



Odisee
DE CO-HOGESCHOOL

Entity Framework



Jens Baetens



Databank

Databanken

- ▣ Verzameling van gegevens die in een bepaalde structuur staan
- ▣ Voorbeelden:
 - SQL Server, MariaDB, MySQL, MongoDB (NoSQL)
- ▣ In deze cursus maken we gebruik van een RDBMS
 - Relational Database Management System
 - Tabellen, Kolommen, Rijen
 - SQL

SQL - Select

- ▣ SELECT columnName FROM tableName WHERE condition GROUP BY columnName HAVING condition ORDER BY columnName ;





SQL - Insert

▣ INSERT INTO tableName (columnName) VALUES (value);





SQL - Delete

- ▣ DELETE FROM tableName WHERE condition;



SQL - Update


▣ UPDATE tableName SET columnName = value WHERE condition;



Databanken in WPF/C#/.NET Framework

Databanken in Visual Studio

- Ga naar Tools > SQL Server > New Query
 - ▬ Indien dit niet lukt: Open de visual studio installer en controleer of het volgende geïnstalleerd is

- 
- A screenshot of the Visual Studio Installer window showing a list of installed components. Each item has a blue checkmark in a box to its left. The components listed are:
- ✓ SQL ADAL runtime
 - ✓ SQL Server Command Line Utilities
 - ✓ SQL Server Data Tools
 - ✓ SQL Server Express 2016 LocalDB
 - ✓ SQL Server ODBC Driver

Databank in VS

- ▣ Kies een server => MSSQLLOCALDB
- ▣ Maak een databank
 - ▬ CREATE DATABASE demo;
- ▣ Controleer in SQL Server Object Browser of de databank bestaat

Databanken aanspreken in C#

▣ ADO.NET

- ActiveX Data Objects
- Framework om queries aan te maken en te sturen naar RDBMS
- Zelf queries schrijven

ADO.NET

0 references

```
private List<string> searchUser(string name)
{
    List<string> names = new List<string>();
    SqlConnection connection = new SqlConnection("data source=(localdb)\\MSSQLLOCALDB; initial catalog=demo");
    SqlCommand command = new SqlCommand($"Select * from dbo.authors where first_name = '{name}';", connection);

    connection.Open();
    SqlDataReader reader = command.ExecuteReader();
    while (reader.Read())
    {
        names.Add($"{reader[1]}");
    }
    reader.Close();
    connection.Close();

    return names;
}
```

Welk probleem kan hier voorvallen?

SQL injection

```
Program.searchUser("Tom'; Drop Table dbo.authors; --");
```

- Aanmaken queries is een kwetsbaarheid
- Technieken om je hiertegen te wapenen zijn
 - SQLParameters
 - Placeholders
- Gevaar is hiermee niet weg dus moet je er steeds opletten



▣ Nadelen

- Foutgevoelig
 - Geen compilatiefouten maar runtime fouten
- Gevoelig voor SQL injection
- Veel repetitief werk
 - Steeds aanmaken van de connectie, queries, ...
 - Veel overhead



Entity Framework

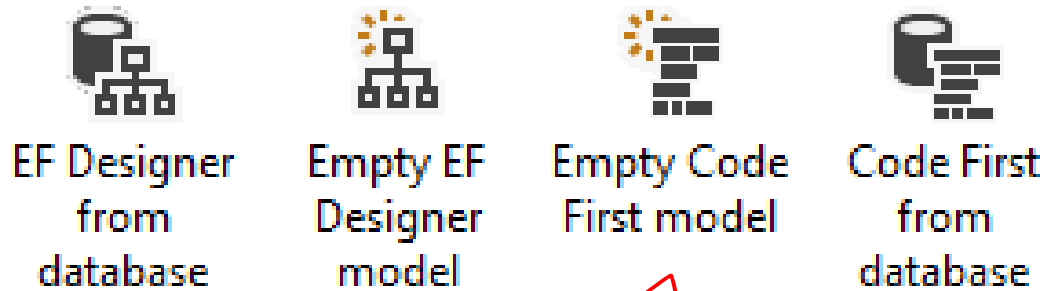
Entity Framework

obj in C# Database

- ORM framework: Object Relational Mapping
 - ▬ SQL tabellen en relaties matchen met objecten in de code
- Geen SQL injection mogelijk
 - ▬ Wordt voor ons gecontroleerd
- Minder repitief^{te} werk
 - ▬ De volledige connectie met de database wordt voor ons gedaan
- Zorg ervoor dat het NugetPackage voor EntityFramework geïnstalleerd is

