <add name="bestellingDB"
    providerName="System.Data.OleDb" connectionString="…"/>
  </connectionStrings>
```



Bepaalt Provider

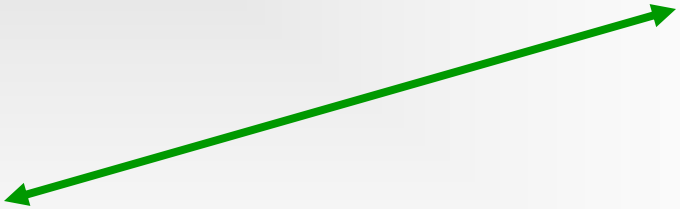
Klasse ConnectionStringSettings

- Eigenschappen van ConnectionStringSettings
  - ProviderName
    - Identificeert de Provider
  - ConnectionString
    - connectiestring

Klasse ConfiguratieManager

- Eigenschap AppSettings
  - Indexer
    - Argument: waarde key-attribuut
    - Resultaat: string, waarde value-attribuut

```
String opdracht =
    ConfigurationManager.AppSettings["SELECT_BESTELLINGEN"];
```



```
<appSettings>
    <add key="SELECT_BESTELLINGEN" value="select * from
    Bestellingen"/>
</appSettings>
```

Verschillende stappen

- Configuratiegegevens inlezen
- Factory voor de provider aanmaken
- Connectie-object aanmaken
- Commando-object aanmaken
- Commando-object uitvoeren → eventueel DataReader overlopen

Factory voor de provider aanmaken

- Maakt ADO.NET-objecten voor een bepaalde provider
  - Connecties, commando's, parameters, data-adapters, …
- Statische methode GetFactory van DbProviderFactories
  - aanmaak DbProviderFactory-object op basis van naam provider (kan namespace zijn, …)

```
DbProviderFactory factory  =
    DbProviderFactories.GetFactory(connStringSet.ProviderName)
    ;
```

Verschillende stappen

- Configuratiegegevens inlezen
- Factory voor de provider aanmaken
- Connectie-object aanmaken
- Commando-object aanmaken
- Commando-object uitvoeren → eventueel DataReader overlopen

Connectie-object aanmaken

- Connectie-objecten aanmaken met DbProviderFactory-object

```
DbConnection connection = factory.CreateConnection();
```

- Connectiestring instellen

```
connection.ConnectionString =
   connStringSet.ConnectionString;
```

ConnectionStringSettings-object

Connectiestring

- Naam/waarde-paren gescheiden door ‘;’
- Namen:
  - Server of Data Source
  - Database
  - Password of Pwd
  - User ID
  - Trusted_Connection (false → User ID & Password)
  - Provider (verplicht bij OleDbConnection)

Alternatief aanmaken connectie-object

- Expliciet provider-klassen gebruiken
  - SqlConnection, OleDbConnection, …
  - Connectiestring
    - Meegeven met constructor
    - Toekennen via eigenschap ConnectionString

```
string connString =
    "Provider=Microsoft.Jet.OLEDB.4.0;User ID=Admin;Data
    Source=C:\\Inetpub\\wwwroot\\vbn\\klantorders.mdb";
 OleDbConnection conn =
    new OleDbConnection(connString);
```

Verschillende stappen

- Configuratiegegevens inlezen
- Factory voor de provider aanmaken
- Connectie-object aanmaken
- Commando-object aanmaken
- Commando-object uitvoeren → eventueel DataReader overlopen

Commando-object aanmaken

- Commando-objecten aanmaken met DbConnection-object

```
DbConnection conn = …
DbCommand command = conn.CreateCommand();
```

- SQL-opdracht instellen

```
command.CommandText =
    ConfigurationManager.AppSettings["SELECT_BESTELLINGEN"];
```

Alternatief aanmaken commando-object

- Expliciet provider-klassen gebruiken
  - SqlCommand
  - OleDbCommand
  - …

