عنوان: **خودکار کردن تعاریف DbSetها در EF Code first**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۲۷ ۲۱:۵
آدرس: www.dotnettips.info
گروه‌ها: Entity framework

پیشنیاز:
تعریف نوع جنریک به صورت متغیر

مطلبی را چندی قبل در مورد نحوه خودکار کردن افزودن کلاس‌های EntityTypeConfiguration به modelBuilder در این سایت مطالعه کردید . در مطلب جاری به خودکار سازی تعاریف مرتبط با DbSetها خواهیم پرداخت.
ابتدا مثال کامل زیر را درنظر بگیرید:

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

namespace MyNamespace
{
    public abstract class BaseEntity
    {
        public int Id { set; get; }
        public string CreatedBy { set; get; }
    }

    public class User : BaseEntity
    {
        public string Name { get; set; }
    }

    public class MyContext : DbContext
    {
        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            var asm = Assembly.GetExecutingAssembly();
            loadEntities(asm, modelBuilder, "MyNamespace");
        }

        void loadEntities(Assembly asm, DbModelBuilder modelBuilder, string nameSpace)
        {
            var entityTypes = asm.GetTypes()
                                 .Where(type => type.BaseType != null &&
                                        type.Namespace == nameSpace &&
                                        type.BaseType.IsAbstract &&
                                        type.BaseType == typeof(BaseEntity))
                                 .ToList();

            var entityMethod = typeof(DbModelBuilder).GetMethod("Entity");
            entityTypes.ForEach(type =>
            {
                entityMethod.MakeGenericMethod(type).Invoke(modelBuilder, new object[] { });
            });
        }
    }

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

        protected override void Seed(MyContext context)
        {
            context.Set<User>().Add(new User { Name = "name-1" });
            context.Set<User>().Add(new User { Name = "name-2" });
            context.Set<User>().Add(new User { Name = "name-3" });
            base.Seed(context);
        }
    }

    public static class Test
```

```
    {
        public static void RunTests()
        {
            Database.SetInitializer(new MigrateDatabaseToLatestVersion<MyContext, Configuration>());
            using (var context = new MyContext())
            {
                var user1 = context.Set<User>().Find(1);
                if (user1 != null)
                    Console.WriteLine(user1.Name);
            }
        }
    }
}
```

توضیحات:

همانطور که ملاحظه می‌کنید در این مثال خبری از تعاریف DbSetها نیست. به کمک Reflection تمام مدل‌های برنامه که از نوع کلاس پایه BaseEntity هستند (روشی مرسوم جهت مدیریت خواص تکراری مدل‌ها) یافت شده (در متد loadEntities) و سپس نتیجه حاصل به صورت پویا به متد جنریک Entity ارسال می‌شود. حاصل، افزوده شدن خودکار کلاس‌های مورد نظر به سیستم EF است.

البته در این حالت چون دیگر کلاس‌های مدل‌ها در MyContext به صورت صریح تعریف نمی‌شوند، نحوه استفاده از آن‌ها را توسط متد Set، در متدهای RunTests و یا Seed، ملاحظه می‌کنید.

نظرات خوانندگان

نویسنده: محسن.د
تاریخ: ۱۳۹۱/۰۸/۲۷ ۲۳:۴۴

البته کد

```
var asm = Assembly.GetExecutingAssembly();
```

در صورتی کار میکنه که مدل‌ها در همین پروژه باشند ولی اگر در یک پروژه جداگانه تعریف شده باشند باید از

