عنوان: **افزودن خودکار کلاس‌های تنظیمات نگاشت‌ها در EF Code first**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۱۰ ۰:۲۱
آدرس: www.dotnettips.info
گروه‌ها: Entity framework

اگر از روش Fluent-API برای تنظیم و افزودن نگاشت‌های کلاس‌ها استفاده کنیم، با زیاد شدن آن‌ها ممکن است در این بین، افزودن یکی فراموش شود یا کلا اضافه کردن دستی آن‌ها در متد OnModelCreating آنچنان جالب نیست. می‌شود این‌کار را به کمک Reflection ساده‌تر و خودکار کرد:

```
void loadEntityConfigurations(Assembly asm, DbModelBuilder modelBuilder, string nameSpace)
        {
            var configurations = asm.GetTypes()
                                    .Where(type => type.BaseType != null &&
                                           type.Namespace == nameSpace &&
                                           type.BaseType.IsGenericType &&
                                           type.BaseType.GetGenericTypeDefinition() ==
typeof(EntityTypeConfiguration<>))
                                    .ToList();

            configurations.ForEach(type =>
               {
                   dynamic instance = Activator.CreateInstance(type);
                   modelBuilder.Configurations.Add(instance);
               });
        }
```

در این متد، در یک اسمبلی مشخص و فضای نامی در آن، به دنبال کلاس‌هایی از نوع EntityTypeConfiguration خواهیم گشت. در ادامه این کلاس‌ها وهله سازی شده و به صورت خودکار به modelBuilder اضافه می‌شوند.

یک مثال کامل که بیانگر نحوه استفاده از متد فوق است:

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

namespace EFGeneral
{
    public class User
    {
        public int UserNumber { get; set; }
        public string Name { get; set; }
    }

    public class UserConfig : EntityTypeConfiguration<User>
    {
        public UserConfig()
        {
            this.HasKey(x => x.UserNumber);
            this.Property(x => x.Name).HasMaxLength(450).IsRequired();
        }
    }

    public class MyContext : DbContext
    {
        public DbSet<User> Users { get; set; }

        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            // modelBuilder.Configurations.Add(new UserConfig());

            var asm = Assembly.GetExecutingAssembly();
            loadEntityConfigurations(asm, modelBuilder, "EFGeneral");
        }

        void loadEntityConfigurations(Assembly asm, DbModelBuilder modelBuilder, string nameSpace)
        {
            var configurations = asm.GetTypes()
                                    .Where(type => type.BaseType != null &&
```

```
                                        type.Namespace == nameSpace &&
                                        type.BaseType.IsGenericType &&
                                        type.BaseType.GetGenericTypeDefinition() ==
typeof(EntityTypeConfiguration<>))
                                    .ToList();

            configurations.ForEach(type =>
               {
                   dynamic instance = Activator.CreateInstance(type);
                   modelBuilder.Configurations.Add(instance);
               });
        }
    }

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

        protected override void Seed(MyContext context)
        {
            context.Users.Add(new User { Name = "name-1" });
            context.Users.Add(new User { Name = "name-2" });
            context.Users.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.Users.Find(1);
                if (user1 != null)
                    Console.WriteLine(user1.Name);
            }
        }
    }
}
```

در این مثال، در متد OnModelCreating بجای اضافه کردن دستی تک تک تنظیمات تعریف شده، از متد loadEntityConfigurations جهت یافتن آن‌ها در اسمبلی جاری و فضای نام مشخصی به نام EFGeneral استفاده شده است.

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۱۳ ۱۳:۵۹

یک نکته تکمیلی

در EF 6 متد ذیل جهت پوشش این مطلب اضافه شده‌است:

```
modelBuilder.Configurations
      .AddFromAssembly(Assembly.GetExecutingAssembly())
```

نویسنده: سدا
تاریخ: ۱۳۹۳/۰۳/۲۱ ۱۳:۴۵

اگر نگاشت در مسیر DomainClasses.Mappings باشه چه تغییری باید ایجاد کنیم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۳/۲۱ ۱۳:۵۳

از AppDomain برای یافتن اسمبلی مدنظر استفاده کنید:

```
var enumerable = AppDomain.CurrentDomain.GetAssemblies()
        .SelectMany(assembly => assembly.GetTypes())
        .Select(type => type.Namespace)
        .Distinct()
        .Where(name => name != null &&
                       name == "DomainClasses.Mappings");
```

نویسنده: سدا
تاریخ: ۱۳۹۳/۰۳/۲۱ ۱۶:۵۶

در حالت اینکه از BaseEntity استفاده شه مشکلی نیست . ولی یه نوع وابستگی میشه... درسته؟
من در حالت BaseEntity جواب گرفتم. اما حالتی که فضای نام رو جستجو کنه خیلی منطقی‌تر و انعطاف پذیرتره.
ولی دقیقا شکل پیاده سازیش به مشکل برخوردم از Dbset خودکار که تو سایت هست استفاده کردم اما موفق نشدم.

امکان داره بگید زمانی که تمامی اسمبلی‌ها با دستور فوق در متغیر تعریف شده ذخیره شدن چطور در ModelBinder ست کنم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۳/۲۱ ۱۷:۹

- کمی بالاتر عنوان شد: modelBuilder.Configurations.AddFromAssembly
- مطلب فوق در مورد کلاس‌های مشتق شده از EntityTypeConfiguration نوشته شد. این کلاس‌ها در EF همیشه به همین شکل هستند؛ مگر اینکه طراحی کل آن تغییر کند.
- در مطلب دیگری که به خودکار کردن تعاریف DbSetها اختصاص داشت، سطر ذیل را از آن مطابق نیاز خودتان، حذف کنید:

```
type.BaseType == typeof(BaseEntity)
```

بررسی فضای نام را هم دارد.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۶ ۲۳:۱۶

یک نکته‌ی تکمیلی

اگر مدل‌ها را نیز به صورت پویا اضافه می‌کنید ، ترتیب این فراخوانی‌ها باید به صورت زیر باشد:
ابتدا modelBuilder.Configurations.AddFromAssembly و بعد modelBuilder.RegisterEntityType