```
OleDbConnection conn = …;
String opdracht = …;
OleDbCommand opdracht = new
    OleDbCommand(opdracht,conn);
```

Command

- Nodig
  - Verbinding: object of string
  - Opdracht
  - Eventueel transactie
- Via constructor
- Of via de eigenschappen
  - CommandText
  - Connection
  - Transaction

Verschillende stappen

- Configuratiegegevens inlezen
- Factory voor de provider aanmaken
- Connectie-object aanmaken
- Commando-object aanmaken
- Commando-object uitvoeren → eventueel DataReader overlopen

Commando-object uitvoeren

- Connectie openen
- Opdracht uitvoeren
- Connectie sluiten

```
DbConnection conn = …
conn.Open();
try {
      … // commando-object uitvoeren
} finally {
    conn.Close();
}
```

Commando-object uitvoeren

- insert, update, delete SQL-opdrachten en DDL-opdrachten
  - Methode ExecuteNonQuery
    - Resultaat: aantal aangepaste rijen
- Zoekopdrachten
  - Methode ExecuteReader
    - Resultaat: DataReader

Zoekopdrachten: DataReader

- Leest gegevens van een databron
- Eén keer lezen van begin tot einde
- Interface IDataReader, abstracte klasse DbDataReader
- Afhankelijk van .Net Data Provider (implementeren IDataReader)
  - SqlDataReader, OleDbDataReader, …
- Teruggeefwaarde van de methode ExecuteReader van het Command-object

DbDataReader: voorbeeld

- Voorbeeld

```
conn.Open();
try {
      DbDataReader reader = command.ExecuteReader();
      while (reader.Read()) {
            DataRow rij = tabelBestel.NewRow();
            rij[0] = reader["BestelID"];
            rij[1] = reader[1];
            rij[2] = reader.GetString(2);
