

# Entity Framework Core (EF Core)

# Inhoud

- Wat is EF CORE
- EF Core Database ondersteuning
- Entity Framework Core vs Entity Framework 6.x
- Het Model (Database Context)
- Querying van gegevens
- Gegevens bewaren
- Stored procedures



# Wat is EF Core (Entity Framework Core)

- = Object-relational mapper (O/RM) dat .NET developers toelaat om met een database te werken dmv .NET objecten.
- =Lichtgewicht, uitbreidbare , en cross-platform versie of Entity Framework.
- EF Core heeft niet automatisch alle functionaliteit van EF6.x.
  - Minder frequent voorkomende functionaliteit wordt niet meer voorzien in EF Core.
- Database providers kunnen via NuGet packages worden geïnstalleerd bv:
  - PM> **Install-Package Microsoft.EntityFrameworkCore.SqlServer**
  - PM> **Install-Package Npgsql.EntityFrameworkCore.PostgreSQL**

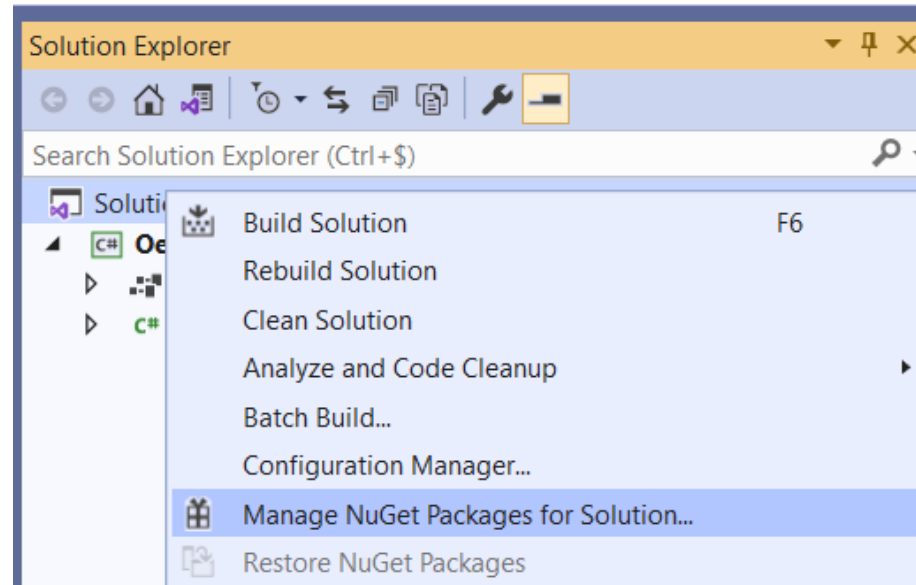
# Hoe gebruiken?

- Providers ondersteund door EF Core:

Database	NuGet Package
SQL Server	<a href="#"><u>Microsoft.EntityFrameworkCore.SqlServer</u></a>
MySQL	<a href="#"><u>MySql.Data.EntityFrameworkCore</u></a>
PostgreSQL	<a href="#"><u>Npgsql.EntityFrameworkCore.PostgreSQL</u></a>
SQLite	<a href="#"><u>Microsoft.EntityFrameworkCore.SQLite</u></a>
SQL Compact	<a href="#"><u>EntityFrameworkCore.SqlServerCompact40</u></a>
In-memory	<a href="#"><u>Microsoft.EntityFrameworkCore.InMemory</u></a>

