



**Vlaamse Dienst voor Arbeidsbemiddeling en Beroepsopleiding**

**C# 2013**

## **ADO.NET Entity Framework Takenbundel**

Deze cursus is eigendom van de VDAB©

## Inhoudsopgave

<b>1</b>	<b>TAKEN</b>	<b>1</b>
1.1	BANK MAKEN	1
1.2	Klanten en hun rekeningen	1
1.3	Zichtrekening toevoegen	1
1.4	Storten	2
1.5	Klant verwijderen	2
1.6	Overschrijven	2
1.7	Klant wijzigen	2
1.8	Personeel	3
1.9	Zichtrekeningen – spaarrekeningen	3
1.10	Totale saldo per klant	3
1.11	Administratieve kost	4
1.12	Code first	4
<b>2</b>	<b>VOORBEELDOPLOSSINGEN</b>	<b>5</b>
2.1	Bank maken	5
2.1.1	De method Storten van de class Rekening	5
2.2	Klanten en hun rekeningen	5
2.3	Nieuwe rekening	6
2.4	Storten	6
2.5	Klant verwijderen	7
2.6	Overschrijven	7
2.6.1	De class SaldoOntoereikendException	7
2.6.2	Een extra method in de partial class Rekening	7
2.6.3	De method Main	8
2.7	Klant wijzigen	8
2.7.1	Aanpassing aan het EDM	8
2.7.2	De method Main	9
2.8	Personeel	9
2.8.1	De method Main en method Afbeelden	9
2.9	Zichtrekeningen – Spaarrekeningen	10
2.9.1	De method Main	10
2.10	Totale saldo per klant	11
2.10.1	De method Main	11

---



<b>2.11</b>	<b>Administratieve kost</b>	<b>11</b>
<b>2.12</b>	<b>Code first</b>	<b>11</b>
2.12.1	Artikelgroep	11
2.12.2	Artikel	12
2.12.3	Leverancier	12
2.12.4	FoodArtikel	12
2.12.5	NonFoodArtikel	12
2.12.6	Properties in de context class	12
<b>3</b>	<b>COLOFON</b>	<b>13</b>

## 1 TAKEN

### 1.1 BANK MAKEN

Hetscript *CreateBank.sql* maakt de database *Bank*. Je voert deze script uit

De database bevat twee tables:

Klanten				Rekeningen			
Column Name	Data Type	Allow Nulls		Column Name	Data Type	Allow Nulls	
 KlantNr	int	<input type="checkbox"/>		 RekeningNr	nvarchar(20)	<input type="checkbox"/>	
Voornaam	nvarchar(50)	<input type="checkbox"/>		KlantNr	int	<input type="checkbox"/>	
				Saldo	decimal(10, 2)	<input type="checkbox"/>	
				Soort	char(1)	<input type="checkbox"/>	

De kolom *Soort* van de table *Rekeningen* bevat:

- S als het een spaarrekening betreft
- Z als het een zichtrekening betreft

Je maakt in Visual Studio een console applicatie.

Je voegt aan deze console applicatie een EDM toe met de entity classes

- *Klant* (die hoort bij de table *Klanten*) en
- *Rekening* (die hoort bij de table *Rekeningen*).

Je maakt in de class *Rekening* een method *Storten*.

- Deze method heeft een parameter *Bedrag*.
- Deze method verhoogt het *Saldo* met dit *Bedrag*.

### 1.2 Klanten en hun rekeningen

Je toont in de console een alfabetische lijst van de klanten.

Je toont per klant zijn naam, zijn rekeningen en het totale saldo van de rekeningen van de klant

Bart

Totaal:0

Homer

345-6789012-12:500,00

Totaal:500,00

Lisa

Totaal:0

Maggie

Totaal:0

Marge

123-4567890-02:1000,00

234-5678901-69:2000,00

Totaal:3000,00

### 1.3 Zichtrekening toevoegen

De gebruiker kan een zichtrekening toevoegen. Je toont eerst een alfabetische lijst van de klanten.

De gebruiker kiest daaruit één klant, door zijn klantnr. in te tikken:

5:Bart

2:Homer

3:Lisa

4:Maggie

1:Marge

KlantNr:

Als de gebruiker geen getal tikt, toon je de foutmelding *Tik een getal*.

Als de gebruiker een onbestaand klantr. intikt, toon je de foutmelding *Klant niet gevonden*.

Je vraagt daarna het rekeningnr. van de nieuwe rekening.