            tabelBestel.Rows.Add(rij);
      }
} finally {
      conn.Close();
}
```

DataReader

- Methodes
  - Read(): verplaatst de cursor één rij naar beneden
  - GetXxx(int volgnummer)
    - Xxx is een type vb. string, …
    - Haalt de gegevens van de kolom bepaald door volgnummer
  - Close(): sluit reader
- Indexer
  - Argument: naam of volgnummer kolom

```
Console.WriteLine(reader[0] + ", " +
    reader["Hoeveelheid"]);
```

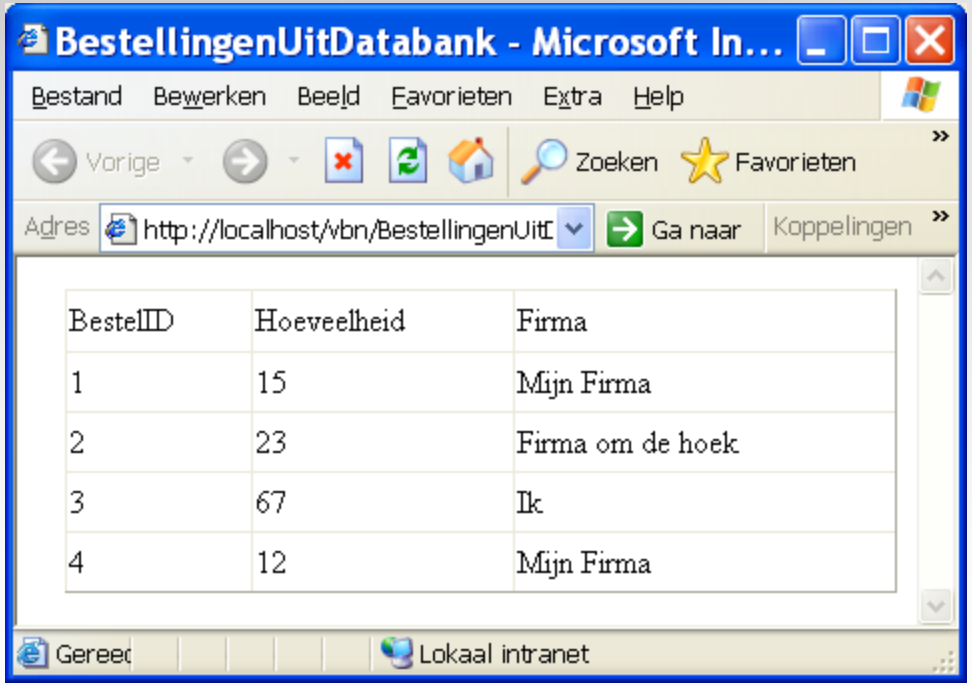
Verschillende stappen

- Configuratiegegevens inlezen
- Factory voor de provider aanmaken
- Connectie-object aanmaken
- Commando-object aanmaken
- Commando-object uitvoeren → eventueel DataReader overlopen

Sleutelobjecten

- Mogelijke acties → sleutelobjecten
  - Connection (verbinding maken)
  - Command (opdrachten uitvoeren)
  - DataReader (gegevens ophalen)
  - DataAdapter (koppeling DataSet en gegevensbank)

Voorbeeld gegevensbank



Gegevens uit de gevensbank inlezen in DataSet

DataSet vullen met gegevens uit databank herbekijken

- BestellingBeheer.cs
  - Methode GetBestellingenMetDataAdapter
- App.config
  - Connectiestring
  - SQL-opdrachten

DataAdapter

- Alternatieve wijze om om te gaan met gegevensbanken in een applicatie → lossere koppeling
  - Ophalen nodige gegevens en bewaren in DataSet
  - Manipuleren DataSet
  - Wijzigingen doorvoeren op gegevensbank
- DataAdapter: brug tussen DataSet en gegevensbank
  - Opvullen DataSet
  - Wijzigen onderliggende databron

DataAdapter-object aanmaken

- DataAdapter-objecten aanmaken met DbProviderFactory-object

