عنوان: EF Code First #6
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۱۹ ۰۰:۱۳:۰۰
آدرس: www.dotnettips.info
گروه‌ها: Entity framework

ادامه بررسی Fluent API جهت تعریف نگاشت کلاس‌ها به بانک اطلاعاتی

در قسمت‌های قبل با استفاده از متادیتا و data annotations جهت بررسی نحوه نگاشت اطلاعات کلاس‌ها به جداول بانک اطلاعاتی آشنا شدیم. اما این موارد تنها قسمتی از توانایی‌های Fluent API مهیا در EF Code first را ارائه می‌دهند. یکی از دلایل آن هم به محدود بودن توانایی‌های ذاتی Attributes بر می‌گردد. برای مثال حین کار با Attributes امکان استفاده از متغیرها یا lambda expressions و امثال آن وجود ندارد. به علاوه شاید عده‌ای علاقمند نباشند تا کلاس‌های خود را با data annotations شلوغ کنند.

در قسمت دوم این سری، مروری مقدماتی داشتیم بر Fluent API. در آنجا ذکر شد که امکان تعریف نگاشت‌ها به کمک توانایی‌های Fluent API به دو روش زیر میسر است:
الف) می‌توان از متد protected override void OnModelCreating در کلاس مشتق شده از DbContext کار را شروع کرد.
ب) و یا اگر بخواهیم کلاس Context برنامه را شلوغ نکنیم بهتر است به ازای هر کلاس مدل برنامه، یک کلاس mapping مشتق شده از EntityTypeConfiguration را تعریف نمائیم. سپس می‌توان این کلاس‌ها را در متد OnModelCreating یاد شده، توسط متد modelBuilder.Configurations.Add جهت استفاده و اعمال، معرفی کرد.

کلاس‌های مدلی را که در این قسمت بررسی خواهیم کرد، همان کلاس‌های User و Project قسمت سوم هستند و هدف این قسمت بیشتر تطابق Fluent API با اطلاعات ارائه شده در قسمت سوم است؛ برای مثال در اینجا چگونه باید از خاصیتی صرفنظر کرد، مسایل همزمانی را اعمال نمود و امثال آن.
بنابراین یک پروژه جدید کنسول را آغاز نمائید. سپس با کمک NuGet ارجاعات لازم را به اسمبلی‌های EF اضافه نمائید.
در پوشه Models این پروژه، سه کلاس تکمیل شده زیر، از قسمت سوم وجود دارند:

```
using System;
using System.Collections.Generic;

namespace EF_Sample03.Models
{
    public class User
    {
        public int Id { set; get; }
        public DateTime AddDate { set; get; }
        public string Name { set; get; }
        public string LastName { set; get; }

        public string FullName
        {
            get { return Name + " " + LastName; }
        }

        public string Email { set; get; }
        public string Description { set; get; }
        public byte[] Photo { set; get; }
        public IList<Project> Projects { set; get; }
        public byte[] RowVersion { set; get; }
        public InterestComponent Interests { set; get; }

        public User()
        {
            Interests = new InterestComponent();
        }
    }
}
```

```
using System;

namespace EF_Sample03.Models
{
    public class Project
    {
        public int Id { set; get; }
        public DateTime AddDate { set; get; }
        public string Title { set; get; }
        public string Description { set; get; }
        public virtual User User { set; get; }
        public byte[] RowVesrion { set; get; }
    }
}
```

```
namespace EF_Sample03.Models
{
    public class InterestComponent
    {
        public string Interest1 { get; set; }
        public string Interest2 { get; set; }
    }
}
```

