عنوان: EF Code First #7
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۲۰ ۱۱:۰۱:۰۰
آدرس: www.dotnettips.info
گروه‌ها: Entity framework

مدیریت روابط بین جداول در EF Code first به کمک Fluent API

EF Code first بجای اتلاف وقت شما با نوشتن فایل‌های XML تهیه نگاشت‌ها یا تنظیم آن‌ها با کد، رویه Convention over configuration را پیشنهاد می‌دهد. همین رویه، جهت مدیریت روابط بین جداول نیز برقرار است. روابط one-to-one، one-to-many، many-to-many و موارد دیگر را بدون یک سطر تنظیم اضافی، صرفا بر اساس یک سری قراردادهای توکار می‌تواند تشخیص داده و اعمال کند. عموما زمانی نیاز به تنظیمات دستی وجود خواهد داشت که قراردادهای توکار رعایت نشوند و یا برای مثال قرار است با یک بانک اطلاعاتی قدیمی از پیش موجود کار کنیم.

مفاهیمی به نام‌های Principal و Dependent

در EF Code first از یک سری واژه‌های خاص جهت بیان ابتدا و انتهای روابط استفاده شده است که عدم آشنایی با آن‌ها درک خطاهای حاصل را مشکل می‌کند:
الف) Principal : طرفی از رابطه است که ابتدا در بانک اطلاعاتی ذخیره خواهد شد.
ب) Dependent : طرفی از رابطه است که پس از ثبت Principal در بانک اطلاعاتی ذخیره می‌شود.
Principal می‌تواند بدون نیاز به Dependent وجود داشته باشد. وجود Dependent بدون Principal ممکن نیست زیرا ارتباط بین این دو توسط یک کلید خارجی تعریف می‌شود.

کدهای مثال مدیریت روابط بین جداول

در دنیای واقعی، همه‌ی مثال‌ها به مدل بلاگ و مطالب آن ختم نمی‌شوند. به همین جهت نیاز است یک مدل نسبتا پیچیده‌تر را در اینجا بررسی کنیم. در ادامه کدهای کامل مثال جاری را مشاهده خواهید کرد:

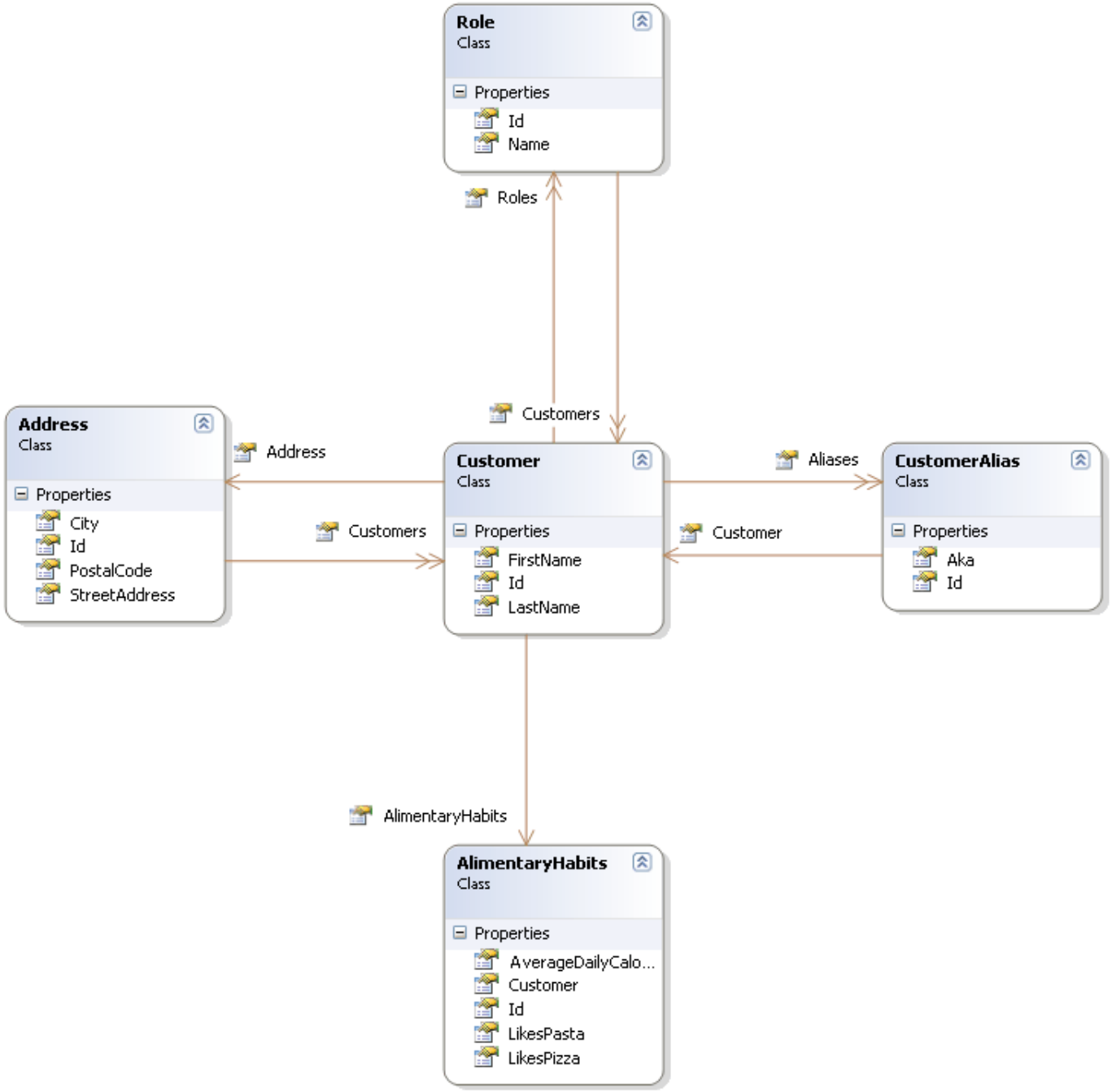
```
using System.Collections.Generic;

namespace EF_Sample35.Models
{
    public class Customer
    {
        public int Id { set; get; }
        public string FirstName { set; get; }
        public string LastName { set; get; }

        public virtual AlimentaryHabits AlimentaryHabits { set; get; }
        public virtual ICollection<CustomerAlias> Aliases { get; set; }
        public virtual ICollection<Role> Roles { get; set; }
        public virtual Address Address { get; set; }
    }
}
```

```
namespace EF_Sample35.Models
{
    public class CustomerAlias
    {
        public int Id { get; set; }
        public string Aka { get; set; }

        public virtual Customer Customer { get; set; }
    }
}
```

```
using System.Collections.Generic;

namespace EF_Sample35.Models
{
    public class Role
    {
        public int Id { set; get; }
        public string Name { set; get; }

        public virtual ICollection<Customer> Customers { set; get; }
    }
}
```

```
namespace EF_Sample35.Models
{
    public class AlimentaryHabits
    {
        public int Id { get; set; }
        public bool LikesPasta { get; set; }
        public bool LikesPizza { get; set; }
        public int AverageDailyCalories { get; set; }

        public virtual Customer Customer { get; set; }
    }
}
```

```
using System.Collections.Generic;

namespace EF_Sample35.Models
{
    public class Address
    {
        public int Id { set; get; }
        public  string City { set; get; }
        public  string StreetAddress { set; get; }
        public  string PostalCode { set; get; }

        public virtual ICollection<Customer> Customers { set; get; }
    }
}
```



همچنین **تعاریف نگاشت‌های برنامه** نیز مطابق کدهای زیر است:

```
using System.Data.Entity.ModelConfiguration;
using EF_Sample35.Models;

namespace EF_Sample35.Mappings
{
    public class CustomerAliasConfig : EntityTypeConfiguration<CustomerAlias>
    {
        public CustomerAliasConfig()
        {
            // one-to-many
            this.HasRequired(x => x.Customer)
                .WithMany(x => x.Aliases)
                .WillCascadeOnDelete();
```

```
        }
    }
}
```

