عنوان: **کار با کلیدهای اصلی و خارجی در EF Code first**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۱۹ ۱۷:۵۱
آدرس: www.dotnettips.info
گروه‌ها: Entity framework

در حین کار با ارتباطات بین اشیاء و جداول، دانستن یک سری از نکات می‌توانند در کم کردن تعداد رفت و برگشت‌های به سرور مؤثر واقع شده و نهایتا سبب بالا رفتن سرعت برنامه شوند. از این دست می‌توان به یک سری نکات ریز همراه با primary-keys و foreign-keys اشاره کرد که در ادامه به آن‌ها پرداخته خواهد شد.
در ابتدا کلاس‌های مدل و Context برنامه را به شکل زیر درنظر بگیرید:

```
using System;
using System.Data.Entity;
using System.Data.Entity.Migrations;

namespace TestKeys
{
    public class Bill
    {
        public int Id { get; set; }
        public decimal Amount { set; get; }
        public virtual Account Account { get; set; }
    }

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

    public class MyContext : DbContext
    {
        public DbSet<Bill> Bills { get; set; }
        public DbSet<Account> Accounts { get; set; }
    }

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

        protected override void Seed(MyContext context)
        {
            var a1 = new Account { Name = "a1" };
            var a2 = new Account { Name = "a2" };

            var bill1 = new Bill { Amount = 100, Account = a1 };
            var bill2 = new Bill { Amount = 200, Account = a2 };

            context.Bills.Add(bill1);
            context.Bills.Add(bill2);
            base.Seed(context);
        }
    }

    public static class Test
    {
        public static void Start()
        {
            Database.SetInitializer(new MigrateDatabaseToLatestVersion<MyContext, Configuration>());
            using (var ctx = new MyContext())
            {
                var bill1 = ctx.Bills.Find(1);
                Console.WriteLine(bill1.Amount);
            }
        }
    }
}
```

در اینجا کلاس صورتحساب و حساب مرتبط به آن تعریف شده‌اند. سپس به کمک DbContext این دو کلاس در معرض دید EF Code

first قرار گرفته‌اند و در کلاس Configuration نحوه آغاز بانک اطلاعاتی به همراه تعدادی رکورد اولیه مشخص شده است.

نحوه صحیح مقدار دهی کلید خارجی در EF Code first

تا اینجا یک روال متداول را مشاهده کردیم. اکنون سؤال این است که اگر بخواهیم اولین رکورد صورتحساب ثبت شده توسط متد Seed را ویرایش کرده و مثلا حساب دوم را به آن انتساب دهیم، بهینه‌ترین روش چیست؟ بهینه‌ترین در اینجا منظور روشی است که کمترین تعداد رفت و برگشت به بانک اطلاعاتی را داشته باشد. همچنین فرض کنید در صفحه ویرایش، اطلاعات حساب‌ها در یک Drop down list شامل نام و id آن‌ها نیز وجود دارد.

روش اول:

```
using (var ctx = new MyContext())
{
     var bill1 = ctx.Bills.Find(1);
     var a2 = new Account { Id = 2, Name = "a2" };
     bill1.Account = a2;
     ctx.SaveChanges();
}
```

این روش مخصوص تازه واردهای EF Code first است و آنطور که مدنظر آن‌ها است کار نمی‌کند.
به کمک متد Find اولین رکورد یافت شده و سپس بر اساس اطلاعات drop down در دسترس، یک شیء جدید حساب را ایجاد و سپس تغییرات لازم را اعمال می‌کنیم. در نهایت اطلاعات را هم ذخیره خواهیم کرد.
این روش به ظاهر کار می‌کنه اما حاصل آن ذخیره رکورد حساب سومی با id=3 در بانک اطلاعاتی است و سپس انتساب آن به اولین صورتحساب ثبت شده.
نتیجه: Id را دستی مقدار دهی نکنید؛ تاثیری ندارد. زیرا اطلاعات شیء جدید حساب، در سیستم tracking مرتبط با Context جاری وجود ندارد. بنابراین EF آن‌را به عنوان یک شیء کاملا جدید درنظر خواهد گرفت، صرفنظر از اینکه Id را به چه مقداری تنظیم کرده‌اید.

