

## Hoofdstuk 5 : Entity Framework - Code First

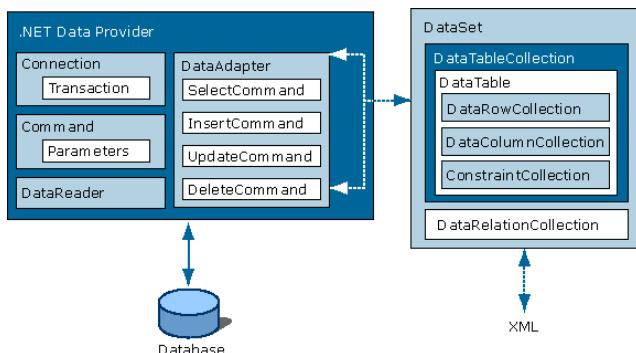
## Hoofdstuk 5 : EF - Code First

1. Inleiding
2. EF Code First : Domain Model
3. EF Code First : DAL
  - DbContext
  - Db Creatie/Init
4. EF Code First : Configuratie EF
  - Gegenereerde db
5. EF Code First : customisatie mapping
  - Verwijderen van conventies
  - Model - Data Annotations
  - Model – Fluent API
6. EF : Linq to Entities
7. EF : DbContext en updates
8. Extra links, tools, referenties

# 1. Inleiding

## ▶ ADO.NET

- Library om volledig zelf de persistentielaag te bouwen



HoGent

3

# 1. Inleiding

## ▶ Entity Framework, versie 5.0

- is een ORM (Object Relational mapper) framework.
- Genereert de persistentielaag
- Infrastructuur om objecten te **mappen** naar database en viceversa
  - Mapt klassen naar tabellen, properties naar kolommen in tabel
  - Mapt objecten naar rijen
  - Mapt associaties naar FK relaties
  - Ondersteunt overerving
- Een **API, Linq to Entities**, voor het opvragen en manipuleren van de objecten. De acties worden vertaald naar queries op de database
  - .Net taal syntax, gecompileerd!
  - Onafhankelijk van de backend SQL dialect, OO taal
- Meer op <http://msdn.microsoft.com/en-US/data/ef>

HoGent

4

# 1. Inleiding

## ► 3 werkwijzen

### ◦ Database First

- Je vertrekt van de database en genereert een model (een .edmx bestand), die dan kan worden gecustomiseerd.

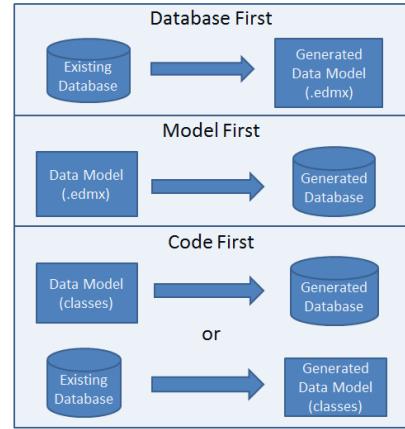
### ◦ Model First

- Je maakt .edmx aan en genereert dan de database.

Nadeel : geen POCO (Plain Old CLR Objects), erven van EntityType

### ◦ Code First

1. Maak domain model aan (POCO's)
2. Aanmaken DAL (Data Access Layer)
  1. Definieer de context
  2. Bepaal db creatie/ instantiatie strategie
3. Maak applicatie en run (afhankelijk van de strategie wordt de database (telkens opnieuw) gecreëerd/gemapped en geïnstantieerd)
4. Pas mapping aan waar nodig



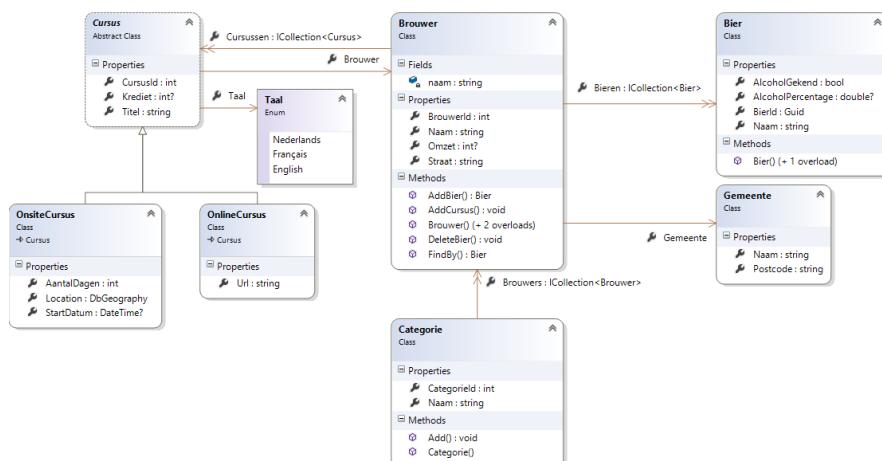
HoGent

5

# 2. EF Code First : Domain Model

## ► Maak domein model aan

### ◦ Open solution [EFExplained.sln](#), bekijk Models > domain.cd



## 2. EF Code First : Domain Model

- ▶ Maak domein model aan
  - Model bevat **POCO's**
    - = Plain Old CLR Objects
    - = persistence-ignorant entity types
  - EF Code First hanteert "**convention over configuration**". Hou je dus aan de conventies (zie volgende slides) bij de aanmaak van het domein model
  - Meer op <http://msdn.microsoft.com/en-US/data/jj679962>

HoGent

7

## 2. EF Code First : Domain Model

- ▶ Conventies
  - Elke klasse wordt een tabel
  - Elke property wordt een kolom in de tabel
  - Primary Keys
  - Enums
  - Spatial data
  - Associaties worden relaties
    1. Associatie in 1 richting navigeerbaar => 1:n relatie in db
      - 1.A. Als associatie navigeerbaar aan de 1 kant
      - 1.B. Als associatie navigeerbaar aan de veel kant
    2. Associatie is bi-directioneel (1:N, N:M, 1:1)
    3. Associatie met klasse zonder primary key
  - Overerving

HoGent

8

## 2. EF Code First : Domain Model

### ▶ Conventies

#### ◦ Elke klasse wordt een tabel

- DB: naam tabel = naam klasse in het meervoud (Engels : + s) 
  - Voorbeeld : Bier klasse -> tabel Biers (Cursus klasse -> tabel Cursus)
- Klasse moet voldoen aan de conventies :
  - Klasse moet **public** zijn.
  - Klasse mag niet **sealed** zijn
  - Een klasse moet een **public or protected** default constructor hebben (constructor zonder parameters).
  - Opm : Je kan configureren welke klassen zullen worden omgezet naar tabellen, en wat de naam van de tabel wordt (zie later)

HoGent

9

## 2. EF Code First : Domain Model

### ▶ Conventies

#### ◦ Elke property wordt een kolom in de tabel

- naam kolom = naam van de property
  - Naam property in klasse Brouwer => kolom Naam in tabel Brouwers
- Datatypes 
  - een string => nvarchar(MAX) en NULL allowed
  - bool, int,... => bit, int, ....NOT NULL
  - bool?, int?...=> bit, int, .... NULL
- Moet getter en setter hebben (setter hoeft niet public te zijn)

HoGent

10

## 2. EF Code First : Domain Model

### ▶ Conventions

#### ◦ Primary keys :

- De property met naam 'Id' of '<classname>Id' wordt de primary key
  - Voorbeeld : In klasse Brouwer : property Id, ID, BrouwerId, BrouwerID
- Als property van type int, long, short => Wordt het eveneens een identity kolom (autonummering)
- Als property van type string => dan geen autonummering, nvarchar(128)
- Guid => Guid 

HoGent

11

## 2. EF Code First : Domain Model

### ▶ Conventions

- Voorbeeld EF mapping Brouwer klasse -> Brouwers tabel
  - => Verdere configuratie adhv Data Annotations of fluent API

```
public class Brouwer
{
    #region Properties

        public int BrouwerId { get; set; }
        private string naam;
        public string Naam { get; set; }

        public string Straat { get; set; }

        public int? Omzet { get; set; }

        public virtual Gemeente Gemeente { get; set; }

        public virtual ICollection<Bier> Bieren { get; set; }

        public virtual ICollection<Course> Courses { get; set; }

    #endregion
}
```



Brouwers			
Column Name	Data Type	Allow Nulls	
BrouwerId	int	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Naam	nvarchar(MAX)	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Straat	nvarchar(MAX)	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Omzet	int	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Gemeente_Postcode	nvarchar(128)	<input checked="" type="checkbox"/>	<input type="checkbox"/>

12

## 2. EF Code First : Domain Model

### ▶ Conventions

#### ◦ Enums :

- Wordt 1 kolom van het type int. Als property nullable dan is NULL allowed anders niet
- De integer waarde van de enum wordt opgeslaan in de tabel
- Voor de enum wordt geen tabel voorzien met de mogelijke waarden van de enum

```
public abstract class Cursus
{
    public int CursusId { get; set; }
    public string Titel { get; set; }
    public int? Krediet { get; set; }
    public Taal Taal { get; set; }
}
```

```
public enum Taal
{
    Nederlands,
    Français,
    English
}
```

Column Name	Data Type	Allow Nulls
CursusId	int	<input type="checkbox"/>
Titel	nvarchar(100)	<input type="checkbox"/>
Krediet	int	<input checked="" type="checkbox"/>
Taal	int	<input type="checkbox"/>

HoGent

13

## 2. EF Code First : Domain Model

### ▶ Conventions

#### ◦ Spatial data

- De positie is een breedte en lengte coördinaat
- De coördinaten zijn decimale graden. Bvb 50.856284 is 50° 51' 22"
- decimale graad= graden + (minuten/60.0) + seconden/3600.0

```
public class OnsiteCursus : Cursus
{
    public DbGeography Location { get; set; }
    public int AantalDagen { get; set; }
    public DateTime StartDatum { get; set; }
}
```

Column Name	Data Type	Allow Nulls
CursusId	int	<input type="checkbox"/>
Titel	nvarchar(100)	<input type="checkbox"/>
Krediet	int	<input checked="" type="checkbox"/>
Taal	int	<input type="checkbox"/>
Url	nvarchar(100)	<input checked="" type="checkbox"/>
Location	geography	<input checked="" type="checkbox"/>
AantalDagen	int	<input checked="" type="checkbox"/>
StartDatum	datetime	<input checked="" type="checkbox"/>

```
, Location = DbGeography.FromText("POINT(50.862882 3.298903)")
```

HoGent

14

## 2. EF Code First : Domain Model

### ▶ Conventies :

#### ◦ **Associaties worden relaties**

- Een associatie wordt in EF een **Navigation property** genoemd
- Steeds **public virtual**
  - **get** accessor moet public zijn (Nodig voor generatie lazy loading proxy en change tracking proxies)
  - **set** hoeft niet public te zijn
- Voor een verzameling gebruik je **ICollection<T>**, anders **T**
- **virtual** : EF maakt proxy klassen aan voor o.a. lazy loading
- Instantieer collections in default constructor

```
public virtual Gemeente Gemeente { get; set; }

public virtual ICollection<Bier> Bieren { get; set; }

public virtual ICollection<Course> Courses { get; set; }
```

```
public Brouwer()
{
    Bieren = new List<Bier>();
    Courses = new List<Course>();
    Omzet = null;
}
```

HoGent

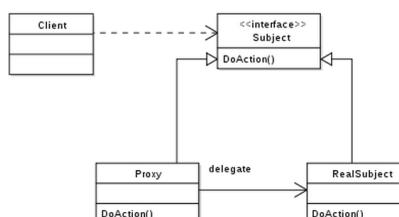
15

## 2. EF Code First : Domain Model

### ▶ Conventies

#### ◦ **Associaties worden relaties**

- **virtual** : EF maakt **proxy** klassen aan voor o.a. **lazy loading**
- Lazy loading betekent bvb
  - Igv opvragen van brouwers, worden enkel de brouwer gegevens opgehaald, nog geen geassocieerde objecten
  - Verwijs je in code naar de property Bieren, dan zal EF de bieren van die brouwer ophalen
  - **Als geen keyword "virtual", dan geen lazy loading!**
- Proxy pattern



HoGent

16

## 2. EF Code First : Domain Model

### ▶ Conventies

#### ◦ **Associaties worden relaties**

1. Associatie in 1 richting navigeerbaar => 1:n relatie in db
  - 1.A. Als associatie navigeerbaar aan de 1 kant
  - 1.B. Als associatie navigeerbaar aan de veel kant
2. Associatie is bi-directioneel
  - 2.1 1..\* associatie
  - 2.2 \*..\* associatie
  - 2.3 1..1 associatie
3. Associatie met klasse zonder primary key

HoGent

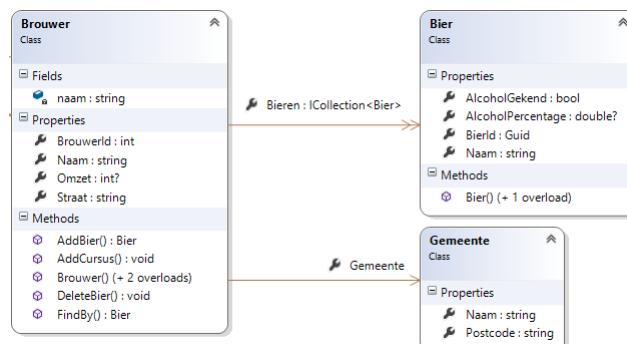
17

## 2. EF Code First : Domain Model

### ▶ Conventies : Associaties worden relaties

#### 1. **Associatie in 1 richting navigeerbaar => 1:n relatie in db**

- Voorbeeld brouwer kent zijn bieren, bier kent zijn brouwer niet (Brouwer heeft een navigation property Bieren, Bier heeft geen navigation property Brouwer)