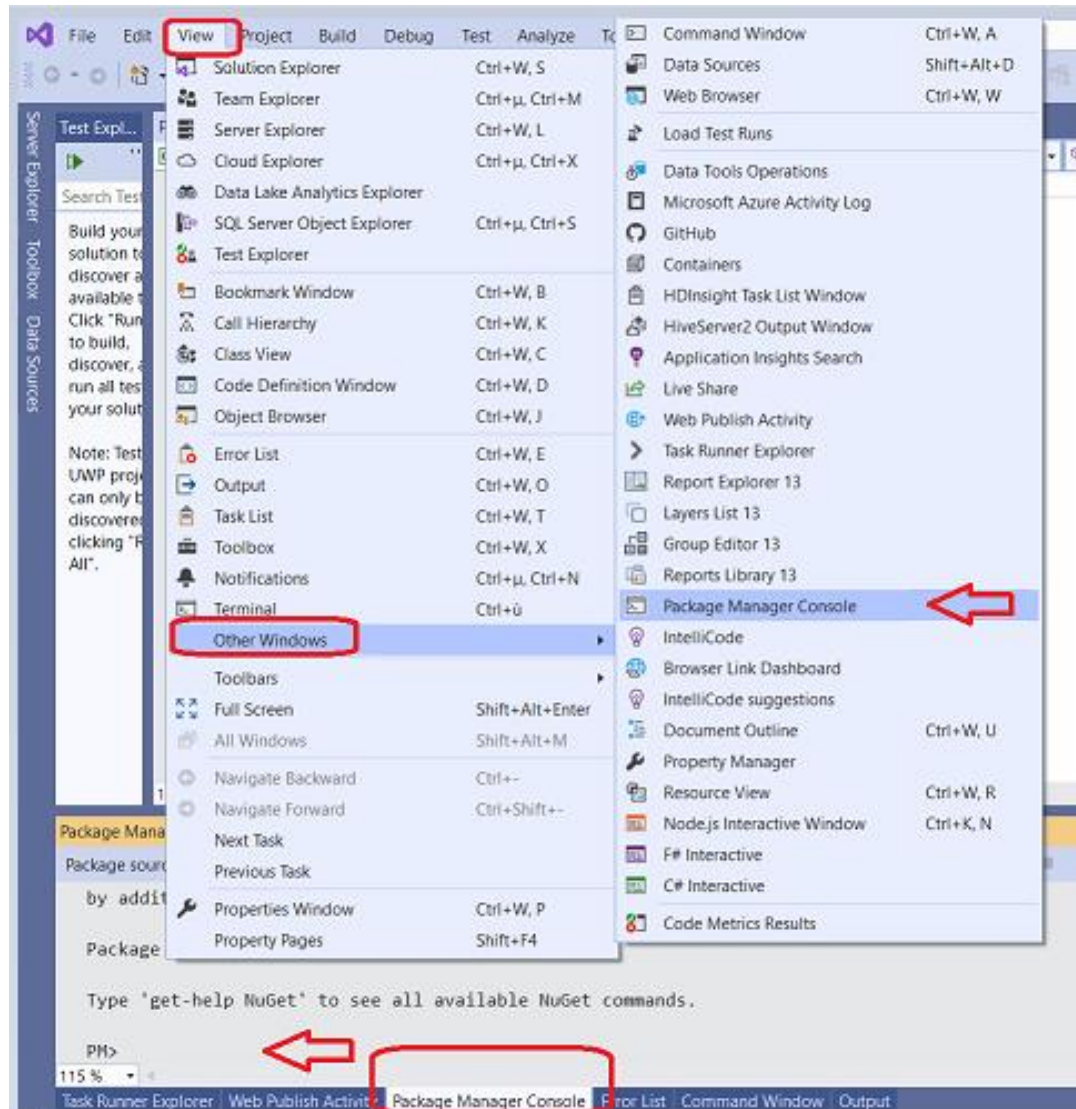
# Hoe gebruiken – Installatie van NuGet Package

- Via NuGet Package Manager UI:



- Of via NuGet Package Manager Console (zie volgende slide):

