Previous Next

Configure One-to-One Relationships using Fluent API in Entity Framework Core

Here you will learn how to configure one-to-one relationships between two entities using Fluent API, if they do not follow EF Core conventions.

Generally, you don't need to configure one-to-one relationships manually because EF Core includes Conventions for One-to-One Relationships. However, if the key or foreign key properties do not follow the convention, then you can use data annotation attributes or Fluent API to configure a one-to-one relationship between the two entities.

Let's configure a one-to-one relationship between the following `Student` and `StudentAddress` entities, which do not follow the foreign key convention.

```
public class Student
{
    public int Id { get; set; }
    public string Name { get; set; }

    public StudentAddress Address { get; set; }
}

public class StudentAddress
{
    public int StudentAddressId { get; set; }
    public string Address { get; set; }
    public string City { get; set; }
    public string State { get; set; }
    public string Country { get; set; }

    public int AddressOfStudentId { get; set; }
    public Student Student { get; set; }
}
```

To configure a one-to-one relationship using Fluent API in EF Core, use the `HasOne`, `WithOne` and `HasForeignKey` methods, as shown below.

```
public class SchoolContext : DbContext
{
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
                    optionsBuilder.UseSqlServer("Server=.\\SQLEXPRESS;Database=EFCore-
SchoolDB;Trusted_Connection=True");
    }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Student>()
            .HasOne<StudentAddress>(s => s.Address)
            .WithOne(ad => ad.Student)
            .HasForeignKey<StudentAddress>(ad => ad.AddressOfStudentId);
    }

    public DbSet<Student> Students { get; set; }
    public DbSet<StudentAddress> StudentAddresses { get; set; }
}
```

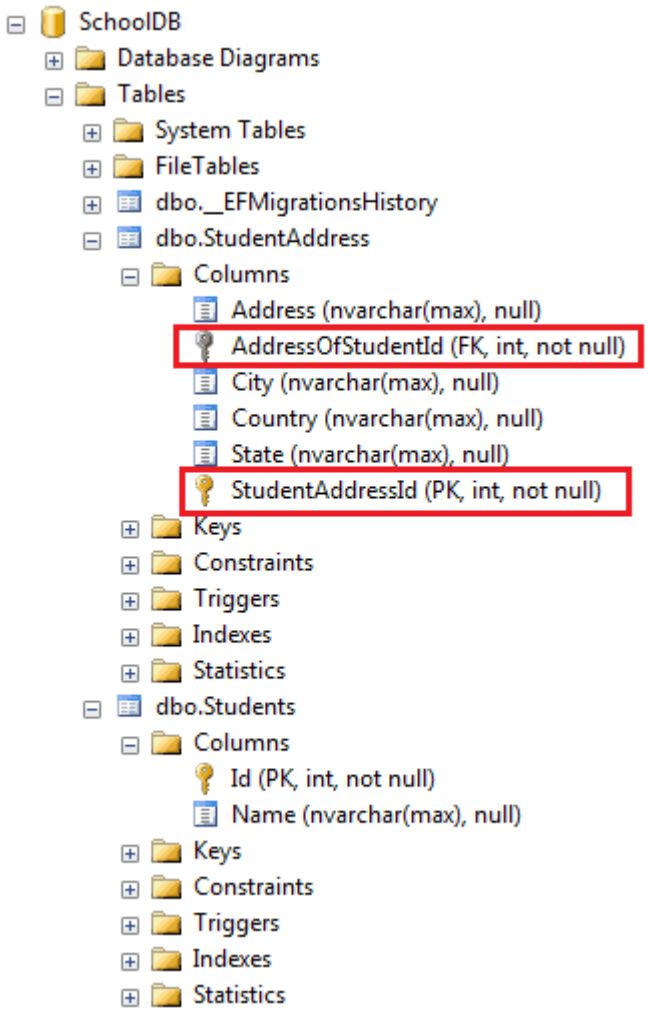
In the above example, the following code snippet configures the one-to-one relationship.

```
modelBuilder.Entity<Student>()
    .HasOne<StudentAddress>(s => s.Address)
    .WithOne(ad => ad.Student)
    .HasForeignKey<StudentAddress>(ad => ad.AddressOfStudentId);
```