HoGent

18

## 2. EF Code First : Domain Model

- ▶ conventies : associaties worden relaties

### 1. Associatie in 1 richting navigeerbaar => 1:n relatie in db

- 1.A. Als associatie navigeerbaar aan de 1 kant
  - 1:n relatie in de DB
  - FK (null) in de tabel aan de N kant, naam FK : naamtafel\_naam PK (tenzij FK ook gedefinieerd in de klasse, dan deze naam)
  - cascading delete = no action 
  - Voorbeeld : Brouwer bevat navigation property Gemeente

```
public class Brouwer
{
    #region Properties

    public int BrouwerId { get; set; }
    private string naam;
    public string Naam { get; set; }

    public string Straat { get; set; }

    public int? Omzet { get; set; }

    public virtual Gemeente Gemeente { get; set; }

    public virtual ICollection<Bier> Bieren { get; set; }

    public virtual ICollection<Course> Courses { get; set; }

    #endregion
}
```



19

## 2. EF Code First : Domain Model

- ▶ conventies : associaties worden relaties

### 1. Associatie in 1 richting navigeerbaar => 1:n relatie in db

- 1.B. Als associatie navigeerbaar aan de \* kant
  - > DB : 1:n relatie, FK(null) in tabel aan N kant, cascading delete = no action
  - Vb. Brouwer bevat navigation property Bieren (ICollection<Bier>)

```
public class Brouwer
{
    #region Properties

    public int BrouwerId { get; set; }
    private string naam;
    public string Naam { get; set; }

    public string Straat { get; set; }

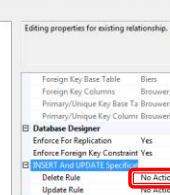
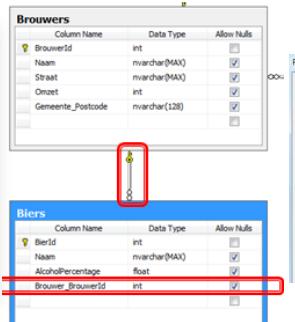
    public int? Omzet { get; set; }

    public virtual Gemeente Gemeente { get; set; }

    public virtual ICollection<Bier> Bieren { get; set; }

    public virtual ICollection<Course> Courses { get; set; }

    #endregion
}
```



HoGent

20

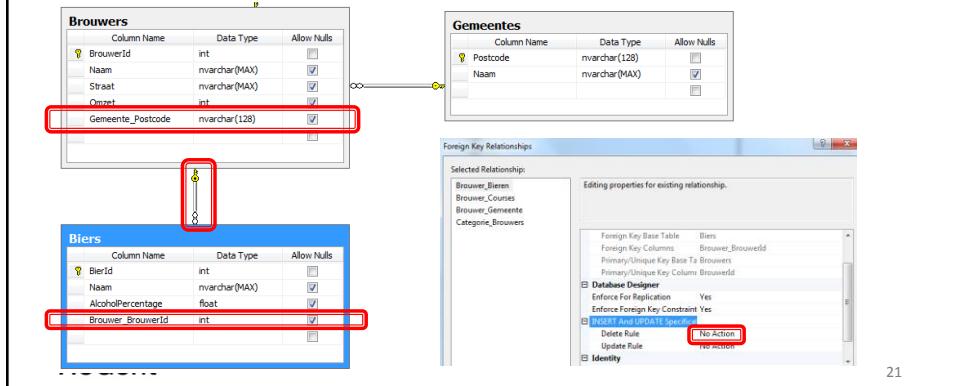
## 2. EF Code First : D

-Maakt 1:n relatie aan  
 -Voegt ook een foreign key toe, aan de N kant (naamTabel\_keynaam), allows null en cascading delete=no action

### Conventies : Associaties worden relaties

#### 1. Associatie in 1 richting navigeerbaar => 1:n relatie in db

- Aan de 1 kant : Brouwer – Gemeente : Brouwer bevat Gemeente
- Aan de \* kant : Brouwer – Bier : Brouwer bevat ICollection<Beer>



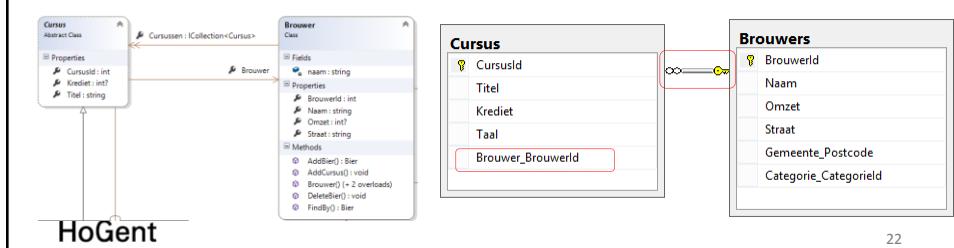
21

## 2. EF Code First : Domain Model

### Conventies : Associaties worden relaties

#### 2. Associatie is bi-directioneel

- = navigeerbaar aan beide kanten
  - ⇒ in DB : overeenkomstige associatie (1:1, 1:N, N:M met tussentabel)
- 2.1 1..\* associatie**
- > DB : 1:n relatie, FK(null) in tabel aan N kant, cascading delete = no action
- Voorbeeld* : Brouwer – Cursus : 1..\*



22

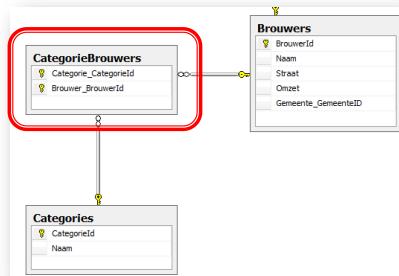
## 2. EF Code First : Domain Model

- ▶ conventies : associaties worden relaties

### 2. Associatie is bi-directioneel

#### 2.2 \*..\* associatie

- DB : bevat intersectietabel met sleutel uit beide tabellen
- Voorbeeld Stel Brouwer – Categorie als volgt gedefinieerd
  - Brouwer : ICollection<Categorie>
  - Categorie : ICollection<Brouwer>



#### 2.3 1..1 associatie

- DB : 1:1 relatie. Plaats FK definieren a.d.h.v. de Fluent API (zie verder)

HoGent

23

## 2. EF Code First : Domain Model

- ▶ conventies :

### 3. Associatie met klasse zonder primary key (is steeds 1:1)

Klasse zonder PK (=value object, geen entity) wordt geen tabel maar wordt onderdeel van de tabel van omvattende klasse.

- Stel klasse Cursus en klasse Adres. Adres heeft geen primary key

```
public abstract class Cursus
{
    public int CursusId { get; set; }
    public string Titel { get; set; }
    public int? Krediet { get; set; }
    public Taal Taal { get; set; }
    public Adres Adres { get; set; }
}

public class Adres
{
    public string Straat { get; set; }
    public string Postcode { get; set; }
    public string Gemeente { get; set; }
}
```

Column Name	Data Type	Allow Nulls
CursusId	int	<input type="checkbox"/>
Titel	nvarchar(100)	<input type="checkbox"/>
Krediet	int	<input checked="" type="checkbox"/>
Taal	int	<input type="checkbox"/>
Adres_Straat	nvarchar(MAX)	<input checked="" type="checkbox"/>
Adres_Postcode	nvarchar(MAX)	<input checked="" type="checkbox"/>
Adres_Gemeente	nvarchar(MAX)	<input checked="" type="checkbox"/>

24

## 2. EF Code First : Domain Model

### ▶ conventies :

#### ◦ Overerving

- Domein : overerving
  - Basisklasse : al dan niet abstract
  - Voorbeeld Cursus basisklasse. Online- en OnsiteCursus zijn subklassen
- DB : Table per Hierarchy (=default voor mappen overerving in tabellen)
  - 1 tabel : met de properties van alle types (zowel van Cursus, als van OnlineCursus, OnsiteCursus)
  - Naam tabel = naam base class (in meervoud)
  - Tabel bevat een extra kolom met de naam "Discriminator" (nvarchar(128)), met als waarden de namen van de (afgeleide) klassen. Hierdoor kan je weten of het record een Cursus, Online- of OnsiteCursus is
    - Discriminator kolom bevat hier dus de waarde Cursus (tenminste als geen abstracte klasse), OnlineCursus of OnsiteCursus
  - Kolom url zal bvb enkel een waarde hebben als Discriminator = OnlineCursus

HoGent

25

## 2. EF Code First : Domain Model

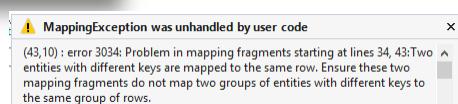
### ▶ conventies :

#### ◦ Overerving

- DB : Table per Hierarchy (=default voor mappen overerving in tabellen)
  - Opmerking (bug in EF5?) : opdat de mapping zou werken, dien je boven primary key property de annotatie [Key] te zetten. Zie later.

```
public abstract class Cursus
{
    [Key]
    public int CursusId { get; set; }
    public string Titel { get; set; }
    public int? Krediet { get; set; }
    public Taal Taal { get; set; }
}
```

- De fout die je anders krijgt :

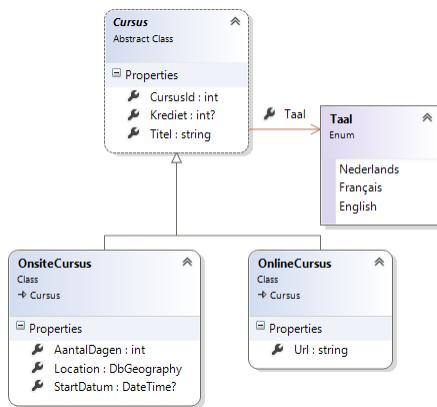


HoGent

## 2. EF Code First : Domain Model

### ▶ Conventies : Overerving

- Domain



HoGent

Database (1 tabel)

Column Name	Data Type	Allow Nulls
CursusId	int	<input type="checkbox"/>
Titel	nvarchar(MAX)	<input checked="" type="checkbox"/>
Krediet	int	<input checked="" type="checkbox"/>
Taal	int	<input type="checkbox"/>
Url	nvarchar(MAX)	<input checked="" type="checkbox"/>
Location	geography	<input checked="" type="checkbox"/>
AantalDagen	int	<input checked="" type="checkbox"/>
StartDatum	datetime	<input checked="" type="checkbox"/>
Discriminator	nvarchar(128)	<input type="checkbox"/>
Brouwer_Brouwerveld	int	<input checked="" type="checkbox"/>

De extra kolom om onderscheid te maken tussen Online- en OnsiteCursus (en Cursus als Cursus niet abstract)

## 3. EF Code First : DAL - DbContext

### ▶ De persistentielaaag

- In .NET spreken we van de DAL (Data Access Layer)
- Hoe ziet dit eruit als we gebruik maken van EF?
  - Benodigde References voor EF : [EntityFramework](#) en [System.Data.Entity](#)
    - Indien niet toegevoegd : Toevoegen met Nuget (zie volgende slide)
    - DAL laag maken we aan in de Models folder > folder DAL
    - Deze folder bevat minstens 2 klassen
      - Klasse die erft van DbContext
      - Initializer klasse

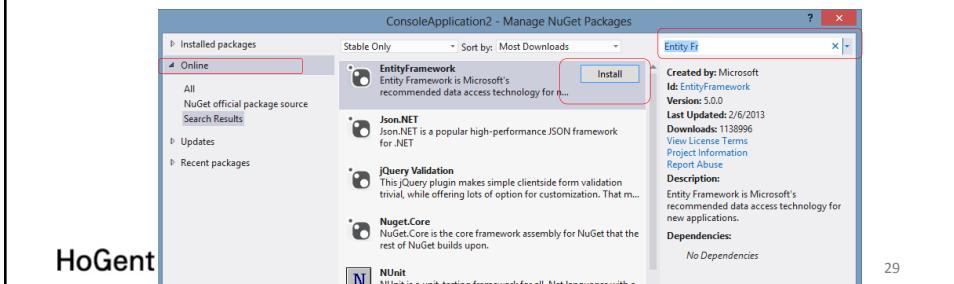
HoGent

28

## 3. EF Code First : DAL - DbContext

### ► De persistentielaaag

- Entity Framework toevoegen aan project (is reeds gebeurd)
  - **Nuget : package installer. Installeert een framework in project**
    - In Solution explorer > rechtsklik op project EFExplained > Manage Nuget Packages...
    - Onderstaand venster opent. Zoek Online naar Entity Framework
    - Eénmaal gevonden klik op Install. Voegt in de References de verwijzing naar EntityFramework en System.Data.Entity toe



## 3. EF Code First : DAL - DbContext

### ► DbContext

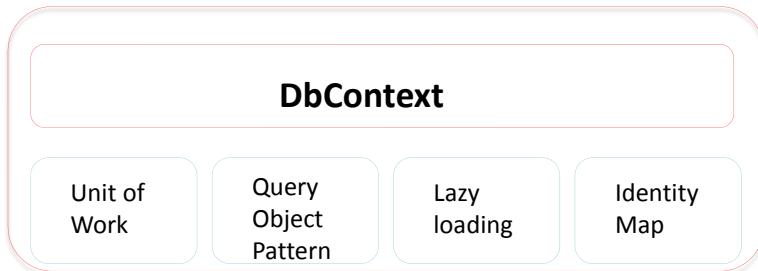
- EF bevat een DbContext klasse

Represents a combination of the Unit-Of-Work and Repository patterns and enables you to query a database and group together changes that will then be written back to the store as a unit.