- Of via NuGet Package Manager Console:



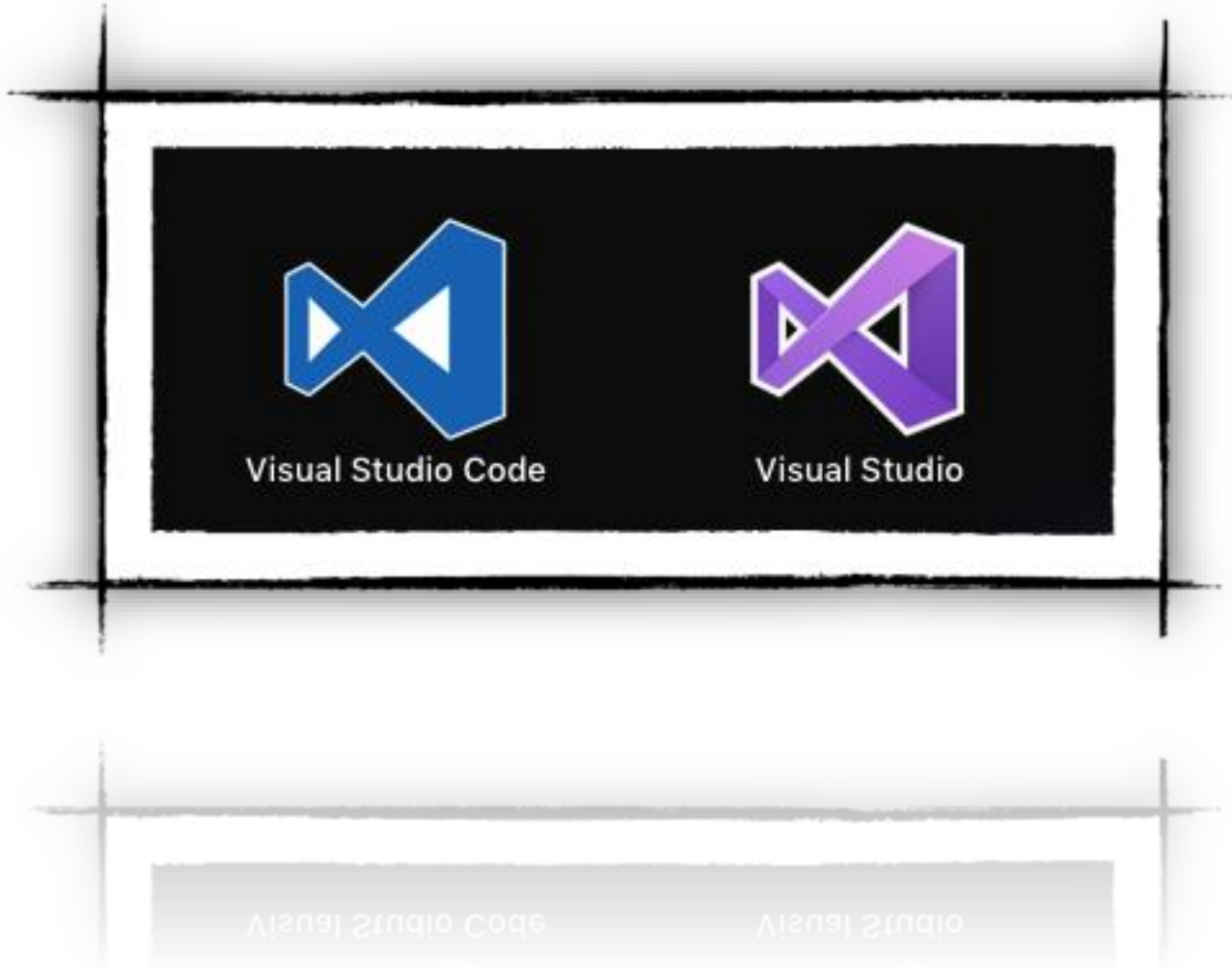
```
PM> Install-Package Microsoft.EntityFrameworkCore.SqlServer
```