روش دوم:

```
using (var ctx = new MyContext())
{
    var bill1 = ctx.Bills.Find(1);
    var a2 = ctx.Accounts.Find(2);
    bill1.Account = a2;
    ctx.SaveChanges();
}
```

اینبار بر اساس Id دریافت شده از Drop down list، شیء حساب دوم را یافته و به صورتحساب اول انتساب می‌دهیم. این روش درست کار می‌کند؛ اما ... بهینه نیست. فرض کنید شیء جاری دارای 5 کلید خارجی است. آیا باید به ازای هر کلید خارجی یکبار از بانک اطلاعاتی کوئری گرفت؟
مگر نه این است که اطلاعات نهایی ذخیره شده در بانک اطلاعاتی متناظر با حساب صورتحساب جاری، فقط یک عدد بیشتر نیست. بنابراین آیا نمی‌شود ما تنها همین عدد متناظر را بجای دریافت کل شیء به صورتحساب نسبت دهیم؟
پاسخ: بله. می‌شود! ادامه آن در روش سوم.

روش سوم:
در اینجا بهترین کار و یکی از best practices طراحی مدل‌های EF این است که طراحی کلاس صورتحساب را به نحو زیر تغییر دهیم:

```
public class Bill
{
        public int Id { get; set; }
        public decimal Amount { set; get; }
```

```
        [ForeignKey("AccountId")]
        public virtual Account Account { get; set; }
        public int AccountId { set; get; }
}
```

به این ترتیب هم navigation property که سبب تعریف رابطه بین دو شیء و همچنین lazy loading اطلاعات آن می‌شود پابرجا خواهد بود و هم توسط خاصیت جدید AccountId که توسط ویژگی ForeignKey معرفی شده است، ویرایش اطلاعات آن دقیقا همانند کار با یک بانک اطلاعاتی واقعی خواهد شد.
اینبار به کمک خاصیت متناظر با کلید خارجی جدول، مقدار دهی و ویرایش کلیدهای خارجی یک شیء به سادگی زیر خواهد بود؛ خصوصا بدون نیاز به رفت و برگشت اضافی به بانک اطلاعاتی جهت دریافت اطلاعات متناظر با اشیاء تعریف شده به صورت navigation property :

```
using (var ctx = new MyContext())
{
    var bill1 = ctx.Bills.Find(1);
    bill1.AccountId = 2;
    ctx.SaveChanges();
}
```

وارد کردن یک شیء به سیستم Tracking

در قسمت قبل عنوان شد که Id را دستی مقدار دهی نکنید، چون تاثیری ندارد. سؤال: آیا می‌شود این شیء ویژه تعریف شده را به سیستم Tracking وارد کرد؟
پاسخ: بلی. به نحو زیر:

```
using (var ctx = new MyContext())
{
     var a2 = new Account { Id = 2, Name = "a2_a2" };
     ctx.Entry(a2).State = System.Data.EntityState.Modified;
     ctx.SaveChanges();
}
```

در اینجا شیء حساب دوم را به صورت دستی و بدون واکشی از بانک اطلاعاتی ایجاد کرده‌ایم. بنابراین از دیدگاه Context جاری هیچ ارتباطی به بانک اطلاعاتی نداشته و یک شیء جدید درنظر گرفته می‌شود (صرفنظر از Id آن). اما می‌توان این وضعیت را تغییر داد. فقط کافی است State آن‌را به نحوی که ملاحظه می‌کنید به Modified تغییر دهیم. اکنون اگر اطلاعات این شیء را ذخیره کنیم، دقیقا حساب با id=2 در بانک اطلاعاتی ویرایش خواهد شد و نه اینکه حساب جدیدی ثبت گردد.

