عنوان: EF Code First #8
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۲۱ ۱۳:۴۷:۰۰
آدرس: www.dotnettips.info
گروه‌ها: Entity framework

ادامه بحث بررسی جزئیات نحوه نگاشت کلاس‌ها به جداول، توسط EF Code first

استفاده از Viewهای SQL Server در EF Code first

از Viewها عموما همانند یک جدول فقط خواندنی استفاده می‌شود. بنابراین نحوه نگاشت اطلاعات یک کلاس به یک View دقیقا همانند نحوه نگاشت اطلاعات یک کلاس به یک جدول است و تمام نکاتی که تا کنون بررسی شدند، در اینجا نیز صادق است. اما ...
الف) بر اساس تنظیمات توکار EF Code first، نام مفرد کلاس‌ها، حین نگاشت به جداول، تبدیل به اسم جمع می‌شوند. بنابراین اگر View ما در سمت بانک اطلاعاتی چنین تعریفی دارد:

```
Create VIEW EmployeesView
AS
SELECT id,
       FirstName
FROM   Employees
```

در سمت کدهای برنامه نیاز است به این شکل تعریف شود:

```
using System.ComponentModel.DataAnnotations;

namespace EF_Sample04.Models
{
    [Table("EmployeesView")]
    public class EmployeesView
    {
        public int Id { set; get; }
        public string FirstName { set; get; }
    }
}
```

در اینجا به کمک ویژگی Table، نام دقیق این View را در بانک اطلاعاتی مشخص کرده‌ایم. به این ترتیب تنظیمات توکار EF بازنویسی خواهد شد و دیگر به دنبال EmployeesViews نخواهد گشت؛ یا جدول متناظر با آن‌را به صورت خودکار ایجاد نخواهد کرد.
ب) View شما نیاز است دارای یک فیلد Primary key نیز باشد.
ج) اگر از مهاجرت خودکار توسط MigrateDatabaseToLatestVersion استفاده کنیم، پیغام خطای زیر را دریافت خواهیم کرد:

```
There is already an object named 'EmployeesView' in the database.
```

علت این است که هنوز جدول dbo.__MigrationHistory از وجود آن مطلع نشده است، زیرا یک View، خارج از برنامه و در سمت بانک اطلاعاتی اضافه می‌شود.
برای حل این مشکل می‌توان همانطور که در قسمت‌های قبل نیز عنوان شد، EF را طوری تنظیم کرد تا کاری با بانک اطلاعاتی نداشته باشد:

```
Database.SetInitializer<Sample04Context>(null);
```

به این ترتیب EmployeesView در همین لحظه قابل استفاده است.
و یا به حالت امن مهاجرت دستی سوئیچ کنید:

```
Add-Migration Init -IgnoreChanges
Update-Database
```

پارامتر IgnoreChanges سبب می‌شود تا متدهای Up و Down کلاس مهاجرت تولید شده، خالی باشد. یعنی زمانیکه دستور Update-Database انجام می‌شود، نه Viewایی دراپ خواهد شد و نه جدول اضافه‌ای ایجاد می‌گردد. فقط جدول dbo.__MigrationHistory به روز می‌شود که هدف اصلی ما نیز همین است.
همچنین در این حالت کنترل کاملی بر روی کلاس‌های Up و Down وجود دارد. می‌توان CreateTable اضافی را به سادگی از این کلاس‌ها حذف کرد.

ضمن اینکه باید دقت داشت یکی از اهداف کار با ORMs، فراهم شدن امکان استفاده از بانک‌های اطلاعاتی مختلف، بدون اعمال تغییری در کدهای برنامه می‌باشد (فقط تغییر کانکشن استرینگ، به علاوه تعیین Provider جدید، باید جهت این مهاجرت کفایت کند). بنابراین اگر از View استفاده می‌کنید، این برنامه به SQL Server گره خواهد خورد و دیگر از سایر بانک‌های اطلاعاتی که از این مفهوم پشتیبانی نمی‌کنند، نمی‌توان به سادگی استفاده کرد.

استفاده از فیلدهای XML اس کیوال سرور

در حال حاضر پشتیبانی توکاری توسط EF Code first از فیلدهای ویژه XML اس کیوال سرور وجود ندارد؛ اما استفاده از آن‌ها با رعایت چند نکته ساده، به نحو زیر است:

```
using System.ComponentModel.DataAnnotations;
using System.Xml.Linq;

namespace EF_Sample04.Models
{
    public class MyXMLTable
    {
        public int Id { get; set; }

        [Column(TypeName = "xml")]
        public string XmlValue { get; set; }

        [NotMapped]
        public XElement XmlValueWrapper
        {
            get { return XElement.Parse(XmlValue); }
            set { XmlValue = value.ToString(); }
        }
    }
}
```

در اینجا توسط TypeName ویژگی Column، نوع توکار xml مشخص شده است. این فیلد در طرف کدهای کلاس‌های برنامه، به صورت string تعریف می‌شود. سپس اگر نیاز بود به این خاصیت توسط LINQ to XML دسترسی یافت، می‌توان یک فیلد محاسباتی را همانند خاصیت XmlValueWrapper فوق تعریف کرد. نکته دیگری را که باید به آن دقت داشت، استفاده از ویژگی NotMapped

می‌باشد، تا EF سعی نکند خاصیتی از نوع XElement را (یک CLR Property) به بانک اطلاعاتی نگاشت کند.