115 %

Task Runner Explorer | Web Publish Activity | Package Manager Console | Error List | Command Window | Output

# Hoe gebruiken?

- .NET Core 3.x is cross-platform, open source en snel. Mogelijkheid om te Programmeren op Mac.

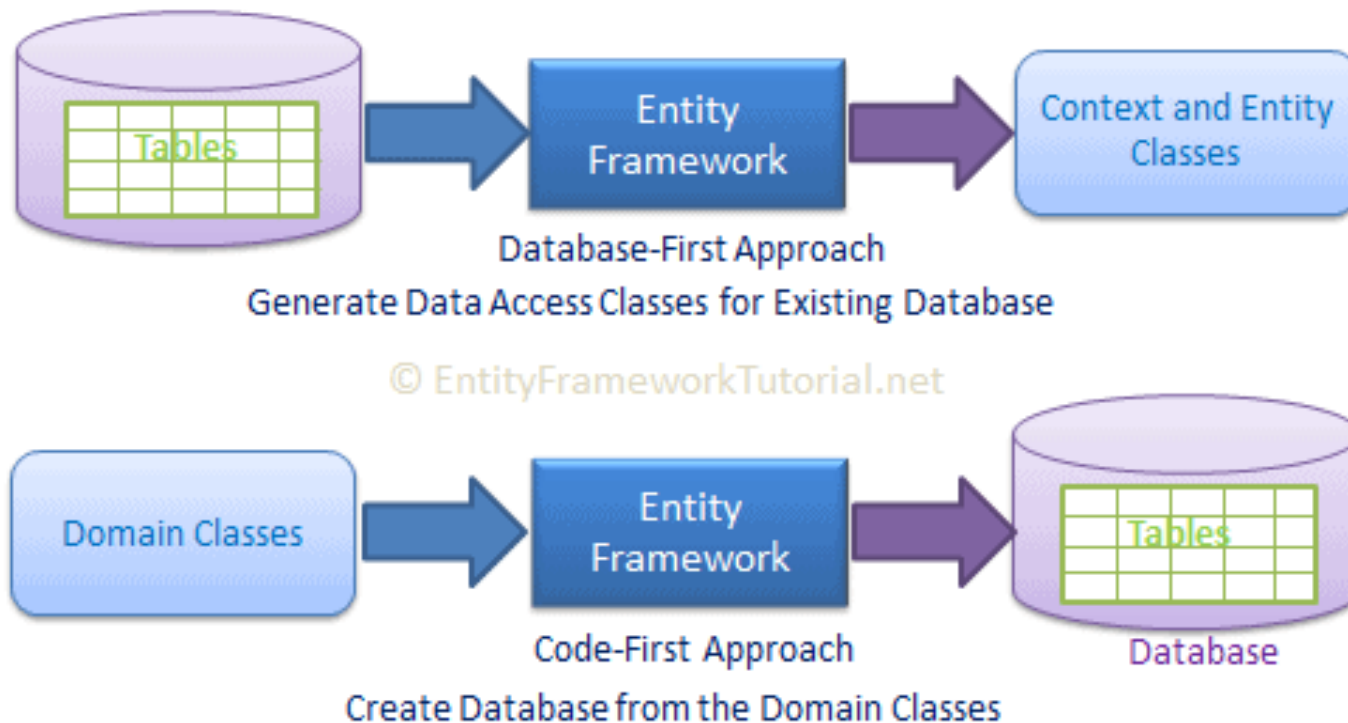


# EF Core Code-first versus Database-first

EF Core ondersteunt 2 soorten aanpak

1) **Code-First**

2) **Database-First**





# EF Core Database-first

## 1) Installeer NuGet Packages:

- ★ [Microsoft.EntityFrameworkCore.SqlServer](#)  
[Microsoft.EntityFrameworkCore.Tools](#)  
[Microsoft.EntityFrameworkCore.Design](#)

2) Run het volgende commando in NuGet Package Manager **Console** (Voor SqlServer db BierenDb): (In het rood staat de ConnectionString)

**Scaffold-DbContext** "Server=.;Database=Data Source=.;Initial Catalog=BierenDb;Integrated Security=True;Trusted\_Connection=True;" Microsoft.EntityFrameworkCore.SqlServer **-OutputDir** Models

**Oefening: Voor bovenstaande 2 stappen uit in een nieuwe Console App (Net.Core)**

# EF Core Database-first- BierenDb Database

The screenshot displays the Visual Studio IDE with the `BierenDbContext.cs` file open. The code defines a `BierenDbContext` class that inherits from `DbContext`. It includes several DbSet properties for `Bierens`, `Brouwers`, `Soortens`, `VwBierenMetBrouwers`, `VwBierenMetSoortens`, `VwBrouwersBeperkt`, and `VwToptiens`. The `OnConfiguring` method is overridden to use a connection string, and the `OnModelCreating` method is also overridden.

```
7 namespace OefScaffoldEFCoreBierenDb.Models
8 {
9     3 references
10     public partial class BierenDbContext : DbContext
11     {
12         0 references
13         public BierenDbContext()
14         {
15         }
16
17         0 references
18         public BierenDbContext(DbContextOptions<BierenDbContext> options)
19         : base(options)
20         {
21         }
22
23         0 references
24         public virtual DbSet<Bieren> Bierens { get; set; }
25
26         0 references
27         public virtual DbSet<Brouwer> Brouwers { get; set; }
28
29         0 references
30         public virtual DbSet<Soorten> Soortens { get; set; }
31
32         0 references
33         public virtual DbSet<VwBierenMetBrouwer> VwBierenMetBrouwers { get; set; }
34
35         0 references
36         public virtual DbSet<VwBierenMetSoorten> VwBierenMetSoortens { get; set; }
37
38         0 references
39         public virtual DbSet<VwBrouwersBeperkt> VwBrouwersBeperkts { get; set; }
40
41         0 references
42         public virtual DbSet<VwToptien> VwToptiens { get; set; }
43
44         0 references
45         protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
46         {
47             if (!optionsBuilder.IsConfigured)
48             {
49                 #warning: To protect potentially sensitive information in your connection string, you should move it out of source code. You can avoid scaffolding the connection string by using the app data store, e.g. the SQL Azure app key, or the 'APPSETTING_CONNECTION_STRING' app setting. Alternatively, you can set a connection string in the app.config file.
50                 optionsBuilder.UseSqlServer("Server=.;Database=Data Source=.;Initial Catalog=BierenDb;Integrated Security=True;Trusted_Connection=True;");
51             }
52         }
53
54         0 references
55         protected override void OnModelCreating(ModelBuilder modelBuilder)
56         {
57         }
58     }
59 }
```

The Solution Explorer on the right shows the project structure for `OefScaffoldEFCoreBierenDb`, including `Dependencies` and `Models` folders. The `Models` folder contains the following files:

- `Bieren.cs`
- `BierenDbContext.cs`
- `Brouwer.cs`
- `Soorten.cs`
- `VwBierenMetBrouwer.cs`
- `VwBierenMetSoorten.cs`
- `VwBrouwersBeperkt.cs`
- `VwToptien.cs`
- `Program.cs`

# Het Model

- een model bestaat uit meerdere entity klassen en een afgeleide klasse van DbContext, waarmee gegevens kunnen worden opgehaald uit en bewaard in de database.