سپس یک پوشه جدید به نام Mappings را به پروژه اضافه نمائید. به ازای هر کلاس فوق، یک کلاس جدید را جهت تعاریف اطلاعات نگاشت‌ها به کمک Fluent API اضافه خواهیم کرد:

```
using System.Data.Entity.ModelConfiguration;
using EF_Sample03.Models;

namespace EF_Sample03.Mappings
{
    public class InterestComponentConfig : ComplexTypeConfiguration<InterestComponent>
    {
        public InterestComponentConfig()
        {
            this.Property(x => x.Interest1).HasMaxLength(450);
            this.Property(x => x.Interest2).HasMaxLength(450);
        }
    }
}
```

```
using System.Data.Entity.ModelConfiguration;
using EF_Sample03.Models;

namespace EF_Sample03.Mappings
{
    public class ProjectConfig : EntityTypeConfiguration<Project>
    {
        public ProjectConfig()
        {
            this.Property(x => x.Description).IsMaxLength();
            this.Property(x => x.RowVesrion).IsRowVersion();
        }
    }
}
```

```
using System.Data.Entity.ModelConfiguration;
using EF_Sample03.Models;
using System.ComponentModel.DataAnnotations;

namespace EF_Sample03.Mappings
{
```

```
    public class UserConfig : EntityTypeConfiguration<User>
    {
        public UserConfig()
        {
            this.HasKey(x => x.Id);
            this.Property(x => x.Id).HasDatabaseGeneratedOption(DatabaseGeneratedOption.Identity);
            this.ToTable("tblUser", schemaName: "guest");
            this.Property(p =>
p.AddDate).HasColumnName("CreateDate").HasColumnType("date").IsRequired();
            this.Property(x => x.Name).HasMaxLength(450);
            this.Property(x => x.LastName).IsMaxLength().IsConcurrencyToken();
            this.Property(x => x.Email).IsFixedLength().HasMaxLength(255); //nchar(128)
            this.Property(x => x.Photo).IsOptional();
            this.Property(x => x.RowVersion).IsRowVersion();
            this.Ignore(x => x.FullName);
        }
    }
}
```

توضیحاتی در مورد کلاس‌های تنظیمات نگاشت‌های خواص به جداول و فیلدهای بانک اطلاعاتی

نظم بخشیدن به تعاریف نگاشت‌ها

همانطور که ملاحظه می‌کنید، جهت نظم بیشتر پروژه و شلوغ نشدن متد OnModelCreating کلاس Context برنامه، که در ادامه کدهای آن معرفی خواهد شد، به ازای هر کلاس مدل، یک کلاس تنظیمات نگاشت‌ها را اضافه کرده‌ایم.
کلاس‌های معمولی نگاشت‌ها ازکلاس EntityTypeConfiguration مشتق خواهند شد و جهت تعریف کلاس InterestComponent به عنوان Complex Type، اینبار از کلاس ComplexTypeConfiguration ارث بری شده است.

تعیین طول فیلدها

در کلاس InterestComponentConfig، به کمک متد HasMaxLength، همان کار ویژگی MaxLength را می‌توان شبیه سازی کرد که در نهایت، طول فیلد nvarchar تشکیل شده در بانک اطلاعاتی را مشخص می‌کند. اگر نیاز است این فیلد nvarchar از نوع max باشد، نیازی به تنظیم خاصی نداشته و حالت پیش فرض است یا اینکه می‌توان صریحا از متد IsMaxLength نیز برای معرفی nvarchar max استفاده کرد.

تعیین مسایل همزمانی

در قسمت سوم با ویژگی‌های ConcurrencyCheck و Timestamp آشنا شدیم. در اینجا اگر نوع خاصیت byte array بود و نیاز به تعریف آن به صورت timestamp وجود داشت، می‌توان از متد IsRowVersion استفاده کرد. معادل ویژگی ConcurrencyCheck در اینجا، متد IsConcurrencyToken است.

تعیین کلید اصلی جدول

اگر پیش فرض‌های EF Code first مانند وجود خاصیتی به نام Id یا ClassName+Id رعایت شود، نیازی به کار خاصی نخواهد بود. اما اگر این قراردادها رعایت نشوند، می‌توان از متد HasKey (که نمونه‌ای از آن‌را در کلاس UserConfig فوق مشاهده می‌کنید)، استفاده کرد.