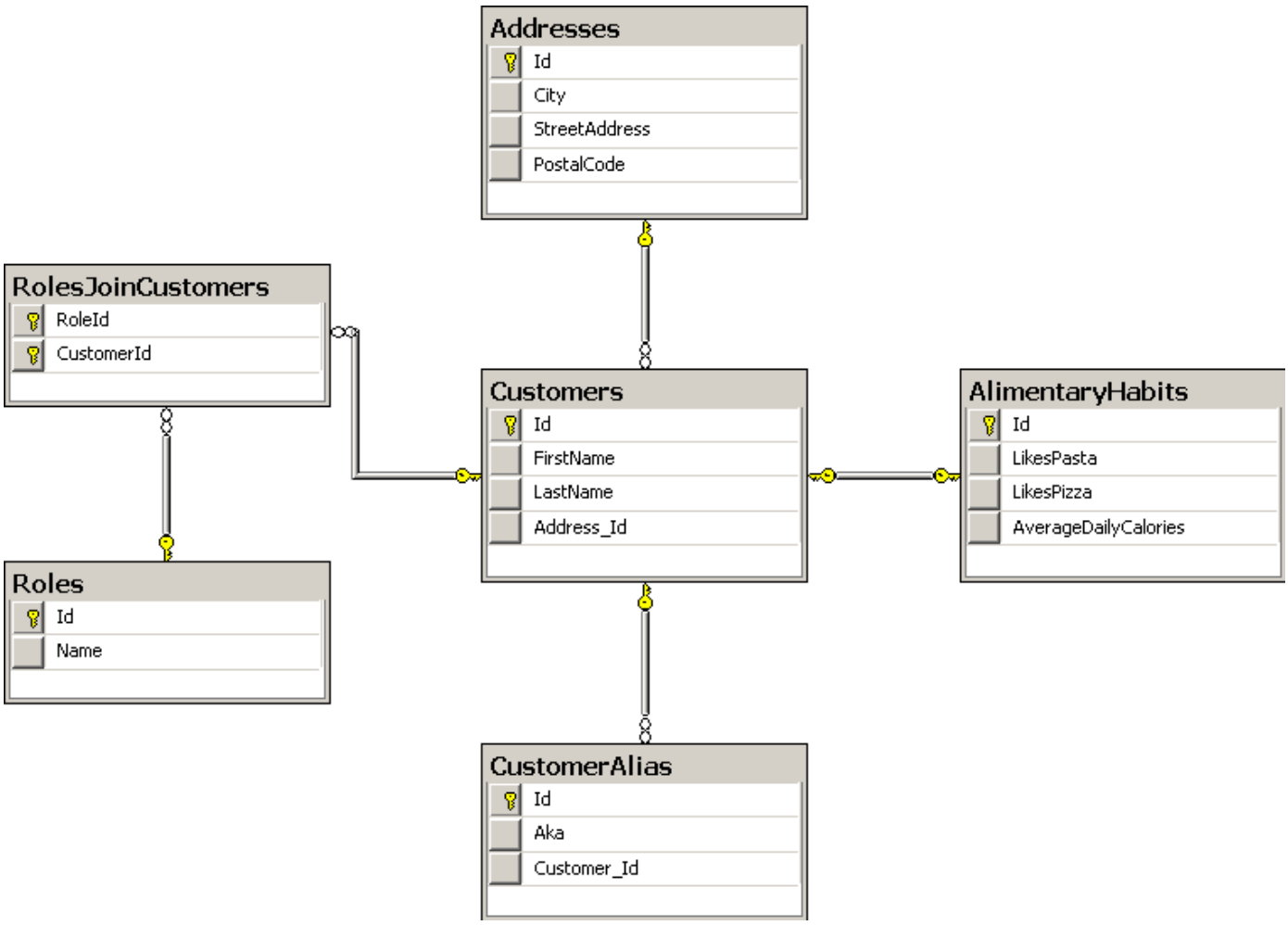
```
using System.Data.Entity.ModelConfiguration;
using EF_Sample35.Models;

namespace EF_Sample35.Mappings
{
    public class CustomerConfig : EntityTypeConfiguration<Customer>
    {
        public CustomerConfig()
        {
            // one-to-one
            this.HasOptional(x => x.AlimentaryHabits)
                .WithRequired(x => x.Customer)
                .WillCascadeOnDelete();

            // many-to-many
            this.HasMany(p => p.Roles)
                .WithMany(t => t.Customers)
                .Map(mc =>
                   {
                       mc.ToTable("RolesJoinCustomers");
                       mc.MapLeftKey("RoleId");
                       mc.MapRightKey("CustomerId");
                   });

            // many-to-one
            this.HasOptional(x => x.Address)
                .WithMany(x => x.Customers)
                .WillCascadeOnDelete();
        }
    }
}
```

به همراه **Context** زیر:

```
using System.Data.Entity;
using System.Data.Entity.Migrations;
using EF_Sample35.Mappings;
using EF_Sample35.Models;

namespace EF_Sample35.DataLayer
{
    public class Sample35Context : DbContext
    {
        public DbSet<AlimentaryHabits> AlimentaryHabits { set; get; }
        public DbSet<Customer> Customers { set; get; }

        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            modelBuilder.Configurations.Add(new CustomerConfig());
            modelBuilder.Configurations.Add(new CustomerAliasConfig());

            base.OnModelCreating(modelBuilder);
        }
    }

    public class Configuration : DbMigrationsConfiguration<Sample35Context>
    {
        public Configuration()
        {
            AutomaticMigrationsEnabled = true;
            AutomaticMigrationDataLossAllowed = true;
        }

        protected override void Seed(Sample35Context context)
        {
            base.Seed(context);
```

```
        }
    }
}
```

که نهایتا منجر به تولید چنین **ساختاری در بانک اطلاعاتی** می‌گردد:



توضیحات کامل کدهای فوق:

تنظیمات روابط one-to-one و یا one-to-zero

زمانیکه رابطه‌ای 0..1 و یا 1..1 است، مطابق قراردادهای توکار EF Code first تنها کافی است یک navigation property را که بیانگر ارجاعی است به شیء دیگر، تعریف کنیم (در هر دو طرف رابطه).
برای مثال در مدل‌های فوق یک مشتری که در حین ثبت اطلاعات اصلی او، «ممکن است» اطلاعات جانبی دیگری (AlimentaryHabits) نیز از او تنها در طی یک رکورد، دریافت شود. قصد هم نداریم یک ComplexType را تعریف کنیم. نیاز است جدول AlimentaryHabits جداگانه وجود داشته باشد.

```
namespace EF_Sample35.Models
{
    public class Customer
    {
        // ...
        public virtual AlimentaryHabits AlimentaryHabits { set; get; }
    }
}
```

```
namespace EF_Sample35.Models
{
    public class AlimentaryHabits
    {
        // ...
        public virtual Customer Customer { get; set; }
    }
}
```

در اینجا خواص virtual تعریف شده در دو طرف رابطه، به EF خواهد گفت که رابطه‌ای، 1:1 برقرار است. در این حالت اگر برنامه را اجرا کنیم، به خطای زیر برخواهیم خورد:

```
Unable to determine the principal end of an association between
the types 'EF_Sample35.Models.Customer' and 'EF_Sample35.Models.AlimentaryHabits'.
The principal end of this association must be explicitly configured using either
the relationship fluent API or data annotations.
```

EF تشخیص داده است که رابطه 1:1 برقرار است؛ اما با قاطعیت نمی‌تواند طرف Principal را تعیین کند. بنابراین باید اندکی به او کمک کرد:

```
using System.Data.Entity.ModelConfiguration;
using EF_Sample35.Models;

namespace EF_Sample35.Mappings
{
    public class CustomerConfig : EntityTypeConfiguration<Customer>
    {
        public CustomerConfig()
        {
            // one-to-one
            this.HasOptional(x => x.AlimentaryHabits)
                .WithRequired(x => x.Customer)
                .WillCascadeOnDelete();
        }
    }
}
```

همانطور که ملاحظه می‌کنید در اینجا توسط متد WithRequired طرف Principal و توسط متد HasOptional، طرف Dependent تعیین شده است. به این ترتیب EF می‌توان یک رابطه 1:1 را تشکیل دهید.
توسط متد WillCascadeOnDelete هم مشخص می‌کنیم که اگر Principal حذف شد، لطفا Dependent را به صورت خودکار حذف کن.

توضیحات ساختار جداول تشکیل شده:
هر دو جدول با همان خواص اصلی که در دو کلاس وجود دارند، تشکیل شده‌اند.