```
DbDataAdapter adapter = factory.CreateDataAdapter();
```

- Zoekopdracht die de adapter gebruikt om gegevens op te halen instellen

```
DbCommand command = …
adapter.SelectCommand = command;
```

Klasse DataAdapter

- Methode Fill
  - Argumenten: DataSet en naam van de tabel in de DataSet
  - Functionaliteit
    - Openen verbinding gegevensbank
    - Query uitvoeren en resultaat bewaren als een tabel van de DataSet (met opgegeven naam)
    - Sluiten verbinding gegevensbank

```
DbDataAdapter adapter = …
DataSet klantDS = new DataSet(DS_ORDERS);
adapter.Fill(klantDS,TABEL_BESTELLING);
```

Alternatief aanmaken DataAdapter-object

- Gebruik maken van de provider specifieke klassen
  - SqlDataAdapter
  - OleDbDataAdapter
  - …
- Argumenten constructor
  - Query + connectie-object

Voorbeeld alternatief

```
OleDbConnection conn = …
string opdracht = ….
OleDbDataAdapter adapter =
    new OleDbDataAdapter(opdracht,conn);
```

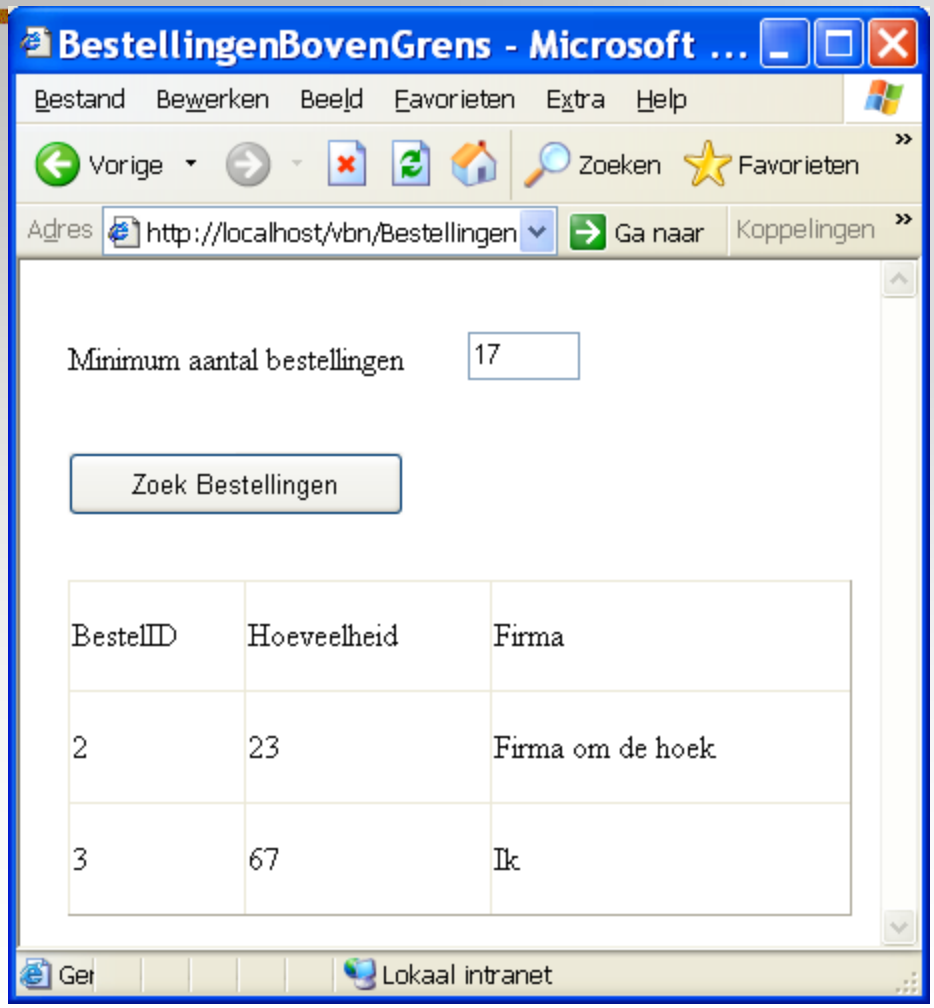
Opmerking

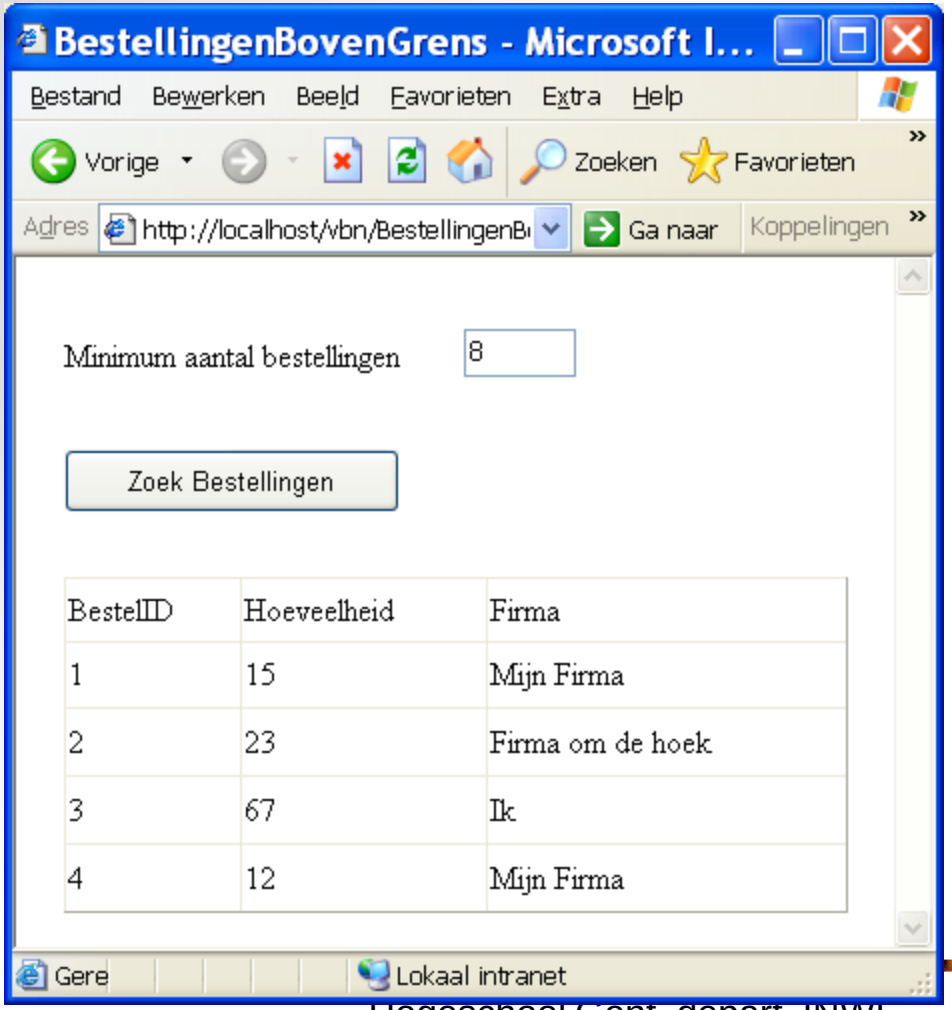
- De volgende objecten kunnen ook visueel aangemaakt worden
  - OleDbConnection
  - OleDbDataAdapter
  - DataSet
- Toolbox → Data

DataReader ↔ DataSet en DataAdapter

- DataReader
  - Enkel resultaten tonen
  - Performanter
  - Wijzigingen worden onmiddellijk doorgevoerd in de gegevensbank
- DataSet
  - Lokaal data bijhouden en bewerken
  - Handig in gebruik (bv. ASP.NET-pagina's)
  - Wijzigingen worden pas later doorgevoerd in de gegevensbank

Voorbeeld opdracht met parameter





Voorbeeld opdracht met parameter

- BestellingBeheer.cs
  - Methode GetBestellingen(int minAantal)
- App.config
  - Connectiestring
  - SQL-opdrachten

Zoekopdracht met parameter

- Werkwijze
  - SQL-opdracht met parameter
  - Command-object aanmaken
  - Parameter aanmaken
  - Parameter toevoegen aan Command-object
  - Parameter een waarde geven

SQL-opdracht met parameter

- Plaats parameter in SQL-opdracht aangeven.
  - Met *@naamParameter*
    - !afhankelijk van provider! (soms : …)
  - ? kan soms ook