نظرات خوانندگان

نویسنده: شهروز جعفری
تاریخ: ۱۳۹۱/۰۴/۱۹ ۱۹:۲۲

سلام
اگر از الگوی unitofwork که پیاده سازیشو درس دادیداستفاده کرده باشیم نحوه استفاده از using ب چه صورت هستش؟

نویسنده: حمید بهروزی
تاریخ: ۱۳۹۱/۰۴/۱۹ ۱۹:۲۴

درود به آقای نصیری عزیز

بر اساس آموزه‌های شما پروژه ای در asp.net mvc4 و ef code first و unitofwork نوشته ام اما برای وهله سازی اینترفیس‌های سرویس و واحد کار برنامه، نمی‌خواهم از کتابخانه structuremap استفاده کنم . پروژه weapsy را مطالعه کردم منتها متوجه نشدم چه جایگزینی برای structuremap برگزیده است . لطفا راهنمایی بفرمایید

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۱۹ ۱۹:۵۲

نیازی نیست به صورت مستقیم فراخوانی شود. کار using رها سازی منابع مرتبط با Context است. اینکار در آنجا در Application_EndRequest به صورت خودکار انجام می‌شود (با کد ObjectFactory.ReleaseAndDisposeAllHttpScopedObjects که نوشته شد).

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۱۹ ۲۰:۳

از Autofac استفاده کرده.

نویسنده: حمید بهروزی
تاریخ: ۱۳۹۱/۰۴/۲۰ ۸:۲۵

سپاس

و این نسبت به structuremap چگونه است ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۲۰ ۸:۴۲

از این دست کتابخانه‌ها زیاد است. استفاده از این‌ها بیشتر سلیقه‌ای است. یک نفر بر اساس سرعت این‌ها رو بررسی می‌کنه یک نفر بر اساس تعداد قابلیت‌ها. در کل نهایتا تمام این‌ها یک هدف رو برآورده می‌کنند و عموما در Syntax بیشتر با هم تفاوت دارند.

نویسنده: محمدی
تاریخ: ۱۳۹۱/۰۴/۲۲ ۱۰:۳۷

```
An object with the same key already exists in the ObjectStateManager. The ObjectStateManager cannot track multiple objects with the same key.
```

وقتی می‌خوام یک entity رو به context اتچ کنم قبلا نباید هیچ entity با این کلید در context وجود داشه باشه مشکل من وقتیه که یک entity رو با ریلیشن های اون اتچ می‌کنم ممکنه یک ریلیشن تکراری وجود داشته باشه که باعث خطای فوق میشه .
یک راه حل اینه که موجودیت‌ها رو به جای attach کردن دوباره از context فراخوانی کنم ولی مطمئنا راه حل اصولی نیست ، ممنون می‌شم راهنمایی کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۲۲ ۱۰:۵۸

در EF Code first می‌شود از این متد جهت بازنویسی مقادیر سیستم tracking استفاده کرد:

```
context.Entry(oldEntity).CurrentValues.SetValues(newEntity)
```

نویسنده: محمدی
تاریخ: ۱۳۹۱/۰۴/۲۲ ۱۱:۲۷

در نظر بگیرید از فرم انتخاب کالا 10 عدد کالا انتخاب کردیم و می‌خواهیم به پیش فاکتور اضافه کنیم و همزمان روی خود کالا هم تغییراتی بدیم و هر کالا هم سه تا ریلیشن داره و... (البته موضوع من سیستم بارنامه و...است و جهت مثال گفتم درج کالا)
قاعدتا باید آپشنی باشه که به بگیم که از ردگیری تغییرات و .. صرف نظر کنه که من از آپشن زیر استفاده کردم ولی برای اتچ تاثیری نداشت.

```
context.Customers.MergeOption = System.Data.Objects.MergeOption.NoTracking;
```