Je moet op dit rekeningnr. geen invoercontrole doen.

Je maakt met deze gegevens een nieuwe zichtrekening die bij de gekozen klant hoort.

Het saldo van de rekening staat op nul.

## 1.4 Storten

De gebruiker kan geld storten op een rekening.

Je vraagt eerst het rekeningnr. van de rekening waarop de gebruiker het geld wil storten. Als de gebruiker een onbestaand rekeningnr. intikt, toon je de foutmelding *Rekening niet gevonden*.

Je vraagt daarna het te storten bedrag. Als de gebruiker geen getal intikt, toon je de foutmelding *Tik een getal*. Als de gebruiker een getal intikt kleiner of gelijk aan nul, toon je de foutmelding *Tik een positief getal*.

## 1.5 Klant verwijderen

De gebruiker kan een klant verwijderen, op voorwaarde dat hij geen rekeningen meer heeft.

Je vraagt het klantr. van de te verwijderen klant.

Als de gebruiker geen getal intikt, toon je de foutmelding *Tik een getal*.

Als de gebruiker een onbestaand klantr. intikt, toon je de foutmelding *Klant niet gevonden*.

Als de gebruiker nog rekeningen heeft, toon je de foutmelding *Klant heeft nog rekeningen*.

## 1.6 Overschrijven

De gebruiker kan geld overschrijven van een rekening naar een andere rekening.

Je vraagt het rekeningnr. van de rekening waarvan het geld wordt overgeschreven.

Je vraagt het rekeningnr. van de rekening naarwaar het geld wordt overgeschreven.

Je vraagt het over te schrijven bedrag.

Als de gebruiker een onbestaand van-rekeningnr. intikt, toon je de foutmelding *Van rekening niet gevonden*.

Als de gebruiker een onbestaand naar-rekeningnr. intikt, toon je de foutmelding *Naar rekening niet gevonden*.

Als de gebruiker geen getal intikt bij bedrag, toon je de foutmelding *Tik een getal*.

Als de gebruiker een getal intikt bij bedrag dat kleiner is of gelijk aan nul, toon je de foutmelding *Tik een positief bedrag*.

Als het saldo van de van-rekening kleiner is dan het over te schrijven bedrag, toon je de foutmelding *Saldo ontoereikend*.

## 1.7 Klant wijzigen

De gebruiker kan de voornaam van een klant corrigeren.

Je vraagt het klantr. van de te wijzigen klant.

Als de gebruiker geen getal intikt, toon je de foutmelding *Tik een getal*.

Als de gebruiker een onbestaand klantr. intikt, toon je de foutmelding *Klant niet gevonden*.

Je vraagt de gecorrigeerde voornaam van de klant.


Tussen het lezen van de klant en het wijzigen van de klant, kan een andere gebruiker dezelfde klant wijzigen. Je vangt deze fout op met optimistic record locking. Je toont dan de foutmelding *Een andere gebruiker wijzigde deze klant*

## 1.8 Personeel

Het script *PersoneelToevoegen.sql* voegt aan de database *Bank* een table *Personeel* toe.

Je voert deze script uit.

De table *Personeel*:

Personeel			
	Column Name	Condensed Type	Nullable
	 PersoneelsNr	int	No
	Voornaam	nvarchar(50)	No
	ManagerNr	int	Yes

Personeelsleden waarvan de kolom *ManagerNr* leeg is, staan hoogst in de hiërarchie.

Je voegt aan je EDM een bijbehorende entity class *Personeelslid* toe.

Je toont alle personeelsleden, beginnend met personeelsleden hoogst in hiërarchie.

Je toont per personeelslid de ondergeschikten.

Iedere keer je een lager niveau toont, toon je een extra tab teken voor de voornaam:

Diane

    Mary

        William

            Andy

            Peter

            Tom

        Gerard

            Loui

            Pamela

            Larry

            Barry

            Martin

        Anthony

            Leslie

            Judy

            Steve

            Foon Yue

            George

        Mami

            Yoshimi

    Jeff

## 1.9 Zichtrekeningen – spaarrekeningen

Je leidt twee derived classes af van de class *Rekening*:

- Zichtrekening De bijbehorende records hebben de waarde Z in de kolom *Soort*
- Spaarrekening De bijbehorende records hebben de waarde S in de kolom *Soort*

Je maakt van de class *Rekening* een abstract class.

Je toont daarna van zichtrekeningen het rekeningnr. en het saldo.