فیلد Id جدول AlimentaryHabits اینبار دیگر Identity نیست. اگر به تعریف قید FK_AlimentaryHabits_Customers_Id دقت کنیم، در اینجا مشخص است که فیلد Id جدول AlimentaryHabits، به فیلد Id جدول مشتری‌ها متصل شده است (یعنی در آن واحد هم primary key است و هم foreign key). به همین جهت به این روش one-to-one association with shared primary key هم گفته می‌شود (کلید اصلی جدول مشتری با جدول AlimentaryHabits به اشتراک گذاشته شده است).

تنظیمات روابط one-to-many

برای مثال همان مشتری فوق را درنظر بگیرید که دارای تعدادی نام مستعار است:

```
using System.Collections.Generic;

namespace EF_Sample35.Models
{
    public class Customer
    {
        // ...
        public virtual ICollection<CustomerAlias> Aliases { get; set; }
    }
}
```

```
namespace EF_Sample35.Models
{
    public class CustomerAlias
    {
        // ...
        public virtual Customer Customer { get; set; }
    }
}
```

همین میزان تنظیم کفایت می‌کند و نیازی به استفاده از Fluent API برای معرفی روابط نیست.
در طرف Principal، یک مجموعه یا لیستی از Dependent وجود دارد. در Dependent هم یک navigation property معرف طرف Principal اضافه شده است.
جدول CustomerAlias اضافه شده، توسط یک کلید خارجی به جدول مشتری مرتبط می‌شود.

سؤال: اگر در اینجا نیز بخواهیم CascadeOnDelete را اعمال کنیم، چه باید کرد؟
پاسخ: جهت سفارشی سازی نحوه تعاریف روابط حتما نیاز به استفاده از Fluent API به نحو زیر می‌باشد:

```
using System.Data.Entity.ModelConfiguration;
using EF_Sample35.Models;

namespace EF_Sample35.Mappings
{
    public class CustomerAliasConfig : EntityTypeConfiguration<CustomerAlias>
    {
        public CustomerAliasConfig()
        {
            // one-to-many
            this.HasRequired(x => x.Customer)
                .WithMany(x => x.Aliases)
                .WillCascadeOnDelete();
        }
    }
}
```

اینکار را باید در کلاس تنظیمات CustomerAlias انجام داد تا بتوان Principal را توسط متد HasRequired به Customer و سپس

dependent را به کمک متد WithMany مشخص کرد. در ادامه می‌توان متد WillCascadeOnDelete یا هر تنظیم سفارشی دیگری را نیز اعمال نمود.
متد HasRequired سبب خواهد شد فیلد Customer_Id، به صورت not null در سمت بانک اطلاعاتی تعریف شود؛ متد HasOptional عکس آن است.

تنظیمات روابط many-to-many

برای تنظیم روابط many-to-many تنها کافی است دو سر رابطه ارجاعاتی را به یکدیگر توسط یک لیست یا مجموعه داشته باشند:

```
using System.Collections.Generic;

namespace EF_Sample35.Models
{
    public class Role
    {
        // ...
        public virtual ICollection<Customer> Customers { set; get; }
    }
}
```

```
using System.Collections.Generic;

namespace EF_Sample35.Models
{
    public class Customer
    {
        // ...
        public virtual ICollection<Role> Roles { get; set; }
    }
}
```

همانطور که مشاهده می‌کنید، یک مشتری می‌تواند چندین نقش داشته باشد و هر نقش می‌تواند به چندین مشتری منتسب شود.
اگر برنامه را به این ترتیب اجرا کنیم، به صورت خودکار یک رابطه many-to-many تشکیل خواهد شد (بدون نیاز به تنظیمات نگاشت‌های آن). نکته جالب آن تشکیل خودکار جدول ارتباط دهنده واسط یا اصطلاحا join-table می‌باشد:

```
CREATE TABLE [dbo].[RolesJoinCustomers](
 [RoleId] [int] NOT NULL,
 [CustomerId] [int] NOT NULL,
)
```

سؤال: نام‌های خودکار استفاده شده را می‌خواهیم تغییر دهیم. چکار باید کرد؟
پاسخ: اگر بانک اطلاعاتی برای بار اول است که توسط این روش تولید می‌شود شاید این پیش فرض‌ها اهمیتی نداشته باشد و نسبتا هم مناسب هستند. اما اگر قرار باشد از یک بانک اطلاعاتی موجود که امکان تغییر نام فیلدها و جداول آن وجود ندارد استفاده کنیم، نیاز به سفارشی سازی تعاریف نگاشت‌ها به کمک Fluent API خواهیم داشت:

```
using System.Data.Entity.ModelConfiguration;
using EF_Sample35.Models;

namespace EF_Sample35.Mappings
{
    public class CustomerConfig : EntityTypeConfiguration<Customer>
    {
        public CustomerConfig()
        {
```

```
            // many-to-many
            this.HasMany(p => p.Roles)
                .WithMany(t => t.Customers)
                .Map(mc =>
                   {
                       mc.ToTable("RolesJoinCustomers");
                       mc.MapLeftKey("RoleId");
                       mc.MapRightKey("CustomerId");
                   });
        }
    }
}
```

تنظیمات روابط many-to-one

در تکمیل مدل‌های مثال جاری، به دو کلاس زیر خواهیم رسید. در اینجا تنها در کلاس مشتری است که ارجاعی به کلاس آدرس او وجود دارد. در کلاس آدرس، یک navigation property همانند حالت 1:1 تعریف نشده است:

```
namespace EF_Sample35.Models
{
    public class Address
    {
        public int Id { set; get; }
        public  string City { set; get; }
        public  string StreetAddress { set; get; }
        public  string PostalCode { set; get; }
    }
}
```

```
using System.Collections.Generic;

namespace EF_Sample35.Models
{
    public class Customer
    {
       // …
        public virtual Address Address { get; set; }
    }
}
```

این رابطه توسط EF Code first به صورت خودکار به یک رابطه many-to-one تفسیر خواهد شد و نیازی به تنظیمات خاصی ندارد. زمانیکه جداول برنامه تشکیل شوند، جدول Addresses موجودیتی مستقل خواهد داشت و جدول مشتری با یک فیلد به نام Address_Id به جدول آدرس‌ها متصل می‌گردد. این فیلد نال پذیر است؛ به عبارتی ذکر آدرس مشتری الزامی نیست.
اگر نیاز بود این تعاریف نیز توسط Fluent API سفارشی شوند، باید خاصیت public virtual ICollection<Customer> Customers به کلاس Address نیز اضافه شود تا بتوان رابطه زیر را توسط کدهای برنامه تعریف کرد:

```
using System.Data.Entity.ModelConfiguration;
using EF_Sample35.Models;

namespace EF_Sample35.Mappings
{
    public class CustomerConfig : EntityTypeConfiguration<Customer>
    {
        public CustomerConfig()
        {
            // many-to-one
            this.HasOptional(x => x.Address)
                .WithMany(x => x.Customers)
```

```
                .WillCascadeOnDelete();
        }
    }
}
```

متد HasOptional سبب می‌شود تا فیلد Address_Id اضافه شده به جدول مشتری‌ها، null پذیر شود.

نظرات خوانندگان

نویسنده: MehdiPayervand
تاریخ: ۱۳۹۱/۰۲/۲۰ ۱۶:۵۳:۵۲

سلام جناب مهندس، تشکر از لطفتون، اگه ممکنه کدهای این سری رو هم توی کدپلس آپ کنین. ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۲۰ ۲۰:۰۵:۳۲

سلام، از اینجا می‌تونید دریافت کنید: (^)

نویسنده: amir
تاریخ: ۱۳۹۱/۰۲/۲۰ ۲۲:۳۹:۰۴

سلام خیلی ممنون بابت مطالب بسیار مفیدتون. اگر ممکنه نحوه ارتباط یک Sql View رو با EF Code First هم توضیح بدید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۲۱ ۰۰:۳۶:۵۳

استفاده از View نکته خاص و اضافه‌تری نداره؛ از این لحاظ که عموما به Viewها به شکل یک جدول فقط خواندنی نگاه می‌شود. بنابراین یک کلاس تعریف کنید حاوی فیلدهای همان View. بعد هم یک data annotations برای مثال Table را بالای این کلاس قرار دهید (اگر نیاز بود از نام خاصی که جزو اصول نامگذاری کلاس‌ها در سی شارپ نیست استفاده کند).

