عنوان: **بررسی تفصیلی رابطه Many-to-Many در EF Code first**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۰۴ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: Entity framework

رابطه چند به چند در مطالب EF Code first سایت جاری، در حد تعریف نگاشت‌های آن بررسی شده، اما نیاز به جزئیات بیشتری برای کار با آن وجود دارد که در ادامه به بررسی آن‌ها خواهیم پرداخت:

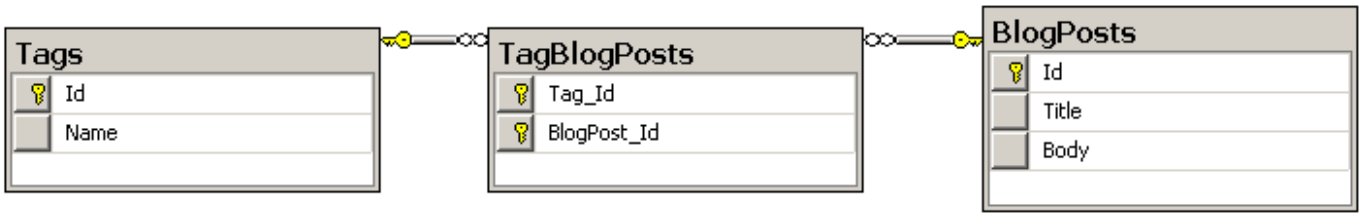
### 1) پیش فرض‌های EF Code first در تشخیص روابط چند به چند

تشخیص اولیه روابط چند به چند، مانند یک مطلب موجود در سایت و برچسب‌های آن؛ که در این حالت یک برچسب می‌تواند به چندین مطلب مختلف اشاره کند و یا برعکس، هر مطلب می‌تواند چندین برچسب داشته باشد، نیازی به تنظیمات خاصی ندارد. همینقدر که دو طرف رابطه توسط یک ICollection به یکدیگر اشاره کنند، مابقی مسایل توسط EF Code first به صورت خودکار حل و فصل خواهند شد:

```
using System;
using System.Linq;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Data.Entity;
using System.Data.Entity.Migrations;
using System.Data.Entity.ModelConfiguration;

namespace Sample
{
    public class BlogPost
    {
        public int Id { set; get; }

        [StringLength(maximumLength: 450, MinimumLength = 1), Required]
        public string Title { set; get; }

        [MaxLength]
        public string Body { set; get; }

        public virtual ICollection<Tag> Tags { set; get; } // many-to-many

        public BlogPost()
        {
            Tags = new List<Tag>();
        }
    }

    public class Tag
    {
        public int Id { set; get; }

        [StringLength(maximumLength: 450), Required]
        public string Name { set; get; }

        public virtual ICollection<BlogPost> BlogPosts { set; get; } // many-to-many

        public Tag()
        {
            BlogPosts = new List<BlogPost>();
        }
    }

    public class MyContext : DbContext
    {
        public DbSet<BlogPost> BlogPosts { get; set; }
        public DbSet<Tag> Tags { get; set; }
    }

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

```
        protected override void Seed(MyContext context)
        {
            var tag1 = new Tag { Name = "Tag1" };
            context.Tags.Add(tag1);

            var post1 = new BlogPost { Title = "Title...1", Body = "Body...1" };
            context.BlogPosts.Add(post1);

            post1.Tags.Add(tag1);

            base.Seed(context);
        }
    }

    public static class Test
    {
        public static void RunTests()
        {
            Database.SetInitializer(new MigrateDatabaseToLatestVersion<MyContext, Configuration>());