- Toegangspoort tot de database
- Verzorgt CRUD : change tracking en persistentie (**SaveChanges** methode)
- Wens je EF te gebruiken dan dien je een klasse aan te maken die erft van **DbContext**
  - In de constructor bepaal je welke database wordt gebruikt
  - Voeg een **DbSet** voor elke aggregate root klasse toe

### 3. EF Code First : DAL - DbContext

#### ▶ DbContext



- Identity Map : houdt bij wat reeds opgehaald uit database
- Lazy loading : laadt enkel het nodige
- UOW : onthoudt alle wijzigingen en zal alle wijziging in 1 transactie persisteren naar de database a.d.h.v. SaveChanges methode
- Query Object pattern : OO taal voor de CRUD

HoGent

31

### 3. EF Code First : DAL - DbContext

#### ▶ DbContext

- DbContext bevat 1 belangrijke methode
  - **SaveChanges** : persisteert alle wijzigingen aan de DbSets in de db, en dit binnen 1 transactie (Unit Of Work).
    - Igv Add => INSERT statement
    - Igv wijzigingen => UPDATE instructie
    - Igv Remove=> DELETE instructie

HoGent

32

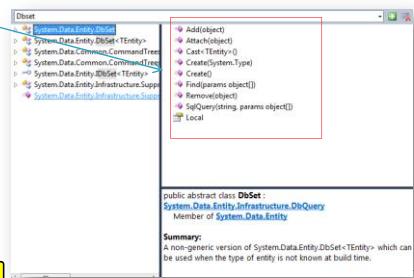
## 3. EF Code First : DAL - DbContext

### ▶ DbContext

#### ◦ **DbSet** (namespace System.Data.Entity)

- lijst van (in memory) objecten van een bepaald type die de persistentielagaar beschikking stelt, die je kan bevragen en persisteren. EF bepaalt wanneer ze uit de db worden opgehaald
- 1 per aggregate root
- CDU methodes + Linq to Entities voor het bevragen ervan

Aggregate root (DDD) = A cluster of associated objects that are treated as a unit for the purpose of data changes. External references are restricted to one member of the Aggregate, designated as the root. A set of consistency rules applies within the Aggregate's boundaries.



Opm : Waarom geen update methode?

33

## 3. EF Code First : DAL - DbContext

### ▶ DbContext

- EF verzorgt de mapping POCO (domain) -> database.
- Maakt hiervoor gebruik van **Type Discovery**. Dit betekent dat EF alle klassen zal mappen
  - waarvoor **DbSet** gedefinieerd in klasse die erft van DbContext
  - waarnaar verwezen wordt via navigation properties
  - Igv overerving is het voldoende een DbSet aan te maken voor de base class of dat er gerefereerd wordt naar de base class
  - Voorbeeld : Stel DbSet voor Brouwer gedefinieerd => dan wordt tabel Brouwers aangemaakt/gemapt, maar door de navigation properties ook tabel Biers, Gemeentes en Cursus. Door overerving bevat Cursus tabel ook Online- en OnsiteCursus. Tabel Categories wordt niet aangemaakt/gemapt tenzij je hier ook een DbSet voor voorziet.

## 3. EF Code First : DAL - DbContext

### ► Klasse die erft van DbContext

- In dit voorbeeld in Folder Models > DAL : **BierhalleContext**
  - Erft van DbContext
  - Bevat DbSet per aggregate root
    - 4 DbSets : Categories, Brouwers, Gemeenten, Cursussen
  - Maak gebruik van **Linq To Entities** om DbSet te bevragen!

```
public class BierhalleContext : DbContext
{
    public DbSet<Brouwer> Brouwers { get; set; }
    public DbSet<Gemeente> Gemeenten { get; set; }
    public DbSet<Categorie> Categories { get; set; }
    public DbSet<Cursus> Cursussen { get; set; }
}
```

HoGent

35

## 3. EF Code First : DAL - DbContext

### ► Klasse die erft van DbContext

- Waar is de database?
  - Als SQL Express instance beschikbaar dan wordt de db daar aangemaakt
  - Anders probeert code first LocalDb aan te maken (een .mdf file in de App\_Data folder)
  - De naam van de database is de fully qualified name : EFExplained.Models.DAL.BierhalleContext.
  - Je kan zelf een naam opgeven voor de database in de constructor, nu wordt de db Bierhalle aangemaakt

```
public class BierhalleContext : DbContext
{
    public BierhalleContext():base("Bierhalle"){}  
}
```

HoGent

36

## 3. EF Code First : DAL - DbContext

### ► Klasse die erft van DbContext

- Waar is de database?
  - Je bent niet beperkt tot lokale SQL Express. Definieer dan een connectiestring in de **web.config**
  - providerName = type database (SQL Server, MySql, Oracle,...)
  - Database = naam van de database
  - Security : of trusted connection (windows authenticatie) of username en wachtwoord meegeven

```
<connectionStrings>
<add name="MyProductContext"
      providerName="System.Data.SqlClient"
      connectionString="Server=.\SQLEXPRESS;Database=Bierhalle;
                        Trusted_Connection=true;" />
</connectionStrings>
```

```
public BierhalleContext() : base("MyProductContext")
```

HoGent

37

## 3. EF Code First : DAL - DbContext

### ► Voorbeeld

```
using System.Data.Entity;
using EFExplained.Models.Domain;

namespace EFExplained.Models.DAL
{
    public class BierhalleContext : DbContext
    {
        public BierhalleContext() : base("Bierhalle")
        {

        }

        public DbSet<Brouwer> Brouwers { get; set; }
        public DbSet<Gemeente> Gemeenten { get; set; }
        public DbSet<Categorie> Categories { get; set; }
        public DbSet<Cursus> Cursussen { get; set; }
    }
}
```

Tip : gebruik snippet ctor-tab voor aanmaken constructor, prop-tab-tab voor properties

In de webapplicatie :  
Brouwers moet direct kunnen worden opgevraagd.  
Bieren vragen we steeds op via Brouwer.  
Vandaar geen DbSet voor Bier

HoGent

38

## EF Code First : DAL- Db Creatie/Init

- ▶ Definieer Database Creatie en Initialisatie strategie
  - Creëer een klasse die erft van  
DropCreateDatabaseAlways<TContext> of  
 CreateDatabaseIfNotExists<TContext> of  
DropCreateDatabaseIfModelChanges<TContext>
    - Dit bepaalt wat er met de database gebeurt, respectievelijk
      - Wordt bij elke run opnieuw gecreëerd
      - Wordt enkel gecreëerd als nog niet bestaat
      - Wordt enkel opnieuw gecreëerd als model gewijzigd is (EF voegt een extra tabel \_MigrationHistory toe aan db om dergelijke zaken te detecteren)
    - Je kan de **Seed** methode overriden en zo de database van data voorzien bij creatie
    - Opm: wens je database 1malig te creëren en daarna enkel de updates toe te voegen : **Code First Migrations**. Meer op <http://msdn.microsoft.com/en-us/data/jj591621>

HoGent

39

## 3. EF Code First : DAL- Db Creatie/Init

- ▶ Definieer Database Creatie en Initialisatie

```
namespace EFExplained.Models.DAL
{
    public class BierhalleInitializer : DropCreateDatabaseAlways<BierhalleContext>
    {
        protected override void Seed(BierhalleContext context)
        {
            Gemeente bavikhove = new Gemeente {Naam = "Bavikhove", Postcode = "8531"};
            Gemeente roeselare = new Gemeente {Naam = "Roeselare", Postcode = "8800"};
            Gemeente puurs = new Gemeente {Naam = "Puurs", Postcode = "2870"};
            Gemeente leuven = new Gemeente {Naam = "Leuven", Postcode = "3000"};
            Gemeente oudenaarde = new Gemeente {Naam = "Oudenaarde", Postcode = "9700"};
            Gemeente affligem = new Gemeente {Naam = "Affligem", Postcode = "1790"};
            List<Gemeente> gemeenten =
                (new Gemeente[] {bavikhove, roeselare, puurs, leuven, oudenaarde, affligem}).ToList();
            gemeenten.ForEach(c => context.Gemeenten.Add(c)); // CRUD methoden van context
            context.SaveChanges();
        }
    }
}
```

Context doet aan ChangeTracking, houdt alle wijzigingen bij tot je SaveChanges aanroeft. Zal dan voor alle wijzigingen INSERT, UPDATE of DELETE instructie creëren=> Transactie

HoGent

40

## 4. EF Code First : Configuratie EF

### ▶ Configuratie EF en generatie DB

- In de console applicatie

```
using System.Data.Entity;

namespace EFEExplained
{
    class Program
    {
        static void Main(string[] args)
        {
            Database.SetInitializer<BierhalleContext>(new BierhalleInitializer());
            BierhalleContext context = new BierhalleContext();
            List<Brouwer> brouwers = context.Brouwers.ToList();
            Console.WriteLine("Database aangemaakt");
            Console.ReadKey();
        }
    }
}
```

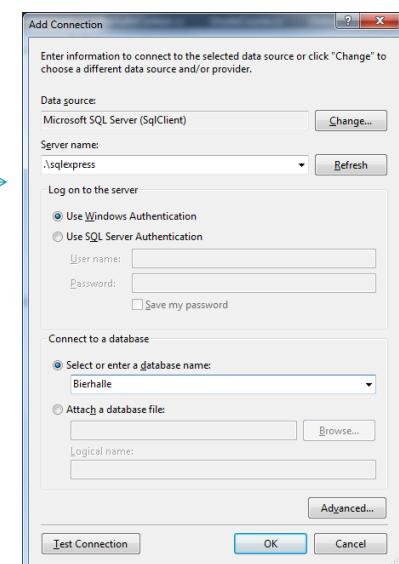
Bepaalt de strategy  
Je moet minstens 1 Ling Query uitvoeren alvorens de database gecreëerd wordt

HoGent

41

## 4. EF Code First : Configuratie EF

- ▶ Run de applicatie
- ▶ Bekijk de database
  - In VS : Server Explorer (View menu > Server Explorer). Klik 
  - In Sql Server 2012 Management Studio Express

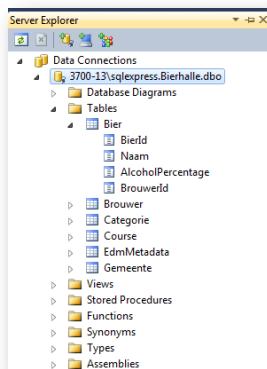


HoGent

42

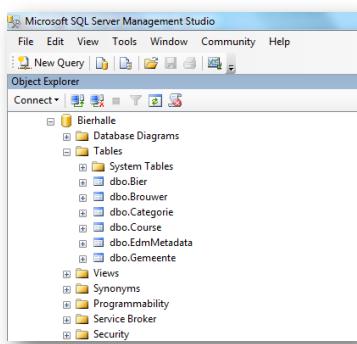
## 4. EF Code First : Configuratie EF

### ▶ Visual Studio : Server Explorer



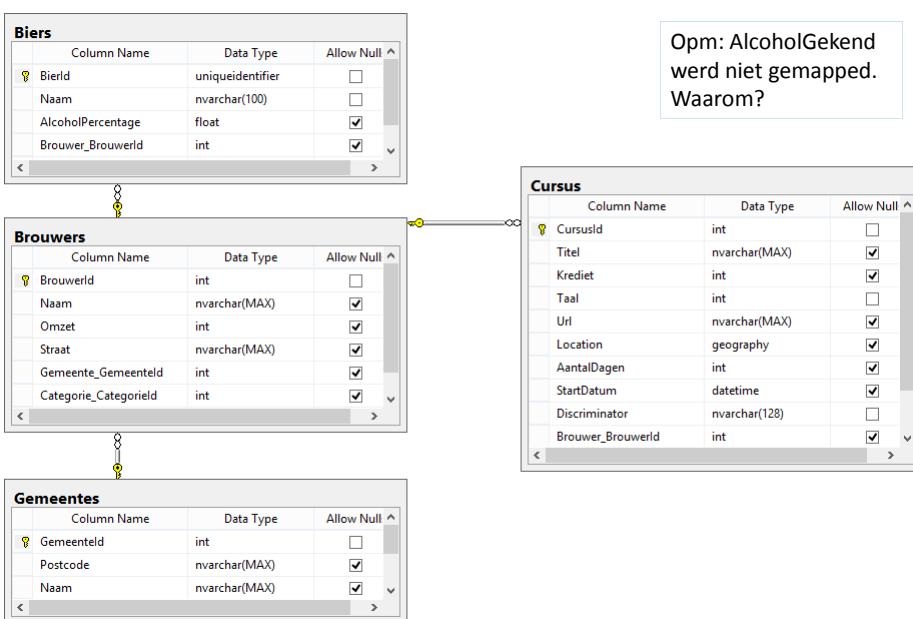
Nadeel : telkens dataconnectie verwijderen bij elke run, anders melding: cannot drop database, database in use

### ▶ SQL Server Management Studio



Ook hier opletten dat je geen design vensters meer openstaan hebt

43



HoGent

44

## 5. EF Code First : customisatie mapping

- ▶ Voldoet de (gegenererde) db niet dan kan je dit verder customizeren
  - Verwijderen van conventies
  - Data Annotaties
    - Annotaties die je toevoegt aan de domeinklasse, die verder specifiëren hoe de mapping dient te gebeuren
  - Fluent API
    - Via code geef je op hoe de mapping verder verfijnd dient te worden

HoGent

45

## 5. EF : Verwijderen conventies

- ▶ Verwijderen van conventies
  - In klasse die erft van DbContext, methode OnModelCreating
  - Voorbeeld : Tabelnamen zijn niet langer meervoud van klassenaam

```
public class BierhalleContext : DbContext {  
    ...  
    protected override void OnModelCreating(DbModelBuilder modelBuilder) {  
        // Configure Code First to ignore PluralizingTableName convention  
        // If you keep this convention then the generated tables will have pluralized names.  
        modelBuilder.Conventions.Remove<PluralizingTableNameConvention>();  
    } }
```

- PluralizingTableNameConvention uit namespace using System.Data.Entity.ModelConfiguration.Conventions;
- Plaats deze code uit commentaar, run, bekijk de db.