و یا موقع attach کردن به اون بگیم از ریلیشن‌ها صرف نظر کن. ویا ریلشین‌ها مورد نظر ما رو باز نویسی کن.
من پروژه رو با Database first شروع کردم البته قراره اونو به Code first تبدیل کنم که هنوز وقت نشده! برای Database first راهکاری نیست؟

یه سوالی که ذهنمو خیلی مشغول کرده چطور می‌تونم یک موجودیت رو از ObjectStateManager حذف کنیم به طوری که چیزی در مورد اون موجودیت ندونه ، با dettach کردن باز هم وضعیت اون موجودیت رو در خودش نگه می‌داره.
ببخشید که طولانی شد و ممنون از حوصله شما بابت پاسخگویی.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۲۲ ۱۱:۵۰

برای حالت database first جهت بازنویسی مقادیر سیستم tracking آن:

```
context.YourEntitySet.ApplyCurrentValues(newEntity)
```

همچنین امکان detach آن هم وجود دارد: (^)

نویسنده: محمدی
تاریخ: ۱۳۹۱/۰۴/۲۲ ۲۳:۵۷

ممنون با کد زیر مشکلم حل شد.

```
foreach (var item in frm.SelectedServices)
                {
                    ObjectStateEntry temp;
                    if (context.ObjectStateManager.TryGetObjectStateEntry(item.Customer.EntityKey, out
```

```
temp))
                    {
                        context.Detach(temp.Entity);
                    }
                    context.Services.Attach(item)
```

;

```
.
                    .
```

```
.
                }
```

نویسنده: عرفان
تاریخ: ۱۳۹۱/۰۶/۱۶ ۱۹:۸

سلام آقای نصیری،
AccountId پراپرتی ای که تو روش سوم به کار بردید اصطلاحاً بهش Foreign Key Property میگن،
حالا سوال بنده اینه که از این Foreign Key Property **فقط** توی رابطه‌های one-to-many(یا many-to-one) استفاده میکنن،درسته؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۱۶ ۱۹:۱۹

در هر رابطه‌ای که نیاز به تعریف کلید خارجی داشته باشد، بهتر است استفاده شود.
مثلا رابطه many-to-many نیازی به این تعریف ندارد چون مدیریت جدول واسط بین دو موجودیت مرتبط که حاوی کلیدهای خارجی به این دو جدول است، خودکار می‌باشد.

نویسنده: عرفان
تاریخ: ۱۳۹۱/۰۶/۱۶ ۱۹:۲۸

درسته،ولی تو رابطه‌های one-to-one(و one-to-zero) هم این روش کاربرد نداره،چون کلید Dependent همین کلید خارجی میشه دیگه،اینم درسته دیگه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۱۶ ۱۹:۴۰

جمله «در هر رابطه‌ای که نیاز به تعریف کلید خارجی داشته باشد، بهتر است استفاده شود.» نسبتا جامع و مانع است. چون در رابطه 1:1 خودبخود کلید اصلی، کلید خارجی اشتراکی هم هست و نیازی به تعریف مجدد آن نیست.

نویسنده: عرفان
تاریخ: ۱۳۹۱/۰۶/۱۶ ۱۹:۴۸

بازم ممنونم.