bv:

```
C# Copy
using Microsoft.EntityFrameworkCore;
using System.Collections.Generic;

namespace Intro
{
    public class BloggingContext : DbContext
    {
        public DbSet<Blog> Blogs { get; set; }
        public DbSet<Post> Posts { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlServer(@"Server=(localdb)\mssqllocaldb;Database=MyDatabase;Trusted_Connection=True;");
        }
    }

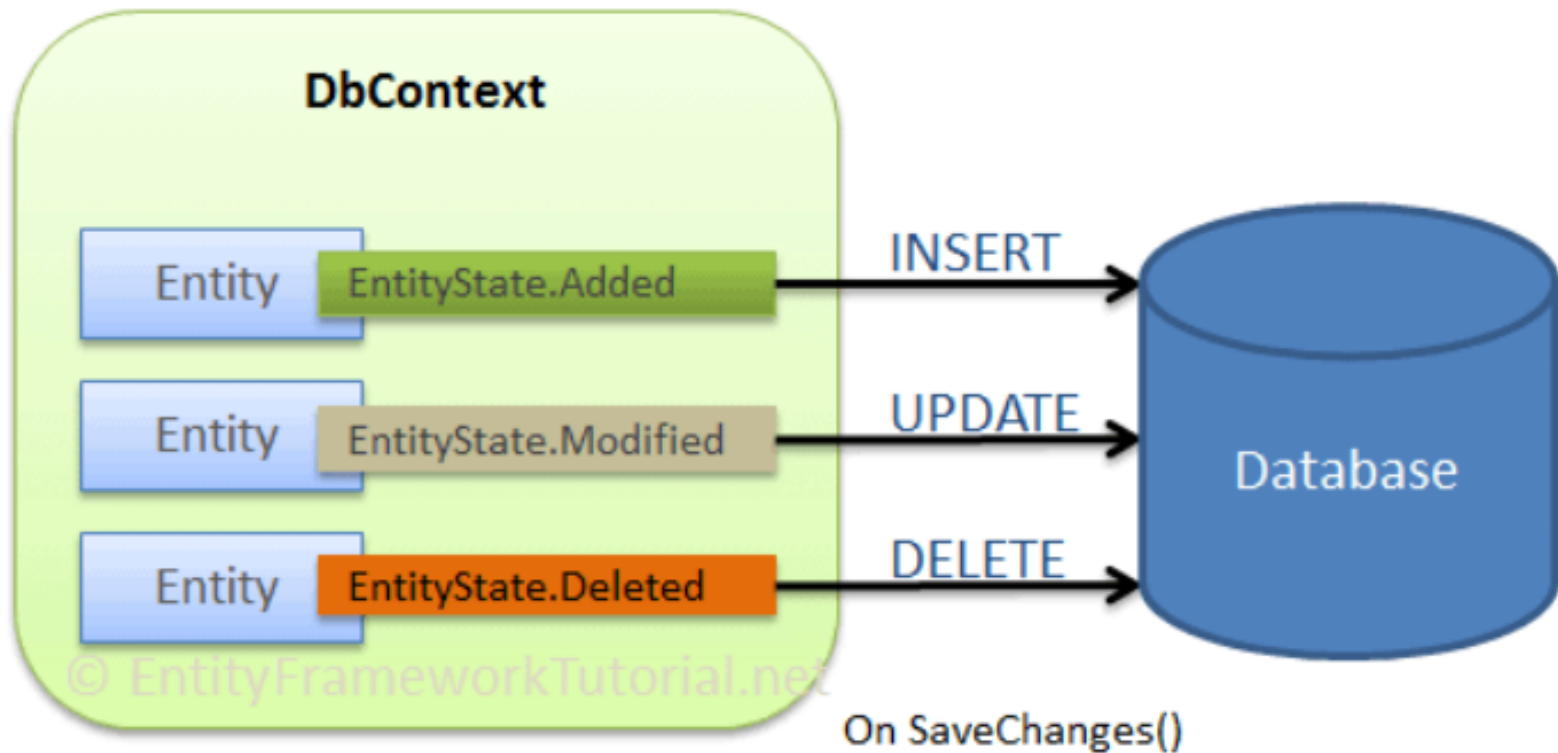
    public class Blog
    {
        public int BlogId { get; set; }
        public string Url { get; set; }

        public List<Post> Posts { get; set; }
    }

    public class Post
    {
        public int PostId { get; set; }
        public string Title { get; set; }
        public string Content { get; set; }

        public int BlogId { get; set; }
        public Blog Blog { get; set; }
    }
}
```

# CRUD (Create/Read/Update/Delete)operaties



# Querying

- Dmv van LINQ (Language Integrated Query) to Entity kunnen gegevens van entity klassen uit de database worden opgehaald

Bv:

```
C# Copy  
  
using (var db = new BloggingContext())  
{  
    var blogs = db.Blogs  
        .Where(b => b.Rating > 3)  
        .OrderBy(b => b.Url)  
        .ToList();  
}
```

**Oefening: Haal de bieren op in de Program.cs van de Console App (Net.Core)  
druk alle biernamen af op de console**

# Querying Bieren van BierenDb

0 references

```
class Program
```

```
{
```

0 references

```
static void Main(string[] args)
```

```
{
```

```
    Console.WriteLine("Hello World!");
```

```
    using(BierenDbContext bierenDb = new BierenDbContext())
```

```
    {
```

```
        List<Bieren> bieren = bierenDb.Bierens.ToList();
```

```
        foreach(Bieren bier in bieren)
```

```
        {
```

```
            Console.WriteLine(bier.Naam);
```

```
        }
```

```
    }
```

```
}
```

```
}
```