```
string query = "select * from Bestellingen where
    Hoeveelheid > @min";
```

Parameter aanmaken en toevoegen aan command

- Parameter-object aanmaken met DbProviderFactory
- Naam en type instellen
- Toevoegen aan verzameling parameters van het commando-object

```
DbParameter parameter = factory.CreateParameter();
parameter.ParameterName = MIN;
parameter.DbType = DbType.Int32;
opdracht.Parameters.Add(parameter);
```

Parameter een waarde geven

- De eigenschap Parameters (type IDataParameterCollection) van IDbCommand
  - Houdt parameters bij
- Selecteer parameter op naam via indexer
- Geef de eigenschap Value van DbParameter een waarde

Geheel getal

```
opdracht.Parameters[MIN].Value = minimumAantal;
```

Parameter

- SQL-opdracht met parameter uitvoeren
- Herhaaldelijk uitvoer van dezelfde opdracht met andere waarden
- Gegevensconversi
- Voorkomen SQL-injectie
- Alternatief
  - Provider-klassen gebruiken: SqlParameter, OleDbParameter, …

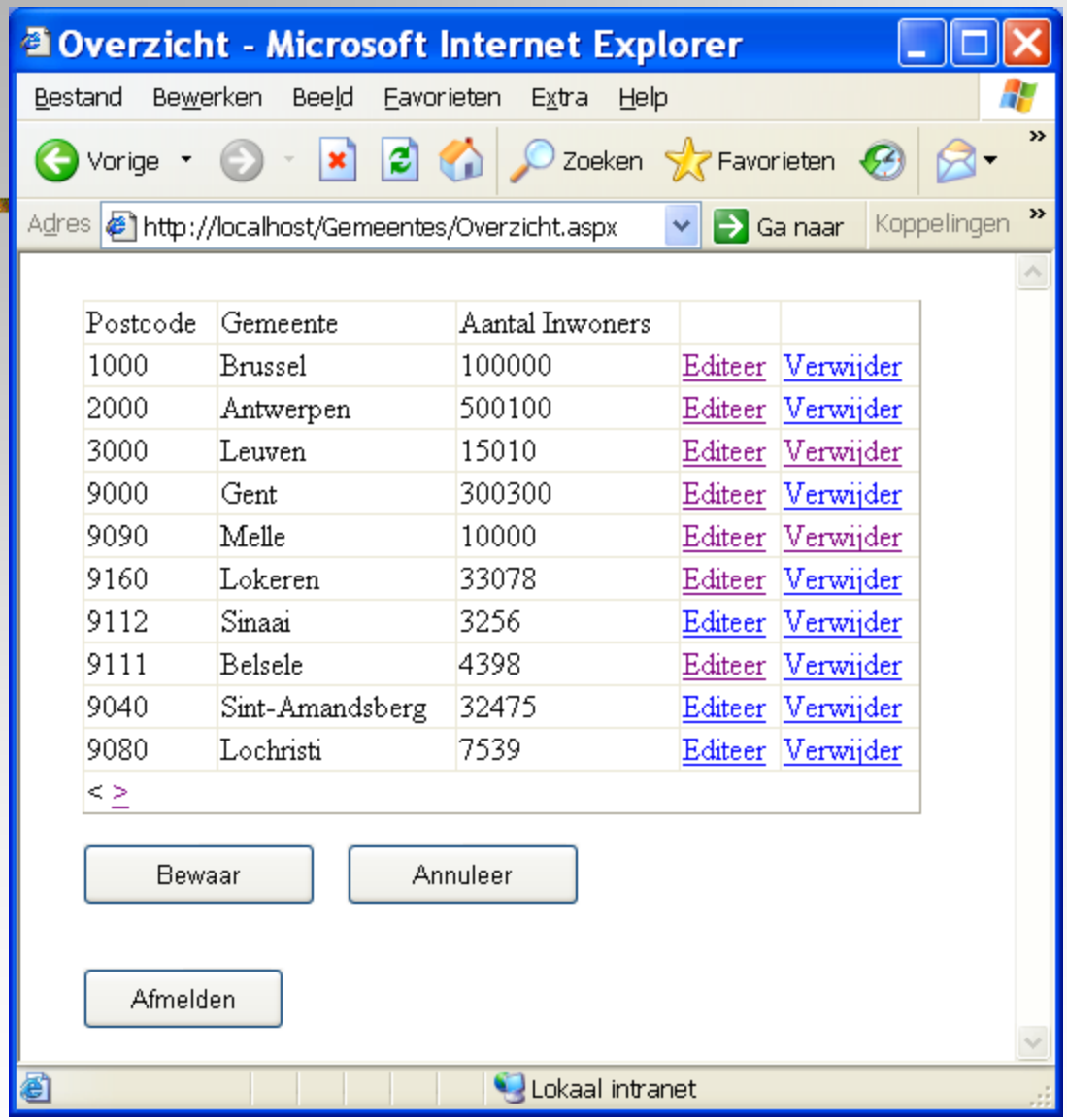
```
OleDbCommand opdracht = …
OleDbParameter param =
    opdracht.Parameters.Add(“@min”,DbType.Int32);
SqlParameter paramNaam = new
    SqlParameter(“@naam”,DbType.String);
```

Parameter

- Constructie
  - Geen argumenten
  - Naam en type