            using (var ctx = new MyContext())
            {
                var post1 = ctx.BlogPosts.Find(1);
                if (post1 != null)
                {
                    Console.WriteLine(post1.Title);
                }
            }
        }
    }
}
```

در این مثال، رابطه بین مطالب ارسالی در یک سایت و برچسب‌های آن به صورت many-to-many تعریف شده است و همینقدر که دو طرف رابطه توسط یک ICollection به هم اشاره می‌کنند، رابطه زیر تشکیل خواهد شد:




در اینجا تمام تنظیمات صورت گرفته بر اساس یک سری از پیش فرض‌ها است. برای مثال نام جدول واسط تشکیل شده، بر اساس تنظیم پیش فرض کنار هم قرار دادن نام دو جدول مرتبط تهیه شده است.
همچنین بهتر است بر روی نام برچسب‌ها، یک ایندکس منحصربفرد نیز تعیین شود: ( ^ و ^ )

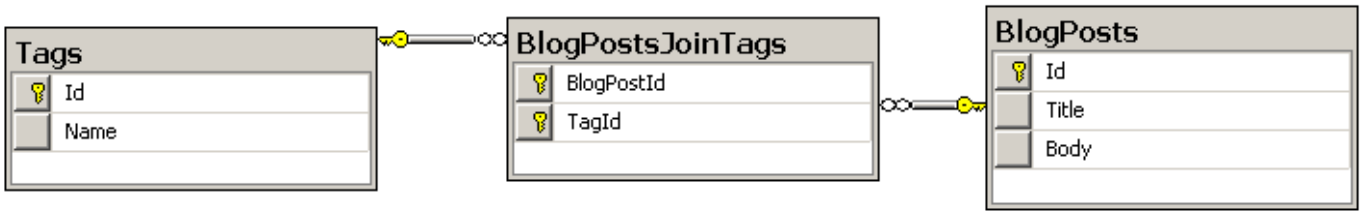
### 2) تنظیم ریز جزئیات روابط چند به چند در EF Code first

تنظیمات پیش فرض انجام شده آنچنان نیازی به تغییر ندارند و منطقی به نظر می‌رسند. اما اگر به هر دلیلی نیاز داشتید کنترل بیشتری بر روی جزئیات این مسایل داشته باشید، باید از Fluent API جهت اعمال آن‌ها استفاده کرد:

```
public class TagMap : EntityTypeConfiguration<Tag>
    {
        public TagMap()
        {
            this.HasMany(x => x.BlogPosts)
                .WithMany(x => x.Tags)
                .Map(map =>
                    {
                        map.MapLeftKey("TagId");
```

```
                    map.MapRightKey("BlogPostId");
                    map.ToTable("BlogPostsJoinTags");
                });
        }
    }

    public class MyContext : DbContext
    {
        public DbSet<BlogPost> BlogPosts { get; set; }
        public DbSet<Tag> Tags { get; set; }

        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            modelBuilder.Configurations.Add(new TagMap());
            base.OnModelCreating(modelBuilder);
        }
    }
```

در اینجا توسط متد Map، نام کلیدهای تعریف شده و همچنین جدول واسط تغییر داده شده‌اند:




### 3) حذف اطلاعات چند به چند

برای حذف تگ‌های یک مطلب، کافی است تک تک آن‌ها را یافته و توسط متد Remove جهت حذف علامتگذاری کنیم. نهایتا با فراخوانی متد SaveChanges، حذف نهایی انجام و اعمال خواهد شد.

```
using (var ctx = new MyContext())
            {
                var post1 = ctx.BlogPosts.Find(1);
                if (post1 != null)
                {
                    Console.WriteLine(post1.Title);
                    foreach (var tag in post1.Tags.ToList())
                        post1.Tags.Remove(tag);
                    ctx.SaveChanges();
                }
            }
```

در اینجا تنها اتفاقی که رخ می‌دهد، حذف اطلاعات ثبت شده در جدول واسط BlogPostsJoinTags است. Tag1 ثبت شده در متد Seed فوق، حذف نخواهد شد. به عبارتی اطلاعات جداول Tags و BlogPosts بدون تغییر باقی خواهند ماند. فقط یک رابطه بین آن‌ها که در جدول واسط تعریف شده است، حذف می‌گردد.

در ادامه اینبار اگر خود post1 را حذف کنیم:

```
var post1 = ctx.BlogPosts.Find(1);
                if (post1 != null)
                {
                    ctx.BlogPosts.Remove(post1);
                    ctx.SaveChanges();
                }
```

علاوه بر حذف post1، رابطه تعریف شده آن در جدول BlogPostsJoinTags نیز حذف می‌گردد؛ اما Tag1 حذف نخواهد شد.
بنابراین دراینجا cascade delete ایی که به صورت پیش فرض وجود دارد، تنها به معنای حذف تمامی ارتباطات موجود در جدول میانی است و نه حذف کامل طرف دوم رابطه. اگر مطلبی حذف شد، فقط آن مطلب و روابط برچسب‌های متعلق به آن از جدول میانی حذف می‌شوند و نه برچسب‌های تعریف شده برای آن.
البته این تصمیم هم منطقی است. از این لحاظ که اگر قرار بود دو طرف یک رابطه چند به چند با هم حذف شوند، ممکن بود با حذف یک مطلب، کل بانک اطلاعاتی خالی شود! فرض کنید یک مطلب دارای سه برچسب است. این سه برچسب با 20 مطلب دیگر هم رابطه دارند. اکنون مطلب اول را حذف می‌کنیم. برچسب‌های متناظر آن نیز باید حذف شوند. با حذف این برچسب‌ها طرف دوم رابطه آن‌ها که چندین مطلب دیگر است نیز باید حذف شوند!

**4) ویرایش و یا افزودن اطلاعات چند به چند**

در مثال فوق فرض کنید که می‌خواهیم به اولین مطلب ثبت شده، تعدادی تگ جدید را اضافه کنیم:

```
var post1 = ctx.BlogPosts.Find(1);
                if (post1 != null)
                {
                    var tag2 = new Tag { Name = "Tag2" };
                    post1.Tags.Add(tag2);
                    ctx.SaveChanges();
                }
```

در اینجا به صورت خودکار، ابتدا tag2 ذخیره شده و سپس ارتباط آن با post1 در جدول رابط ذخیره خواهد شد.

در مثالی دیگر اگر یک برنامه ASP.NET را درنظر بگیریم، در هربار ویرایش یک مطلب، تعدادی Tag به سرور ارسال می‌شوند. در ابتدای امر هم مشخص نیست کدامیک جدید هستند، چه تعدادی در لیست تگ‌های قبلی مطلب وجود دارند، یا اینکه کلا از لیست برچسب‌ها حذف شده‌اند:

```
//نام تگ‌های دریافتی از کاربر
                var tagsList = new[] { "Tag1", "Tag2", "Tag3" };

                //بارگذاری یک مطلب به همراه تگ‌های آن
                var post1 = ctx.BlogPosts.Include(x => x.Tags).FirstOrDefault(x => x.Id == 1);
                if (post1 != null)
                {
                    //ابتدا کلیه تگ‌های موجود را حذف خواهیم کرد
                    if (post1.Tags != null && post1.Tags.Any())
                        post1.Tags.Clear();

                    //سپس در طی فقط یک کوئری بررسی می‌کنیم کدامیک از موارد ارسالی موجود هستند
                    var listOfActualTags = ctx.Tags.Where(x => tagsList.Contains(x.Name)).ToList();
                    var listOfActualTagNames = listOfActualTags.Select(x => x.Name.ToLower()).ToList();

                    //فقط موارد جدید به تگ‌ها و ارتباطات موجود اضافه می‌شوند
                    foreach (var tag in tagsList)
                    {
                        if (!listOfActualTagNames.Contains(tag.ToLowerInvariant().Trim()))
                        {
                            ctx.Tags.Add(new Tag { Name = tag.Trim() });
                        }
                    }
                    ctx.SaveChanges(); // ثبت موارد جدید

                    //موارد قبلی هم حفظ می‌شوند
                    foreach (var item in listOfActualTags)
                    {
                        post1.Tags.Add(item);
                    }
                    ctx.SaveChanges();
                }
```

در این مثال فقط تعدادی رشته از کاربر دریافت شده است، بدون Id آن‌ها. ابتدا مطلب متناظر، به همراه تگ‌های آن توسط متد Include دریافت می‌شود. سپس نیاز داریم به سیستم ردیابی EF اعلام کنیم که اتفاقاتی قرار است رخ دهد. به همین جهت تمام

تگ‌های مطلب یافت شده را خالی خواهیم کرد. سپس در یک کوئری، بر اساس نام تگ‌های دریافتی، معادل آن‌ها را از بانک اطلاعاتی دریافت خواهیم کرد؛ کوئری tagsList.Contains به where in در طی یک رفت و برگشت، ترجمه می‌شود:

```
SELECT
[Extent1].[Id] AS [Id],
[Extent1].[Name] AS [Name]
FROM [dbo].[Tags] AS [Extent1]
WHERE [Extent1].[Name] IN (N'Tag1',N'Tag2',N'Tag3')
```

آن‌هایی که جدید هستند به بانک اطلاعاتی اضافه شده (بدون نیاز به تعریف قبلی آن‌ها)، آن‌هایی که در لیست قبلی برچسب‌های مطلب بوده‌اند، حفظ خواهند شد.
لازم است لیست موارد موجود را (listOfActualTags) از بانک اطلاعاتی دریافت کنیم، زیرا به این ترتیب سیستم ردیابی EF آن‌ها را به عنوان رکوردی جدید و تکراری ثبت نخواهد کرد.

**5) تهیه کوئری‌های LINQ بر روی روابط چند به چند**

الف) دریافت یک مطلب خاص به همراه تمام تگ‌های آن:

```
ctx.BlogPosts.Where(p => p.Id == 1).Include(p => p.Tags).FirstOrDefault()
```

ب) دریافت کلیه مطالبی که شامل Tag1 هستند:

```
var posts = from p in ctx.BlogPosts
                from t in p.Tags
                where t.Name == "Tag1"
                select p;
```

و یا :

```
var posts = ctx.Tags.Where(x => x.Name == "Tag1").SelectMany(x => x.BlogPosts);
```

## نظرات خوانندگان

نویسنده: MehRad
تاریخ: ۱۳۹۱/۱۲/۰۴ ۱۳:۲۶

با سلام؛ اگر بخواهیم به جدول tagblogpost یک آیتم دیگه اضافه کنیم مثلا تاریخ رو به این جدول اضافه کنیم باید به چه شکل این کار رو انجام دهیم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۰۴ ۱۳:۵۷

پشتیبانی نمی‌شود. این جدول واسط در رابطه many-to-many به صورت داخلی توسط EF مدیریت می‌شود و از دسترس برنامه نویس خارج است.
البته یک راه حل برای آن در اینجا مطرح شده. رابطه دیگر many-to-many نیست. دو رابطه one-to-many تشکیل شده به جدول واسط.

نویسنده: حسین
تاریخ: ۱۳۹۱/۱۲/۰۵ ۱۳:۵۴

سلام . یه سوالی که به نظرم رسید و شما نگفتین رو می‌خوام بپرسم. الان توی این مثال اگه ما بخوایم به یه پست جدید تگی رو که موجوده اختصاص بدیم باید چی کار کنیم چون اگه به این صورت عمل کنیم

