Felipe Gavilán

Software Engineering

APRIL 14, 2019
GAVILANCH

Entity Framework Core: Foreign key linked with a non-primary key

How can we establish a one-to-one relationship between two models, if we do not want to use the primary key to link the foreign key?

That is, suppose we have the following models:

```
1  public class Menu
2  {
3    [Key]
4    public int Id { get; set; }
5    public string MenuCode { get; set; }
6    public Route Route { get; set; }
7  }
8
9  public class Route
10 {
11   [Key]
12   public int Id { get; set; }
13   public string MenuCode { get; set; }
14   public string Name { get; set; }
15   public Menu Menu { get; set; }
16 }
```

We want to establish the one-to-one relationship between them, however, we want the foreign key of Route "MenuCode" to be linked to the "MenuCode" field of the Menu class.

If we try to do the following using the Fluent API in the data context:

```
modelBuilder.Entity<Menu>()
.HasOne(x => x.Route)
.WithOne(x => x.Menu)
.HasForeignKey<Route>(x => x.MenuCode);
```

We'll get the following error:

The relationship from 'Menu.Route' to 'Route.Menu' with foreign key properties {'MenuCode' : string} cannot target the primary key {'Id' : int} because it is not compatible. Configure a principal key or a set of compatible foreign key properties for this relationship.

This error occurs because you are trying to link MenuCode of the class Route with Id of the class Menu. The problem is that MenuCode is a string and Id is an integer, therefore, they are not compatible. What we want is to link MenuCode of Route with MenuCode of Menu.

The solution is to configure the **PrincipalKey**. The PrincipalKey will allow us to define the reference key with a **unique restriction** which will be the **destination** of the relationship. That is to say, with the PrincipalKey we can say that we want to link our foreign key MenuCode of the class Route, with the field MenuCode of the class Menu.

We emphasize that the reference column of the other table must have unique values, that is, it must have a unique constraint. The unique constraint means that values cannot be repeated in different entries in a column. Once we configure a field as PrincipalKey, the unique constraint will automatically be applied.

We can configure the PrincipalKey in the following manner:

```
modelBuilder.Entity<Menu>()
.HasOne(x => x.Route)
.WithOne(x => x.Menu)
.HasPrincipalKey<Menu>(x => x.MenuCode)
.HasForeignKey<Route>(x => x.MenuCode);
```

Now with this, we are marking MenuCode of the Menu class as the principal key, which means that it will be the destiny of the one-to-one relationship, in other words, the link will be made between the Route's MenuCode and the Menu's MenuCode. From here we can add our migrations and proceed with the development of our application.

Summary

- With Entity Framework Core we can define relationships between our models
- We can use HasPrincipalKey to define with which field we want to link our foreign key