# Querying

- Ophalen van gerelateerde gegevens
  - **"Eager loading"**: gerelateerde gegevens worden uit database gehaald dmv **.Include(...)**

```
C# Copy
using (var context = new BloggingContext())
{
    var blogs = context.Blogs
        .Include(blog => blog.Posts)
        .ToList();
}
```

- **"Explicit loading"**: gerelateerde gegevens worden uit database gehaald dmv **.Reference(...).Load()**

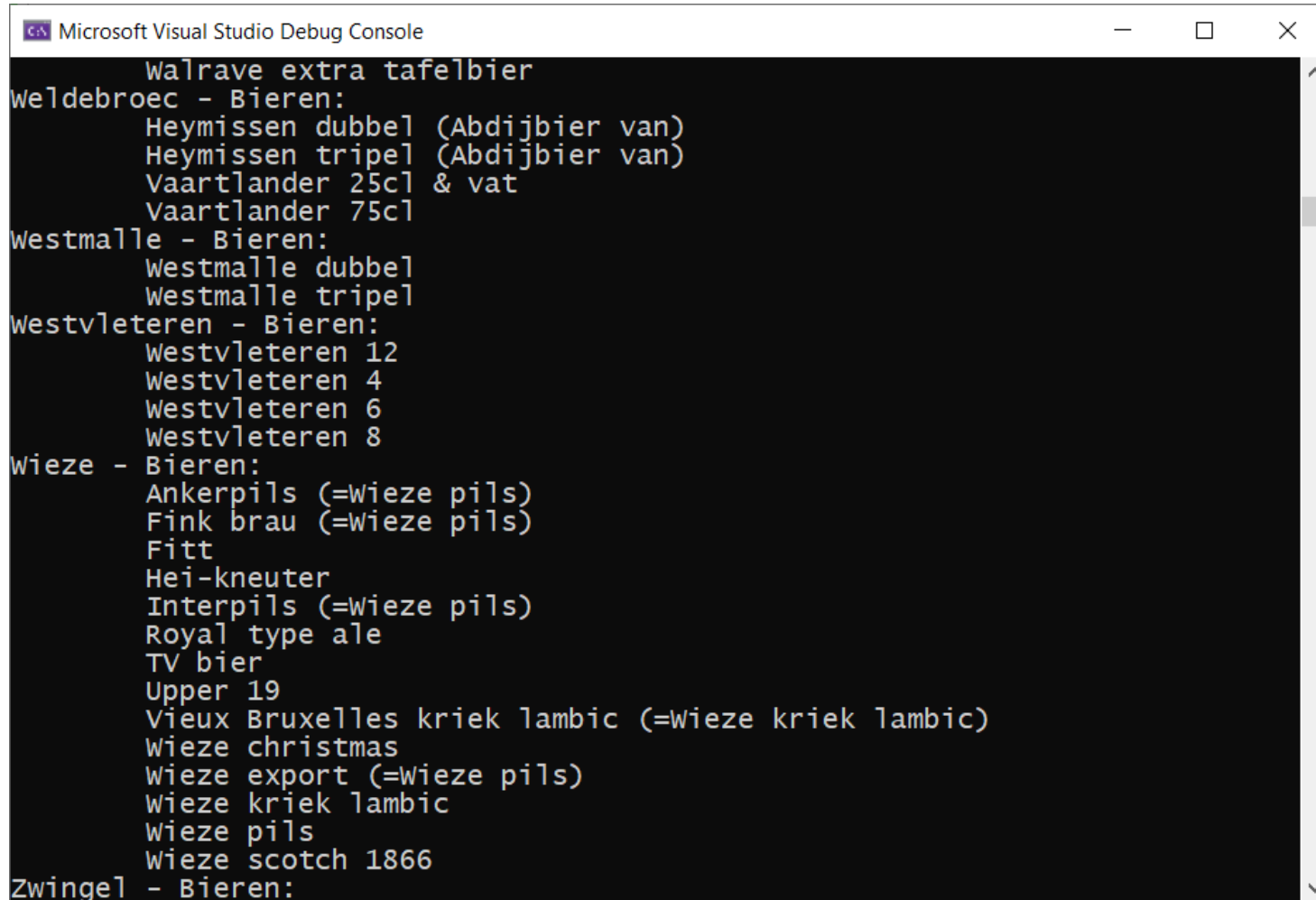
```
C# Copy
using (var context = new BloggingContext())
{
    var blog = context.Blogs
        .Single(b => b.BlogId == 1);

    context.Entry(blog)
        .Collection(b => b.Posts)
        .Load();

    context.Entry(blog)
        .Reference(b => b.Owner)
        .Load();
}
```