```
post1.Tags.Add(tag);
```

باید مشخص بشه این تگ یه new object هستش یا خیر یا تگی هست که یبار از طریق context انتخاب شده و داخل حافظه موجوده . اگه این تگ قبلا از طریق context آورده شده باشه آیا به صورت تگی که ازقبل در دیتابیس موجوده به پست مورد نظر تخصیص داده میشه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۰۵ ۱۴:۶

توضیح دادم با مثال: «... در مثالی دیگر اگر یک برنامه ASP.NET را درنظر بگیریم ...»
listOfActualTags از بانک اطلاعاتی دریافت شده؛ بر اساس مواردی که موجود بوده.
به این ترتیب چون این تگ‌ها به سیستم ردیابی EF وارد می‌شوند و همچنین post1.Tags.Clear در ابتدای کار فراخوانی شده، استفاده از متد post1.Tags.Add(item) سبب ثبت مورد تکراری نخواهد شد.
کلا EF هر آیتمی رو که Id آن‌را از طریق دریافت اطلاعات از بانک اطلاعاتی در سیستم ردیابی خودش داشته باشه، جدید و تکراری ثبت نمی‌کنه. برای نمونه در حالت new Tag استفاده شده، این موارد جدید ثبت می‌شوند چون Id از قبل ثبت شده‌ای ندارند.
برای توضیحات بیشتر مراجعه کنید به مطلب نحوه استفاده از کلیدهای خارجی در EF . (حتی می‌شود یک شیء را بدون واکشی از دیتابیس به سیستم ردیابی وارد کرد؛ البته اگر Id آن‌را داشته باشید)

نویسنده: مرتضی
تاریخ: ۱۳۹۱/۱۲/۲۸ ۲۳:۱

سلام آقای نصیری بنده یه سوال داشتم بهترین روش برای پیاده سازی رابطه n به m برای طراحی صفحه ادمین چیه؟ببنید بنده توبه قسمت ادمین سایت یک صفحه برای هر جدول طراحی میکنم که از طریق اون بتونم اطلاعات جدول رو حذف و یا بروزرسانی و یا رکورد جدید وارد کنم حالا برای طراحی صفحه ای که جداول اون رابطه n به m دارن دچار مشکل شدم .مثلا فرض بکنید که جدول پست 1000 رکورد داره و جدول tags هم 500 رکورد .حالا برای وارد کردن یک رکورد در داخل جدول رابط باید ID یک رکورد پست و همچنین ID یک رکورد tags رو توبه جدول رابط قرار بدیم بنده خودم از dropdownlist استفاده کردم برای اینکار

که چون تعداد رکوردهای خیلی زیاده این روش جواب گو نیست به نظر شما بهترین روش چیه؟ اگه منظورم رو متوجه نشدید بگید توضیح بیشتری بدم.ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۲۸ ۲۳:۹

من در سایت جاری برای تعریف روابط چند به چند از افزونه TagIt استفاده کردم.

نویسنده: ناصر طاهری
تاریخ: ۱۳۹۲/۰۳/۱۸ ۱:۲۶

سلام
در قسمت :

```
//فقط موارد جدید به تگ‌ها و ارتباطات موجود اضافه می‌شوند
foreach (var tag in data.Tags)
{
     if (!listOfActualTagNames.Contains(tag.ToLowerInvariant().Trim()))
     {
            _tag.Add(new Tag { Title = tag.Trim() });
      }
}
 _uow.SaveChanges(); // ثبت موارد جدید
```

فقط تگ‌ها در جدول خودشون ذخیره میشن و ارتباطی با پست مربوطه ایجاد نمیشه.
آیا نباید به کد زیر تغییر داد؟