تعیین فیلدهای تولید شده توسط بانک اطلاعاتی

به کمک متد HasDatabaseGeneratedOption، می‌توان مشخص کرد که آیا یک فیلد Identity است و یا یک فیلد محاسباتی ویژه و یا هیچکدام.

تعیین نام جدول و schema آن

اگر نیاز است از قراردادهای نامگذاری خاصی پیروی شود، می‌توان از متد ToTable جهت تعریف نام جدول متناظر با کلاس جاری استفاده کرد. همچنین در اینجا امکان تعریف schema نیز وجود دارد.

تعیین نام و نوع سفارشی فیلدها

همچنین اگر نام فیلدها نیز باید از قراردادهای دیگری پیروی کنند، می‌توان آن‌ها را به صورت صریح توسط متد HasColumnName معرفی کرد. اگر نیاز است این خاصیت به نوع خاصی در بانک اطلاعاتی نگاشت شود، باید از متد HasColumnType کمک گرفت. برای مثال در اینجا بجای نوع datetime، از نوع ویژه date استفاده شده است.

معرفی فیلدها به صورت nchar بجای nvarchar

برای نمونه اگر قرار است هش کلمه عبور در بانک اطلاعاتی ذخیره شود، چون طول آن ثابت می‌باشد، توصیه شده‌است که بجای nvarchar از nchar برای تعریف آن استفاده شود. برای این منظور تنها کافی است از متد IsFixedLength استفاده شود. در این حالت طول پیش فرض 128 برای فیلد درنظر گرفته خواهد شد. بنابراین اگر نیاز است از طول دیگری استفاده شود، می‌توان همانند سابق از متد HasMaxLength کمک گرفت.

ضمنا این فیلدها همگی یونیکد هستند و با n شروع شده‌اند. اگر می‌خواهید از varchar یا char استفاده کنید، می‌توان از متد IsUnicode با پارامتر false استفاده کرد.

معرفی یک فیلد به صورت null پذیر در سمت بانک اطلاعاتی

استفاده از متد IsOptional، فیلد را در سمت بانک اطلاعاتی به صورت فیلدی با امکان پذیرش مقادیر null معرفی می‌کند.

البته در اینجا به صورت پیش فرض byte arrayها به همین نحو معرفی می‌شوند و تنظیم فوق صرفا جهت ارائه توضیحات بیشتر در نظر گرفته شد.

صرفنظر کردن از خواص محاسباتی در تعاریف نگاشت‌ها

با توجه به اینکه خاصیت FullName به صورت یک خاصیت محاسباتی فقط خواندنی، در کدهای برنامه تعریف شده است، با استفاده از متد Ignore، از نگاشت آن به بانک اطلاعاتی جلوگیری خواهیم کرد.

معرفی کلاس‌های تعاریف نگاشت‌ها به برنامه

استفاده از کلاس‌های Config فوق خودکار نیست و نیاز است توسط متد modelBuilder.Configurations.Add معرفی شوند:

```
using System.Data.Entity;
using System.Data.Entity.Migrations;
using EF_Sample03.Mappings;
using EF_Sample03.Models;

namespace EF_Sample03.DataLayer
{
    public class Sample03Context : DbContext
    {
        public DbSet<User> Users { set; get; }
        public DbSet<Project> Projects { set; get; }

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

            //modelBuilder.ComplexType<InterestComponent>();
            //modelBuilder.Ignore<InterestComponent>();

            base.OnModelCreating(modelBuilder);
        }
    }

    public class Configuration : DbMigrationsConfiguration<Sample03Context>
    {
        public Configuration()
        {
            AutomaticMigrationsEnabled = true;
            AutomaticMigrationDataLossAllowed = true;
        }

        protected override void Seed(Sample03Context context)
        {
            base.Seed(context);
```

```
        }
    }
}
```