نویسنده: امیرحسین جلوداری
تاریخ: ۱۳۹۱/۰۸/۲۰ ۱۱:۲۰

سلام ...
این مدلو ببینید :

```
public class FileUpload
{
    public int FileUploadId { get; set; }
```

```
    public string FileName { get; set; }

}

public class Company
{
    public int CompanyId { get; set; }
    public string Name { get; set; }

    public FileUpload Logo { get; set; }
    public int? LogoId { get; set; }

    public FileUpload Catalog { get; set; }
    public int? CatalogId { get; set; }
}

public class Ads
{
    public int AdsId { get; set; }
    public string Name { get; set; }

    public FileUpload Picture { get; set; }
    public int? PictureId { get; set; }
}

public class TestContext : DbContext
{
    public DbSet<Company> Companies { get; set; }
    public DbSet<Ads> Adses { get; set; }
    public DbSet<FileUpload> FileUploads { get; set; }

    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        modelBuilder.Conventions.Remove<PluralizingTableNameConvention>();
        modelBuilder.Entity<Ads>()
            .HasOptional(a => a.Picture)
            .WithMany()
            .HasForeignKey(a => a.PictureId)
            .WillCascadeOnDelete();

        modelBuilder.Entity<Company>()
           .HasOptional(a => a.Logo)
           .WithMany()
           .HasForeignKey(a => a.LogoId)
           .WillCascadeOnDelete();

        modelBuilder.Entity<Company>()
           .HasOptional(a => a.Catalog)
           .WithMany()
           .HasForeignKey(a => a.CatalogId)
           .WillCascadeOnDelete();
    }
}
```

اینو اگه ازش migration بگیرین متوجه اروراش میشین
من میخوام هر شرکت بتونه به صورت optional لوگو یا کاتالوگ داشته باشه و همین طور قسمت تبلیغات هم همین طور!
همین طور موقعی هم که مثلا یه شرکت پاک میشه لوگو و کاتالوگشم تو جدول FileUpload پاک شه (cascade delete)
همین طور هر رکورد FileUpload رو بتونم به صورت مستقیم حذف کنم
میشه این کارا که گفتمو کرد؟! چون واقعا یه همچین چیزی نیازه !

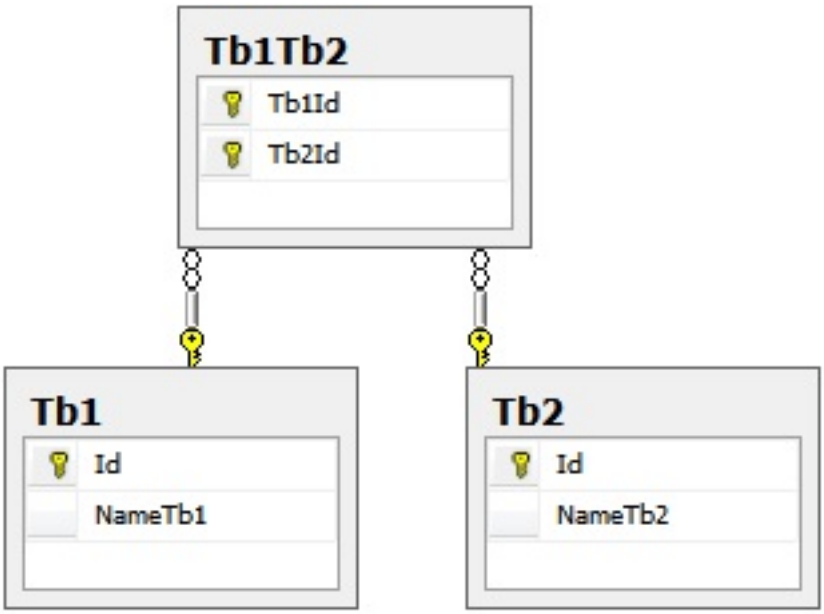
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۲۰ ۱۴:۰

طراحی رو می‌تونید ساده‌تر کنید با قابلیت توسعه بعدی. کلاس Ads رو حذف کنید. خواص لوگو و کاتالوگ رو هم حذف کنید. یک خاصیت به نام FileType به کلاس FileUpload اضافه کنید که می‌تونه تبلیغ، کاتالوگ، لوگو و بسیاری موارد دیگر که در آینده اضافه خواهند شد، باشد. بنابراین این FileType نیاز به یک کلاس جداگانه خواهد داشت برای مدیریت بهتر. استفاده از Enum هم پیشنهاد نمی‌شود چون توسط برنامه و کاربر قابل ویرایش نیست. در آخر یک خاصیت لیستی File هم از نوع FileUpload به کلاس شرکت اضافه کنید.