## 1.10 Totale saldo per klant

Hetscript *TotaleSaldoPerKlantMaken.sql*. voegt aan de database *Bank* een view *TotaleSaldoPerKlant* toe. Je voert deze script uit.

De view bevat 3 kolommen: *KlantNr*, *Voornaam*, *TotaleSaldo*

(het totale saldo van de rekeningen van één klant)

Voeg aan het EDM een entity toe die bij deze view hoort.

Toon daarna per klant de voornaam en het totale saldo van die klant.

### 1.11 Administratieve kost

Het script *AdministratieveKostMaken.sql* maakt in de database *Bank* een stored procedure *AdministratieveKost*. Je voert deze script uit.

De stored procedure bevat één invoerparameter: *Kost* (decimal).

De stored procedure vermindert het saldo van alle rekeningen met het bedrag in deze invoerparameter *Kost*.

Je vraagt aan de gebruiker het bedrag van de kost.

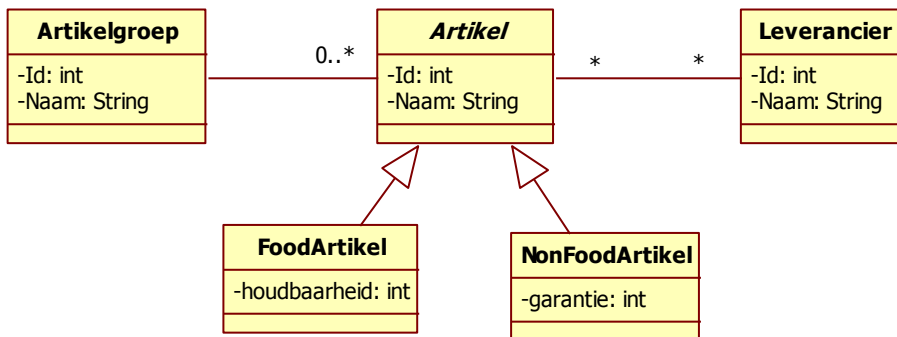
Als de gebruiker geen getal intikt, toon je de foutmelding *Tik een getal*.

Anders roep je de stored procedure op, waarbij je het ingetikte getal als parameter meegeeft.

Je toont aan de gebruiker het aantal aangepaste rekeningen.

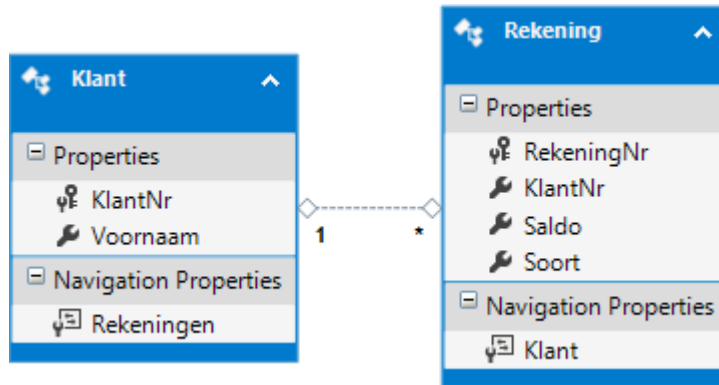
### 1.12 Code first

Je maakt in een nieuw project volgende classes met code first en een bijbehorende database



## 2 VOORBEELDOPLOSSINGEN

### 2.1 Bank maken



- De class *Klanten* is hernoemd naar *Klant*
- De class *Rekeningen* is hernoemd naar *Rekening*
- De navigation property *Klanten* van de class *Rekening* is hernoemd naar *Klant*.

#### 2.1.1 De method *Storten* van de class *Rekening*

```

namespace EFTaken
{
    public partial class Rekening
    {
        public void Storten(decimal bedrag)
        {
            this.Saldo += bedrag;
        }
    }
}
  
```

### 2.2 Klanten en hun rekeningen

```

static void Main(string[] args)
{
    using (var entities = new BankEntities())
    {
        var query = from klant in entities.Klanten.Include("Rekeningen")
                    orderby klant.Voornaam
                    select klant;
        foreach (var klant in query)
        {
            Console.WriteLine(klant.Voornaam);
            var totaleSaldo = Decimal.Zero;
            foreach (var rekening in klant.Rekeningen)
            {
                Console.WriteLine("{0}: {1}", rekening.RekeningNr, rekening.Saldo);
                totaleSaldo += rekening.Saldo;
            }
            Console.WriteLine("Totaal: {0}", totaleSaldo);
            Console.WriteLine();
        }
    }
}
  