HoGent

46

## 5. EF : Model – Data Annotations

- ▶ Data Annotations : verder verfijnen van mapping
  - Add References : [System.ComponentModel.DataAnnotations](#) en [EntityFramework](#) (bevatten beide annotaties)
  - Add using System.ComponentModel.DataAnnotations(.Schema) en/of EntityFramework
  - Plaats annotatie boven betreffende property/klasse in domain
    - [Table] : boven een klasse, specificeert de tabelnaam
    - [Column] : boven een property. Specificeert kolomnaam, volgnr (Order) van kolom en database datatype (TypeName)
    - [Key] : sleutel. Igv samengestelde sleutel ook Column annotatie met Order attribuut [Key,Column(Order=0)]
    - [DatabaseGenerated] : boven een property om aan te geven hoe de database waarde gegenereerd wordt (Identity, Computed or None)
    - [Required] : verplicht
    - [StringLength] : vaste lengte
    - [MaxLength]/[MinLength] : maximale/minimale lengte
    - [NotMapped] : de property of klasse wordt niet opgenomen in gegenereerde db schema

HoGent

47

## 5. EF : Model – Data Annotations

### ▶ Data Annotaties

- Momenteel

```
public class Gemeente
{
    public int Gemeenteld { get; set; }
    public string Postcode { get; set; }
    public string Naam { get; set; }
}
```

EF  
generiert

Gemeentes	Column Name	Data Type	Allow Null
Gemeenteld	int		
Postcode	nvarchar(MAX)		
Naam	nvarchar(MAX)		

- We wensen dit verder te verfijnen :
  - tabelnaam Gemeente ipv Gemeentes
  - Postcode : steeds 4 posities en is sleutel (en dus not null)
  - Naam : maximale lengte 100, verplicht
- Pas de klasse Gemeente aan

HoGent

48

## 5. EF : Model – Data Annotations

### ► Data Annotations

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
```

```
namespace EFExplained.Models.Domain
{
    [Table("Gemeente")]
    public class Gemeente
    {
        //public int GemeentId { get; set; }
        [Key, StringLength(4), Column(TypeName = "nchar")]
        public string Postcode { get; set; }
        [Required, MaxLength(100)]
        public string Naam { get; set; }
    }
}
```



Gemeente		
Column Name	Data Type	Allow Nulls
Postcode	nchar(4)	<input type="checkbox"/>
Naam	nvarchar(100)	<input type="checkbox"/>

HoGent

49

## 5. EF : Model – Data Annotations

- [DatabaseGenerated]
  - boven een property om aan te geven hoe de database waarde gegenereerd wordt (Identity, Computed or None)
  - Voorbeeld : tabel Bier
    - Stel BierId een GUID (ipv int)

```
[DatabaseGenerated(DatabaseGeneratedOption.Identity)]
public Guid BierId { get; set; }
```

HoGent

50

## 5. EF : Model – Data Annotations

- [ForeignKey] : geplaatst boven een navigation property om aan te geven welke property de bijhorende foreign key is van deze associatie. Of geplaatst boven foreign key property en bevat dan de naam van de geassocieerde navigation property
- [InverseProperty] : boven een navigation property om de property aan te duiden die het andere einde van de associatie voorstelt (igv associatie met zichzelf)

```
public class Brouwer
{
    #region Properties
    public int BrouwerId { get; set; }
    private string naam;
    public string Naam { ... }
    public string Straat { get; set; }
    public int? Omzet { get; set; }
    [ForeignKey("Gemeente")]
    public string Postcode { get; set; }
    public virtual Gemeente Gemeente { get; set; }
    public virtual ICollection<Bier> Bieren { get; set; }
```

Een voorbeeld van ForeignKey : Brouwer – Gemeente. Vaak in webapplicaties wordt in de klasse niet alleen de associatie maar ook de foreign key bijgehouden (vereenvoudigt data binding) Je moet hier niet opnieuw StringLength en TypeName definiëren. Neemt EF over uit Gemeente.

51

## 5. EF : Model – Data Annotations

- [ForeignKey] - opmerking
  - In MVC applicaties voegt men vaak de foreign key toe, dit vereenvoudigt de databinding (zie later).
  - Als nullable type (bvb int?, long?, string)=> 0..1 kant in database, null allowed en cascading delete = no action.
  - Als geen nullable type (bvb int, string met annotatie required) => 1 kant in database, not null en cascading delete = delete. Cascading delete kan je aanpassen met fluent api
- Daar we DDD (Domain Driven Design) toepassen, gaan we hier geen gebruik van maken

## 5. EF : Model – Data Annotations

- [ConcurrencyCheck] : last in wins. Anders optimistic concurrency (geen locking tussen opvragen en update, wel controle van bepaalde properties mogelijk bij update. Gebruik hiervoor best een TimeStamp kolom. Indien je andere kolom wenst met deze annotatie opmaken) 
- [Timestamp] : de timestamp kolom. Een kolom met deze annotatie, wordt door EF automatisch geconfigureerd als ConcurrencyCheck en DatabaseGeneratedPattern=Computed. Datatype Byte[]
- Pas de klasse Brouwer aan

```
public class Brouwer {  
    ....  
    // When you specify the Timestamp attribute,  
    // the Entity Framework configures ConcurrencyCheck and  
    // DatabaseGeneratedPattern=Computed.  
    [Timestamp] public Byte[] Timestamp { get; set; }  
}
```

HoGent

53

## 5. EF : Model – Data Annotations

- Opmerking rond Concurrency
  - In multi-user omgeving, indien meerdere gebruikers dezelfde records wijzigen
  - Last-in-wins updating :
    - Enkel gebruiken indien kans op collision heel klein

```
"update Brouwers set naam=@naam, adres=@adres, ... where brouwernr=@brouwernr";
```

- Timestamp-based updating (optimistische locking)
  - In tabel hou je timestamp kolom bij met tijdstip laatste wijziging. Indien deze gewijzigd is sinds het opvragen van gegevens (daar haal je timestamp ook op), dan mislukt de update

```
"update Brouwers set naam=@naam, adres=@adres,TimeStamp=@TimeStamp, ...  
where brouwernr=@brouwernr andTimeStamp=@oud_TimeStamp;
```

- Ofwel eigen kolom definiëren hiervoor (ConcurrencyCheck)

HoGent

54

## 5. EF : Model – Data Annotations

- [ComplexType] : boven een klasse, als dit een complex type is = geen key, enkel groepering van properties. Moet in andere klasse naar verwezen worden. Wordt **geen** aparte tabel in de database. De props uit het complexe type worden dan in parent table opgenomen.

```
public class Cursus {  
    public Cursus() { Details = new Details();}  
    public int CursusID { get; set; }  
    public string Title { get; set; }  
    public Details Details { get; set; }  
}  
[ComplexType]  
public class Details {  
    public System.DateTime Time { get; set; }  
    public string Location { get; set; }  
    public string Days { get; set; }  
}
```

Cursus table bevat dan de kolommen CursusID, Title, Details\_Time, Details\_Location, Details\_Days

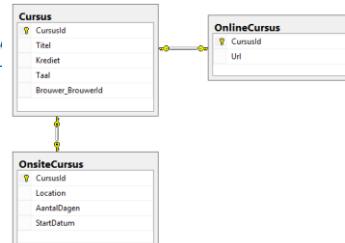
HoGent

55

## 5. EF : Model – Data

### Inheritance

- Table per Hierarchy (=default)
- Table per type
  - 1 tabel per type
  - Gebruik annotatie Table om afgeleide klassen naar een tabel te mappen



```
public class Cursus {  
    [Key]  
    public int CursusId { get; set; } // Primary key  
    public string Titel { get; set; }  
    ...  
}  
[Table("OnlineCursus")]  
public class OnlineCursus : Cursus {  
    public string URL { get; set; }  
}
```

HoGent

56

## 5. EF : Model – Data Annotations

- ▶ Oefening : Pas domeinmodel aan (voeg de nodige using statements toe) en run opnieuw
  - Brouwer
    - Naam : verplicht, max 100 char
    - Straat = max 100 lang
  - Bier
    - BierId : GUID, door database gegenereerd
    - Naam : verplicht, max 100 char
  - Categorie
    - Naam : verplicht, max 100 char
  - Cursus
    - Alle tekstvelden maximaal 100 lang
    - Titel verplicht
    - Krediet is optioneel (Hoe kan je dit aanpassen in model zonder gebruik te maken van data annotatie)

HoGent

57

## 5. EF : Model – Fluent API

- ▶ Fluent API
  - Indien conventions en annotaties onvoldoende, of indien je liever je model niet vervult met annotaties, gebruik fluent api
  - Aanpassen in **OnModelCreating** methode in klasse die erft van DbContext
    - Parameter van type DbModelBuilder: voor mappen van POCO naar db schema

```
protected override void OnModelCreating(DbModelBuilder modelBuilder)
{
    modelBuilder.Entity<Cursus>()
        .Property(c => c.Titel)
        .IsRequired().HasMaxLength(100);
```

*De property Titel in  
klasse Cursus : is  
verplicht en is max 100  
char lang in de db*

- Mapping mogelijk van
  - Properties
  - Types
  - Associates

HoGent

58

## 5. EF : Model – Fluent API

### ▶ Mapping properties

```
//Primary key  
modelBuilder.Entity<Brouwer>().HasKey(b => b.BrouwerId);  
  
//Samengestelde primary key (Stel Cursus samengestelde sleutel brouwerid, volgnr)  
modelBuilder.Entity<Cursus>() .HasKey(c => new { c.BrouwerId, c.SeqNbr });  
  
//Lengte en datatype  
modelBuilder.Entity<Brouwer>().Property(b=> Naam) .HasMaxLength(50);  
modelBuilder.Entity<Gemeente>().Property(c => c.Postcode)  
    .HasColumnType("nchar").HasMaxLength(5);  
  
//Required (NOT NULL)  
modelBuilder.Entity<Brouwer>().Property(b=> b.Naam) .IsRequired();  
  
//Identity  
modelBuilder.Entity<Gemeente>().Property(t => t.Postcode)  
.HasDatabaseGeneratedOption(DatabaseGeneratedOption.None);  
.....
```

## 5. EF : Model – Fluent API

### ▶ Mapping Properties

```
//NotMapped  
modelBuilder.Entity<Brouwer>() .Ignore(b => b.AantalBieren);  
  
//Default voor string is nvarchar. Anders wordt het ascii tekenset (varchar)  
modelBuilder.Entity<Brouwer>() .Property(b => b.Name) .IsUnicode(false);  
  
//Optimistic concurrency  
modelBuilder.Entity<Brouwer>() .Property(b => b.Timestamp) .IsConcurrencyToken();  
  
En verder  
IsMaxLength : de database provider's max  
IsFixedLength, IsVariableLength  
HasColumnName, HasColumnOrder, HasColumnType
```

## 5. EF : Model – Fluent API

### ▶ Mapping Types

```
//klasse is een complex type  
modelBuilder.ComplexType<Details>();  
  
// klasse wordt niet opgenomen in gegenereerde db schema  
modelBuilder.Ignore<OnlineCursus>();  
  
//Specifieren van de tabelnaam  
modelBuilder.Entity<Brouwer>() .ToTable("t_Brouwer");
```

- Oefening : verwijder annotaties bij Gemeente en vervang door fluent api.

HoGent

61

## 5. EF : Model – Fluent API

### ▶ Data Annotaties

```
[Table("Gemeente")]  
public class Gemeente  
{  
    //public int GemeentId { get; set; }  
    [Key, StringLength(4), Column(TypeName = "nchar")]  
    public string Postcode { get; set; }  
    [Required, MaxLength(100)]  
    public string Naam { get; set; }  
    public virtual ICollection<Brouwer> Brouwers { get; set; }  
}
```

### OF Fluent Api

```
public class Gemeente  
{  
    //public int GemeentId { get; set; }  
    public string Postcode { get; set; }  
    public string Naam { get; set; }  
}  
  