```
var asm = Assembly.GetAssembly(typeof(DomainModels.BaseEntity));
```

استفاده کرد . که DomainModels.BaseEntity یکی از کلاس‌های موجود در اسمبلی مربوط به مدلها باید باشد .

نویسنده: ali mazinani
تاریخ: ۱۳۹۱/۰۸/۲۸ ۱۴:۳۹

سلام آیا این کار سربار اضافی نداره؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۲۸ ۱۶:۱۴

نه. اینکار فقط یکبار در آغاز برنامه انجام می‌شود. نتیجه کار هم توسط EF کش خواهد شد.

نویسنده: hoseini
تاریخ: ۱۳۹۱/۱۲/۰۴ ۲۳:۲۴

سلام
این متد کار افزودن نگاشت‌های کلاس‌ها رو هم انجام میده ؟
و اگر بخواد انجام بده باید چه تغییراتی روی کد انجام بشه؟
ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۰۴ ۲۳:۳۶

« افزودن خودکار کلاس‌های تنظیمات نگاشت‌ها در EF Code first »

نویسنده: شایان مرادی
تاریخ: ۱۳۹۱/۱۲/۲۵ ۱۵:۵۱

سلام
اگر مدلها در چندین پروژه بودن چطور؟
مثلا به صورت ماژول هر ماژول مدل خود را دارد

نویسنده: وحید نصیری

تاریخ: ۱۳۹۱/۱۲/۲۵ ۱۷:۱۷

الف)
- تمام مدل‌های شما باید از کلاس مشخصی مثلا BaseEntity مشتق شوند.
- یا در تنظیمات برنامه مشخص کنید در حین Reflection چه فضای نامی باید جستجو شود.
- و یا مثلا مانند ASP.NET MVC اگر کلاسی نامش به عبارت خاصی ختم شد و همچنین از کلاس پایه خاصی نیز مشتق شده بود آنگاه بررسی شود.
ب)
- مرحله بعد اندکی ویرایش متد loadEntities است جهت خواندن مدل‌های واقع شده مثلا در یک فضای نام خاص یا مشتق شده از یک کلاس پایه خاص و سپس افزودن خودکار آن‌ها به modelBuilder همانند چیزی که در مثال فوق پیاده سازی شده.
در مثال بالا فقط یک اسمبلی جستجو می‌شود؛ اگر نیاز است تمام اسمبلی‌های بارگذاری شده در برنامه جستجو شوند، روش کار مراجعه به AppDomain است:

```
foreach (Assembly currentassembly in AppDomain.CurrentDomain.GetAssemblies())
{
   Type t = currentassembly.GetType("typeName", false, true);
    if (t != null) {return currentassembly.FullName;}
}
```

نویسنده: شایان مرادی
تاریخ: ۱۳۹۱/۱۲/۲۵ ۱۸:۱۱

ممنونم از جوابتون

AppDomain.CurrentDomain.GetAssemblisl اسمبلی‌ها را از پوشه Bin اضافه می‌نماید یا اینکه جستجویی در کل Libaryها انجام می‌دهد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۲۵ ۱۸:۲۰

خیر. چیزی را اضافه یا بارگذاری نمی‌کند. فقط در AppDomain برنامه، کل اسمبلی‌ها و ماژول‌های بارگذاری شده موجود را لیست می‌کند .

نویسنده: شایان مرادی
تاریخ: ۱۳۹۱/۱۲/۲۶ ۱۸:۲۳

سلام
من به این طریق Dbsetها رو اضافه میکنم و مشکلی پیش نمیاد . اما وقتی میخوام نگاشت‌ها را به وسیله توضیحات این لینک انجام بدم به مشکل میخورم
من ابتدا Dbsetها را با این کد شما اضافه میکنم سپس تابع اضافه کردن نگاشت‌ها را اجرا میکنم
اما به مشکل میخورم
ممنون میشم اگه راهنمایی نمایید به چه شکل باید هم موجودیت‌ها و هم نگاشت‌های آن‌ها را همزمان انجام داد
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۲۶ ۱۹:۱

عکس عمل کنید. ابتدا تنظیمات، بعد موجودیت‌ها اضافه شوند.
ضمنا اگر احیانا گذارتان به انجمنی افتاد این «به مشکل برخوردم» رو باید توضیح بدید. باید خطای حاصل رو ذکر کنید (علاوه بر روشی که طی شده).

نویسنده: مهدی فرهانی
تاریخ: ۱۳۹۲/۰۱/۰۹ ۱۳:۵۹

با سلام
چندوقتی هست که دارم از این روش برای برنامه هام استفاده میکنم و هیچ مشکلی نداشتم تا اینکه دیشب با یک مشکل عجیب برخورد کردم و مشکل این بود که زمان ساخت جداول به Property که به صورت NotMaped در کلاس Base تعریف کرده بودم ایراد گرفت.پس از بررسی که انجام دادم متوجه شدم که ترتیب Map کردن Entityها باعث این مشکل میشه.
http://entityframework.codeplex.com/workitem/481

این لینک مشکلی که گزارش شده، راه حلی که نوشته شده این که ترتیب Map کردن کلاس‌ها باید تغییر کند.این درحالی است که در این روش هیچ ترتیبی در نظر گرفته نمیشه و براساس خواندن کلاس‌ها از اسمبلی ساخته میشه.
برای رفع این مشکل اومدم و یک ویژگی ترتیب به کلاسهایی که میدونستم ترتیبشون مهم هست اضافه کردم و زمان خواندن کلاس‌ها از اسمبلی مورد نظر اونها مرتب کردم و مشکل حل شد.
آیا راه حل بهتری وجود داره یا نه ؟