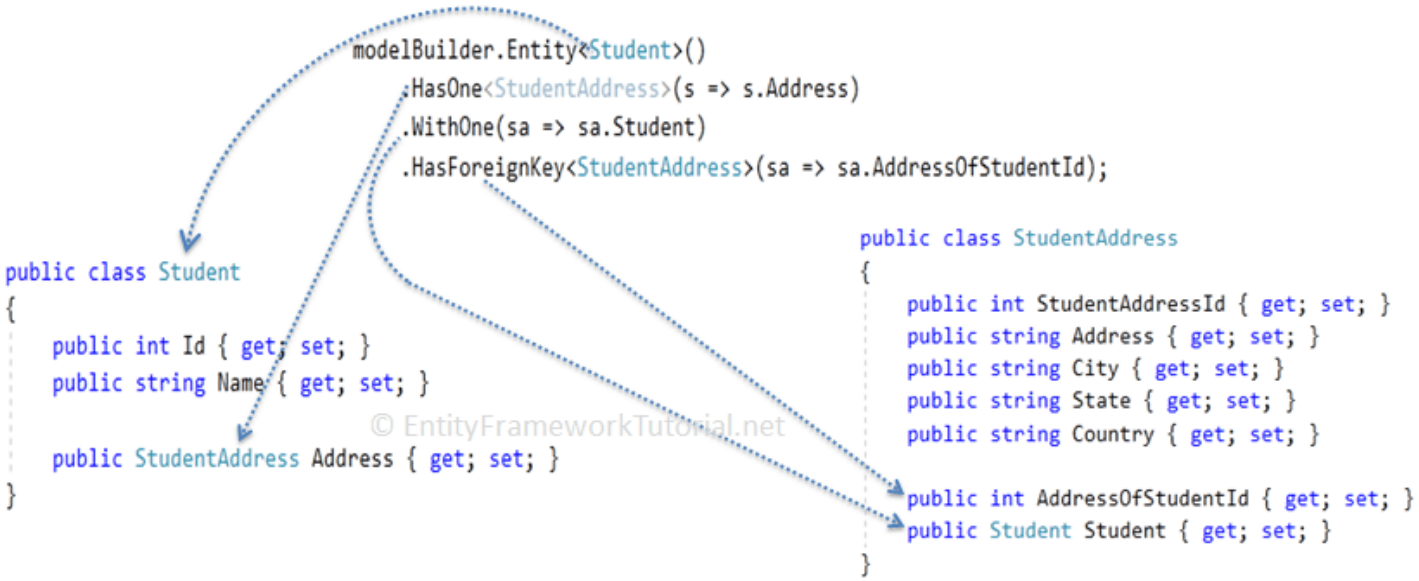
Let's understand it step by step.

- `modelBuilder.Entity<Student>()` starts configuring the `Student` entity.
- The `.HasOne<StudentAddress>(s => s.Address)` method specifies that the `Student` entity includes one `StudentAddress` reference property using a lambda expression.
- `.WithOne(ad => ad.Student)` configures the other end of the relationship, the `StudentAddress` entity. It specifies that the `StudentAddress` entity includes a reference navigation property of `Student` type.
- `.HasForeignKey<StudentAddress>(ad => ad.AddressOfStudentId)` specifies the foreign key property name.

Now, to reflect this in the database, execute migration commands, `add-migration <name>` and `update-database`. The database will include two tables with one-to-one relationship as shown below.



The following figure illustrates the Fluent API configuration for a one-to-one relationship.



You can start configuring with the `StudentAddress` entity in the same way, as below.

```
modelBuilder.Entity<StudentAddress>()
    .HasOne<Student>(ad => ad.Student)
    .WithOne(s => s.Address)
    .HasForeignKey<StudentAddress>(ad => ad.AddressOfStudentId);
```

Thus, you can configure a one-to-one relationship in entity framework core.

Previous Next

ENTITYFRAMEWORKTUTORIAL

Learn Entity Framework using simple yet practical examples on EntityFrameworkTutorial.net for free. Learn Entity Framework DB-First, Code-First and EF Core step by step. While using this site, you agree to have read and accepted our terms of use and privacy policy.

feedback@entityframeworktutorial.net

TUTORIALS

- EF Basics
- EF Core
- EF 6 DB-First
- EF 6 Code-First

E-MAIL LIST

Subscribe to EntityFrameworkTutorial email list and get EF 6 and EF Core Cheat Sheets, latest updates, tips & tricks about Entity Framework to your inbox.

Email address GO

We respect your privacy.

HOME PRIVACY POLICY ADVERTISE WITH US

© 2020 EntityFrameworkTutorial.net. All Rights Reserved.