protected override void OnModelCreating(DbModelBuilder modelBuilder)  
{  
    modelBuilder.Entity<Gemeente>().ToTable("Gemeente");  
    modelBuilder.Entity<Gemeente>().HasKey(g => g.Postcode);  
    modelBuilder.Entity<Gemeente>().Property(g => g.Postcode)  
        .HasMaxLength(4).HasColumnType("nchar");  
    modelBuilder.Entity<Gemeente>().Property(g => g.Naam)  
        .IsRequired().HasMaxLength(100);  
}
```

Gemeente	Column Name	Data Type	Allow Nulls
Postcode	nchar(4)	<input type="checkbox"/>	
Naam	nvarchar(100)	<input type="checkbox"/>	

HoGent

62

## 5. EF : Model – Fluent API

### ▶ Mapping Associaties

- = Mappen van een associatie in het domein met een relatie in DB.
- Enkel de bi-directionele associaties (in beide richting navigeerbaar, zoals Category-Brouwer), waarvoor ook de foreign keys zijn opgenomen in de domeinklassen, kan EF correct mappen naar een relatie. In alle andere gevallen past EF default conventies toe. Dit kan je verder verfijnen via de Fluent API
- Default zal EF
  - Een niet bi-directionele associatie (een associatie in 1 richting navigeerbaar) mappen naar 1:n relatie in de database
  - Als foreign key ontbreekt in de domeinklasse gaat EF een foreign-key kolom toevoegen in de tabel aan de N-kant. De naam van de FK kolom is [naam Tabel] – [naam key] , allows null. Voor de relatie geldt cascading = no action

HoGent

63

## 5. EF : Model – Fluent API

### ▶ Voorbeeld

- Het domein



- Wordt door EF default gemapt in de database naar
  - 1:n relatie
  - Foreign key : Brouwer\_BrouwerId, NULL
  - Cascade delete : no action
- Het zou moeten gemapt worden naar
  - 1:n relatie
  - Foreign key : **BrouwerId**, NOT NULL
  - Cascade delete : **delete**



Aanpassen  
via Fluent  
API

HoGent

64

## 5. EF : Model – Fluent API

### ▶ Mapping Associaties

- Mappen van een associatie in het domein

A bevat de navigation property



- Naar een relatie in de database



- Relational impedance Mismatch :

- in domein is associatie vaak maar in 1 richting navigeerbaar (multipliciteit is maar aan 1 kant gekend), in database steeds in beide richtingen navigeerbaar (de multipliciteiten zijn aan beide kanten gekend)
- Veel op veel is mogelijk in domein. In database worden dit 2 1:n relaties met een intersectietabel.

HoGent

65

## 5. EF : Model – Fluent API

### ▶ Mapping Associaties

- Voorbeeld in domein

Brouwer bevat de navigation property Bieren



- Moet in de database vertaald worden naar een relatie



- ?: in het domein weten we niet of een Bier tot meerdere Brouwers kan behoren, of steeds tot 1 of mogelijks tot geen Brouwer behoort. Via de fluent api kunnen we dit definiëren. Ook de FK en cascading delete kunnen we opgeven via fluent api

HoGent

66

## 5. EF : Model – Fluent API

### ▶ Mapping Associaties

A bevat de navigation property



- Je vertrekt van de klasse met de navigation property. Bvb modelBuilder.Entity<A>()
- Met fluent API kan je tot 4 properties opgeven om een associatie te mappen naar een DB relatie.
  1. navigation property (required) →
  2. inverse navigation property (optional) →
  3. foreign key properties (optional) : naam van foreign key, of composite key
  4. WillCascadeOnDelete : true of false

HoGent

67

## 5. EF : Model – Fluent API

### ▶ Mapping Associaties

1. navigation property (required) →

Multipliciteit x betekent 1 instantie van A is gekoppeld met x instanties B

- Als x=1 => HasRequired(x=>x.NavigationPropertyName)
- Als x=0..1 => HasOptional(x=>x.NavigationPropertyName)
- Als x=n => HasMany(x=>x.CollectionNavigationPropertyName)

#### • Voorbeeld

- Brouwer bevat navigation property Gemeente
  - modelBuilder.Entity<Brouwer>().HasRequired(b=>b.Gemeente);
  - modelBuilder.Entity<Brouwer>().HasOptional(b=>b.Gemeente);
- Brouwer bevat navigation property Bieren (ICollection<Bier>)
  - modelBuilder.Entity<Brouwer>().HasMany(b=>b.Bieren);

Entiteit Brouwer bevat de navigation property Bieren

Naam van de navigation property

HoGent

68

## 5. EF : Model – Fluent API

### ▶ Mapping Associaties

#### 2. inverse navigation property (optional)

- $y \Rightarrow 1$  instantie van B is gekoppeld aan  $y$  instanties van A
- De mogelijke waarden voor  $y$  zijn afhankelijk van de waarde van  $x$
- $x = .HasRequired(...)$ 
  - $y=n \Rightarrow \text{WithMany}()$  : A krijgt een foreign key
  - $y=0..1 \Rightarrow \text{WithOptional}()$  : A krijgt foreign key
  - $y=1 \Rightarrow$  Nu kan EF niet bepalen welke tabel de primaire tabel is, dus opgeven of de Principal (=A) of de Dependant (=B) de primaire tabel is
    - $\text{WithRequiredPrincipal}() \Rightarrow A = \text{primaire tabel}, B \text{ bevat FK}$
    - $\text{WithRequiredDependant}() \Rightarrow B = \text{primaire tabel}, A \text{ bevat FK}$
- Voorbeeld : Brouwer – Gemeente (is verplicht)



HoGent

69

## 5. EF : Model – Fluent API

### ▶ Mapping Associaties

#### 2. inverse navigation property (optional)

- Voorbeeld 1 : Brouwer – Gemeente (is verplicht) EN Gemeente bevat **geen** navigation property naar Brouwer

modelBuilder.Entity<Brouwer>()

*HasRequired want 1  
Brouwer heeft juist 1  
Gemeente*

.HasRequired(b => b.Gemeente)  
.WithMany();



*WithMany want Gemeente heeft meerdere Brouwers. In WithMany kan je **geen** navigational property meegeven want **niet** gedefinieerd in domeinklasse Gemeente.*

- Voorbeeld 2 : Cursus – Brouwer (verplicht) EN Brouwer bevat **wel een** navigation property naar Cursus

modelBuilder.Entity<Cursus>()

*HasRequired want 1  
Cursus is gekoppeld  
aan juist 1 Brouwer*

.HasRequired(c => c.Brouwer)  
.WithMany(b=>b.Cursussen);

*WithMany want Brouwer kan meerdere Cursussen hebben. In WithMany moet je de navigational property meegeven want gedefinieerd in domeinklasse Brouwer.*

HoGent

70

## 5. EF : Model – Fluent API

### ▶ Mapping Associaties

#### 2. inverse navigation property (optional)

- `x = .HasOptional(...)`
  - $y=n \Rightarrow \text{WithMany}()$  : en A krijgt ook een foreign key
  - $y=1 \Rightarrow \text{WithRequired}()$  : B krijgt foreign key
  - $y=0..1 \Rightarrow$  Als de relatie optional is kan EF niet bepalen welke tabel de primaire tabel is
    - `WithOptionalPrincipal()` : A = primaire tabel, B bevat de FK
    - `WithOptionalDependant()` : B = primaire tabel, A bevat FK
- Voorbeeld : Brouwer – Gemeente (Stel dat dit optioneel is)

```
modelBuilder.Entity<Brouwer>()
    .HasOptional(b => b.Gemeente)
    .WithMany();
```

HoGent

71

## 5. EF : Model – Fluent API

### ▶ Mapping Associaties

#### 2. inverse navigation property (optional)

- `.HasMany`
  - `WithMany()`
  - `WithOptional()`
  - `WithRequired()`

- Voorbeeld : Category – Brouwer. Category bevat navigation prop Brouwers, dus we vertrekken vanuit Categorie

```
modelBuilder.Entity<Categorie>()
    .HasMany(b => b.Brouwers)
    .WithMany();
```

HasMany : 1 Categorie bevat  
meerdere Brouwers  
WithMany : 1 Brouwer meerdere  
Categoriën

- Voorbeeld : Brouwer – Bier. Brouwer bevat navigation property, dus we vertrekken vanuit Brouwer

```
modelBuilder.Entity<Brouwer>()
    .HasMany(b => b.Bieren)
    .WithRequired();
```

HasMany : 1 Brouwer heeft  
meerdere Bieren  
WithRequired : 1 Bier behoort  
(verplicht) tot juist 1 Brouwer

HoGent

72

## 5. EF : Model – Fluent API

### ▶ Mapping Associaties

#### 3. Foreign key(optional)

- Foreign key niet opgenomen in domeinklasse
  - Map(x=>x.MapKey("naam foreign key in database"));
  - Voorbeeld : brouwer bevat enkel navigation prop Gemeente  
modelBuilder.Entity<Brouwer>()
    - .HasRequired(b => b.Gemeente)
    - .WithMany()
    - .Map(m=>m.MapKey("Postcode"));
- Foreign key is gedefinieerd in domeinklasse
  - => HasForeignKey((x=>x.ForeignPropertyName))
  - Voorbeeld : brouwer bevat navigation prop Gemeente en OOK property Postcode  
modelBuilder.Entity<Brouwer>()
    - .HasRequired(b => b.Gemeente)
    - .WithMany()
    - .HasForeignKey(b=>b.Postcode);

HoGent

73

## 5. EF : Model – Fluent API

### ▶ Mapping Associaties

#### 4. Cascading

- WillCascadeOnDelete(true/false)
- Vb. Brouwer-Gemeente. Verwijderen van Gemeente zal niet lukken zolang er brouwers in gemeente gevestigd zijn

```
modelBuilder.Entity<Brouwer>()
    .HasRequired(b => b.Gemeente)
    .WithMany()
    .Map(m=>m.MapKey("Postcode"))
    .WillCascadeOnDelete(false);
```

- Als je WillCascadeOnDelete(true) definieert worden alle brouwers uit gemeente ook verwijderd als een gemeente wordt verwijderd

HoGent

74

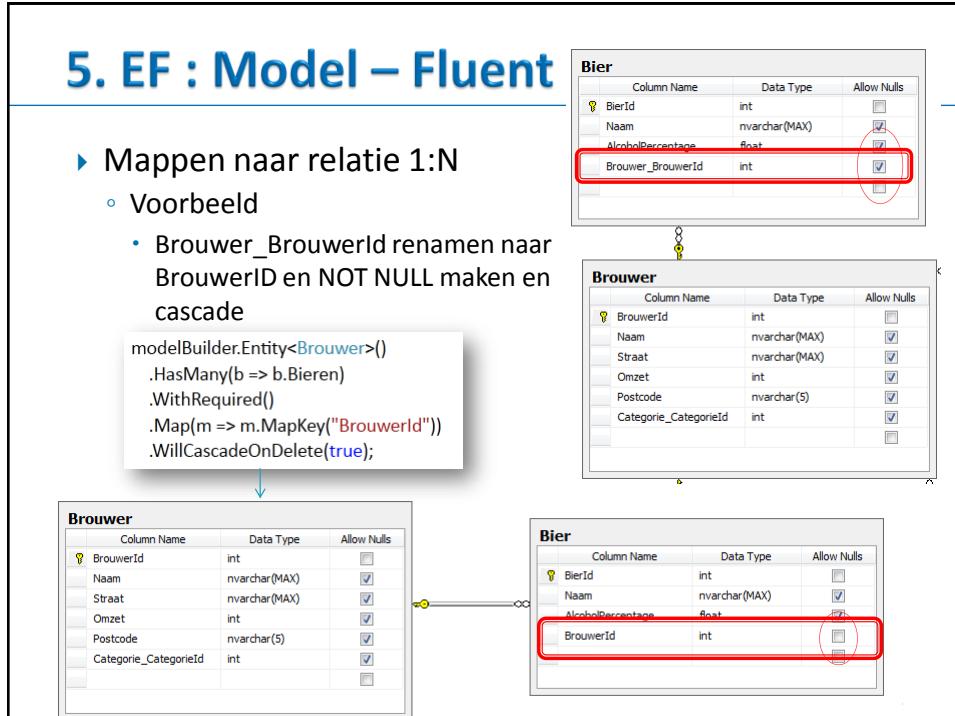
## 5. EF : Model – Fluent

### ► Mappen naar relatie 1:N

#### ◦ Voorbeeld

- Brouwer\_BrouwerId renamen naar BrouwerID en NOT NULL maken en cascade

```
modelBuilder.Entity<Brouwer>()
    .HasMany(b => b.Bieren)
    .WithRequired()
    .Map(m => m.MapKey("BrouwerId"))
    .WillCascadeOnDelete(true);
```



## 5. EF : Model – Fluent API

### ► Mappen naar relatie 1:N

- Map de relatie Brouwer – Cursus.
  - 1..n
  - FK BrouwerId, NOT NULL
  - On delete cascade
- Map de relatie Brouwer – Gemeente
  - 1..n
  - FK Postcode, NULL
  - On delete no action

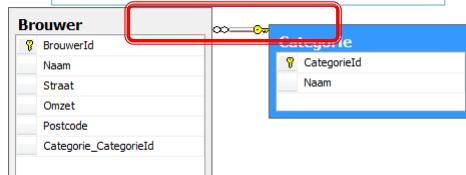
## 5. EF : Model – Fluent API

### ► Mappen naar relatie N:M

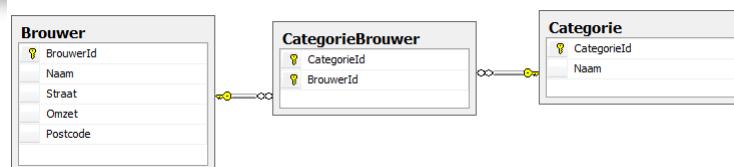
- Voorbeeld
  - Brouwer-Categorie

```
modelBuilder.Entity<Categorie>()
    .HasMany(b => b.Brouwers)
    .WithMany()
    .Map(m =>
{
    m.ToTable("CategorieBrouwer");
    m.MapLeftKey("CategorieId");
    m.MapRightKey("BrouwerId");
});
```

De default mapping door EF => 1:n



Door de fluent api, mapt EF nu naar



HoGent

//

## 5. EF : Model – Fluent API

### ► Mapping Types : Overerving

```
//Mappen van table-per-hierarchy (TPH) inheritance. De discriminator kolom krijgt
de naam "Type" en waarden "OL", "OS"
modelBuilder.Entity<Cursus>()
    .Map<OnlineCursus>(m => m.Requires("Type").HasValue("OL"))
    .Map<OnsiteCursus>(m => m.Requires("Type").HasValue("OS"));
```

Column Name	Data Type	Allow Nulls
CourseId	int	<input type="checkbox"/>
Title	nvarchar(MAX)	<input checked="" type="checkbox"/>
Credits	int	<input checked="" type="checkbox"/>
Url	nvarchar(MAX)	<input checked="" type="checkbox"/>
Location	nvarchar(MAX)	<input checked="" type="checkbox"/>
Days	nvarchar(MAX)	<input checked="" type="checkbox"/>
Time	datetime	<input checked="" type="checkbox"/>
Brouwer_BrouwerId	int	<input checked="" type="checkbox"/>
Type	nvarchar(128)	<input checked="" type="checkbox"/>

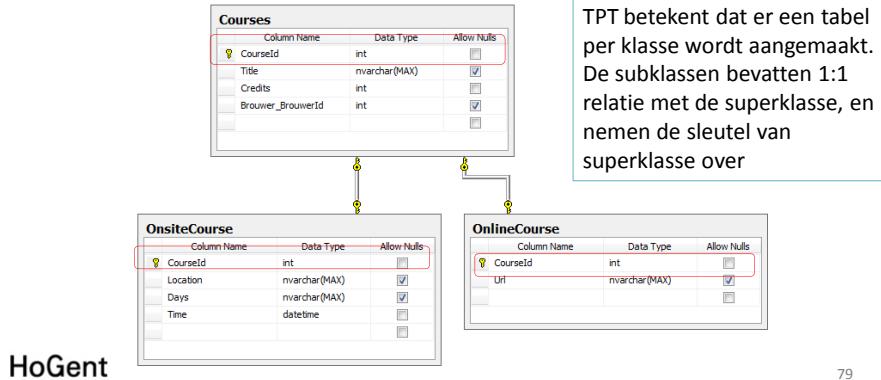
HoGent

78

## 5. EF : Model – Fluent API

### ▶ Mapping Types : Overerving

```
//Mappen van table-per-type (TPT) inheritance  
modelBuilder.Entity<Cursus>().ToTable("Cursus");  
modelBuilder.Entity<OnsiteCursus>().ToTable("OnsiteCursus");
```



79

## 5. EF : Model – Fluent API

### ▶ Mapping Types : Overerving

```
//Mappen van table-per-concrete class (TPC) inheritance. Per concrete klasse heb je een tabel. Cursus moet dan een abstracte klasse zijn.
```