  - Naam en waarde
- Eigenschappen
  - ParameterName
  - Value
  - DbType
    - Opsommingen: DbType, SqlDbType, OleDbType,…

IDataParameterCollection

- Methode Add
- Indexer
  - Argument: string
  - Teruggeefwaarde afhankelijk van het type: object, SqlParameter, OleDbParameter



- Editeerbare tabel
- Achterliggende DataSet wordt veranderd
- Indien veranderingen OK → doorsturen naar gegevensbank

Principe

- Tabel toont gegevens van een DataSet
- DataSet wordt veranderd in de loop van het programma
  - Deze veranderingen zijn niet definitief
    - Methode RejectChanges van DataRow
    - Oorspronkelijke rij wordt bijgehouden
  - De rij wordt gemarkeerd
    - Verwijderd
    - Aangepast
    - …

    → Opsomming verschillende statussen: DataRowState

Principe

- DataAdapter
  - Opvullen DataSet
  - Aanpassingen DataSet doorvoeren in gegevensbank
    - Voor elke gemarkeerde rij wordt een opdracht uitgevoerd
  - Heeft 4 Command-objecten
    - SelectCommand, InsertCommand, UpdateCommand en DeleteCommand (Eigenschappen!!)
    - Bepalen hoe wijzigingen aan de DataSet vertaald worden in de gegevensbank

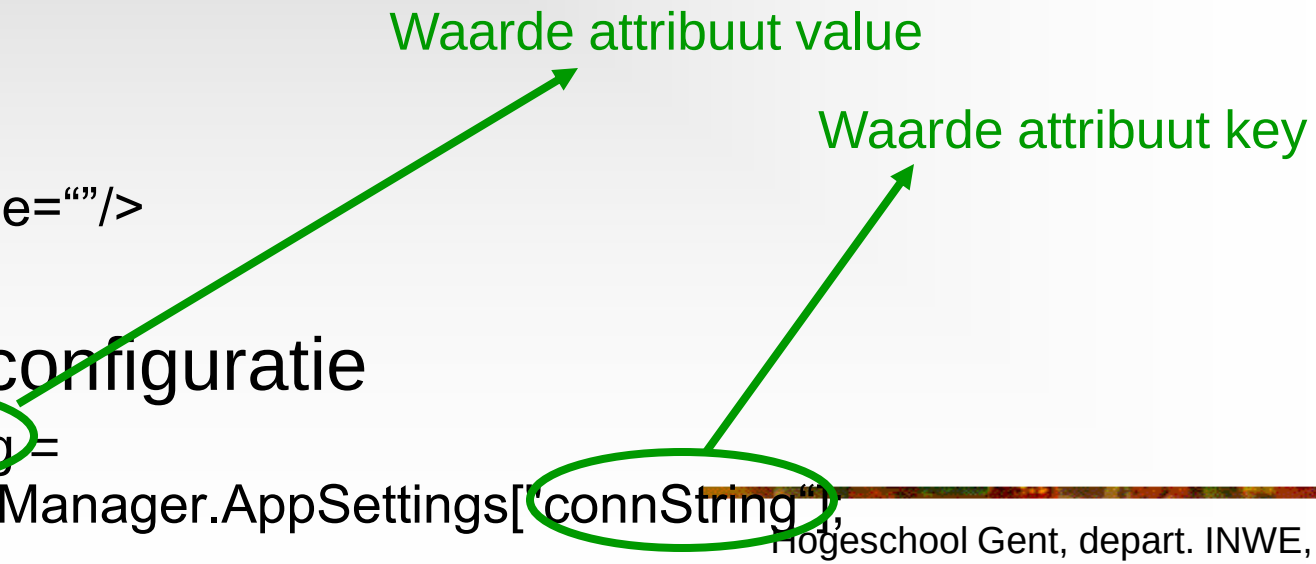
Configuratie: Web.config

- ConnectieString
- SQL-opdrachten
  - Select
  - Update
  - Delete
- Syntax