# Oefening – Ophalen gerelateerde gegevens

- **Oefening BierenDb op "Eager loading": Haal alle Brouwers op en hun bieren dmv .Include(...) en druk deze af op de console:**

A screenshot of the Microsoft Visual Studio Debug Console window. The window has a title bar with the Visual Studio logo and the text "Microsoft Visual Studio Debug Console". The console output is as follows:

```
Walrave extra tafelbier
Weldebroec - Bieren:
    Heymissen dubbel (Abdijbier van)
    Heymissen tripel (Abdijbier van)
    Vaartlander 25cl & vat
    Vaartlander 75cl
Westmalle - Bieren:
    Westmalle dubbel
    Westmalle tripel
Westvleteren - Bieren:
    Westvleteren 12
    Westvleteren 4
    Westvleteren 6
    Westvleteren 8
Wieze - Bieren:
    Ankerpils (=Wieze pils)
    Fink brau (=Wieze pils)
    Fitt
    Hei-kneuter
    Interpils (=Wieze pils)
    Royal type ale
    TV bier
    Upper 19
    Vieux Bruxelles kriek lambic (=Wieze kriek lambic)
    Wieze christmas
    Wieze export (=Wieze pils)
    Wieze kriek lambic
    Wieze pils
    Wieze scotch 1866
Zwingel - Bieren:
```



# Oefening – Ophalen gerelateerde gegevens

- Oefening BierenDb op "**Eager loading**": Haal alle Brouwers op en hun bieren dmv **.Include(...)**

```
using (BierenDbContext bierenDb = new BierenDbContext())
{
    List<Brouwer> brouwers = bierenDb.Brouwers.Include(b => b.Bierens).ToList();
    foreach (Brouwer brouwer in brouwers)
    {
        Console.WriteLine(brouwer.BrNaam + " - Bierens: ");
        foreach (Bieren bier in brouwer.Bierens)
        {
            Console.WriteLine("\t" + bier.Naam);
        }
    }
}
```

# Toevoegen van nieuwe gegevens

- Gegevens worden aangemaakt, gewijzigd en verwijderd in the database dmv instantie van de entity klassen

• Bv:

```
C# Copy  
  
using (var db = new BloggingContext())  
{  
    var blog = new Blog { Url = "http://sample.com" };  
    db.Blogs.Add(blog);  
    db.SaveChanges();  
}
```

**Oefening BierenDb. Voeg een nieuw bier toe met naam "TESTBIER" met soort "Alcoholvrij" (zoek eerst het SoortNr op in Soorten) aan de Brouwer "Zwingel" (zoek eerst deze Brouwer op via BrNaam). Bewaar de gegevens in de BierenDb**

```
using (BierenDbContext bierenDb = new BierenDbContext())  
{  
    //Zoek soort met naam 'alcoholvrij'  
    Soorten soort = bierenDb.Soortens.Where(s => s.Soort.ToLower() == "alcoholvrij").FirstOrDefault();  
    if (soort == null)  
    {  
        Console.WriteLine("Soort 'alcoholvrij' niet gevonden");  
        return;  
    }  
    //Zoek brouwer met naam 'Zwingel'  
    //Voeg alcoholvrij Bier aan brouwer Zwingel Toe met naam 'TESTBIER'  
    //Bewaar nieuw bier in database  
}
```

# Toevoegen van nieuwe gegevens - oplossing

**Oefening BierenDb. Voeg een nieuw bier toe met naam "TESTBIER" met soort "Alcoholvrij" (zoek eerst het SoortNr op in Soorten) aan de Brouwer "Zwingel" (zoek eerst deze Brouwer op via BrNaam). Bewaar de gegevens in de BierenDb.**

**Tip: voor nieuw BierNr (géén autoincrementele kolom), gebruik:**

```
int newbierNr = bierenDb.Bierens.Max(b => b.BierNr) + 1;
```

```
using (BierenDbContext bierenDb = new BierenDbContext())
{
    //Zoek bier met naam 'alcoholvrij'
    Soorten soort = bierenDb.Soortens.Where(s => s.Soort.ToLower() == "alcoholvrij").FirstOrDefault();
    if (soort == null)
    {
        Console.WriteLine("Soort 'alcoholvrij' niet gevonden");
        return;
    }
    //Zoek brouwer met naam 'Zwingel'
    Brouwer brouwer = bierenDb.Brouwers.Where(s => s.BrNaam.ToLower() == "zwingel").FirstOrDefault();
    if (brouwer == null)
    {
        Console.WriteLine("Brouwer 'Zwingel' niet gevonden");
        return;
    }
    //Voeg alcoholvrij Bier aan brouwer Zwingel Toe met naam 'TESTBIER'
    int newbierNr = bierenDb.Bierens.Max(b => b.BierNr) + 1;
    Bieren nieuwBier = new Bieren { BierNr = newbierNr, Naam = "TESTBIER", BrouwerNr = brouwer.BrouwerNr, SoortNr = soort.SoortNr, Alcohol = 0.0 };
    //Bewaar nieuw bier in database
    bierenDb.Bierens.Add(nieuwBier);
    bierenDb.SaveChanges();
}
```