در اینجا کلاس Context برنامه مثال جاری را ملاحظه می‌کنید؛ به همراه کلاس Configuration مهاجرت خودکار که در قسمت‌های قبل بررسی شد.
در متد OnModelCreating نیز می‌توان یک کلاس را از نوع Complex معرفی کرد تا برای آن در بانک اطلاعاتی جدول جداگانه‌ای تعریف نشود. اما باید دقت داشت که اینکار را فقط یکبار می‌توان انجام داد؛ یا توسط کلاس InterestComponentConfig و یا توسط متد modelBuilder.ComplexType. اگر هر دو با هم فراخوانی شوند، EF یک استثناء را صادر خواهد کرد.

و در نهایت، قسمت آغازین برنامه اینبار به شکل زیر خواهد بود که از آغاز کننده MigrateDatabaseToLatestVersion (قسمت چهارم این سری) نیز استفاده کرده است:

```
using System;
using System.Data.Entity;
using EF_Sample03.DataLayer;

namespace EF_Sample03
{
    class Program
    {
        static void Main(string[] args)
        {
            Database.SetInitializer(new MigrateDatabaseToLatestVersion<Sample03Context,
Configuration>());

            using (var db = new Sample03Context())
            {
                var project1 = db.Projects.Find(1);
                if (project1 != null)
                {
                    Console.WriteLine(project1.Title);
                }
            }
        }
    }
}
```

ضمنا رشته اتصالی مورد استفاده تعریف شده در فایل کانفیک برنامه نیز به صورت زیر تعریف شده است:

```
<connectionStrings>
    <clear/>
    <add
       name="Sample03Context"
       connectionString="Data Source=(local);Initial Catalog=testdb2012;Integrated Security = true"
       providerName="System.Data.SqlClient"
      />
/connectionStrings>
```

در قسمت‌های بعد مباحث پیشرفته‌تری از تنظیمات نگاشت‌ها را به کمک Fluent API، بررسی خواهیم کرد. برای مثال روابط ارث بری، many-to-many و ... چگونه تعریف می‌شوند.

نظرات خوانندگان

نویسنده: ایلیا اکبری فرد
تاریخ: ۱۳۹۱/۰۲/۱۹ ۱۷:۰۶:۱۹

سلام. بسیار ممنون بابت مقالات مفید.
چند تاسوال. من یک پروژه class library جدا برای DomainClasses و DataLayer ایجاد کردم ولی نمی توانم آنها را به پروژه سیلورلایت خود reference دهم .
دوم اینکه validation در codeFirst را نمی توان بطور کامل در fluent api پیاده سازی کرد و حتما باید annotation بکار برد. نمیشود بطورکامل از fluent api استفاده کرد .

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۱۹ ۱۹:۳۲:۴۸

- بحث سیلورلایت جدا است. سیلورلایت یک فناوری سمت کاربر است. مثل جاوا اسکریپت. برای دسترسی به سرور نیاز دارد با وب سرویس کار کند. متداول‌ترین آن WCF RIA Services است که نگارش‌های جدید آن امکان استفاده از EF Code first را هم دارد. بنابراین مستقیما نمی‌تونید از «هیچ» ORM ایی در سیلورلایت «مستقیما» استفاده کنید؛ اما ... لایه سرویس سمت سرور شما این امکان را دارد. در مورد WCF RIA Services قبلا مطلب نوشتم (البته مربوط به database first است؛ در آن زمان که نوشته شده): (^) (قسمت 26)
- این رو به نظر در قسمت‌های قبل ذکر کردم که فقط پیغام‌های خطا رو نمی‌تونید اینجا ذکر کنید و گرنه حداکثر طول و فیلداجباری و غیره همان اثر را دارد.