Entity Framework

- 4 manieren om met dit framework te werken in .Net Framework



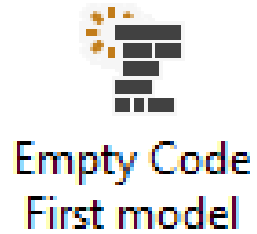
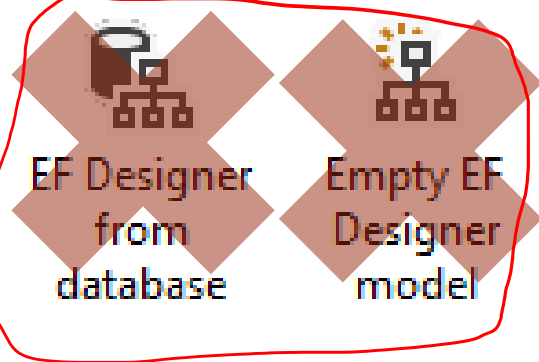
Zelf beginnen
code te schrijven
→ database wordt
voor jou aangemaakt

Binnen dit vak geven
we hieraan de voorkeur

Entity Framework

- 4 manieren om met dit framework te werken in .Net Framework

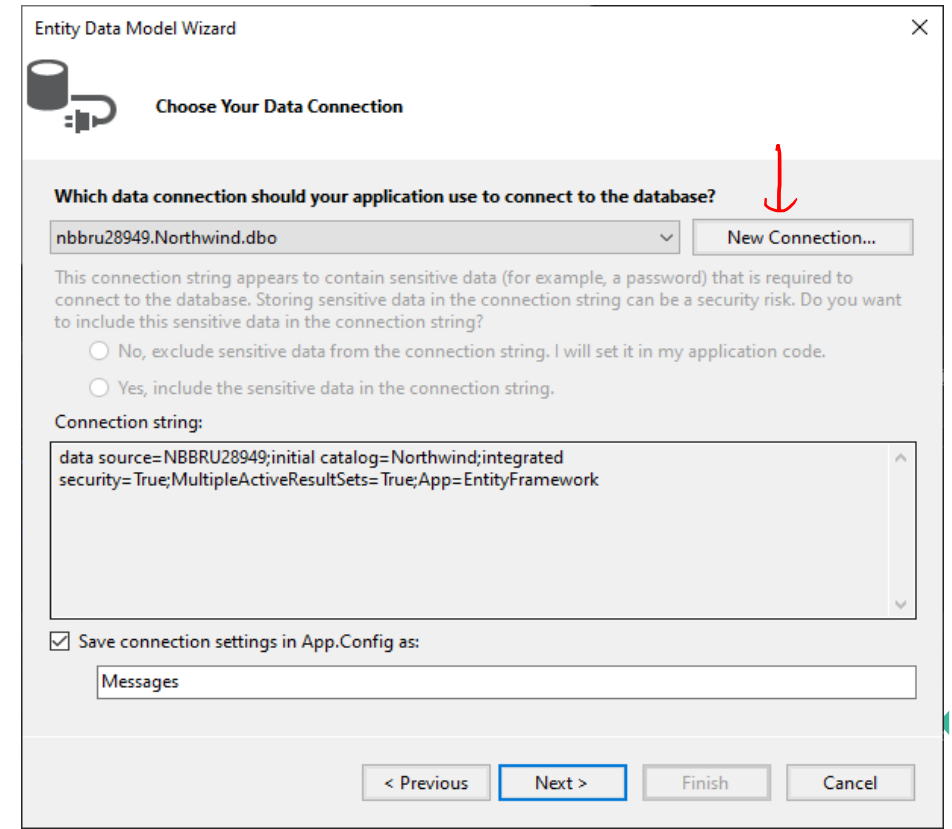
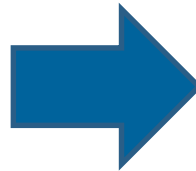
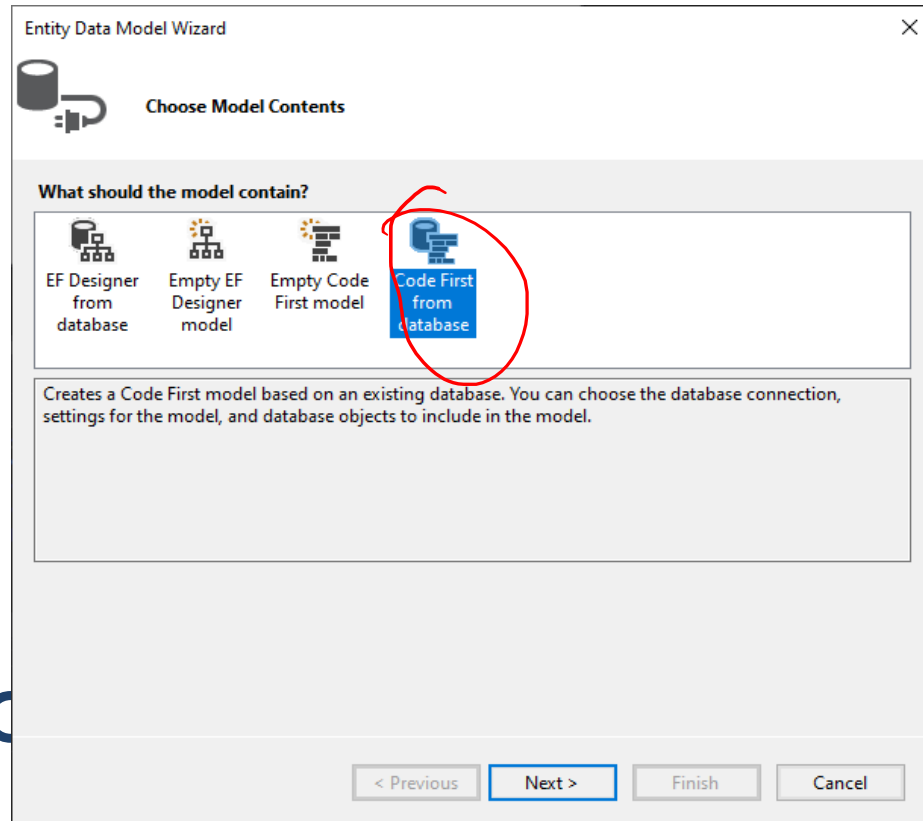
Net een GUI