```



## 2.3 Nieuwe rekening

```
static void Main(string[] args)
{
    using (var entities = new BankEntities())
    {
        var query = from klant in entities.Klanten orderby klant.Voornaam select klant;
        foreach (var klant in query)
        {
            Console.WriteLine("{0}: {1}", klant.KlantNr, klant.Voornaam);
        }
        try
        {
            Console.Write("KlantNr:");
            var klantNr = int.Parse(Console.ReadLine());
            var klant = entities.Klanten.Find(klantNr);
            if (klant == null)
            {
                Console.WriteLine("Klant niet gevonden");
            }
            else
            {
                Console.Write("RekeningNr:");
                var rekeningNr = Console.ReadLine();
                var rekening = new Rekening {RekeningNr = rekeningNr, Soort = "Z"};
                klant.Rekeningen.Add(rekening);
                entities.SaveChanges();
            }
        }
        catch (FormatException)
        {
            Console.WriteLine("Tik een getal");
        }
    }
}
```

## 2.4 Storten

```
static void Main(string[] args)
{
    using (var entities = new BankEntities())
    {
        Console.Write("RekeningNr:");
        var rekeningNr = Console.ReadLine();
        var rekening = entities.Rekeningen.Find(rekeningNr);
        if (rekening == null)
        {
            Console.WriteLine("Rekening niet gevonden");
        }
        else
        {
            try
            {
                Console.Write("Bedrag:");
                var bedrag = decimal.Parse(Console.ReadLine());
                if (bedrag <= Decimal.Zero)
                {
                    Console.WriteLine("Tik een positief bedrag");
                }
                else
                {
                    rekening.Storten(bedrag);
                    entities.SaveChanges();
                }
            }
            catch (FormatException)
            {
                Console.WriteLine("Tik een bedrag");
            }
        }
    }
}
```

## 2.5 Klant verwijderen

```
static void Main(string[] args)
{
    try
    {
        Console.Write("KlantNr:");
        var klantNr = int.Parse(Console.ReadLine());
        using (var entities = new BankEntities())
        {
            var klant = entities.Klanten.Find(klantNr);
            if (klant == null)
            {
                Console.WriteLine("Klant niet gevonden");
            }
            else
            {
                if (klant.Rekeningen.Count != 0)
                {
                    Console.WriteLine("Klant heeft nog rekeningen");
                }
                else
                {
                    entities.Klanten.Remove(klant);
                    entities.SaveChanges();
                }
            }
        }
    }
    catch (FormatException)
    {
        Console.WriteLine("Tik een getal");
    }
}
```

## 2.6 Overschrijven

### 2.6.1 De class SaldoOntoereikendException

```
using System;
namespace EFTaken
{
    public class SaldoOntoereikendException:Exception
    {
    }
}
```

### 2.6.2 Een extra method in de partial class Rekening

```
public void Overschrijven(Rekening naarRekening, decimal bedrag)
{
    if (this.Saldo < bedrag)
    {
        throw new SaldoOntoereikendException();
    }
    else
    {
        this.Saldo -= bedrag;
        naarRekening.Saldo += bedrag;
    }
}
```

### 2.6.3 De method Main

```
static void Main(string[] args)
{
    Console.WriteLine("RekeningNr. van rekening:");
    var vanRekeningNr = Console.ReadLine();
    Console.WriteLine("RekeningNr. naar rekening:");
    var naarRekeningNr = Console.ReadLine();
    try
    {
        Console.WriteLine("Bedrag:");
        var bedrag = decimal.Parse(Console.ReadLine());
        if (bedrag <= decimal.Zero)
            Console.WriteLine("Tik een positief bedrag");
        else
        {
            var transactionOptions = new System.Transactions.TransactionOptions
            {IsolationLevel=System.Transactions.IsolationLevel.RepeatableRead};
            using (var transactionScope = new
                System.Transactions.TransactionScope(
                    System.Transactions.TransactionScopeOption.Required,
                    transactionOptions))
            {
                using (var entities = new BankEntities())
                {
                    var vanRekening = entities.Rekeningen.Find(vanRekeningNr);
                    if (vanRekening == null)
                        Console.WriteLine("Van rekening niet gevonden");
                    else
                    {
                        var naarRekening = entities.Rekeningen.Find(naarRekeningNr);
                        if (naarRekening == null)
                            Console.WriteLine("Naar rekening niet gevonden");
                        else
                        {
                            try
                            {
                                vanRekening.Overschrijven(naarRekening, bedrag);
                                entities.SaveChanges();
                                transactionScope.Complete();
                            }
                            catch (SaldoOntoereikendException)
                            {
                                Console.WriteLine("Saldo ontoereikend");
                            }
                        }
                    }
                }
            }
        }
    }
    catch (FormatException)
    {
        Console.WriteLine("Tik een bedrag");
    }
}
```