نویسنده: amir
تاریخ: ۱۳۹۱/۰۲/۲۱ ۱۳:۲۰:۵۶

سلام دیتابیس من به صورت خلاصه از دو تا جدول تشکیل شده یکی مشتریان (Customers) و یکی هم دریافت و پرداخت ها (Payments). حالا میخوام وقتی یک مشتری حذف میشه کل دریافت پرداخت هاش هم حذف بشه. از کد زیر استفاده کردم ولی نمیدونم چرا تو دیتابیس یه فیلد اضافی Customer_ID رو به جدول دریافت پرداخت هام اضافه میکنه در حالی که من خودم یه فیلد CustomerID گذاشته بودم!

```
modelBuilder.Entity()
.HasRequired(s => s.Customer)
.WithMany()
.WillCascadeOnDelete();
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۲۱ ۱۳:۳۵:۱۲

لطف تعریف دقیق کلاس‌های مدلتون رو اینجا قرار بدید و لینک بدید: pastebin.com

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۲۱ ۱۳:۵۴:۴۷

ضمنا در قسمت هشتم (قسمت بعدی)، در مورد Self Referencing Entity بحث شده. نگاشت شما خیلی شبیه به آن است. در کل نیاز است تعریف دقیق مدل‌ها و روابط تعریف شده رو دید.

نویسنده: amir
تاریخ: ۱۳۹۱/۰۲/۲۱ ۱۵:۱۵:۳۸

اینقدر باهاش ور رفتم تا درست شد (با استفاده از متد HasForeignKey). الان مشکلی که دارم در مورد Cascade Update هستش.
اگر اینم تو قسمت های بعدی توضیح بدید عالی میشه.

فقط یه خواهش دیگه که داشتم (البته مرتبط با این موضوع نیست)، اینکه اگر وقت کردید در مورد WPF و MVVM مطالب بیشتری بذارید چون این دو موضوع الان خیلی طرفدار دارن ولی یه خورده سخت هستن!.

من روزی چند بار به سایت شما میام و از مطالب مفیدتون استفاده میکنم. واقعاً ازتون ممنونم.

نویسنده: محمد شهریاری
تاریخ: ۱۳۹۱/۰۵/۱۷ ۱۷:۳۷

د صورتی که یک رابطه many-to - many داشته باشیم و ابتدا مثل رابطه Role , Customer ذخیره سازی ارتباط این جداول به چه صورت است ؟
در حالت عادی برای هر ذخیره سازی هم اطلاعات Role و هم Customer ذخیره می‌گردد .
آیا باید یه Entity واسط هم در نظر بگیریم و پس از ثبت اطلاعات مثلا Customer با ID مربوط به Role از طریق Entity واسط اطلاعات ارتباط این Entity 2 ذخیره شود ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۱۷ ۱۹:۵۸

خیر. مدیریت این جدول واسط کاملا خودکار است.

نویسنده: محمد شهریاری
تاریخ: ۱۳۹۱/۰۵/۲۲ ۱۵:۳

با تشکر از پاسخ دهی شما به سوالات؛ موقع Create درست اعمال می‌شود، اما هنگام Edit جدول واسط به روز نمی‌گردد.
مثلا برای دو جدول User , Role که نقش‌های یک کاربر بوسیله یک string[] به اکشن Edit پاس داده شده
کد مربوطه به صورت زیر می‌باشد

```
[HttpPost]
        public ActionResult Edit(User user, string[] tags)
        {
            if (ModelState.IsValid)
            {
                List<Role> roles = new List<Role>();
                foreach (var item in tags)
                {
                    Role role = db.Roles.Find(long.Parse(item));
                    roles.Add(role);
                }
                user.Roles = roles;

                db.Entry(user).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(user);
        }
```

اما جدول واسط در این قسمت به روز نمی‌شود . متاسفانه چیز خاصی در این رابطه پیدا نکردم و مجددا مزاحم شما شدم .

با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۲۲ ۱۶:۳۴

قبل از ویرایش این سطر را اضافه کنید

```
if(user.Roles != null && user.Roles.Any())
   user.Roles.Clear();
```

همچنین اگر itemهای دریافتی primary key هستند می‌تونید از متد attach بجای مراجعه به بانک اطلاعاتی، استفاده کنید:

```
ctx.Roles.Attach(new Role { Id = item });
```

نویسنده: محمد شهریاری
تاریخ: ۱۳۹۱/۰۵/۲۲ ۲۳:۲۸

سلام
در action مربوط به Edit که در بالا امده است فیلد Roles برابر null می‌باشد . دلیل این رو نمی‌دانم شاید مشکل از bind فرم هست اما entity که در متد Get مقدار دهی میشه Roles مقدار دارد .
در مورد attach میشه توضیح بدید . البته با این هم نتونستم کاری انجام بدم .
در اخر ناچار شدم ابتدا User رو یک بار find کنم و سپس با مقدار مدل مقدار دهی کنم به نظر خودم که کار درستی نیست .
خوشحال میشم که با راه حل اساسی آشنا بشم .
امکانش هست .

کدها رو به این صورت تغییر دادم که مشکلم برطرف شد ولی فکر میکنم که بدون find هم باید راه حلی باشه که بلد نیستم

```
[HttpPost]
        public ActionResult Edit(User user, string[] tags)
        {
            User Currentuser = db.Users.Find(user.Id);

            Currentuser.FirstName = user.FirstName;
            Currentuser.LastName = user.LastName;
            if (ModelState.IsValid)
            {
                List<Role> roles = new List<Role>();
                if (Currentuser.Roles != null && Currentuser.Roles.Any())
                    Currentuser.Roles.Clear();
                foreach (var item in tags)
                {
                    Role role = db.Roles.Find(long.Parse(item));
                    roles.Add(role);
                }
                Currentuser.Roles = roles;

                db.Entry(Currentuser).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(Currentuser);
        }
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۲۳ ۰:۳۷

اگر با attach جواب نگرفتید، لیست نقش‌ها رو با یک رفت و برگشت هم می‌توان بدست آورد:

```
var listOfActualRoles = db.Roles.Where(x => tags.Contains(x.Id)).ToList();
```

نویسنده: alireza
تاریخ: ۱۳۹۱/۱۰/۲۱ ۹:۰

سلام وحید جان ممنون از این همه لطف

من یک پروژه را با CodeFirst شروع کردم اما یه جایی اشتباه کردم فکر کنم اشتباهم توی یکی از Mappingها باشه. اگه لطف کنید ببینید مشکل چیه.بدون استفاده از Mapping مشکلی نیست و دیتا بیس با روابطی که میخوام ایجاد میشه اما وقتی از Mapping استفاده میکنم با این خطا مواجه میشم:

```
{"Sequence contains more than one matching element"}
```

چندتا کلاس‌ها به شکل زیر هست:

```
public class Driver
    {
        public int Id { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string NationalCode { get; set; }
        public string CellPhone { get; set; }
        public string LicenseNumber { get; set; }
        public bool IsDriverAssistance { get; set; }

        [InverseProperty("Driver")]
        public virtual ICollection<Transference> Transferences { get; set; }
        [InverseProperty("DriverAssistance")]
        public virtual ICollection<Transference> TransferencesForAssistance { get; set; }
        [InverseProperty("Driver")]
        public virtual ICollection<Tanker> Tankers { get; set; }
        [InverseProperty("DriverAssistance")]
        public virtual ICollection<Tanker> TankersForAssistance { get; set; }

    }
```

```
public class Transference
    {
        public string Id { get; set; }
        public DateTime Date { get; set; }
        public Int16 Lytrazh { get; set; }
        public bool IsEMS { get; set; }
        public DateTime LoadingDate { get; set; }
        public DateTime DeliveryDate { get; set; }
        [InverseProperty("Transferences")]
        public virtual Driver Driver { get; set; }
        [InverseProperty("TransferencesForAssistance")]
        public virtual Driver DriverAssistance { get; set; }
        public virtual TypeOfTanker TypesOfTanker { get; set; }
        public virtual Tanker Tanker { get; set; }
        public virtual Consumer Consumer { get; set; }

    }
```