- In .Net core zijn er echter twee minder

Code first from database

- Eerst database aanmaken
- Daarna code genereren op basis van de database



Code first from database

■ Maak indien nodig een nieuwe connectie aan

Kies de servernaam
Op je lokale machine normaal gezien
(localdb)\MSSQLLOCALDB.
Deze wordt niet altijd automatisch herkend

Kies de database die je wil gebruiken.

Connection Properties

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient) Change...

Server name: (localdb)\MSSQLLOCALDB Refresh

Log on to the server

Authentication: Windows Authentication

User name: Password: Save my password

Connect to a database

☒ Select or enter a database name: Messages

☐ Attach a database file: Browse...

Logical name:

Test Connection OK Cancel Advanced...

Code First from database

- ▣ Kies de te overnemen tabellen
- ▣ Als dit goed gaat, worden een aantal klassen aangemaakt

Entity Data Model Wizard

Choose Your Database Objects and Settings

Which database objects do you want to include in your model?

☒ Tables

☐ Views

☒ Pluralize or singularize generated object names

☒ Include foreign key columns in the model

☐ Import selected stored procedures and functions into the entity model

< Previous Next > **Finish** Cancel

Code First without database

- ▣ Wanneer er nog geen database is
 - ▣ Database wordt gemaakt op basis van de klassen
 - ▣ Heeft in dit vak de voorkeur
-
- ▣ Er is ook een wizard hiervoor maar het is zo eenvoudig dat het niet nodig is

Code First without database

▣ Volgende stappen gaan we uitvoeren

- ▬ NuGet Packages installeren
- ▬ DbContext aanmaken
- ▬ connectionString instellen
- ▬ Aanmaken aantal dataklassen met *→ User*
 - ▣ Default constructor
 - ▣ Public property met getter en setter voor elk field/member in je dataklasse

Code first without databases – NuGet Packages

▣ .Net Framework applicatie

- EntityFramework

▣ .Net Core

- EntityFrameworkCore
- EntityFrameworkCore.SqlServer
- EntityFrameworkCore.Tools

Code first without databases – DbContext

- ▣ Deze klasse spreekt de database aan via het entity framework
- ▣ Maak een klasse aan die ervan overerft

0 references
`class UserContext: DbContext`
`{`
`...`
`}`

≈ Database

Code first without databases – Connectionstring

- Geef in de DbContext – klasse aan met welke database er moet gecommuniceerd worden

- Via connectionstring

- Waar? (Ip-adres)
- Hoe connecteren we (Username, pass, ...)
- Welke taal gebruikt de database
- Hoe noemt de database

- De structuur van deze string is afhankelijk van de data provider

- SQL: "Data Source=TestServer;Initial Catalog=Pubs;User ID=Dan;Password=training"
Server Database Username wachtwoord

Code first without databases – Connectionstring

- ▣ "Data Source=TestServer;Initial Catalog=Pubs;User ID=Dan;Password=training"
 - ▬ Datasource
 - Server waarop de databank draait
 - ▬ Initial catalog
 - Databank op de server die we gebruiken
 - ▬ User ID
 - Username om in te loggen (enkel nodig indien ze beveiligd is)
 - ▬ Password
 - Password om in te loggen (enkel nodig indien ze beveiligd is)