پ.ن.
این نوع سؤالات شخصی را لطفا در انجمن‌ها پیگیری کنید.

نویسنده: کیا
تاریخ: ۱۳۹۱/۰۸/۲۱ ۱۳:۱۶

من 3تا جدول زیر رو در بانک ساختم :



و کلاسها به صورت زیر تعریف کردم:

```
public class Tb1
    {
        public Tb1()
        {
            ListTb2 = new List<Tb2>();
        }
        public int Id { get; set; }
        public string NameTb1 { get; set; }

        public virtual ICollection<Tb2> ListTb2 { get; set; }
    }
    public class Tb2
    {
        public Tb2()
        {
            ListTb1 = new List<Tb1>();
        }
        public int Id { get; set; }
        public string NameTb2 { get; set; }

        public virtual ICollection<Tb1> ListTb1 { get; set; }
    }
```

و همینطور mapping :

```
public class Tb1Map : EntityTypeConfiguration<Tb1>
    {
        public Tb1Map()
        {
            this.HasKey(x => x.Id);

            this.HasMany(x => x.ListTb2)
```

```
                .WithMany(xx => xx.ListTb1)
                .Map
                (
                    x =>
                        {
                            x.MapLeftKey("Tb1Id");
                            x.MapRightKey("Tb2Id");
                            x.ToTable("Tb1Tb2");
                        }
                );

        }
    }

    public class Tb2Map : EntityTypeConfiguration<Tb2>
    {
        public Tb2Map()
        {
            this.HasKey(x => x.Id);
        }
    }
```

موقعی که در برنامه به صورت زیر استفاده می‌کنم:

```
var sv1 = new TableService<Tb1>(_uow);
            var sv2 = new TableService<Tb2>(_uow);

            var t1 = new Tb1 { NameTb1 = "T111" };
            sv1.Add(t1);
            //var res1= _uow.SaveChanges();

            var t2 = new Tb2 { NameTb2 = "T222" };
            sv2.Add(t2);
            //var res2 = _uow.SaveChanges();

            t1.ListTb2.Add(t2);
            var result = _uow.SaveChanges();
```

هنگام SaveChanges این خطا رو 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 entity cannot be identified
as the source of the exception. Handling of exceptions while saving can be made easier by exposing
foreign key properties in your entity types. See the InnerException for details.
```

همراه با innerException زیر:

```
{"The INSERT statement conflicted with the FOREIGN KEY constraint \"FK_Tb1Tb2_Tb2\". The conflict
occurred in database \"dbTest\", table \"dbo.Tb2\", column 'Id'.\r\nThe statement has been
terminated."}
```

در واقع همینطور که مشخصه من می‌خوام اون جدول رابطه رو در codeFirst حذف کنم یجورایی و رابطه رو بین 2 جدول اصلی بیارم. کجای کارم اشتباهه؟ و راهکارش چیه؟
من با پروفایلر هم نگاه کردم همه چی تا آخر داره پیش می‌ره!
(آیا ForeignKey رو باید طور دیگه ای تعریف کنم؟)
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۲۱ ۱۳:۵۷

مثال شما رو به صورت مستقل اجرا کردم، جداول ساخته شدند، رکوردها در هر سه جدول ثبت شدند و مشکلی مشاهده نمیشه.
اگر برای دیتابیس موجود قصد دارید mapping تعریف کنید ممکن است کلیدهای تعریف شده در آن کم یا زیاد باشند. بهتر است یک خروجی مستقل از کلاس‌های فوق تهیه کنید (اجازه بدید EF دیتابیس را تولید کند) و بعد با کار خودتون مقایسه کنید که چه

چیزهایی را کم و زیاد دارد.
اگر code-first عمل می‌کنید و دیتابیس قرار است از روی کدهای فوق تهیه شود، تمام نگاشت‌ها را حذف کنید (کلاس‌های Map تعریف شده را)، EF به راحتی روابط man-to-many را تشخیص داده و کلیدهای خارجی و جدول واسط را تهیه می‌کند. نام‌های پیش فرض آن هم از نظر من بسیار مناسب است و نیازی به تغییر ندارند. (تنها تغییری که با بودن کلاس‌های Map فوق حاصل میشه، تعیین نام فیلدهای جدول واسط است و زمانیکه code-first کار می‌کنید این نام‌ها مهم نیستند؛ چون با LINQ نهایتا قرار است کار کنید و خواص کلاس‌ها)

نویسنده: کیا
تاریخ: ۱۳۹۱/۰۸/۲۱ ۱۴:۲۷

ولی با این مثالی که زدم برای من رکوردها ثبت نشدند که! :-؟

اما این راههایی که گفتید رو رفتم بیشترشو و در نهایت مشکل با حذف خط 9 حل شد.
فکر کنم مشکل از 2بار ثبت شدن **موجودیتی تکراری از Tb2** در این جدول باشه! (گرچه علت رو نفهمیدم فکر کنم هنوز!) یکبار در خط 9 که مستقیما موجودیت را به نشانه Added علامت گذاری می‌کنم در ChangeTracker :