```
modelBuilder.Entity<Cursus>()  
    .Property(c => c.CursusID)  
    .HasDatabaseGeneratedOption(DatabaseGeneratedOption.None);  
modelBuilder.Entity<OnsiteCursus>()  
    .Map(m =>  
    {  
        m.MapInheritedProperties();  
        m.ToTable("OnsiteCursus");  
    });  
modelBuilder.Entity<OnlineCursus>()  
    .Map(m =>  
    {  
        m  
            .MapInheritedProperties();  
        m.ToTable("OnlineCursus");  
    });
```

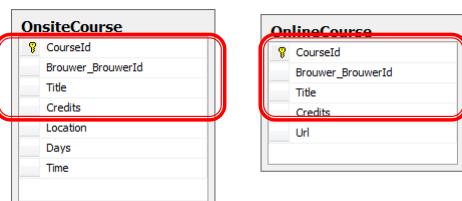
**MapInheritedProperties**  
remaps all properties that were inherited from the base class to new columns in the table for the derived class.

HoGent

80

## 5. EF : Model – Fluent API

### ▶ Mapping Types : Overerving



**MapInheritedProperties**  
remaps all properties that were inherited from the base class to new columns in the table for the derived class.

- Dit betekent dat er geen Cursus tabel meer is. Merk op dat id geen identity meer is. (Oplossing : Guid)
- In Brouwer dien je dan ook de collectie te veranderen in (anders foutmeldingen)

```
public virtual ICollection<OnlineCourse> OnlineCourses { get; set; }  
public virtual ICollection<OnsiteCourse> OnsiteCourses { get; set; }
```

HoGent

81

## 5. EF : Model – Fluent API

### ▶ Mapping Types : overige

```
//Entity splitting : mappen van 1 entiteit naar meerdere tabellen  
modelBuilder.Entity<Department>()  
    .Map(m =>  
    {  
        m.Properties(t => new { t.DepartmentID, t.Name });  
        m.ToTable("Department");  
    })  
    .Map(m =>  
    {  
        m.Properties(t => new { t.DepartmentID, t.Administrator, t.StartDate,  
            t.Budget });  
        m.ToTable("DepartmentDetails");  
    });
```

**Properties**  
to map a subset of properties to a specific table

HoGent

82

## 5. EF : Model – Fluent API

- ▶ Mapping Types : overige

```
//Mappen van meerdere entiteiten naar 1 tabel
modelBuilder.Entity<OfficeAssignment>()
    .HasKey(t => t.InstructorID);
modelBuilder.Entity<Instructor>()
    .HasRequired(t => t.OfficeAssignment)
    .WithRequiredPrincipal(t => t.Instructor);
modelBuilder.Entity<Instructor>()
    .ToTable("Instructor");
modelBuilder.Entity<OfficeAssignment>().
    ToTable("Instructor");
```

HoGent

83

## 5. EF : Model – Fluent API

- ▶ Mapping kan ook in aparte klasse (1 per entiteit)
  - Maak folder Mapper aan in DAL folder. Creëer de klasse

```
using System.Data.Entity.ModelConfiguration;
using EFExplained.Models.Domain;

namespace EFExplained.Models.DAL.Mapping
{
    public class GemeenteMapper : EntityTypeConfiguration<Gemeente>
    {
        public GemeenteMapper()
        {
            Property(g => g.Naam).IsRequired().HasMaxLength(100);
            Property(g => g.Postcode).HasMaxLength(4).IsFixedLength();
            HasKey(g => g.Postcode);
        }
    }
}
```

HoGent

84

## 5. EF : Model – Fluent API

- ▶ Mapping kan ook in aparte klasse (1 per entiteit)
  - Pas dan BierhalleContext aan

```
protected override void OnModelCreating(DbModelBuilder modelBuilder)
{
    //modelBuilder.Entity<Gemeente>().Property(g => g.Naam).IsRequired().HasMaxLength(100);
    //modelBuilder.Entity<Gemeente>().Property(g => g.Postcode).HasMaxLength(4).IsFixedLength();
    //modelBuilder.Entity<Gemeente>().HasKey(g => g.Postcode);
    modelBuilder.Configurations.Add(new GemeenteMapper());
}
```

HoGent

85

## 5. EF : Model – Fluent API

- ▶ Toevoegen van indexen,....
  - Kan in de seeding methode

```
protected override void Seed(BierhalleContext context)
{
    context.Database.ExecuteSqlCommand("CREATE INDEX Brouwers_Postcode ON Brouwer (postcode);
```

HoGent

86

## 5. EF Code First

► Tip :

- Maak een klassen aan
- Run dan de applicatie, bekijk database
- Per klasse
  - Voeg mapping toe
  - Run, bekijk database
- Voeg de mapping klasse per klasse toe, en run telkens na het mappen van een klasse. Fouten zijn anders heel moeilijk te vinden!!

HoGent

87

## 5. EF Code First

**EF kan adhv de POCO's, Data Annotations en Fluent API de database genereren/of mappen.**

HoGent

88

## 6. EF : Linq to Entities

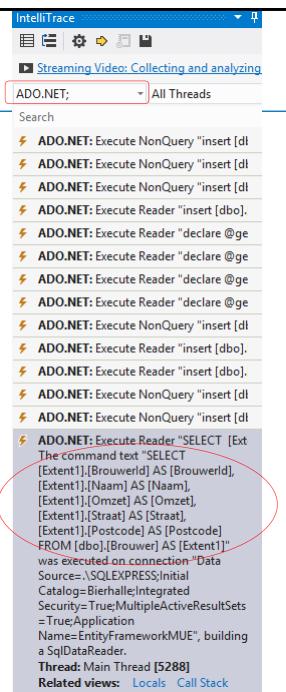
- ▶ LINQ to Entities
  - Bevragen van de database
  - Communicatie met de DbContext
    - Aggregates, projecties, filteren, sorteren, ... mogelijk
  - Strongly typed queries!
  - Retourneert : Entiteiten
- ▶ Achter de schermen genereert EF de overeenkomstige queries.
  - Gebruik een profiler om de gegenereerde queries te bekijken en zo de performantie in het oog te houden
    - SQL Profiler (geen onderdeel van SQL server Express Editie)
    - Entity Framework Profiler  
<http://www.hibernatingrhinos.com/products/EFProf> (Trial voor 30 dagen (zeer goed))
    - IntelliTrace window

HoGent

89

## 6. EF : Linq to Entities

- ▶ Gebruik van IntelliTrace window
  - Debug > IntelliTrace
  - Selecteer als Categorie ADO.NET
  - Zo kan je de gegenereerde queries bekijken



HoGent

90

## 6. EF : Linq to Entities

### ► Gebruik van EF Profiler

- Start Profiler (EFProf.exe)
- In de applicatie
  - Voeg referentie toe naar

```
References
  EntityFramework
  HibernatingRhinos.Profiler.Appender
```

- Voeg lijn code toe in Program.cs

```
static void Main(string[] args)
{
    HibernatingRhinos.Profiler.Appender.EntityFramework.EntityFrameworkProfiler.Initialize();
}
```

- Zorg dat db niet opnieuw gecreëerd wordt

```
public class BierhalleInitializer : CreateDatabaseIfNotExists<BierhalleContext>
```

- Run de applicatie

HoGent

91

## 6. EF : Linq to Entities

### ► Zie program.cs,

- plaats ReadDbContext(context); uit commentaar

### ► Opvragen alle brouwers (alle kolommen)

```
Console.WriteLine("\n---Opvragen alle brouwers---");
brouwers = context.Brouwers.OrderBy(b => b.Naam);
foreach (Brouwer br in brouwers)
    Console.WriteLine(br.Naam);
```

```
SELECT
[Extent1].[BrouwerId] AS [BrouwerId],
[Extent1].[Naam] AS [Naam],
[Extent1].[Straat] AS [Straat],
[Extent1].[Omzet] AS [Omzet],
[Extent1].[Postcode] AS [Postcode]
FROM [dbo].[Brouwer] AS [Extent1]
ORDER BY [Extent1].[Naam] ASC
```

### ► Opvragen alle brouwers (specifieke kolommen)

```
Console.WriteLine("\n---Opvragen alle brouwers/specifieke kolommen---");
var brouwers2 = context.Brouwers.
    Select(b => new {Naam = b.Naam, Omzet = b.Omzet});
foreach (var br in brouwers2)
    Console.WriteLine(br.Naam);
```

```
SELECT
1 AS [C1],
[Extent1].[Naam] AS [Naam],
[Extent1].[Omzet] AS [Omzet]
FROM [dbo].[Brouwer] AS [Extent1]
```

HoGent

92

## 6. EF : Linq to Entities

### ► Opvragen 1 brouwer (vb met id 1)

```
Console.WriteLine("\n---Opvragen brouwer met id 1---");
brouwer = context.Brouwers.SingleOrDefault(b => b.BrouwerId == 1);
Console.WriteLine("Brouwer met id 1 : " + brouwer.Naam);
```

```
return (from brouwer in brouwers
        where brouwer.BrouwerNr == brouwerId
        select brouwer).SingleOrDefault();
```

OF

```
SELECT TOP (2)
[Extent1].[BrouwerId] AS [BrouwerId],
[Extent1].[Naam] AS [Naam],
[Extent1].[Straat] AS [Straat],
[Extent1].[Omzet] AS [Omzet],
[Extent1].[Postcode] AS [Postcode]
FROM [dbo].[Brouwer] AS [Extent1]
WHERE 1 = [Extent1].[BrouwerId]
```

- `SingleOrDefault()` : retourneert null als brouwer niet bestaat. Throwt exception als er meer dan 1 brouwer aan criterium voldoet
- `Single()` : Exception als brouwer niet bestaat
- `FirstOrDefault()` : neemt eerste brouwer die aan criterium voldoet. Null als brouwer niet bestaat. Geeft geen fout als er meerdere brouwers aan criterium voldoen. (is performanter dan `SingleOrDefault()`)
- `Find(1)` : zoekt op ID

HoGent

93

## 6. EF : Linq to Entities

### ► Filteren

#### ◦ 1 klasse

```
Console.WriteLine("\n---Opvragen alle brouwer waarvan de naam met b begint---");
brouwers = context.Brouwers.Where(b => b.Naam.StartsWith("b")).OrderBy(b => b.Naam).ToList();
foreach (Brouwer br in brouwers)
    Console.WriteLine(br.Naam);
```

```
SELECT
[Extent1].[BrouwerId] AS [BrouwerId],
[Extent1].[Naam] AS [Naam],
[Extent1].[Straat] AS [Straat],
[Extent1].[Omzet] AS [Omzet],
[Extent1].[Postcode] AS [Postcode]
FROM [dbo].[Brouwer] AS [Extent1]
WHERE [Extent1].[Naam] LIKE N'b%'
ORDER BY [Extent1].[Naam] ASC
```

#### ◦ Via navigational props => JOIN!!

```
Console.WriteLine("\n---Opvragen alle brouwer met gemeente Roeselare---");
brouwers = context.Brouwers.Where(b => b.Gemeente.Naam == "Roeselare").OrderBy(b => b.Naam);
foreach (Brouwer br in brouwers)
    Console.WriteLine(br.Naam);
```

```
SELECT
[Extent1].[BrouwerId] AS [BrouwerId],
[Extent1].[Naam] AS [Naam],
[Extent1].[Straat] AS [Straat],
[Extent1].[Omzet] AS [Omzet],
[Extent1].[Postcode] AS [Postcode]
FROM [dbo].[Brouwer] AS [Extent1]
INNER JOIN [dbo].[Gemeente] AS [Extent2] ON [Extent1].[Postcode] = [Extent2].[Postcode]
WHERE [Roeselare] = [Extent2].[Naam]
ORDER BY [Extent1].[Naam] ASC
```

Distinct() : geeft enkel de verschillende terug

HoGent

94

## 6. EF : Linq to Entities

- Filteren op \* associatie : ANY (subQuery)

- Brouwers met een bier waarvan de naam met een B begint.  
Gebruiken associatie \*

```
Console.WriteLine("\n---Oprragen alle Brouwers met een bier waarvan de naam met een B begint. ---");
brouwers =
    from br in context.Brouwers
    where br.Bieren.Any(b => b.Naam.ToUpper().StartsWith("B"))
    select br;
foreach (Brouwer br in brouwers)
    Console.WriteLine(br.Naam);
```

```
SELECT
[Extent1].[BrouwerId] AS [BrouwerId],
[Extent1].[Naam] AS [Naam],
[Extent1].[Straat] AS [Straat],
[Extent1].[Omzet] AS [Omzet],
[Extent1].[Postcode] AS [Postcode]
FROM [dbo].[Brouwer] AS [Extent1]
WHERE EXISTS (SELECT
    1 AS [C1]
    FROM [dbo].[Bier] AS [Extent2]
    WHERE ([Extent1].[BrouwerId] = [Extent2].[BrouwerId]) AND (UPPER([Extent2].[Naam]) LIKE N'B%')
)
```

Heel wat methodes :

- Contains
- Count
- Except
- Intersect
- ...

HoGent

95

## 6. EF : Linq to Entities

### ► EF gebruikt **Lazy loading**

- = Laden van gerelateerde data, enkel wanneer je er expliciet om vraagt

```
//lazy loading
Console.WriteLine("\n---Lazy loading---");
brouwers = context.Brouwers.OrderBy(b => b.Naam).ToList();
foreach (Brouwer br in brouwers)
    Console.WriteLine(br.Naam + " : " + ((br.Gemeente!=null) ? br.Gemeente.Naam : ""));
```

- Daar je nu informatie nodig hebt over gemeente moet EF terug naar de database

HoGent

Pag. 96

## 6. EF : Linq to Entities

- ▶ EF gebruikt Lazy loading
  - Query bekijken in EF profiler