```
<appSettings>
<add key="" value=""/>
</appSettings>
```

- Ophalen configuratie

```
String connString =
    ConfigurationManager.AppSettings["connString"];
```



Aanmaak DataAdapter

- In Global.asax.cs
- Maakt gebruik van de specifieke DataAdapter OleDbAdapter
- Methode MaakAdapter

Aanmaak DataAdapter

- SelectCommand
  - Ingesteld bij constructie data-adapter
  - Primaire sleutels ook ophalen

```
adapter.MissingSchemaAction =
    MissingSchemaAction.AddWithKey;
```

- UpdateCommand
  - Hoe gegevens van een aangepaste rij doorsturen naar de gegevensbank
- DeleteCommand
  - Hoe gegevens van een verwijderde rij verwijderen in de gegevensbank

Aanmaak Command-object

- Constructor
  - SQL-opdracht (String)
  - OleDbConnection-object

```
String connString =
    ConfigurationManager.AppSettings["connString“];
OleDbConnection conn = new OleDbConnection(connString);
String update = ConfigurationManager.AppSettings["update“];
OleDbCommand command = new OleDbCommand(update,
    conn);
```

Aanmaak Command-object

- Instellen parameters
  - Aanmaak parameter
    - Naam
    - Type

```
param = command.Parameters.Add("@naam",
    OleDbType.VarChar);
```

  - Verbinden met een kolom in de DataSet

```
param.SourceColumn = "naam“;
```

  - Aangeven welke versie gebruikt moet worden bij uitvoeren opdracht
    - Oude ↔ nieuwe waarde

```
param.SourceVersion = DataRowVersion.Current;
```

OleDbParameter

- Eigenschap SourceColumn
  - Welke kolom bevat de waarde van de parameter
- Eigenschap SourceVersion
  - bepaalt welke versie van de kolom gebruikt wordt bij het uitvoeren van de opdracht

  → opsomming DataRowVersion
    - Current, Original, …

Wijzigingen doorvoeren

- Markeren van elke rij die toegevoegd, gewijzigd of verwijderd wordt
- Ongedaan maken wijzigingen
  - Methode RejectChanges van DataRow
- De methode Update van de DataAdapter past alle gemarkeerde rijen uit de DataSet aan in de gegevensbank

```
adapter.Update(gemeenteDS);
```

Zoeken van een rij in een DataSet

- De sleutels moeten ook opgehaald worden
  - Instellen in DataAdapter
- DataRowCollection
  - Methode Find
    - Argument: object of tabel van objecten die de primaire sleutel voorstellen
    - Resulaat: DataRow horende bij de opgegeven primaire sleutel

```
DataRow dr =
   gemeenteDS.Tables[0].Rows.Find(postcode);
```

Transaction

- Verschillende gegevensbankopdrachten moeten samen uitgevoerd worden
- Analoog verschillende .Net Data Providers
- Verschillende stappen
  - Starten transactie → transactie-object
  - Transactie-object toekennen aan Command-objecten
  - Uitvoeren opdrachten
  - Commit - Rollback

Transaction

```
// verbinding met gegevensbank
SqlConnection conn;
...
conn.Open();

// start transactie
SqlTransaction trans = conn.BeginTransaction();

// commando aanmaken en toevoegen aan transactie
SqlCommand opdracht = new SqlCommand();
opdracht.Connection = conn;
opdracht.Transaction = trans;
```

Transaction

```
// opdrachten uitvoeren
try {
    opdracht.CommandText = "Insert into Region (RegionID,
      RegionDescription) VALUES (100, 'Description')";
    opdracht.ExecuteNonQuery();
    opdracht.CommandText = "Insert into Region (RegionID,
      RegionDescription) VALUES (101, 'Description')";
    opdracht.ExecuteNonQuery();
    trans.Commit();
} catch(Exception e) {
    trans.Rollback();
} finally {
    conn.Close();
}
```

Informatie over ADO.NET

- Accessing Data with ADO.NET (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconaccessingdatawithadonet.asp)
- ADO.NET Overview (http://samples.gotdotnet.com/quickstart/aspplus/doc/adoplusoverview.aspx)
- .Net Data Access Architecture Guide (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/daag.asp)