**Oefening EF Deel 3: HondenRescue**



**1. Start**  
  
Vertrek vanuit het startbestand H06 HRStartbestand. Deze navigatiebalk is reeds voorzien.   

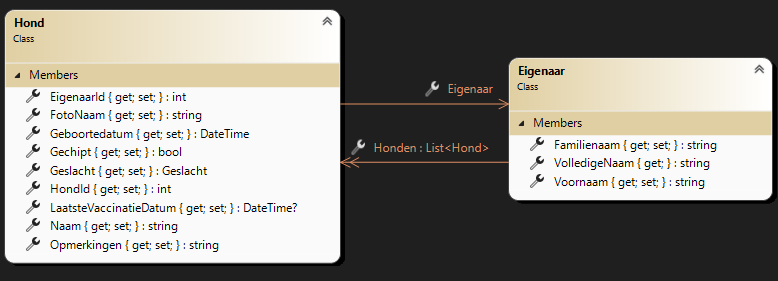

We beginnen eerst met de **one** to many te implementeren.

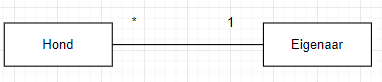
**2. Models en data**

**2A Nieuw klassenmodel: Eigenaar**  
**2B Database-Model Eigenaar**

Er is een nieuwe model **Eigenaar** dat dient toegevoegd te worden.

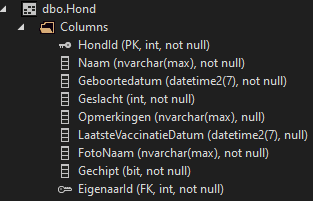
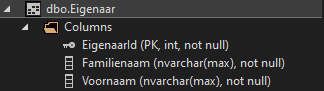
Gebruik het gegeven model om deze classe aan te maken en de properties te voorzien en Eveneens de relaties te verankeren in de klasse.  
  
In het model van Eigenaar voorzie je bijkomend een getter die VolledigeNaam concateneert met als output: LIEFOGEN Jef  
Klassendiagram:



Datamodel:  


**2C (Database) migrations**  
  
Je voert deze wijzingen toe door een migratie te maken en deze te updaten naar de database. Gebruik bij voorkeur als naam: MIG2

Controle resultaat:



**2C Data voorzien**

In de vertreksolution is een sql-bestand data-H06.sql voorzien, importeer dit in de database. Deze zal bestaande data bijwerken en nieuwe data toevoegen. Indien er eventuele query fouten optreden, analyseer of je model wel overeenkomt.

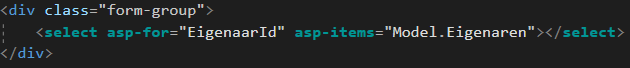
**3. Relaties**

Voorzie de relaties met Fluent API in OnModelCreating op professionele wijze.

**4. Nieuwe hond toevoegen**

Voorzie een dropdown om een eigenaar te kunnen selecteren. De eigenaren zijn dus gebaseerd op de **one to many relatie.**

Tip: Voor een dropdown te maken, kan je inspiratie halen uit vorige oefening. Er bestaan nog efficiëntere snellere manieren. Geïnteresseerd?   
Controller:  
View (en in je viewModel een SelectList van Eigenaren):



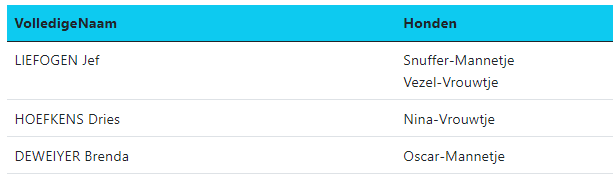
Resultaat:



**5. Bestaande hond aanpassen**

Voorzie nèt als in de Create een dropdown om een eigenaar te kunnen selecteren, indien er al een eigenaar is toegewezen, is deze default geselecteerd in de dropdown.

**6. Eigenaar en hun honden (/Hond/EigenaarEnHonden)**



**7. Many to many**

In bovenstaande situatie kon een hond eigenaar meerdere honden hebben, maar één hond had altijd maar één eigenaar. In deze nieuwe situatie wordt het mogelijk dat een hond **meerdere eigenaren** kan hebben. Je maakt **een kopie (bv. MijnOPL\_H6\_1\_op\_veel) naar map/zip van je voorgaande oplossing,** want we gaan deze omzetten naar een andere relatievorm, maar je wil al je harde inzet niet kwijt.

(!) Probeer deze niet in de vorige solution te bouwen en beide te combineren, want dat gaat niet in deze casus.

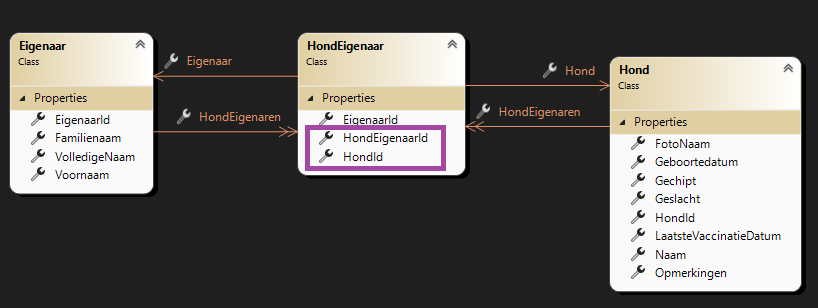
Verwijder dus de relaties van de one to many in de models en Fluent API. Verwijder de bestaande HondController. Maak een nieuwe HondController aan met DI naar de Database. Alle CUD-operaties moeten niet meer geïmplementeerd worden. We gaan weldra enkel de READ-operatie(s) benutten.

**6A Nieuw klassenmodel: HondEigenaar**  
**6B Database-Model HondEigenaar**

Er is een nieuwe associatie-entiteit Hond**Eigenaar** dat dient toegevoegd te worden.

Gebruik het gegeven klasse- of datamodel om deze classe aan te maken en de properties te voorzien en eveneens de relaties te verankeren in de klasse.

Voorzie ook de relaties met Fluent API in OnModelCreating op professionele wijze.  
  
De FK’s: integers zijn paars gemarkeerd.

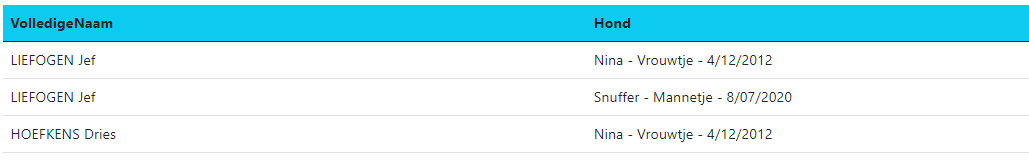


Voer een migratie uit op de database en controleer of de associatie-entiteit aanwezig is. Laad de data in.

**6C View browser output**

Url: Hond/EigenarenHonden/

Zorg voor logica code in de controller, alsook een viewmodel en een view. Hieronder zie je het uiteindelijke resultaat van de view dat je dient te bereiken. Voorzie dus een many to many dat volgende resultaat oplevert:



Zo heeft LIEFOGEN – Jef: 2 honden, waarvan Nina en deze 2 eigenaren heeft namelijk.

**7. Uitbreiding**

Verbeter de width in bovenstaande tabel, daarnaast is het label VolledigeNaam niet netjes. Maak het professioneler. Bedenk een creatieve manier om de hond zijn gegevens ook nog efficiënter af te beelden.