```
SELECT ... FROM [dbo].[Brouwer] AS [Extent1]
SELECT ... FROM [dbo].[Gemeente] AS [Extent1] WHERE [Extent1].[Postcode] = '8531'
SELECT ... FROM [dbo].[Gemeente] AS [Extent1] WHERE [Extent1].[Postcode] = '2870'
SELECT ... FROM [dbo].[Gemeente] AS [Extent1] WHERE [Extent1].[Postcode] = '3000'
SELECT ... FROM [dbo].[Gemeente] AS [Extent1] WHERE [Extent1].[Postcode] = '9700'
```

	7	4 ms / 5 ms
1	39 ms / 92 ms	
1	11 ms / 14 ms	
1	7 ms / 8 ms	
1	4 ms / 5 ms	

```
SELECT [Extent1].[BrouwerId] AS [BrouwerId],
       [Extent1].[Naam]      AS [Naam],
       [Extent1].[Straat]    AS [Straat],
       [Extent1].[Omzet]     AS [Omzet],
       [Extent1].[Postcode]  AS [Postcode]
  FROM [dbo].[Brouwer] AS [Extent1]
 ORDER BY [Extent1].[Naam] ASC
```

```
SELECT [Extent1].[Postcode] AS [Postcode],
       [Extent1].[Naam]      AS [Naam]
  FROM [dbo].[Gemeente] AS [Extent1]
 WHERE [Extent1].[Postcode] = '8531' /* @EntityKeyValue1 */
```

Opgelet : lazy loading : extra roundtrip  
naar database. 100 Brouwers betekent 101 queries!

HoGent

97

## 6. EF : Linq to Entities

- ▶ EF gebruikt Lazy loading

- Oplossing Include

1 query

```
Console.WriteLine("\n---Include---");
brouwers = context.Brouwers.Include(b=>b.Gemeente).OrderBy(b => b.Naam).ToList();
foreach (Brouwer br in brouwers)
    Console.WriteLine(br.Naam + " :" + ((br.Gemeente != null) ? br.Gemeente.Naam : ""));
```

```
SELECT ... FROM [dbo].[Brouwer] AS [Extent1] left outer join [d...
```

7

53 ms / 55 ms

```
Details    Alerts    Stack Trace
1 | SELECT [Extent1].[BrouwerId] AS [BrouwerId],
2 |       [Extent1].[Naam]      AS [Naam],
3 |       [Extent1].[Straat]    AS [Straat],
4 |       [Extent1].[Omzet]     AS [Omzet],
5 |       [Extent1].[Postcode]  AS [Postcode],
6 |       [Extent2].[Postcode]  AS [Postcode1],
7 |       [Extent2].[Naam]      AS [Naam1]
8 |   FROM [dbo].[Brouwer] AS [Extent1]
9 |   LEFT OUTER JOIN [dbo].[Gemeente] AS [Extent2]
10 |      ON [Extent1].[Postcode] = [Extent2].[Postcode]
11 | ORDER BY [Extent1].[Naam] ASC
```

Para

HoGent

98

## 6. EF : Linq to Entities

- Hou de performantie in het oog!
  - Zonder Include : 5 queries naar de database
  - Met Include : 1 query
  - N+1 Anti-pattern : meer op

<http://efprof.com/Learn/Alerts>SelectNPlusOne>

```
SELECT ... FROM [dbo].[Brouwer] AS [Extent1]
SELECT ... FROM [dbo].[Gemeente] AS [Extent1] WHERE [Extent1].[Postcode] = '8531'
SELECT ... FROM [dbo].[Gemeente] AS [Extent1] WHERE [Extent1].[Postcode] = '2870'
SELECT ... FROM [dbo].[Gemeente] AS [Extent1] WHERE [Extent1].[Postcode] = '3000'
SELECT ... FROM [dbo].[Gemeente] AS [Extent1] WHERE [Extent1].[Postcode] = '9700'
```

```
SELECT ... FROM [dbo].[Brouwer] AS [Extent1] left outer join [d...
```

Details   Alerts   Stack Trace

```
1  SELECT [Extent1].[BrouwerId] AS [BrouwerId],
2      [Extent1].[Naam] AS [Naam],
3      [Extent1].[Straat] AS [Straat],
4      [Extent1].[Omzet] AS [Omzet],
5      [Extent1].[Postcode] AS [Postcode],
6      [Extent2].[Postcode] AS [Postcode1],
7      [Extent2].[Naam] AS [Naam1]
8  FROM [dbo].[Brouwer] AS [Extent1]
9      LEFT OUTER JOIN [dbo].[Gemeente] AS [Extent2]
10     ON [Extent1].[Postcode] = [Extent2].[Postcode]
11 ORDER BY [Extent1].[Naam] ASC
```

HoGent

99

### Meerdere includes : Categorie met brouwers en gemeente

```
Console.WriteLine("\n--Include--");
cat = context.Categoriees
    .Include(b => b.Brouwers.Select(g => g.Gemeente))
    .ToList();
foreach (Categorie c in cat)
{
    Console.WriteLine(c.Naam);
    foreach (Brouwer b in c.Brouwers)
        Console.WriteLine(b.Naam + " " + (b.Gemeente != null ? b.Gemeente.Naam : ""));
}
```

```
SELECT [Project1].[CategorieId] AS [CategorieId],
[Project1].[Naam] AS [Naam],
[Project1].[C1] AS [C1],
[Project1].[BrouwerId] AS [BrouwerId],
[Project1].[Naam1] AS [Naam1],
[Project1].[Straat] AS [Straat],
[Project1].[Omzet] AS [Omzet],
[Project1].[Postcode] AS [Postcode],
[Project1].[Postcode1] AS [Postcode1],
[Project1].[Naam2] AS [Naam2],
[Project1].[Naam3] AS [Naam3]
FROM (
    SELECT [Extent1].[CategorieId] AS [CategorieId],
           [Extent1].[Naam] AS [Naam],
           [Join2].[BrouwerId] AS [BrouwerId],
           [Join2].[Naam1] AS [Naam1],
           [Join2].[Straat] AS [Straat],
           [Join2].[Omzet] AS [Omzet],
           [Join2].[Postcode] AS [Postcode],
           [Join2].[Postcode1] AS [Postcode1],
           [Join2].[Naam2] AS [Naam2],
           CASE
               WHEN ((Join2).[CategorieId] IS NULL) THEN CAST(NULL AS int)
               ELSE 1
           END AS [C1]
    FROM [dbo].[Categorie] AS [Extent1]
    LEFT OUTER JOIN (SELECT [Extent2].[CategorieId] AS [CategorieId],
                           [Extent2].[BrouwerId] AS [BrouwerId],
                           [Extent2].[Naam] AS [Naam],
                           [Extent2].[Straat] AS [Straat],
                           [Extent2].[Omzet] AS [Omzet],
                           [Extent2].[Postcode] AS [Postcode1],
                           [Extent3].[Postcode] AS [Postcode2],
                           [Extent3].[Naam] AS [Naam3]
                      FROM [dbo].[CategorieBrouwer] AS [Extent2]
                      INNER JOIN [dbo].[Brouwer] AS [Extent3]
                         ON [Extent3].[BrouwerId] = [Extent2].[BrouwerId]
                      LEFT OUTER JOIN [dbo].[Gemeente] AS [Extent4]
                         ON [Extent3].[Postcode] = [Extent4].[Postcode]) AS [Join2]
        ON [Extent1].[CategorieId] = [Join2].[CategorieId]
)
```

Je kan ook meerdere  
Includes in 1 instructie  
gebruiken

100

## 6. EF : Linq to Entities

### ▶ Overerving

- TypeOf

```
Console.WriteLine("\n---Overerving---");
brouwer = context.Brouwers.SingleOrDefault(b => b.BrouwerId == 1);
var q = brouwer.Courses.OfType<OnlineCourse>().ToList();
```

```
SELECT ... FROM [dbo].[Course] AS [Extent1] WHERE ([Extent1].[BrouwerId] = 1) and ...
Details Stack Trace
1 SELECT [Extent1].[CourseId] AS [CourseId],
2     [Extent1].[Discriminator] AS [Discriminator],
3     [Extent1].[Title] AS [Title],
4     [Extent1].[Credits] AS [Credits],
5     [Extent1].[Url] AS [Url],
6     [Extent1].[Location] AS [Location],
7     [Extent1].[Days] AS [Days],
8     [Extent1].[Time] AS [Time],
9     [Extent1].[BrouwerId] AS [BrouwerId]
10    FROM [dbo].[Course] AS [Extent1]
11   WHERE ([Extent1].[BrouwerId] = 1 /* @EntityKeyValue1 */)
12      AND ([Extent1].[Discriminator] IN ('OnlineCourse', 'OnsiteCourse'))
```

HoGent

101

## 6. EF : Linq to Entities

### ▶ Oefening (zie program.cs, DoExercisesDbContext())

1. Alle brouwers waarvan omzet gekend is en omzet >= 500000
2. Alle brouwers met meer dan 4 bieren. Toon naam brouwer
3. Alle brouwers uit leuven. Toon naam en aantal bieren

HoGent

102

## 6. EF : Linq to Entities

- Oefening : Let op voor de performantie bij query 3

```
Console.WriteLine("\n---Alle brouwers uit Leuven, naam en aantal bieren---");
brouwers = from b in context.Brouwers
           where b.Gemeente.Naam == "Leuven"
           select b;
foreach (Brouwer br in brouwers)
    Console.WriteLine(br.Naam + " " + br.Bieren.Count());
```

1 brouwer =>  
2 queries

```
SELECT ... FROM [dbo].[Brouwer] AS [Extent1] inner join [dbo].[...] WHERE N'Leuven' = [Extent2].[Naam]
SELECT ... FROM [dbo].[Bier] AS [Extent1] WHERE [Extent1].[BrouwerId] = 5
```

Aantal rijen

1
5

```
brouwers = from b in context.Brouwers.Include(b=>b.Bieren)
           where b.Gemeente.Naam == "Leuven"
           select b;
foreach (Brouwer br in brouwers)
    Console.WriteLine(br.Naam + " " + br.Bieren.Count());
```

1 brouwer =>  
1 query, maar  
1 rij per bier in  
het resultaat

```
SELECT ... FROM (Select [Extent1].[BrouwerId] AS [BrouwerId], [...
```

5

HoGent

103

## 6. EF : Linq to Entities

- Oefening : Let op voor de performantie bij query 3

```
var brouwers3 = from b in context.Brouwers.Include(b => b.Bieren)
                 where b.Gemeente.Naam == "Leuven"
                 select new { Naam = b.Naam, AantalBieren = b.Bieren.Count() };
foreach (var br in brouwers3)
    Console.WriteLine(br.Naam + " " + br.AantalBieren.ToString());
```

1 query, 1 rij per  
brouwer

```
SELECT ... FROM [dbo].[Brouwer] AS [Extent1] inner join [dbo].[...] WHERE N'Leuven' = [Extent2].[Naam]
```

1

```
Details Stack Trace
1 SELECT [Extent1].[BrouwerId] AS [BrouwerId],
2     [Extent1].[Naam] AS [Naam],
3     (SELECT COUNT(1) AS [A1]
4      FROM [dbo].[Bier] AS [Extent3]
5      WHERE [Extent1].[BrouwerId] = [Extent3].[BrouwerId]) AS [C1]
6 FROM [dbo].[Brouwer] AS [Extent1]
7     INNER JOIN [dbo].[Gemeente] AS [Extent2]
8     ON [Extent1].[Postcode] = [Extent2].[Postcode]
9 WHERE N'Leuven' = [Extent2].[Naam]
```

HoGent

104

## 7. EF : DbContext en updates

- ▶ DbContext verzorgt de object Tracking
  - Bij opvragen van objecten (na een select query) worden deze in de Cache (**Identity Map**) geplaatst.
  - De DbContext houdt in de cache voor elke entiteit 2 objecten bij
    - Het object (entiteit) zelf
    - De ObjectStateEntry : nodig voor change tracking
      - Originale waarde van object
      - EntityState : unchanged, added, updated, deleted
  - SaveChanges : persisteert wijzigingen naar database
    - Alle entiteiten in cache worden overlopen. Als EntityState verschilt van unchanged wordt insert, update of delete instructie gegenereerd. Enkel de gewijzigde properties worden gesascadeerd.
    - = **Unit of Work** (alle wijzigingen tegelijk) en **TransactieBeheer** (alle wijzigingen lukken of worden gerollbackt)

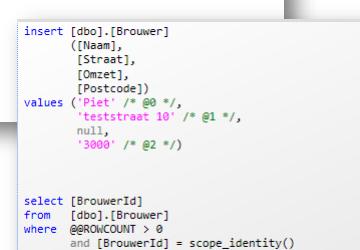
HoGent

105

## 7. EF : DbContext en updates

- ▶ Toevoegen

```
Console.WriteLine("--Add--");
Gemeente leuven = context.Gemeenten.SingleOrDefault(g => g.Naam == "Leuven");
Brouwer piet = new Brouwer("Piet");
piet.Straat = "teststraat 10";
piet.Gemeente = leuven;
context.Brouwers.Add(piet);
context.SaveChanges();
```



```
insert [dbo].[Brouwer]
([Naam],
[Straat],
[Omzet],
[Postcode])
values ('Piet' /* @0 */,
'teststraat 10' /* @1 */,
null,
'3000' /* @2 */)

select [BrouwerId]
from [dbo].[Brouwer]
where @@ROWCOUNT > 0
and [BrouwerId] = scope_identity()
```

```
begin transaction with isolation level: ReadCommitted
[INSERT INTO [dbo].[Brouwer] ...
SELECT ... FROM [dbo].[Brouwer] WHERE @@ROWCOUNT > 0 and [BrouwerId] = scope_identity()]
commit transaction
```

HoGent

106

## 7. EF : DbContext en updates

### ▶ Toevoegen (aan collectie)

```
Console.WriteLine("\n---Add in ICollection---");
Course course = new OnlineCourse() {Title = "test", Url = "http://test@hogent.be"};
brouwer = context.Brouwers.SingleOrDefault(b => b.BrouwerId == 1);
brouwer.Courses.Add(course);
context.SaveChanges();
```