نویسنده: Hassan
تاریخ: ۱۳۹۱/۰۲/۲۰ ۱۰:۱۷:۴۷

Code generatorها تمامی لایه ها را می سازند. Entity Framework Code First می تواند لایه های DAL و BLL را بسازد یا Add lon برای این کار وجود دارد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۲۰ ۱۰:۵۰:۵۵

یکی از مهم‌ترین اهداف EF Code first این است که با زیرساخت‌های یک ORM آشنا شوید. نباید سؤال بپرسید database first که مسایل همزمانی رو اعمال می‌کنه؛ ولی اطلاع نداشته باشید که پشت صحنه آن در تنظیمات خواص یک فیلد یا جدول، چه امکاناتی وجود دارد و چه مسایلی از چشم شما دور مانده است. این فرق کسی است که اول کد می‌نویسد و طراحی می‌کند (code first)، با کسی که فقط وابسته است به یک سری ابزار که سازوکار درونی آن‌ها را نمی‌داند (database first).
بنابراین سؤال اینجا است که آیا وظیفه‌ی یک ORMاست که برای شما کدنویسی لایه‌های مختلف را انجام دهد؟ یا اینکه اومدید اینجا یک سطح بالاتر رو تجربه کنید؟
البته ابزار هم وجود دارد مانند MVC Scaffolding که بر مبنای EF code first کار می‌کند و یک Code generator است برای ASP.NET MVC . ولی هدف از این مباحث چیز دیگری است.

نویسنده: Mohammadreza Shakeri
تاریخ: ۱۳۹۱/۰۲/۲۰ ۱۲:۵۵:۲۳

سلام.ممنون به خاطر مطالبی که قرار میدید.
من در قسمت دوم به خطایی برخورد کردم که تو این قسمت هم دوباره با این خطا مواجه شدم.

Inconsistent accessibility: property type 'System.Data.Entity.DbSet' is less accessible than property ''EF_Sample03.DataLayer.Sample03Context.Projects

اشکال کار چی می تونه باشه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۲۰ ۱۳:۲۱:۳۱

پاسخ دقیق نیاز به بررسی کدهای شما دارد ولی ... اشکال کار فقط در سطح دسترسی کلاس‌ها و خواص تعریف شده می‌تواند باشد. برای مثال تعدادی رو مثلا internal class تعریف کردید تعداد دیگر رو public class و از این نوع موارد.

نویسنده: peyman
تاریخ: ۱۳۹۱/۰۴/۰۶ ۱۹:۴۶

سلام آقای نصیری . وقتی EF code First رو در این وب سایت سرچ میکنم تمامی سری آموزشی EF code First بجز شماره 5 نمایش داده میشه ! و در واقع نتونستم شماره 5 رو ببینم .

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۰۶ ۲۱:۲۰

از تگ مربوطه استفاده کنید.

نویسنده: رضا
تاریخ: ۱۳۹۱/۰۵/۲۸ ۱۲:۳۰

سلام آقای نصیری - میخواستم بدونم نحوه ایجاد Unique Constraint روی فیلدهای دیتابیس با روش Code First به چه شکلی است؟
دیتابیس من Sql Ce 4.0 SP1 هستش و از Entity Framework 5.0 هم استفاده میکنم.
ممنون.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۲۸ ۱۲:۳۸

در متد Seed، متد زیر را فراخوانی کنید:

```
private static void createUniqueIndex(MyContext context, string tableName, string fieldName)
{
  try
  {
     context.Database.ExecuteSqlCommand("CREATE UNIQUE INDEX [IX_Unique_" + tableName + "_" + fieldName
+ "] ON [" + tableName + "]([" + fieldName + "] ASC);");
  }
  catch { }
}
```

نویسنده: رضا
تاریخ: ۱۳۹۱/۰۵/۳۰ ۱۹:۴۹

واقعاً ممنون آقای نصیری.
حالا اگر بخوایم Unique Constraint رو همزمان روی دو فیلد اعمال کنیم به چه صورت خواهد بود؟ یعنی من توی جدول دیتابیسم CustomerId و ProductId دارم که میخوام هیچ دو رکوردی نباشه که این مقادیرش تکراری باشه.
بی نهایت ممنون.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۳۰ ۱۹:۵۴

مراجعه کنید به قسمت هشتم، بحث composite keys .