## 2.7 Klant wijzigen

### 2.7.1 Aanpassing aan het EDM

Je wijzigt de property *Concurrency Mode* van de property *Voornaam* van de class *Klant* naar *Fixed*.



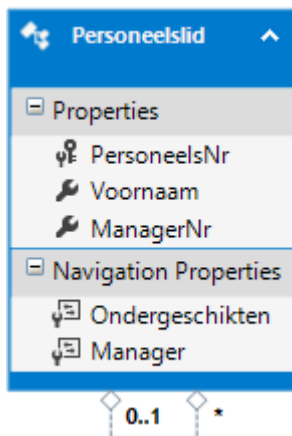
Opmerking: Je hoeft dit niet te doen voor de property *KlantNr*.

De bijbehorende kolom is een autonummer kolom en kan dus niet gewijzigd worden bij het wijzigen van een record.

### 2.7.2 De method Main

```
static void Main(string[] args)
{
    Console.WriteLine("KlantNr:");
    try
    {
        var klantNr = int.Parse(Console.ReadLine());
        using (var entities = new BankEntities())
        {
            var klant = entities.Klanten.Find(klantNr);
            if (klant == null)
            {
                Console.WriteLine("Klant niet gevonden");
            }
            else
            {
                Console.WriteLine("Voornaam:");
                klant.Voornaam = Console.ReadLine();
                entities.SaveChanges();
            }
        }
    }
    catch (DbUpdateConcurrencyException)
    {
        Console.WriteLine("Een andere gebruiker wijzigde deze klant");
    }
    catch (FormatException)
    {
        Console.WriteLine("Tik een getal");
    }
}
```

## 2.8 Personeel

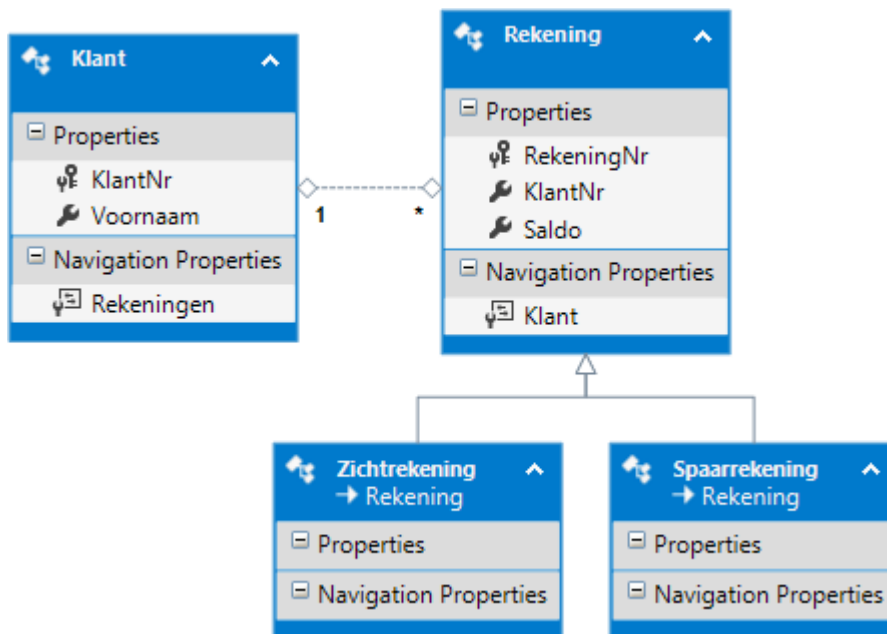


### 2.8.1 De method Main en method Afbeelden

```
class Program
{
    static void Main(string[] args)
    {
        using (var entities = new BankEntities())
        {
            var hoogstenInHierarchie =
                (from personeelslid in entities.Personeel
                 where personeelslid.Manager == null
                 select personeelslid).ToList();
            new Program().Afbeelden(hoogstenInHierarchie, 0);
        }
    }
}
```

```
void Afbeelden(List<Personeelslid> personeel, int insprong)
{
    foreach (var personeelslid in personeel)
    {
        Console.WriteLine(new String('\t', insprong));
        Console.WriteLine(personeelslid.Voornaam);
        if (personeelslid.Ondergeschikten.Count != 0)
        {
            Afbeelden(personeelslid.Ondergeschikten.ToList(), insprong + 1);
        }
    }
}
```