و همچنین اگر علاقمند هستید که این قابلیت به صورت توکار اضافه شود، می‌توانید اینجا رای دهید !

نحوه تعریف Composite keys در EF Code first

کلاس نوع فعالیت زیر را درنظر بگیرید:

```
namespace EF_Sample04.Models
{
    public class ActivityType
    {
        public int UserId { set; get; }
        public int ActivityID { get; set; }
    }
}
```

در جدول متناظر با این کلاس، نباید دو رکورد تکراری حاوی شماره کاربری و شماره فعالیت یکسانی باهم وجود داشته باشند. بنابراین بهتر است بر روی این دو فیلد، یک کلید ترکیبی تعریف کرد:

```
using System.Data.Entity.ModelConfiguration;
using EF_Sample04.Models;

namespace EF_Sample04.Mappings
{
    public class ActivityTypeConfig : EntityTypeConfiguration<ActivityType>
    {
        public ActivityTypeConfig()
        {
            this.HasKey(x => new { x.ActivityID, x.UserId });
        }
    }
}
```

در اینجا نحوه معرفی بیش از یک کلید را در متد HasKey ملاحظه می‌کنید.

یک نکته:
اینبار اگر سعی کنیم مثلا از متد db.ActivityTypes.Find با یک پارامتر استفاده کنیم، پیغام خطای «The number of primary key values passed must match number of primary key values defined on the entity» را دریافت خواهیم کرد. برای رفع آن باید هر دو کلید، در این متد قید شوند:

```
var activity1 = db.ActivityTypes.Find(4, 1);
```

ترتیب آن‌ها هم بر اساس ترتیبی که در کلاس ActivityTypeConfig، ذکر شده است، مشخص می‌گردد. بنابراین در این مثال، اولین پارامتر متد Find، به ActivityID اشاره می‌کند و دومین پارامتر به UserId.

بررسی نحوه تعریف نگاشت جداول خود ارجاع دهنده (Self Referencing Entity)

سناریوهای کاربردی بسیاری را جهت جداول خود ارجاع دهنده می‌توان متصور شد و عموما تمام آن‌ها برای مدل سازی اطلاعات

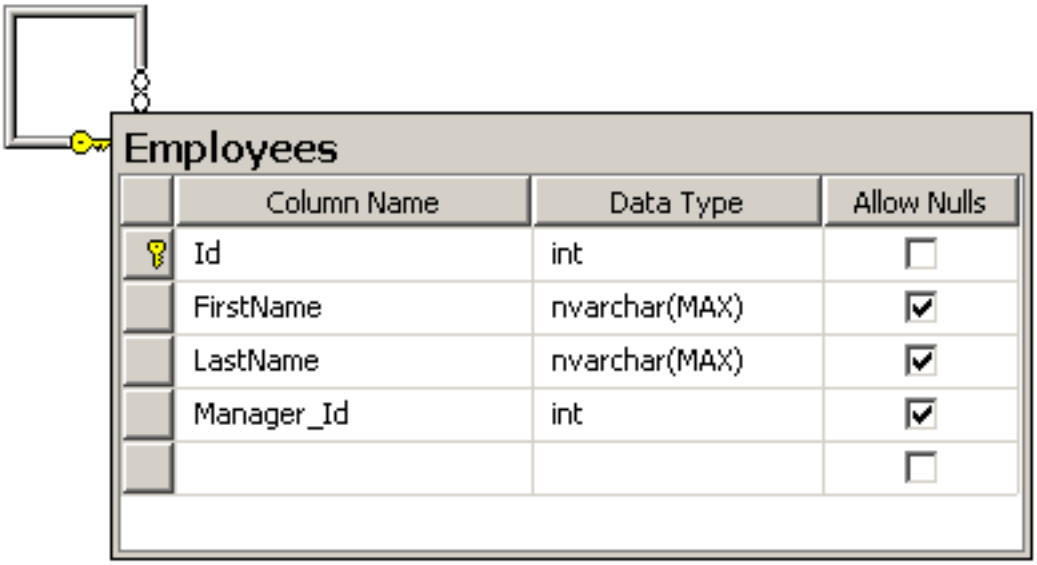
چند سطحی کاربرد دارند. برای مثال یک کارمند را درنظر بگیرید. مدیر این شخص هم یک کارمند است. مسئول این مدیر هم یک کارمند است و الی آخر. نمونه دیگر آن، طراحی منوهای چند سطحی هستند و یا یک مشتری را درنظر بگیرید. مشتری دیگری که توسط این مشتری معرفی شده است نیز یک مشتری است. این مشتری نیز می‌تواند یک مشتری دیگر را به شما معرفی کند و این سلسله مراتب به همین ترتیب می‌تواند ادامه پیدا کند.
در طراحی بانک‌های اطلاعاتی، برای ایجاد یک چنین جداولی، یک کلید خارجی را که به کلید اصلی همان جدول اشاره می‌کند، ایجاد خواهند کرد؛ اما در EF Code first چطور؟

```
using System.Collections.Generic;

namespace EF_Sample04.Models
{
    public class Employee
    {
        public int Id { set; get; }
        public string FirstName { get; set; }
        public string LastName { get; set; }

        //public int? ManagerID { get; set; }
        public virtual Employee Manager { get; set; }
    }
}
```