```
sv2.Add(t2);
```

و یکبار در خط 12 که همان موجودیت را در navigation property مربوط به موجودیتی از Tb1 مقدار دهی می‌کنم و در خط بعدش که SaveChanges را فراخوانی می‌کنم :

```
t1.ListTb2.Add(t2);
```

(گرچه فکر می‌کردم یک خطای منطقی باید رخ بدهد و حداقل 2بار بصورت duplicate و با id متفاوت ثبت بشه نه اینکه به اینصورت خطا بگیره)

من CodeFirst کار می‌کنم که البته بیشتر اوقات بانک ساخته شده هست و باید به بانک موجود mapping کنم. بیشتر می‌خواستم نحوه عملکرد ef رو روی روابط چند-ب-چند ببینم و اینکه روی یک بانک از قبل آماده به چه صورت باید انجامش بدم. از Ef PowerTools کمک می‌گیرم و روی مسایل حول اینها مشغول تست و کلنجار هستم. مخصوصا حالت‌های مختلف در روابط و جداول رابطه.
تشکر مجدد

نویسنده: علیرضا
تاریخ: ۱۳۹۲/۰۱/۲۱ ۹:۲۰

سلام؛ من با این روشی که شما فرمودید کار کردم ولی موقع آپدیت با خطای زیر متوقف میشه.
A referential integrity constraint violation occurred: The property values that define the referential constraints are not consistent between principal and dependent objects in the relationship.

دقیقا هم زمانی که می‌خوام State رو به Modified تغییر بدم این خطا رخ می‌ده.اما وقتی که هم foreign key رو مقدار می‌دم و هم مقدار شی navigation property رو از DB واکشی می‌کنم و مقدار دهی می‌کنم درست کار می‌کنه.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۲۱ ۹:۵۰

- کدهای این مطلب قبل از ارسال، آزمایش شده‌اند و همین کدهایی را که ملاحظه می‌کنید بدون مشکل کار می‌کنند.
- روش صحیح سؤال پرسیدن این است که مشخص کنید چه مدلی، چه کدی، چه مقادیری در حال استفاده هستند تا این خطا رخ

داده. (امکان باز تولید یک استثناء رو باید بتونید فراهم کنید)
- این خطا زمانی رخ می‌ده که مقادیر کلید اصلی و کلید خارجی در حال ثبت یکی نباشند.

نویسنده: محمد صاحب
تاریخ: ۱۳۹۲/۱۱/۱۲ ۱۵:۰

برای مواردی که کلید اصلی Identity نباشه راه حلی هست ؟

کد