```
SELECT ... FROM [dbo].[Brouwer] AS [Extent1] WHERE 1 = [Extent1].[BrouwerId]
SELECT ... FROM [dbo].[Course] AS [Extent1] WHERE ([Extent1].[BrouwerId] = 1) and ...
begin transaction with isolation level: ReadCommitted
INSERT INTO [dbo].[Course] ...
SELECT ... FROM [dbo].[Course] WHERE @@ROWCOUNT > 0 and [CourseId] = scope_identity()
commit transaction
```

Alle cursussen van  
brouwer1 worden  
eerst opgehaald

```
Details Stack Trace
1| SELECT [Extent1].[CourseId] AS [CourseId],
2|   [Extent1].[Discriminator] AS [Discriminator],
3|   [Extent1].[Title] AS [Title],
4|   [Extent1].[Credits] AS [Credits],
5|   [Extent1].[Url] AS [Url],
6|   [Extent1].[Location] AS [Location],
7|   [Extent1].[Days] AS [Days],
8|   [Extent1].[Time] AS [Time],
9|   [Extent1].[BrouwerId] AS [BrouwerId]
10|  FROM [dbo].[Course] AS [Extent1]
11| WHERE ((Extent1).[BrouwerId] = 1 /* @EntityKeyValue1 */)
AND ((Extent1).[Discriminator] IN ('OnlineCourse', 'OnsiteCourse'))
```

107

## 7. EF : DbContext en updates

### ▶ Wijzigen

```
Console.WriteLine("\n---Wijzigen---");
Gemeente roeselare = context.Gemeenten.SingleOrDefault(g => g.Postcode == "8800");
piet = context.Brouwers.FirstOrDefault(b => b.Naam == "Piet");
piet.Straat = "smidsestraat 10";
piet.Gemeente = roeselare;
context.SaveChanges();
```

```
SELECT ... FROM [dbo].[Gemeente] AS [Extent1] WHERE N'8800' = [Extent1].[Postcode]
SELECT ... FROM [dbo].[Brouwer] AS [Extent1] WHERE N'Piet' = [Extent1].[Naam]
begin transaction with isolation level: ReadCommitted
UPDATE [dbo].[Brouwer] ... WHERE ([BrouwerId] = 8)
commit transaction
```

```
Details Stack Trace
1| update [dbo].[Brouwer]
2| set [Straat] = 'smidsestraat 10' /* @0 */,
3|   [Postcode] = '8800' /* @1 */
4| where ([BrouwerId] = 8 /* @2 */)
```

HoGent

108

## 7. EF : DbContext en updates

### ▶ Verwijderen

Cascading deletes worden uitgevoerd!

```
Console.WriteLine("\n---Verwijderen---");
Console.WriteLine("aantal brouwers voor delete : " + context.Brouwers.Count());
piet = context.Brouwers.FirstOrDefault(b => b.Naam == "Piet");
context.Brouwers.Remove(piet);
context.SaveChanges();
Console.WriteLine("aantal brouwers na delete : " + context.Brouwers.Count());
```

```
SELECT ... FROM (Select COUNT(I) AS [A1] From [dbo].[Brouwer] A...
SELECT ... FROM [dbo].[Brouwer] AS [Extent1] WHERE N'Piet' = [Extent1].[Naam]
begin transaction with isolation level: ReadCommitted
DELETE FROM [dbo].[Brouwer] WHERE ([BrouwerId] = 8)
commit transaction
SELECT ... FROM (Select COUNT(I) AS [A1] From [dbo].[Brouwer] A...
```

Details Stack Trace

```
1| delete [dbo].[Brouwer]
2| where ([BrouwerId] = 8 /* @0 */)
```

HoGe

109

## 7. EF : DbContext en updates

### ▶ Transactie : verschillende updates samen

```
Console.WriteLine("\n---Transactie---");
roeselare = context.Gemeente.SingleOrDefault(g => g.Postcode == "8800");
Brouwer brouwer1 = context.Brouwers.SingleOrDefault(b => b.BrouwerId==1);
brouwer1.Straat = "smidsestraat 10";
Brouwer brouwer2 = context.Brouwers.SingleOrDefault(b => b.BrouwerId == 2);
brouwer2.Straat = "teststraat 10";
context.SaveChanges();
```

```
begin transaction with isolation level: ReadCommitted
UPDATE [dbo].[Brouwer] ... WHERE ([BrouwerId] = 1)
UPDATE [dbo].[Brouwer] ... WHERE ([BrouwerId] = 2)
commit transaction
```

Details Stack Trace

```
1| update [dbo].[Brouwer]
2| set [Straat] = 'smidsestraat 10' /* @0 */
3| where ([BrouwerId] = 1 /* @1 */)
```

HoGent

110

## 7. EF : DbContext en updates

- ▶ Object Context verzorgt ook de associaties

```
Console.WriteLine("\n---Onderhouden van associaties--");
Console.WriteLine("Aantal gemeenten voor :" + context.Gemeenten.Count());
brouwer = context.Brouwers.SingleOrDefault(b => b.BrouwerId == 1);
Gemeente gent = new Gemeente() { Postcode = "9000", Naam = "Gent" };
brouwer.Gemeente = gent;
context.SaveChanges();
```

```
begin transaction with isolation level: ReadCommitted
INSERT INTO [dbo].[Gemeente] ...
UPDATE [dbo].[Brouwer] ... WHERE ([BrouwerId] = 1)
commit transaction
SELECT ... FROM (Select COUNT(1) AS [A1] From [dbo].[Gemeente] ...
```

Details Stack Trace

```
1 update [dbo].[Brouwer]
2 set [Postcode] = '9000' /* @0 */
3 where ([BrouwerId] = 1 /* @1 */)
```

HoGent

111

## 7. EF : DbContext en updates

- ▶ Object Context verzorgt ook de associaties

- Opgelet met deletes

```
Console.WriteLine("\n---Onderhouden van associaties - Delete --");
brouwer = context.Brouwers.SingleOrDefault(b => b.BrouwerId == 1);
Bier bier = brouwer.Bieren.First();
brouwer.DeleteBier(bier);
context.SaveChanges();
```

```
public void DeleteBier(Bier bier)
{
    if (!Bieren.Contains(bier))
        throw new ArgumentException(string.Format("{0} is geen bier van {1}", bier.Naam, this.Naam));
    Bieren.Remove(bier);
}
```

- Delete verwijdert hier de associatie, niet het bier.

HoGent

112

## 7. EF : DbContext en updates

### ► Object Context verzorgt ook de associaties

- Runtime Fout : Reden associatie tussen Brouwer en Bier. Een bier moet gekoppeld zijn aan een Brouwer

The screenshot shows a code editor with C# code and a 'DbUpdateException was unhandled' dialog box. The code attempts to delete a Brouwer entity and its associated Bieren entities. A tooltip points to the problematic part of the code where the foreign key BrouwerId is null.

```
Console.WriteLine("\n--Onderhouden van associaties - Delete --");
brouwer = context.Brouwers.SingleOrDefault(b => b.BrouwerId == 1);
Bier bier = brouwer.Bieren.First();
brouwer.DeleteBier(bier);
context.SaveChanges();
//opkuis
brouwer.Bieren.Add(bier);
context.SaveChanges();
//opkuis
#endregion
Console.ReadKey();
```

**DbUpdateException was unhandled**

An error occurred while saving entities that do not expose foreign key properties for their relationships. The EntityEntries property will return null because a single entity cannot be identified as the source of the exception. Handling of exceptions while saving can be made easier by exposing foreign key properties in your entity types. See the InnerException for details.

Troubleshooting tips:

modelBuilder.Entity<Brouwer>()
 .HasMany(b => b.Bieren)
 .WithRequired().Map(m => m.MapKey("BrouwerId"))
 .WillCascadeOnDelete(false);

113

## 7. EF : DbContext en updates

### ► Object Context verzorgt ook de associaties

- 2 oplossingen

- Oplossing 1 : Mapping tussen Brouwer en Bier aanpassen. Maar dan blijft bier in de database bestaan, foreign key BrouwerId bevat dan de null waarde.

The screenshot shows a tooltip for the 'WithOptional()' method in the Entity configuration code. This method is used to handle the nullable foreign key relationship.

```
modelBuilder.Entity<Brouwer>()
    .HasMany(b => b.Bieren)
    .WithOptional().Map(m => m.MapKey("BrouwerId"))
    .WillCascadeOnDelete(false);
```

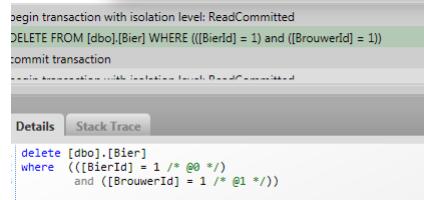
## 7. EF : DbContext en updates

- ▶ Object Context verzorgt ook de associaties

- 2 oplossingen

- Oplossing 2 : Bier zelf ook verwijderen uit de database. Maar dan moet je in BierhalleContext een DbSet toevoegen

```
Console.WriteLine("\n---Onderhouden van associaties - Delete --");
brouwer = context.Brouwers.SingleOrDefault(b => b.Brouwervl == 1);
Bier bier = brouwer.Bieren.First();
brouwer.DeleteBier(bier);
context.Bieren.Remove(bier);
context.SaveChanges();
```



```
begin transaction with isolation level: ReadCommitted
DELETE FROM [dbo].[Bier] WHERE ((BierId) = 1) and ([Brouwervl] = 1)
commit transaction
```

```
public DbSet<Brouwer> Brouwers { get; set; }
public DbSet<Gemeente> Gemeenten { get; set; }
public DbSet<Categorie> Categories { get; set; }
public DbSet<Bier> Bieren { get; set; }
```

115

## 7. EF : DbContext en updates

- ▶ DbContext verzorgt ook de bidirectionele relaties en FK en bijhorende associatie, na SaveChanges!

- Je kan werken via FK of via associatie

```
Console.WriteLine("\n---Update via gemeente--");
brouwer = context.Brouwers.FirstOrDefault(b => b.Gemeente!=null);
Console.WriteLine("Postcode voor : " + brouwer.Postcode);
Console.WriteLine("Gemeente voor : " + brouwer.Gemeente.Naam);
Gemeente gemeente = context.Gemeenten.SingleOrDefault(g => g.Postcode == "8800");
brouwer.Gemeente = gemeente;
Console.WriteLine("Postcode na instellen gemeente " + gemeente.Naam + " - " + gemeente.Postcode + ": " + brouwer.Postcode);
context.SaveChanges();
Console.WriteLine("Savechanges Postcode na instellen gemeente " + gemeente.Naam + " - " + gemeente.Postcode);
```

```
---Update via gemeente--
Postcode voor : 2870
Gemeente voor : Puurs
Postcode na instellen gemeente Roeselare-8800: 2870
Savechanges Postcode na instellen gemeente Roeselare-8800: 8800

---Update via postcode--
Gemeente na instellen postcode 9700 Oudenaarde : Roeselare
Savechanges Gemeente postcode 9700 Oudenaarde : Oudenaarde
```

HoGen

116

## 7. EF : DbContext en updates

### ▶ Oefening

- Voeg een nieuwe Brouwer toe uit de gemeente Brugge. Als de gemeente Brugge nog niet bestaat, maak de gemeente aan.

HoGent

117

## 8. Extra's : Interessante links

- ▶ Code First Migrations : <http://msdn.microsoft.com/en-us/data/jj591621> en <http://msdn.microsoft.com/en-us/data/jj554735>
- ▶ Performance considerations :  
<http://msdn.microsoft.com/en-us/data/hh949853>
- ▶ Concurrency : <http://msdn.microsoft.com/en-us/data/jj592904>

HoGent

118

## 8. Extra tools

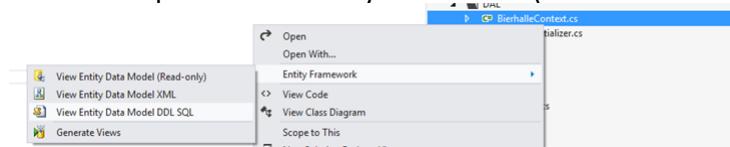
- ▶ Microsoft SQL Server Data Tools
  - <http://msdn.microsoft.com/en-us/data/tools.aspx>
  - Extra View : View > Sql Server Object Explorer

The screenshot shows two windows side-by-side. On the left is the 'EFExplained' tool showing the 'Bier [Design]' table with columns: BierId (uniqueidentifier, primary key), Naam (nvarchar(100)), AlcoholPercentage (float), and BrouwerId (int). On the right is the 'Sql Server Object Explorer' showing the database structure for 'Bierhalle'. It includes tables like dbo.Bier, dbo.Brouwer, dbo.Categorie, dbo.Cursus, dbo.GemengdeCursus, dbo.OnlineCursus, and dbo.OmselCursus. A tooltip on the right provides instructions for using the tool.

Opm : geeft problemen met DropCreateDbAlways. Bij runnen steeds Exception database in use. Opl : Delete de db Bierhalle in Sql Server Object Explorer. Vink close existing connections ook aan

## 8. Extra tools

- ▶ EF Power Tools
  - Download :  
<http://visualstudiogallery.msdn.microsoft.com/72a60b14-1581-4b9b-89f2-846072eff19d>
  - Rechtsklik op Context > Entity Framework (=EF Power Tool)



- Kies View Entity Data Model DDL SQL
- Meer op  
<http://msdn.microsoft.com/en-us/data/jj593170>

```
create table [dbo].[Bier] (
    [BierId] [int] not null identity,
    [Naam] [nvarchar](max) null,
    [AlcoholPercentage] [float] null,
    [BrouwerId] [int] not null,
    primary key ([BierId])
);

create table [dbo].[Brouwer] (
    [BrouwerId] [int] not null identity,
    [Naam] [nvarchar](max) not null,
    [Straat] [nvarchar](max) null,
    [Omzet] [int] null,
    [Geactiveerd] [bit] not null
);
```

## 8. Referenties

- ▶ EF : <http://msdn.microsoft.com/en-us/data/ef.aspx>
- ▶ Pluralsight : Entity Framework 4.1 – Code First
- ▶ Tutorial : <http://www.asp.net/mvc/tutorials/getting-started-with-ef-using-mvc/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application>