نویسنده: محسن کریمی
تاریخ: ۱۳۹۱/۰۸/۰۳ ۱۰:۵۵

سلام

```
using System.ComponentModel.DataAnnotations;
```

در کلاس userconfig باید کامل بشه به این:

```
using System.ComponentModel.DataAnnotations.Schema;
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۰۳ ۱۱:۲۲

در **5** EF جای یک سری از کلاس‌ها تغییر کرده. مثلا ویژگی‌های ComplexType، ForeignKey و ... به فضای نام System.ComponentModel.DataAnnotations.Schema منتقل شده‌اند. در همین حد تغییر جهت کامپایل مجدد کد کفایت می‌کند.

نویسنده: یاسر مرادی
تاریخ: ۱۳۹۱/۰۸/۰۳ ۱۲:۲۶

در مورد ASP.NET Web API و UpShot و اینها که از EF 4.3 تو خودشون استفاده کردند چی ؟
متاسفانه دیگه نمی‌شه با assembly binding بشون بگیم که از EF 5 استفاده کنید
چون Runtime خطا می‌دن و مثلا می‌گن که System.ComponentMode.DataAnnotation.ForeignKey در Entity Framework 5 نیست !
بله نیست، چون رفته به DLL مربوط به Component Model.Data Annotation
راه حلی جز گرفتن سورس کد upshot و Build مجددش هست ؟
و غیر از عقب گرد به EF 3
ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۰۳ ۱۳:۲۵

- البته EF 5 فقط یک نام تجاری است. نگارش اسمبلی آن 4.4.0.0 است.
- assembly binding هم باید کار کنه چون فضای نام System.ComponentModel.DataAnnotations.Schema داخل خود اسمبلی جدید EF هست؛ هرچند به ظاهر جزئی از یک اسمبلی دیگر به نظر می‌رسد، که در عمل اینطور نیست. به این ترتیب امکان استفاده از EF5 در برنامه‌های دات نت 4 هم هست.

نویسنده: یوسف
تاریخ: ۱۳۹۲/۰۵/۱۱ ۱۸:۲۰

سلام.
اگر بخواهیم برای فیلدی که آن را Identity کرده‌ایم، مقادیر Seed و Step را تغییر دهیم ، مثلاً هر دو را از یک به منفی یک تنظیم کنیم، راهکار چیست؟

نویسنده: وحید نصیری

تاریخ: ۱۳۹۲/۰۵/۱۱ ۱۹:۲۶

برای هر قابلیتی که در تنظیمات نگاشت‌ها وجود ندارد و سفارشی است باید در تعاریف Migrations در متد Seed آن، SQL مرتبط را ذکر کنید. مانند « ایجاد ایندکس منحصربفرد در EF Code first » و یا « ایندکس منحصر به فرد با استفاده از Data Annotation در EF Code First ». اصول و روش کار یکی است؛ فقط کوئری SQL ایی که باید اجرا شود، بنابر نیاز تفاوت می‌کند.

نویسنده: احمدعلی شفیعی
تاریخ: ۱۳۹۳/۰۹/۱۷ ۲۲:۵۸

سلام. توی EF 6 متد HasKey کجاست؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۱۸ ۰:۱۰

در همان فضای نام System.Data.Entity.ModelConfiguration

نویسنده: احمدعلی شفیعی
تاریخ: ۱۳۹۳/۰۹/۱۸ ۰:۱۳

بله ممنون. من اشتباها بجای EntityTypeConfiguration از ComplexTypeConfiguration استفاده می‌کردم که توی اون HasKey وجود نداشت.