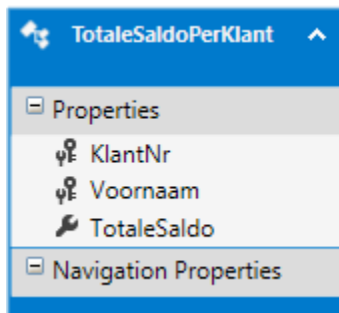
## 2.9 Zichtrekeningen – Spaarrekeningen



### 2.9.1 De method Main

```
static void Main(string[] args)
{
    using (var entities = new BankEntities())
    {
        var query = from rekening in entities.Rekeningen
                     where rekening is Zichtrekening
                     select rekening;
        foreach (var zichtrekening in query)
        {
            Console.WriteLine("{0}: {1}",
                               zichtrekening.RekeningNr, zichtrekening.Saldo);
        }
    }
}
```

## 2.10 Totale saldo per klant



De property *Entity Set Name* is hernoemd naar *TotaleSaldosPerKlant*

### 2.10.1 De method Main

```
static void Main(string[] args)
{
    using (var entities = new BankEntities())
    {
        var query = from totaleSaldoPerKlant
                     in entities.TotaleSaldosPerKlant
                     orderby totaleSaldoPerKlant.Voornaam
                     select totaleSaldoPerKlant;
        foreach (var totaleSaldoPerKlant in query)
            Console.WriteLine("{0}: {1}", totaleSaldoPerKlant.Voornaam,
                              totaleSaldoPerKlant.TotaleSaldo);
    }
}
```

## 2.11 Administratieve kost

```
static void Main(string[] args)
{
    try
    {
        Console.WriteLine("Kost:");
        var kost = decimal.Parse(Console.ReadLine());
        using (var entities = new BankEntities())
        {
            Console.WriteLine("{0} rekeningen aangepast",
                              entities.AdministratieveKost(kost));
        }
    }
    catch (FormatException)
    {
        Console.WriteLine("Tik een getal");
    }
}
```

## 2.12 Code first

### 2.12.1 Artikelgroep

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
namespace CodeFirstCursus
{
    [Table("Artikelgroepen")]
    public class Artikelgroep
    {
        public int Id { get; set; }
        public string Naam { get; set; }
        public ICollection<Artikel> Artikels { get; set; }
    }
}
```

### 2.12.2 Artikel

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
namespace CodeFirstCursus
{
    [Table("Artikels")]
    public abstract class Artikel
    {
        public int Id { get; set; }
        public string Naam { get; set; }
        public virtual Artikelgroep Artikelgroep { get; set; }
        public int ArtikelgroepId { get; set; }
        public ICollection<Leverancier> Leveranciers {get;set;}
    }
}
```

### 2.12.3 Leverancier

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
namespace CodeFirstCursus
{
    [Table("Leveranciers")]
    public class Leverancier
    {
        public int Id { get; set; }
        public string Naam { get; set; }
        public ICollection<Artikel> Artikels { get; set; }
    }
}
```

### 2.12.4 FoodArtikel

```
using System.ComponentModel.DataAnnotations.Schema;
namespace CodeFirstCursus
{
    [Table("FoodArtikels")]
    public class FoodArtikel:Artikel
    {
        public int Houdbaarheid { get; set; }
    }
}
```

### 2.12.5 NonFoodArtikel

```
using System.ComponentModel.DataAnnotations.Schema;
namespace CodeFirstCursus
{
    [Table("NonFoodArtikels")]
    public class NonFoodArtikel:Artikel
    {
        public int Garantie { get; set; }
    }
}
```

### 2.12.6 Properties in de context class

```
public DbSet<Artikelgroep> Artikelgroepen { get; set; }
public DbSet<Artikel> Artikels { get; set; }
public DbSet<Leverancier> Leveranciers { get; set; }
```

---

## 3 COLOFON

**Sectorverantwoordelijke:**

**Cursusverantwoordelijke:**

**Medewerkers:**

Hans Desmet

**Versie:**

27/11/2014

**Nummer dotatielijst:**