فکر کنم مشکل از این کلاس زیر باشه:

```
public class TransferenceConfig : EntityTypeConfiguration<Transference>
    {
        public TransferenceConfig()
        {
            // one-to-many
            this.HasRequired(x => x.Consumer)
                .WithMany(x => x.Transferences);

                // one-to-many
            this.HasRequired(x => x.TypesOfTanker)
                .WithMany(x => x.Transferences);

            // one-to-many
            this.HasRequired(x => x.Tanker)
                .WithMany(x => x.Transferences);
```

```
            // one-to-many
            this.HasRequired(x => x.Driver)
                .WithMany(x => x.Transferences);

            // one-to-many
            this.HasRequired(x => x.DriverAssistance)
                .WithMany(x => x.Transferences);

        }
    }
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۲۱ ۱۸:۲۳

- مشکل کلاس کانفیگ فوق در این است که از یک طرف InverseProperty تعریف کردید، از طرف دیگر در حالت تنظیمات Fluent، این مورد رعایت نشده. مثلا DriverAssistance باید به TransferencesForAssistance (مطابق InverseProperty تعریف شده) مرتبط می‌شد و الی آخر (الان همگی به یک مورد مرتبط شدن).
- در کل نیازی به کلاس کانفیگ فوق ندارید. حذفش کنید. EF می‌تونه روابط one-to-many رو بدون کانفیگ خاصی تشخیص بده. علت وجود قسمت هفتم، اعمال یک سری تنظیمات اضافه‌تر است نسبت به تنظیمات پیش فرض. مثلا اگر از نام‌های پیش فرض خرسند نیستید، اینجا می‌تونید توسط Fluent API خیلی از این موارد رو سفارشی سازی کنید و تغییر بدید. البته شرطش هم این است که از ICollection برای معرفی موارد one-to-many استفاده کنید (که اینکار در کلاس Driver انجام شده، همچنین یک سر دیگر آن به صورت virtual در کلاس مقابل وجود دارد. به علاوه مطلب نحوه تعریف صحیح کلیدهای خارجی را هم اضافه کنید تا طراحی بهتری داشته باشید).

نویسنده: علیرضا پایدار
تاریخ: ۱۳۹۱/۱۰/۲۱ ۲۱:۲۷

ممنون از پاسخ.
درست می‌فرمایید من میتونستم کلاس کانفیگ را حذف کنم اما میخواستم با کانفیگ تست کنم که نتونستم.
البته تنظیمات اضافه هم قراره وقتی این مشکل رفع شد اضافه نمایم مثل تنظیم حداکثر طول فیلد،یا عنوان مناسب برای کلید خارجی و NOT NULL از این جور تنظیمات که خودتون توی مطالب قبلی ارائه نمودید.
فرمودید:"- مشکل کلاس کانفیگ فوق در این است که از یک طرف InverseProperty تعریف کردید، از طرف دیگر در حالت تنظیمات Fluent، این مورد رعایت نشده. مثلا DriverAssistance باید به TransferencesForAssistance (مطابق InverseProperty تعریف شده) مرتبط می‌شد و الی آخر (الان همگی به یک مورد مرتبط شدن). "
منظور اینه به شکل زیر تبدیل بشه:

```
public class TransferenceConfig : EntityTypeConfiguration<Transference>
   {
       public TransferenceConfig()
       {
           // one-to-many
           this.HasRequired(x => x.Consumer)
               .WithMany(x => x.Transferences);

               // one-to-many
           this.HasRequired(x => x.TypesOfTanker)
               .WithMany(x => x.Transferences);

           // one-to-many
           this.HasRequired(x => x.Tanker)
               .WithMany(x => x.Transferences);

           // one-to-many
           this.HasRequired(x => x.Driver)
               .WithMany(x => x.Transferences);
 //-------------------------------------------------------------------------------------
           // one-to-many
           this.HasRequired(x => x.DriverAssistance)
               .WithMany(x => x.TransferencesForAssistance);
//-------------------------------------------------------------------------------------

       }
   }
```

بخشی که بین 2 تا خط نقطه چین قرار گرفته.بله؟
بجای اینکه از InverseProperty <-- Attribute بشه آیا معادلی توی Fluent API داره؟

بازم ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۲۱ ۲۱:۳۳

زمانیکه از Fluent API استفاده می‌کنید نیازی به ذکر Attributes ندارید؛ چون طرفین یک ارتباط و ریز مشخصات آن‌ها با کدنویسی دقیقا (و بدون هیچگونه قرارداد خاصی) مشخص می‌شوند. یک نمونه رو در مثال شما عنوان کردم، مثلا DriverAssistance از یک کلاس به TransferencesForAssistance کلاس دیگر مرتبط شده. به این ترتیب نیازی به ذکر ویژگی خاصی برای مشخص سازی مجدد آن نیست (چون دیگر جای حدس و گمان و پیش‌فرضی باقی نمی‌ماند. صریحا تنظیمات مشخص شده‌اند). برای سایر موارد هم باید به همین ترتیب عمل کنید.

نویسنده: مسعود
تاریخ: ۱۳۹۱/۱۱/۰۲ ۹:۲۱

سلام
مدیریت روابط n-n در کلاسهای Poco به چه صورت است؟ من یک برنامه tier-2 دارم و برای Entity هایم state گرفتم و سمت کلاینت وضعیت entityها رو مدیریت میکنم و سمت سرور اونا رو روی DBContext اعمال میکنم.
دو کلاس Role و Permission دارم که باهم رابطه n-n دارند؛ حال اگه یک Role چند Permission داشته باشه و بخوام یکی از اونا رو حذف کنم و یا آپدیت کنم بهم پیغام خطا میده(An error occurred while saving entities that do not expose foreign key properties for their relationships. The EntityEntries property will return null because a single e ...) و داخل inner exception هم مینویسه (Violation of PRIMARY KEY constraint 'PK_dbo.SecurityRolePermission'. Cannot insert duplicate key in object 'dbo.SecurityRolePermission'
یعنی در حالتهای آپدیت و حذف هم میخواد insert انجام بده (با ef profiler هم چک کردم)البته فکر کنم تا حدودی معلوم باشه چرا اینکار رو میکنه؛ چون مدیریتی روی state اون کلاس واسط(RolePermission) انجام نمیشه، ممکنه بگید چطوری این مشکل رفع میشه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۰۲ ۱۳:۳۰

در مطلب « کار با کلیدهای اصلی و خارجی در EF Code first » توضیح دادم چرا و در چه صورتی رکورد تکراری تولید میشه.

نویسنده: سارا زرمهر
تاریخ: ۱۳۹۱/۱۱/۰۳ ۱۲:۱

سلام
توی کلاس Context ما کدام موجودیت‌ها در Dbset میگزاریم؟ در این جا چرا شما فقط AlimentaryHabits وCustomer رو توی Context گذاشتید؟

```
namespace EF_Sample35.DataLayer
{
 public class Sample35Context : DbContext
 {
 public DbSet<AlimentaryHabits> AlimentaryHabits { set; get; }
 public DbSet<Customer> Customers { set; get; }
 ...
 ...
 }
}
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۰۳ ۱۲:۱۷

- قرار دادن تمام کلاس‌های شرکت کننده در تشکیل جداول، حالت پیش فرض و معمول است. از این جهت که برای ثبت اطلاعات جداگانه در هر کدام نیاز به DbSet متناظر خواهد بود.
+ EF توانایی یافتن روابط و تشکیل جداول متناظر را بر اساس روابط بین کلاس‌ها، دارا است. اگر به تصویر اسکیمای حاصل دقت کنید این مساله مشهود است.
- در کل در این «مثال» ذکر دو مورد جهت برآوردن مقصود توضیحات داده شده کافی بوده.