Code first without databases – Connectionstring

- ▣ Kan ingesteld worden in de `app.config` file
- ▣ Xml file om configuraties in te stellen

```
<configuration>
  <entityFramework>
    <providers>
      <provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer"/>
    </providers>
  </entityFramework>
  <connectionStrings>
    <add name="users" connectionString="data source = (localdb)\MSSQLLOCALDB; initial catalog=users;" providerName="System.Data.SqlClient"/>
  </connectionStrings>
</configuration>
```



Name om later de connectionstring terug te vinden



Connectionstring met alle info



Providername, deze geeft aan welk type db we gebruiken

Code first without databases – Connectionstring

- Geef de connectionstring mee in de constructor van de DbContext klasse

```
1 reference
class UserContext: DbContext
{
    0 references
    public UserContext(): base("users")
    {
        // users is de naam van de connectionstring, see app.config
    }
}
```

naam v.d. conn string

Code first without databases – User dataklasse

- ▣ DbContext ≈ Database
- ▣ Dataklassen ≈ Tabellen (Type DbSet<T>)
- ▣ Properties in deze klassen ≈ kolommen

```
1 reference
class DbContext: DbContext
{
    0 references
    public UserContext((): base("users")
    {
        // users is de naam van de connectionstring, see app.config
    }

    0 references
    public DbSet<User> users { get; set; }
}
```

≈ Tabel
→ elke user hiervan
≈ Roster
Rij

Kolommen →

```
1 reference
class User
{
    private string firstName;
    private string lastName;
    private int id;

    0 references
    public string FirstName { get { return firstName; } set { firstName = value; } }
    0 references
    public string LastName { get { return lastName; } set { lastName = value; } }
    0 references
    public int Id { get { return id; } set { id = value; } }
}
```

Code first without databases – User dataklasse

- ▣ `DbSet<T>` is een soort *lijst* *Enumerable*
 - ▬ Elk item in de lijst stelt een rij of record voor

Code first without databases – User dataklasse

■ In het vak data en informatieverwerking gezien dat een table een aantal **constrains** kan hebben

- Primary key
- Not Null

• Foreign Key

Nog niet gedaan

■ 2 manieren om dit toe te voegen

- Data annotaties voor de eenvoudigere zaken
- Fluent API voor complexere zaken

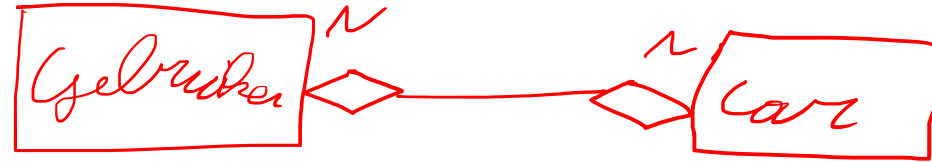
← Tot nu toe enkel deze manier

Data annotaties

- ▣ [Key]
 - ▬ Primary key (verplicht)
- ▣ [Column("blog_id")]
 - ▬ Andere naam dan property
- ▣ [Column(TypeName="varchar(200)")]
 - ▬ Bepaald type in de databank
- ▣ [MaxLength(500)]
 - ▬ Max lengte van string
- ▣ [Required]
 - ▬ Not null
- ▣ [NotMapped]
 - ▬ By default krijgen alle public properties een veld in de databank



Fluent API



- Dit wordt toegevoegd in de (te overridden) OnModelCreating methode
 - In DbContext
 - Meestal niet nodig maar hier kan alles specifiek ingesteld worden

Gebruikers

ID	FN	LN	Car
			1
			2

Cars

ID	Mark	Model

Gebruiker ID	Car ID
1	2
1	3
1	4
2	2
2	4

How modeller je List<Car> cars?

Fluent API - examples

De tabel waarop we
wijzigingen willen doen.

Instellen van een
primary key bestaande
uit 2 velden

0 references

```
protected void OnModelCreating(DbModelBuilder modelBuilder)
{
    modelBuilder.Entity<User>()
        .HasKey(user => new {user.FirstName, user.LastName});

    modelBuilder.Entity<User>()
        .HasMany<Car>(user => user.Cars)
        .WithMany(car => car.Users);

    modelBuilder.Entity<User>()
        .HasOptional<Car>(user => user.FavoriteCar);
}
```

Instellen van een
veel op veel
relatie

Instellen van een
optionele waarde
FavoriteCar

Fluent API - examples

```
modelBuilder.Entity<Car>().Property(car => car.Model)
    .HasMaxLength(50)
    .IsOptional()
    .HasColumnName("reeks");
```

Instellen van een maximale
lengte

Instellen dat kolom model
optioneel is

Instellen dat we de naam
reeks gebruiken in de
database