```
namespace TestKeys
{
    class Program
    {
        public class Bill
        {
            [DatabaseGenerated(DatabaseGeneratedOption.None)]
            public string Id { get; set; }
            public decimal Amount { set; get; }
            [ForeignKey("AccountId")]
            public virtual Account Account { get; set; }
            public string AccountId { set; get; }
        }

        public class Account
        {
            [DatabaseGenerated(DatabaseGeneratedOption.None)]
            public string Id { get; set; }
            public string Name { get; set; }
        }

        public class MyContext : DbContext
        {
            public DbSet<Bill> Bills { get; set; }
            public DbSet<Account> Accounts { get; set; }
        }


        public class BillFromWebsrv
        {
            public string Id { get; set; }
            public decimal Amount { set; get; }
            public DateTime DateTime { get; set; }

            public Account Account { get; set; }

        }



        static void Main(string[] args)
        {
            Database.SetInitializer(new DropCreateDatabaseIfModelChanges<MyContext>());
            using (var ctx = new MyContext())
            {

                foreach (var dummyBill in DummyBills())
                {
                    var bl = new Bill { Id = dummyBill.Id, Amount = dummyBill.Amount, Account =
dummyBill.Account };

                    ctx.Bills.Add(bl);
                }
                ctx.SaveChanges();
            }


        }

        public static List<BillFromWebsrv> DummyBills()
        {
            return new List<BillFromWebsrv>
            {
                new BillFromWebsrv
                {
```

```
                    Id = "1",
                    Amount = 1231,
                    DateTime = DateTime.Now,
                    Account = new Account {Id = "1", Name = "ac1"}
                },
                new BillFromWebsrv
                {
                    Id = "2",
                    Amount = 1232,
                    DateTime = DateTime.Now,
                    Account = new Account {Id = "2", Name = "ac2"}
                },
                new BillFromWebsrv
                {
                    Id = "3",
                    Amount = 1233,
                    DateTime = DateTime.Now,
                    Account = new Account {Id = "2", Name = "ac2"}
                },
                new BillFromWebsrv
                {
                    Id = "4",
                    Amount = 1134,
                    DateTime = DateTime.Now,
                    Account = new Account {Id = "3", Name = "ac3"}
                }
            };
        }
    }
}
```

ارور

```
{"Violation of PRIMARY KEY constraint 'PK_dbo.Accounts'. Cannot insert duplicate key in object
'dbo.Accounts'. The duplicate key value is (2).\r\nThe statement has been terminated."}
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۱۲ ۱۸:۴۹

Account هایی که اضافه می‌کنید تحت نظر Context نیستند؛ یک شیء ساده هستند که عنوان کردید، Add اش کن؛ EF هم دقیقا همینکار را انجام داده. زمانیکه به سومین رکورد با اکانتی دارای id=2 می‌رسد، کار متوقف می‌شود چون این id قبلا در رکورد قبلی سعی شده در بانک اطلاعاتی ذخیره شود.

بنابراین باید Context را بررسی کرد که Account در حال اضافه شدن، آیا قبلا در کش Local آن وجود خارجی داشته یا خیر. به این صورت:

```
using (var ctx = new MyContext())
            {
                foreach (var dummyBill in DummyBills())
                {
                    var account = dummyBill.Account;

                    var entry = ctx.Entry<Account>(account);
                    if (entry.State == EntityState.Detached)
                    {
                        var attachedEntity = ctx.Accounts.Local.SingleOrDefault(e => e.Id ==
account.Id);
                        if (attachedEntity != null)
                        {
                            // یعنی قبلا تحت نظر زمینه جاری قرار گرفته و نیازی به ثبت مجدد آن نیست
                            account = attachedEntity;
                        }
                    }

                    var bl = new Bill { Id = dummyBill.Id, Amount = dummyBill.Amount, Account = account
};
                    ctx.Bills.Add(bl);
                }
                ctx.SaveChanges();
            }
```

اگر قبلا تحت نظر قرار گرفته (در کش Accounts.Local موجود است)، باید از همان وهله موجود در Context استفاده شود و نه اینکه یک شیء جدید منقطع را درخواست داد که مجددا ثبت شود.