نویسنده: حسین غلامی
تاریخ: ۱۳۹۱/۱۱/۱۹ ۱۷:۴۲

سلام؛
در SQL SERVER قسمتی داریم به نام INSERT And UPDATE Specification که دو گزینه‌ی Delete Rule و Update Rule برای ارتباط‌ها وجود دارند.
در اینجا شما WillCascadeOnDelete رو بیان کردید ولی من هر چه دنبال متدی متناظر با Update Rule میگردم وجود نداره.
یه چیزی مثل : WillCascadeOnUpdate
آیا در EF ساپورت نمیشه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۱۹ ۱۸:۱۷

خیر. ON UPDATE CASCADE عموما به معنای به روز رسانی primary key است که در جداول دیگر ارجاع دارد و از دیدگاه EF یک کلید اصلی، read only است.
به نظر من کار درستی انجام دادن. زمانیکه نیاز به به‌روز رسانی primary key دارید یعنی طراحی بانک اطلاعاتی شما یا نرمال نیست یا مشکل داره.
البته می‌تونید رویه ذخیره شده درست کنید برای اینکارها و دستی مسایل رو به زور اعمال کنید ولی به صورت پیش فرض خارج از سیستم Tracking آن که به صورت خودکار اطلاعات اشیاء مرتبط را به روز می‌کند، Cascade Update دیگری وجود ندارد.

نویسنده: مرتضی
تاریخ: ۱۳۹۱/۱۱/۲۱ ۱۸:۳۲

سلام

من هم با نظر شما در مورد بروز رسانی کلید اصلی موافقم.

اما واقعیت متفاوت تره.

من تو یه نرم افزار شماره پرسنلی که نوعش هم عدد صحیح بود کردم کلید اصلی.

بعد از 6 ماه متوجه شدن که شماره پرسنلی یکی از کارمندان رو اشتباه وارد کردن!

البته من از ef database first استفاده میکردم که اونجا هم مثل بقیه ormها اجازه update کردن کلید اصلی رو نمیده.

نویسنده: سعید

تاریخ: ۱۳۹۱/۱۱/۲۱ ۱۹:۴۹

همه این‌ها به طراحی بر می‌گرده. می‌تونستید شماره پرسنلی رو به صورت unique تعریف کنید و کلید اصلی رو یک فیلد auto increment، تا مشکل نداشته باشید. مثل آدرس ایمیل کاربران در یک بانک اطلاعاتی. این آدرس باید منحصربفرد باشه به ازای یک کاربر در سایت. یک کاربر باز هم می‌تونه از این نوع فیلدهای unique داشته باشه در یک جدول. مثل نام کاربر و یا مثل کد ملی.

نویسنده: مرتضی
تاریخ: ۱۳۹۱/۱۱/۲۱ ۲۰:۱۷

سعید جان میدونم که اینکار میشه- مطمئنا شماره ملیشون رو uniqe کردم!

ما برای انتخاب کلید اصلی دو حالت داریم -

1- استفاده از کلیدهای طبیعی مثل شماره پرسنلی

2- استفاده از کلیدهای جانشین مثل یک فیلد identity - این حالت موقعی استفاده میشه که کلید طبیعی نداشته باشیم

نویسنده: سعید
تاریخ: ۱۳۹۱/۱۱/۲۱ ۲۰:۲۸

زمانیکه شماره پرسنلی رو تبدیل به کلید اصلی می‌کنید یعنی تکرار داده در جداول مختلفی که به آن ارتباط پیدا می‌کنند و نیاز به آپدیت تمام جداول مرتبط با تغییر حتی یک نقطه در آن. به نظر شما این نوع دیتابیس نرمال شده است؟

نویسنده: مرتضی
تاریخ: ۱۳۹۱/۱۱/۲۱ ۲۱:۱۴

تکرار داده در جداول مختلفی که به آن ارتباط پیدا می‌کنند ??

تکرار چه داده ای؟

سعید جان مطمئنا کلید شما تو جداول برای رابطه استفاده میشه حالا چه طبیعی باشه چه جانشین.

با فرض اینکه با 10 جدول ارتباط داشته باشه با n تعداد رکورد . حالا 6 ماهی یک بار این اتفاق چه ایرادی داره؟

بگذریم . موفق باشید

نویسنده: سعید
تاریخ: ۱۳۹۱/۱۱/۲۱ ۲۲:۲۵

- تکرار داده‌ای که قرار هست ویرایش بشه. فرض کن که یک لینک از کارمند مورد نظر ایجاد شده با شماره پرسنلی که pk است. الان این لینک یاد داشت شده دیگه کار نمی‌کنه. این یک مثال ساده است. یا به روز رسانی تمام رکوردهای یک جدول با یک update که تریگر هم روش تعریف شده ممکنه مشکل ساز بشه یا اصلا کار نکنه.

- در ef هر موجودیت نیاز به یک کلید منحصربفرد داره برای شناسایی و از این کلید در سیستم ردیابی خودش استفاده می‌کنه. اگر این کلید قرار باشه تغییر کنه، ef نیاز داره تا یک وهله جدید رو خلق کنه و نمونه قبلی رو نابود. به این ترتیب عملکردش بهم می‌خوره و مجبور میشه رکورد جدید ثبت کنه بجای آپدیت قبلی. البته این کارها هم بدعتی نیست چون طراحی اون برای اساس

اصول domain driven design انجام شده.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۲۲ ۱۵:۱

اگر نیاز به یک مثال تکمیلی مشابه دیگر دارید که اکثر روابط در آن مطرح شده باشد به مطلب زیر مراجعه کنید:

Creating a More Complex Data Model for an ASP.NET MVC Application

نویسنده: مسعود2
تاریخ: ۱۳۹۱/۱۱/۲۲ ۱۵:۶

ممنون ولی متأسفانه اونجا هم این رابطه (1..0 - 1..0) رو نگفته.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۲۲ ۱۷:۱۹

- در مثال قسمت هفتم، رابطه مشتری با عادت‌های آن «one-to-zero- **or** -one» است. یک مشتری رو میشه تعریف کرد، صرفنظر از عادت‌های ویژه او؛ البته نه برعکس.
- در مثال سایت MVC ، رابطه دپارتمان و ادمین آن هم «one-to-zero-or-one» است. به این نحو هم تعریف شده

```
modelBuilder.Entity<Department>()
    .HasOptional(x => x.Administrator);
```

بدون اضافه کردن قسمت WithRequired و با داشتن یک Id نال پذیر در کلاس دپارتمان:

```
public int? InstructorID { get; set; }
```

اگر به تعاریف آن دقت کنید، کلاس Instructor رابطه‌ای با دپارتمان نداره اما رابطه دپارتمان با ادمین آن که از نوع Instructor است نال پذیر تعریف شده تا رابطه «یک به صفر یا یک» میسر شود.
- رابطه کلاس‌های Instructor و OfficeAssignment مثال سایت MVC مانند مثال قسمت هفتم فوق است و متد WithRequired رو ذکر کرده.

نویسنده: مسعود2
تاریخ: ۱۳۹۱/۱۱/۲۲ ۱۸:۳۱

یعنی میفرمایید من چه رابطه ای بین این دو کلاس تعریف کنم تا رابطه 1..0-1..0 داشته باشم؟

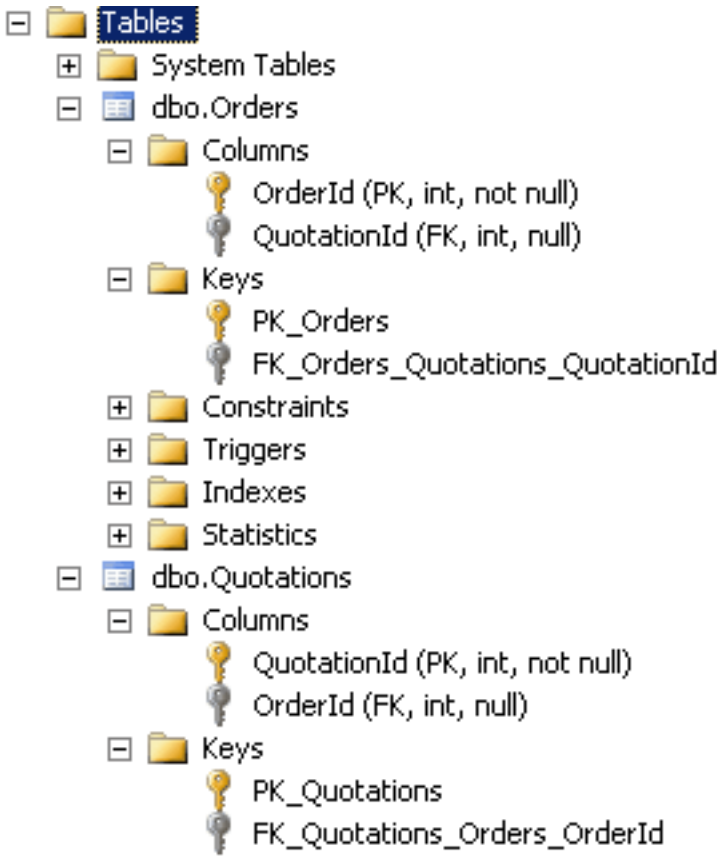
```
public class Order
    {
        public int OrderId { get; set; }
        public virtual Quotation Quotation { get; set; }
    }
    public class Quotation
    {
        public int QuotationId { get; set; }
        public virtual Order Order { get; set; }
    }
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۲۲ ۲۱:۴۴

با این تنظیمات