```
if (!listOfActualTagNames.Contains(tag.ToLowerInvariant().Trim()))
 {
      var tags = new Tag { Title = tag.Trim() };
       _tag.Add(tags);
       posts.Tags.Add(tags);
  }
```

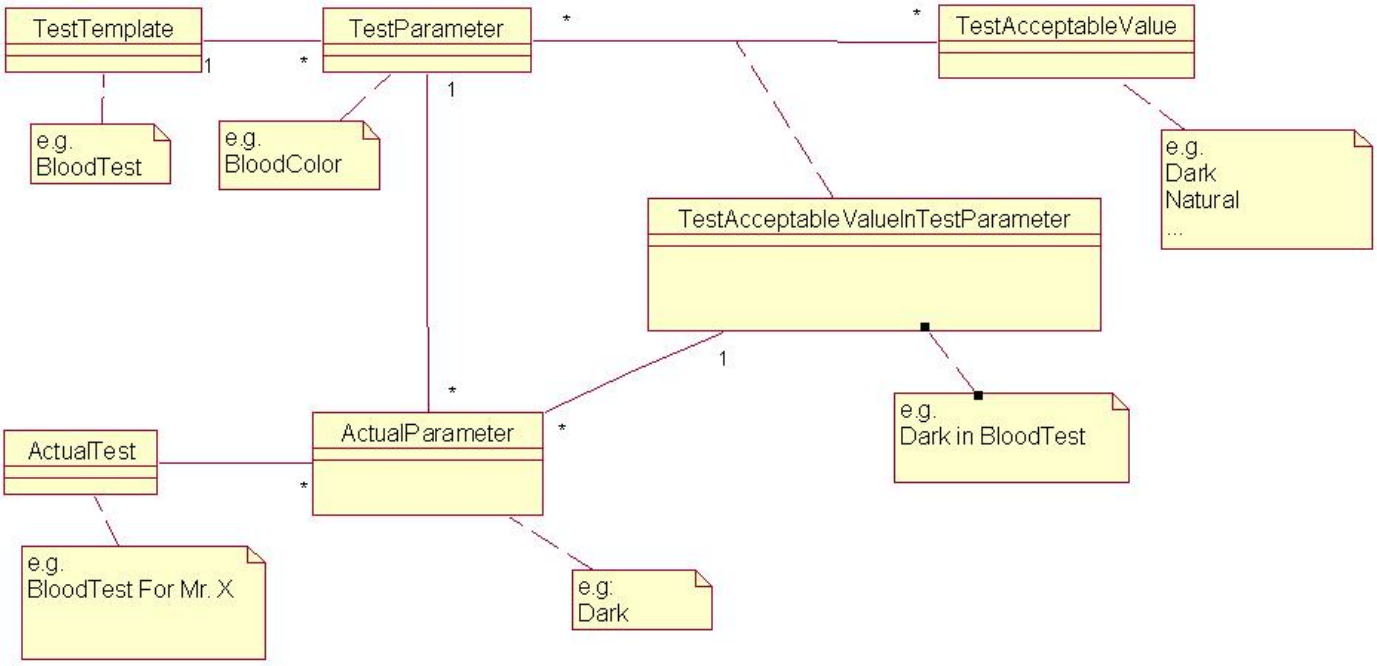
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۱۸ ۱۰:۷

درسته. اصلاحیه کدهای مطلب فوق:

```
var newTag = ctx.Tags.Add(new Tag { Name = tag.Trim() });
post1.Tags.Add(newTag);
```

نویسنده: مسعود2
تاریخ: ۱۳۹۲/۰۹/۲۳ ۱۷:۱۹

فرض کنید من لازم دارم در یک برنامه مدیریت آزمایشگاه؛ کاربر بتونه ساختار تستها رو تعریف کنه و بگه اون تست چه پارامترهایی داره و مقادیر مجاز هر پارامتر چی‌ها هست و بعدش بر اساس این ساختار تعریف شده، مقادیر مشاهده شده در آزمایش واقعی رو برای این تست ثبت کنه بنابر این من این کلاسها رو طراحی کرده ام:




تا اینجا ساختار تست که شامل چند پارامتر با مقادیر مجازشون هست رو میشه با این کلاسها تعریف کرد.
حالا فرض کنید برای یک تست تعریف شده با این ساختار میخواهیم مقادیر مشاهده شده واقعی رو ثبت کنیم(ActualTest,ActualParameterValue). وقتی بخوام رنگ خون رو از لیست رنگهای مجاز تعریف شده برای این پارامتر انتخاب کنم، قاعدتا بایستی id مربوط به جدول واسط(TestAcceptableValueInTestParameter) به عنوان id رنگ انتخاب شده در جدول مقدار واقعی پارامتر ثبت بشه. به عبارت دیگه من با id کلاسی کار دارم که اگر با روش many-to-many ذکر شده برای EF کار کنم، همچین کلاسی جزو مدلهای من نیست. آیا EF راهکاری برای این موارد داره؟
ممنون.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۲۳ ۱۸:۲۹

چرا رابطه TestParameter و TestAcceptedValue به صورت many-to-many تعریف شده؟ رنگ خون چندین مقدار دارد، اما عکس آن صادق نیست. یعنی یک رنگ خون را نمی‌شود به چندین TestParameter مختلف مانند قند خون یا سطح فلان هورمون انتساب داد.
مثال ساده آن کاربر و نقش‌های او است. یک کاربر می‌تواند چندین نقش داشته باشد (نویسنده، ادیتور و غیره). یک نقش می‌تواند به چندین کاربر منتسب شود (مثلا نقش ادیتور را می‌شود به ده‌ها کاربر انتساب داد). یعنی می‌شود از هر طرف این رابطه، یک رکورد را به چندین رکورد طرف دیگر ربط منطقی داد. اما در حالت مداخل یک آزمایش و مقادیر مجاز جهت یک مدخل، اینچنین نیست و رابطه one-to-many است.

نویسنده: مسعود2
تاریخ: ۱۳۹۲/۰۹/۲۳ ۱۹:۵۹

فرض کنید آزمایش دیگری غیر از آزمایش خون تعریف شود، مثلا آزمایش ادرار؛ اونوقت این رنگ(و نه رنگ خون) در اونجا هم مورد استفاده پیدا میکند. بنابر این رابطه بایستی many-to-many باشد. یا حتی میتوان برای TestParameter واحد اندازه گیری(UOM) در نظر گرفت که برای آن هم همین مساله رخ میدهد(چون هر پارامتر تست میتواند چندین واحد اندازه گیری داشته باشد و هر واحد اندازه گیری در مورد چندین پارامتر تست استفاده شود).

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۲۳ ۲۱:۵

- در مورد واحدهای اندازه‌گیری منطقی است.
- « آیا EF راهکاری برای این موارد داره؟ » مراجعه کنید به پاسخ اولین نظر مطرح شده. کمی بالاتر.

نویسنده: احمدعلی شفیعی
تاریخ: ۱۳۹۳/۰۹/۱۹ ۰:۱۹

سلام. من ساختاری شبیه به این دارم:

```
public class Person
{
        //...
        public virtual IList<Center> PreferedCenters { get; set; }
        public virtual IList<Center> ActiveCenters { get; set; }

        public Person()
        {
            PreferedCenters = new List<Center>();
            ActiveCenters = new List<Center>();
        }
}
```

و کلا Center هم به شکل زیره:

```
public class Center
{
    //...
    public virtual IList<Person> Persons { get; set; }

    public Center()
    {
        Persons = new List<Person>();
    }
}
```

مشکلی که دارم اینه که منطقا باید دوتا رابطه‌ی Many-to-many تشکیل بشه: ActiveCenters و PreferedCenters. ولی توی جدول خروجی EF، فقط ستونی به اسم Center_ID ساخته می‌شه و وقتی هم که می‌خوام به سیستم چیزی اضافه کنم اروری شبیه به این می‌گیرم:

```
An unhandled exception of type 'System.InvalidOperationException' occurred in EntityFramework.dll

Additional information: Multiplicity constraint violated. The role 'Center_Persons_Source' of the
relationship 'Yarigaran.DataLayer.Center_Persons' has multiplicity 1 or 0..1.
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۱۹ ۰:۵۵

- به ازای هر سر رابطه‌ی چند به چند، باید یک ICollection در دو طرف وجود داشته باشد (ICollection حالت پیش‌فرض EF است و نه IList). در حالت شما، 2 مورد در کلاس شخص و دو مورد در کلاس مرکز. در غیر اینصورت رابطه‌ها یک به چند تفسیر می‌شوند. به علاوه در این حالت مشخص نیست که هر خاصیت به کدام خاصیت باید متصل شود چون InverseProperty ذکر نشده‌است.
- زمانیکه بیش از یک ستون یک جدول قرار است با یک جدول خاص دیگر رابطه‌ی چند به چند داشته باشند، نیاز به چندین جدول واسط خواهد بود که باید به صورت صریح مشخص شوند :

```
modelBuilder.Entity<Person>()
    .HasMany(u => u.PreferedCenters)
    .WithMany(t => t.Persons)
    .Map(x =>
    {
        x.MapLeftKey("PersonId");
        x.MapRightKey("CenterId");
        x.ToTable("Center_Persons1"); // جدول واسط یک
    });
```

```
modelBuilder.Entity<Person>()
    .HasMany(u => u.ActiveCenters)
    .WithMany(t => t.Persons)
    .Map(x =>
    {
        x.MapLeftKey("PersonId");
        x.MapRightKey("CenterId");
        x.ToTable("Center_Persons2"); // جدول واسط دو
    });
```