# Wijzigen bestaande gegevens

**Wijzig de naam van TESTBIER in TESTALCOHOLVRIJ Bier en bewaar de wijziging in de database. Zoek eerst TESTBIER op in de database**

```
using (BierenDbContext bierenDb = new BierenDbContext())
{
    //Zoek bier met naam 'TESTBIER'
    Bieren bier = bierenDb.Bierens.Where(b => b.Naam.ToUpper() == "TESTBIER").FirstOrDefault();
    if (bier == null)
    {
        Console.WriteLine("Bier 'TESTBIER' niet gevonden");
        return;
    }
    bier.Naam = "TESTALCOHOLVRIJ";
    bierenDb.Bierens.Update(bier);
    bierenDb.SaveChanges();
}
```

# Verwijderen van bestaande gegevens

Verwijder TESTBIER uit de database.

```
using (BierenDbContext bierenDb = new BierenDbContext())
{
    //Zoek bier met naam 'TESTALCOHOLVRIJ'
    Bieren bier = bierenDb.Bierens.Where(b => b.Naam.ToUpper() == "TESTALCOHOLVRIJ").FirstOrDefault();
    if (bier == null)
    {
        Console.WriteLine("Bier 'TESTALCOHOLVRIJ' niet gevonden");
        return;
    }
    //Verwijder bier
    bierenDb.Bierens.Remove(bier);
    bierenDb.SaveChanges();
}
```

# Stored Procedures

- Gebruik **FromSqlRaw** methode die SQL queries en stored procs kan uitvoeren.

Beperkingen, bv:

- SQL queries kunnen enkel gebruikt worden om entity types terug te geven die deel uitmaken van het model.

```
string name = "Bill";  
var context = new SchoolContext();  
var students = context.Students.FromSqlRaw($"Select * from Students").ToList();
```

- Voor INSERT, UPDATE, DELETE queries: **ExecuteSqlRaw**.
  - Geef een geheel getal terug dat aangeeft hoeveel rijen erin zijn betrokken
  - ExecuteSqlRaw moet worden gebruikt op context.**Database**

```
var context = new SchoolContext();  
var rowsAffected = context.Database.ExecuteSqlRaw("Update Students set FirstName = 'Bill' where StudentId = 1;");
```

- Voor stored procedures: **ExecuteSqlRaw**.

```
var context = new SchoolContext();  
var param = new SqlParameter() { ParameterName = "@FirstName", SqlDbType = System.Data.SqlDbType.VarChar,  
    Direction = System.Data.ParameterDirection.Input, Size = 50, Value = "Bill" };  
var students = context.Database.ExecuteSqlRaw("GetStudents @FirstName", param).ToList();
```

# Stored Procedures - Oefening

- Gebruik **FromSqlRaw** methode om alle biergegevens op te vragen, omgekeerd alfabetisch gesorteerd op Naam via een SQL query
- Maak gebruik van **ExecuteSqlCommand** om de stored procedure **sp\_GeefBierenVoorSoortNr** uit te voeren waarin een parameter **@SoortNr** wordt meegegeven bv 22 (Geuze) en die alle biergegevens van alle bieren van deze soort teruggeeft.

```
using (BierenDbContext bierenDb = new BierenDbContext())
{
    //Voer RAW Sql query uit:
    List<Bieren> bieren = bierenDb.Bierens.FromSqlRaw("SELECT * from Bieren").OrderByDescending(b => b.Naam).ToList();
    foreach (Bieren bier in bieren)
    {
        Console.WriteLine(bier.Naam);
    }
    //bieren.ForEachAsync(b => Console.WriteLine(b.Naam));

    var param = new SqlParameter()
    {
        ParameterName = "@SoortNr",
        SqlDbType = System.Data.SqlDbType.Int,
        Direction = System.Data.ParameterDirection.Input,
        Value = 22 //Geuze
    };
    List<Bieren> geuzeBieren = bierenDb.Bierens.FromSqlRaw("sp_GeefBierenVoorSoortNr @SoortNr", param).ToList();
    foreach (Bieren bier in geuzeBieren)
    {
        Console.WriteLine(bier.Naam);
    }
}
```

# Referenties

<https://www.entityframeworktutorial.net/efcore>

[http://sd.blackball.lv/library/Mastering\\_Entity\\_Framework\\_Core\\_2.0.pdf](http://sd.blackball.lv/library/Mastering_Entity_Framework_Core_2.0.pdf)