```
public class OrderMap : EntityTypeConfiguration<Order>
    {
        public OrderMap()
        {
            this.HasOptional(x => x.Quotation)
                .WithOptionalPrincipal()
                .Map(x => x.MapKey("OrderId"));
        }
    }

    public class QuotationMap : EntityTypeConfiguration<Quotation>
    {
        public QuotationMap()
        {
            this.HasOptional(x => x.Order)
                .WithOptionalPrincipal()
                .Map(x => x.MapKey("QuotationId"));
        }
    }
```

با این خروجی



نویسنده: مسعود2
تاریخ: ۱۳۹۱/۱۱/۲۳ ۲۲:۲۶

ممنون، جواب داد.

نویسنده: سعید یزدانی
تاریخ: ۱۳۹۱/۱۱/۲۷ ۱۹:۱۳

چرا برای مشخص کردن Principal، همیشه در کلاس maping جدولی که در رابطه Dependent است باید اقدام کنیم . ایا قانون تو کار خود ef است ؟

نویسنده: میهمان
تاریخ: ۱۳۹۱/۱۱/۳۰ ۱۷:۵۹

با سلام
اگر ما این چنین ساختار را داشته باشیم مانند مثال ذکر شده در این پست ، در مورد قسمت زیر

```
public CustomerConfig()
        {
            // many-to-many
            this.HasMany(p => p.Roles)
                .WithMany(t => t.Customers)
                .Map(mc =>
                   {
                       mc.ToTable("RolesJoinCustomers");
                       mc.MapLeftKey("RoleId");
                       mc.MapRightKey("CustomerId");
                   });
        }
}
```

با حذف یک Customer و یا Role ، رکوردهای مربوط به آن در جدول RolesJoinCustomers به صورت خود کار حذف می‌شود؟ اگر نه ، راه حل چیست ؟ چون من برای حذف رکورد ، با error برخورد میکنم
مورد دوم : اگر یک customer به عنوان مثال با تعدادی customer دیگر در ارتباط باشد(با تعدادی customer نه با یک customer) در این صورت چگونه باید پیاده سازی شود ؟ در صورت یک Customer ، رکوردهای مربوط به آن به صورت خود کار حذف می‌شود؟ اگر نه ، راه حل چیست ؟
با تشکر فراوان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۰۴ ۹:۴۶

در مطلب « بررسی تفصیلی روابط many-to-many » بیشتر توضیح دادم.

نویسنده: نوید
تاریخ: ۱۳۹۱/۱۲/۰۷ ۱۰:۰

سلام؛ اگر ما جداولی داشته باشیم که ارتباط many-to-many داشته باشند مثل کالا و انبار و بخواهیم که یک اطلاعات اضافی ای در جدول واسط آنها ذخیره شود (مثلا تعداد کالاها در هر انبار) . در این شرایط باید جدول واسط را خودمان ایجاذ کنیم یا این کار را نیز میتوان توسط EF-Code First انجام داد.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۰۷ ۱۰:۲۲

- در مطلب « بررسی تفصیلی روابط many-to-many » توضیح داده شده.
- تعداد ستون‌های جدول واسط خارج از دسترس شما است و توسط EF مدیریت می‌شود.
- تعداد کالا در هر انبار را یا کوئری بگیرید که روش انجام کار در انتهای مطلب یاد شده با مثال عنوان شده یا اینکه یک فیلد محاسبه شده در جدول انبار ایجاد کنید و نه در جدول واسط.
تعریف این فیلد اضافی در جدول واسط بی‌معنا است؛ چون در این جدول، در هر سطر، رابطه بین یک کالا و یک انبار ذخیره می‌شود. بنابراین نگهداری تعداد کل کالاهای یک انبار خارج از مسئولیت یک ردیف این جدول واسط است.

نویسنده: dream
تاریخ: ۱۳۹۱/۱۲/۱۸ ۱۲:۳

با سلام؛
من یه مشکلی داشتم ، می‌خواستم بدونم برای اینکه بتونیم بین سه جدول ارتباط many-to-many داشته باشیم چه جوری باید پیاده سازی کنیم،
مثلا کلاس "دانشجو"و"درس" و استاد" که ID هر کدوم توی یه جدول واسط قرار بگیره مثل ارتباط Many-to-many بین دو جدول
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۱۸ ۱۲:۳۴

- نیازی به رابطه many-to-many در تمام حالات مثال شما نیست.
رابطه دانشجو و درس چند به چند است.
رابطه درس و استاد چند به چند است.
نیازی نیست بین استاد و دانشجو رابطه مستقیمی تعریف شود.
نیاز به جدول چهارمی وجود دارد به نام «واحدهای اخذ شده» که در اینجا ID یک درس و یک استاد و یک دانشجو ثبت می‌شود.
رابطه‌ها هم یک به چند است. یک دانشجو چند واحد اخذ شده می‌تواند داشته باشد. یک استاد چند واحد ارائه شده را می‌تواند اداره کند.

+ مراجعه کنید به بحث بررسی تفصیلی رابطه چند به چند و کامنت‌های آن و لینکی که در آن به راه حل خاصی اشاره شده که کار جدول واسط را شبیه سازی می‌کند با دو رابطه یک به چند.

نویسنده: محبوبه محمدی
تاریخ: ۱۳۹۲/۰۲/۱۰ ۱۱:۸

سلام.ممنون از توضیحات خوبتون.
من یک رابطه **many-to-one** بین جداول **Project** و **ProjectRow** دارم که به این صورت map شده:

```
HasOptional ( c => c.Project ).WithMany (c => c.ProjectRowCollection).HasForeignKey(c =>
c.ProjectID).WillCascadeOnDelete();
```

حالا وقتی میخوام یک ProjectRow رو حذف کنم به این صورت عمل میکنم:

```
ProjectRowCollection.Remove(ProjectRowItem);
```

اما وقتی یک ردیف پروژه رو حذف میکنم به جای اینکه ردیف رو از جدول حذف کنه فقط کلید خارجی رو NULL میکنه مگر اینکه مستقیم از خود ProjectRow ردیف رو حذف کنم.مشکل از کجا میتونه باشه؟!
ممنون از اینکه وقت گذاشتید و خوندید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۱۰ ۱۱:۳۷

- علت به Optional و Required بودن روابط بر می‌گردد. حالت Required یعنی فرزند، بدون والد نمی‌تواند وجود داشته باشد؛ برعکس حالت optional. بنابراین فقط در حالت Required حذف فرزندان در صورت حذف والد صورت خواهد گرفت.
- جزئیات بیشتر از زبان طراحان EF:
Deleting orphans with Entity Framework
+ طراحی پیش فرض است؛ مطابق مستندات MSDN:

```
If a foreign key on the dependent entity is nullable, Code First does not set cascade delete on the
relationship,
and when the principal is deleted the foreign key will be set to null.
```

نویسنده: محبوبه محمدی
تاریخ: ۱۳۹۲/۰۲/۱۰ ۱۲:۱۶

خیلی ممنونم.

نویسنده: محبوبه محمدی
تاریخ: ۱۳۹۲/۰۴/۰۴ ۱۴:۳۴

راستش هنوز این مشکل برای من حل نشده،اینطور که من فهمیدم برای حذف یه فرزند باید مستقیما خودش رو حذف کنیم،با استفاده از حذف کردنش از Collection مربوط به والدش نمیشه،درسته؟ببینید نمیخوام والد رو حذف کنم،میخام از طریق لیستی که از فرزندها توی والد هست یکی از فرزندها رو حذف کنم.آیا امکانش هست؟