```
var entityTypes = modelAssembly.GetTypes()
                                    .Where(type => type.Namespace != null
                                        && (type.Namespace.StartsWith(domainNamespace)
                                            && type.CustomAttributes.All(c => c.AttributeType != 
typeof(ComplexTypeAttribute))
                                            && !type.IsAbstract && !type.IsEnum)
                                        )
                                    .OrderBy(c => c.CustomAttributes.Any(
                                        x => x.AttributeType == typeof (EntityOrderAttribute)) ? 
c.GetCustomAttribute<EntityOrderAttribute>().OrderNumber : 100).ToList();
```

نویسنده: محسن
تاریخ: ۱۳۹۲/۰۱/۱۰ ۹:۹

این مساله در حالت معمولی تعریف DbSetها هم پیش میاد. یعنی الزاما محدود به روش گفته شده در این مطلب نیست. خودشون هم پذیرفتن که یک باگ است و در حال رفعش هستند.

نویسنده: پدرام جباری
تاریخ: ۱۳۹۲/۰۲/۱۰ ۲۱:۲۷

با سلام

من برای پروژه ای که در حال انجام اون هستم چون تعداد جداول زیاد هست برای آنکه زمان نمونه ساختن هر context کاهش پیدا کنه، برای هر بخش از پروژه Context متفاوتی ایجاد کردم تا زمان نگاشت تنظیمات جداول کاهش پیدا کنه ، البته یک Context کلی هست که اون وظیفه ساخت دیتابیس رو داره (داخل EF 6 این امکان هست که یک دیتابیس از چند Context متفاوت ایجاد شده باشه ، ولی خوب برای این کار تو EF5 باید یک Context کلی داشت که مشکلی ایجاد نشه)
به نظر شما کار درستی هست ؟ اصلا تاثیری داره ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۱۰ ۲۱:۵۳

- بحث جاری در مورد EF 5 هست. کل مباحثی که در سایت مطرح شده در مورد تا قبل از EF 6 است . (هرچند هدف اصلی EF6 از بحث چند Context ایی پوشش دادن ایجاد جداول مختلف به ازای Schemaهای مختلف است. پیشتر dbo بیشتر مد نظر بود (پشتیبانی از تک schema به ازای یک دیتابیس). الان پوشش چندین Schema با هم در طی چندین Context مختلف در یک بانک اطلاعاتی)
- تاثیری نداره. نگاشت‌ها فقط یکبار در آغاز کار برنامه انجام می‌شوند و بعد کش خواهند شد. هر بار وهله سازی context به

معنای انجام چند باره نگاشت‌ها و یافتن و برقراری آن‌ها نیست. اتفاقا این وهله سازی‌های ثانویه پس از آغاز برنامه، فوق العاده هم سریع هستند.
خلاصه اینکه مباحث مطرح شده در مطلب جاری فقط در آغاز برنامه اجرا می‌شوند و نه به ازای هر بار وهله سازی تک Context برنامه.
- در مورد اینکه چرا باید یک کلاس Context در برنامه داشت اینجا توضیح دادم . بحث الگوی واحد کار مهم‌ترین آن‌ها است.

نویسنده: پدرام جباری
تاریخ: ۱۳۹۲/۰۲/۱۰ ۲۳:۴۳

ممنون بابت جوابتون، واقعا نمی‌دونم چرا همیشه فکر می‌کردم به این صورت نیست و هر بار بخش نگاشت‌ها انجام میشه، تقریبا بیشتر مقاله‌های مربوط به افزایش سرعت رو در EF رو در وبسایت مطالعه کردم و کل روند انجام کار مفهوم شد برام، شاید بیشتر به خاطر عدم اطمینانم به EF بود که همچین فکری کردم.
واقعا ممنون بابت مطالبتون که در وبسایت قرار دادید خیلی به من کمک کرده از همه نظر.

نویسنده: بهنام حقی
تاریخ: ۱۳۹۳/۰۱/۲۸ ۱۲:۳۶

سلام
من یه مشکلی دارم توی این قسمت.
توی پروژه از خودکار سازی نگاشت‌ها و همین بخش استفاده کردم که در نهایت پیغام خطای زیر رو میده:

```
The entity type User is not part of the model for the current context.
```

قسمت نگاشت‌ها همانند همین که تو سایت گفتین استفاده کردم. پروژه شامل سه بخش Domain و Model و Console هست برای تست این دو بخش. قسمت خودکار سازی نگاشت‌ها بدون خود کارسازی تعاریف DbSetها به درستی کار میکنه، وقتی این بخش رو پیاده میکنم پیغام خطای بالا رو میده تو متد Seed واسه دیتاهای پیش فرض User.

```
namespace Test.Domain
{
    public abstract class BaseEntity
    {
        public int Id { set; get; }
    }
}

namespace Test.Domain.Entities
{
    public class User : BaseEntity
    {
        public int UserNumber { get; set; }
        public string Name { get; set; }
    }
}

namespace Test.Domain.Mappings
{
    public class UserMap : EntityTypeConfiguration<User>
    {
        public UserMap()
        {
            this.HasKey(x => x.UserNumber);
            this.Property(x => x.Name).HasMaxLength(450).IsRequired();
        }
    }
}
```

```
loadEntities(asm, modelBuilder, "Test.Domain");
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۲۸ ۱۲:۵۱

درسته. چون کلاس User در فضای نام Test.Domain قرار ندارد.
بررسی انجام شده به صورت type.Namespace == nameSpace است ولی حالت مدنظر شما type.Namespace. **StartsWith** nameSpace است.

نویسنده: بهنام حقی
تاریخ: ۱۳۹۳/۰۱/۲۸ ۱۳:۱۱

خیلی ممنون. برای اینکه همه اسمبلی هارو بگرده، و یه اسمبلی خاص رو بهش معرفی نکنیم، چطور از این تابع استفاده کنم:

```
foreach (Assembly asm in AppDomain.CurrentDomain.GetAssemblies())
{
  //...
}
```

تابع زیر رو به ازای هر فضای نام که میده باید فراخونی کنم؟

```
loadEntities(asm, modelBuilder, "نام فضا");
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۲۸ ۱۳:۲۲

بله. ولی این حالت مقرون به صرفه نیست. چون تعداد اسمبلی‌های بارگذاری شده در یک App domain زیاد است و شامل موارد پایه و سیستمی دات نت هم هست (با هزاران فضای نام). در این حالت این جستجو زمان زیادی را به خود اختصاص خواهد داد. بهتر است لیست یک سری اسمبلی مشخص را درنظر داشته باشید تا اینکه کل سیستم را جستجو کنید.

نویسنده: سدا
تاریخ: ۱۳۹۳/۰۳/۲۱ ۱۷:۱۵

اگر بخوایم خودکار Dbsetها تعریف شه و Models و Mapping در پروژه ای دیگه باشه، متود Seed که نیاز مستقیم به Context و Dbsetها داره. یعنی بر فرض بخوایم یک User به Context.Users اضافه کنیم امکان پذیر نیست درسته؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۳/۲۱ ۱۷:۲۲

از متد Set استفاده کنید. در مثال فوق در متد protected override void Seed نمونه‌اش ارائه شده:

```
context.Set<User>().Add(new User { Name = "name-1" });
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۶ ۲۳:۱۴

یک نکته‌ی تکمیلی

متد **RegisterEntityType** modelBuilder. به نگارش‌های اخیر EF برای افزودن پویای مدل‌ها و موجودیت‌ها اضافه شده‌است و دیگر نیازی به روش Reflection مطرح شده‌ی در این مطلب نیست.