```
Project.ProjectRowCollection.Remove(ProjectRowItem);
```

یعنی من مجبورم تو مثال بالا مستقیم ProjectRowItem رو حذف کنم با استفاده از دستور بالا نمیشه؟!

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۴/۰۴ ۱۸:۴

- در هر حالتی، زمانیکه یک شیء کامل را به همراه Id تحت نظر سیستم ردیابی آن دارید (مثل ProjectRowItem)، ساده‌ترین راه حذف آن، ابتدا حذف آن از DbSet مرتبط، مانند ctx.ProjectRowItems.Remove و بعد فراخوانی SaveChanges است (جهت اعمال نهایی تغییرات به بانک اطلاعاتی).
- این شیء اگر تحت نظر سیستم ردیابی نباشد، فراخوانی متد Remove اثری نخواهد داشت. اطلاعات بیشتر : ^ و ^
- زمانیکه فقط یک شیء تحت ردیابی را از یک لیست حذف می‌کنید، این مورد فقط به معنای null کردن ID آن است؛ چون فرمان اصلی حذف خود شیء صادر نشده است. فقط دیگر علاقمند نیستید که این رابطه برقرار باشد.

نویسنده: سارا محمدی
تاریخ: ۱۳۹۲/۱۲/۰۱ ۱۰:۲

ممنون از مطلب خوبتان
من رابطه مشتری و آدرس را متوجه نشدم یعنی هر آدرس میتواند برای چند مشتری باشد و آیا کلید جدول آدرس کلید خارجی هم هست اگه ممکنه بیشتر توضیح بدید ممنون

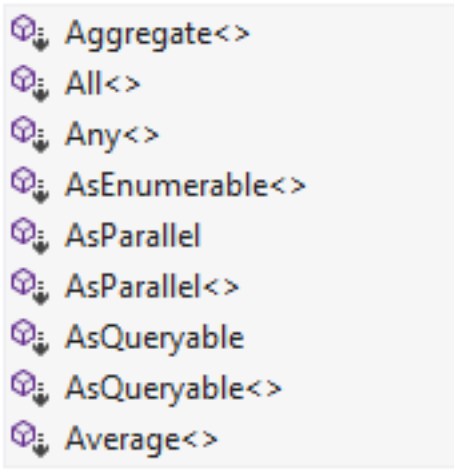
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۰۱ ۱۰:۹

قسمت «تنظیمات روابط many-to-one» و تصویر ساختار ذیل عبارت «که نهایتا منجر به تولید چنین ساختاری در بانک اطلاعاتی می‌گردد » را مطالعه کنید.

نویسنده: حمید حسین وند
تاریخ: ۱۳۹۳/۰۱/۲۴ ۱۶:۴۹

نتونستم از روابط استفاده کنم. (یک به چند)

```
using System;
using System.Collections.Generic;
using System.Data.Entity.ModelConfiguration;
using System.Linq;
using System.Text;

namespace DomainClasses.EntityConfiguration
{
    public class UserConfig : EntityTypeConfiguration<User>
    {
        public UserConfig()
        {
            HasRequired(x=>x.Username).WithMany(x=>x.

        }
    }
}
```



علت اینکه میخوام از رابطه one to more استفاده کنم اینه که چندین جدول دیگه دارم که همه شون مرتبط به جدول User هستند.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۲۴ ۱۷:۲

- در قسمت HasRequired که Username نباید تعریف شود. در اینجا یک سر دیگر رابطه باید معرفی گردد. همان روابط و کلاس‌هایی که به صورت virtual در کدها آمده.

- در متن ذکر کردم «همین میزان تنظیم کفایت می‌کند و نیازی به استفاده از Fluent API برای معرفی روابط نیست.»

برای بسیاری از تنظیمات EF Code first، اگر پیش فرض‌های آن‌را رعایت کنید، نیازی به هیچگونه تنظیم اضافه‌تری ندارید. مثلا برای رابطه one-to-many فقط کافی است در دو سر رابطه (نه فقط یک سر آن)، تنظیمات زیر را داشته باشید:

```
// یک سایت که چندین بلاگ دارد
public class Site
{
    public int Id { get; set; }
    public string Name { get; set; }

    public virtual ICollection<Blog> Blogs { set; get; }
}

public class Blog
{
    public int Id { get; set; }
    public string Name { set; get; }

    [ForeignKey("SiteId")]
    public virtual Site Site { get; set; }
    public int SiteId { set; get; }
```

```
}
```

همین مقدار کافی است و پیش فرض‌ها را پوشش می‌دهد. تنظیمات Fluent برای زمانی است که می‌خواهید پیش‌فرض‌ها را بازنویسی کنید. مثلا نام جدول خودکار تشکیل شده توسط آن مدنظر شما نیست. یا حالت بسیار خاصی از روابط مانند مدل‌های خودارجاع دهنده باید تشکیل شود و در این حالت فقط حالت Fluent است که پاسخگوی یک چنین سناریوهایی است.

نویسنده: حمید حسین وند
تاریخ: ۱۳۹۳/۰۱/۲۴ ۱۷:۳۳

بله شما درست می‌فرمایید اما اگه بخوام وقتی رکوردی از جدول User حذف میشه رکوردهای مربوط به این یوزر در جداول دیگه (حدود 5 جدول) حذف بشه باید از WillCascadeOnDelete استفاده کنم.
می‌تونم به صورت زیر استفاده کنم اما میخوام ببینم چرا نمی‌تونم به صورت یک به چند استفاده کنم.

```
HasMany(x => x.Ads).WithRequired(x => x.User).WillCascadeOnDelete();
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۲۴ ۱۷:۴۵

- در قسمت HasRequired که Username نباید تعریف شود. در اینجا یک سر دیگر رابطه باید معرفی گردد. همان روابط و کلاس‌هایی که به صورت virtual در کدها آمده. HasRequired با IsRequired متفاوت است.
+ حذف آبشاری به صورت پیش فرض فعال است (برای مواردی که کلید خارجی نال پذیر نیست). نیازی به فعال سازی دستی آن نیست.

نویسنده: حمید حسین وند
تاریخ: ۱۳۹۳/۰۱/۲۵ ۱۰:۷

سلام؛ کد زیر رو نوشتم طبق اون چیزی که توی این لینک که داده بودید اما وقتی حذف انجام میدم فقط کلیدهای خارجی رو نال میکنه و خود رکورد رو از جدول اصلی حذف میکنه.

```
modelBuilder.Conventions.Remove<OneToManyCascadeDeleteConvention>();
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۲۵ ۱۰:۳۹

- کدهایی که در ماخذ رسمی ذکر شدند ، برای حذف پیش فرض‌های EF هست. به صورت پیش فرض OneToManyCascadeDeleteConvention وجود دارد. اگر بخواهید در همه جا آن‌را حذف کنید،
modelBuilder.Conventions.Remove را بر روی آن فراخوانی کنید. اگر نیاز است فقط در یک رابطه‌ی خاص این مورد حذف شود از متد WillCascadeOnDelete با پارامتر false استفاده کنید.
- همچنین مطابق این ماخذ: اگر کلید خارجی مدنظر نال پذیر باشد (مانند نوع‌های nullable صریح و یا string و امثال آن)، حذف آبشاری را اعمال نمی‌کند. فقط یک سر رابطه را نال کرده و آن‌را حذف می‌کند.

```
If a foreign key on the dependent entity
```

is nullable

```
, Code First does
not set cascade delete on the relationship, and when the principal is
deleted the foreign key will be set to null
```

```
If a foreign key on the dependent entity
```

is not nullable

```
, then Code First sets cascade delete on the relationship
```

نویسنده: جواد
تاریخ: ۱۳۹۳/۰۲/۰۵ ۱۰:۳۶

سلام؛ من یک جدول دارم که در اون کتاب‌های امانت گرفته شده را ثبت می‌کنم.یعنی این که در اون کلید خارجی کتاب و همچنین کلید خارجی اون عضو وجود دارد.حالا وقتی که می‌خواهم یک رکورد را از این جدول حذف بکنم به ارور زیر بر می‌خورم.

.The object cannot be deleted because it was not found in the ObjectStateManager

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۰۵ ۱۱:۳۲

یعنی سیستم tracking اطلاعی از وجود شیء شما ندارد (^ و ^).