در این کلاس، خاصیت Manager دارای ارجاعی است به همان کلاس؛ یعنی یک کارمند می‌تواند مسئول کارمند دیگری باشد. برای تعریف نگاشت این کلاس به بانک اطلاعاتی می‌توان از روش زیر استفاده کرد:

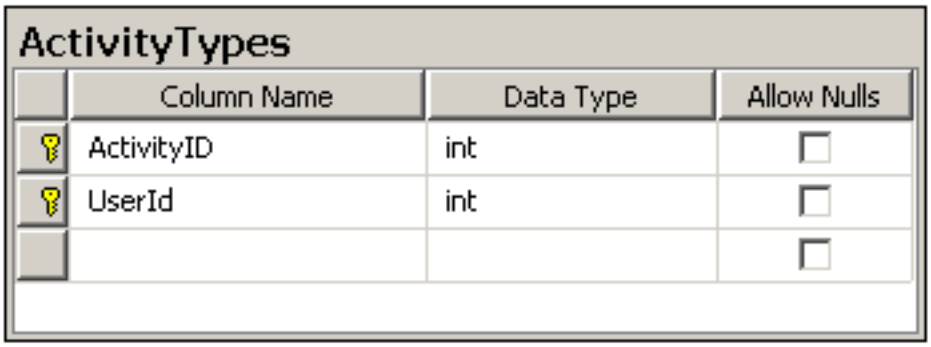
```
using System.Data.Entity.ModelConfiguration;
using EF_Sample04.Models;

namespace EF_Sample04.Mappings
{
    public class EmployeeConfig : EntityTypeConfiguration<Employee>
    {
        public EmployeeConfig()
        {
            this.HasOptional(x => x.Manager)
                .WithMany()
                //.HasForeignKey(x => x.ManagerID)
                .WillCascadeOnDelete(false);
        }
    }
}
```

با توجه به اینکه یک کارمند می‌تواند مسئولی نداشته باشد (خودش مدیر ارشد است)، به کمک متد HasOptional مشخص کرده‌ایم که فیلد Manager_Id را که می‌خواهی به این کلاس اضافه کنی باید نال پذیر باشد. توسط متد WithMany طرف دیگر رابطه مشخص شده است.
اگر نیاز بود فیلد Manager_Id اضافه شده نام دیگری داشته باشد، یک خاصیت nullable مانند ManagerID را که در کلاس Employee مشاهده می‌کنید، اضافه نمائید. سپس در طرف تعاریف نگاشت‌ها به کمک متد HasForeignKey، باید صریحا عنوان کرد که این خاصیت، همان کلید خارجی است. از این نکته در سایر حالات تعاریف نگاشت‌ها نیز می‌توان استفاده کرد، خصوصا اگر از یک بانک اطلاعاتی موجود قرار است استفاده شود و از نام‌های دیگری بجز نام‌های پیش فرض EF استفاده کرده است.



Employees

	Column Name	Data Type	Allow Nulls
🔑	Id	int	☐
	FirstName	nvarchar(MAX)	☑
	LastName	nvarchar(MAX)	☑
	Manager_Id	int	☑
			☐



ActivityTypes

	Column Name	Data Type	Allow Nulls
🔑	ActivityID	int	☐
🔑	UserId	int	☐
			☐

مثال‌های این سری رو از این آدرس هم می‌تونید دریافت کنید: (^)

نظرات خوانندگان

نویسنده: Dsictco
تاریخ: ۱۳۹۱/۰۲/۲۱ ۲۰:۰۹:۱۷

سلام
با تشکر از آموزش های مفید شما
آیا امکان Bind کردن گرید به EF وجود دارد؟منظورم مثل ADO.NET. اونجا راحت می تونیم هر فیلدی که میخواهیم به هر ستونی Bind کنیم، این کار رو با EF هم میشه انجام داد؟یا باید کدنویسی کنیم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۲۱ ۲۲:۵۰:۴۹

این مورد به توانایی‌های LINQ شما بر می‌گردد. در اینجا کوئری‌ها رو باید با LINQ نوشت و سپس مباحث Projection و امثال آن برای تهیه لیست مورد نظر جهت Bind به گریدها می‌تونه مدنظر باشه.
یک قسمت رو به مروری سریع به کوئری نوشتن در EF اختصاص خواهم داد.

نویسنده: مهمان
تاریخ: ۱۳۹۱/۰۷/۲۲ ۹:۴۷

سلام در قسمت Self Referencing Entity اگر بخواهیم کلید خارجی نام دیگری داشته باشد طبق گفته شما که عمل کردم خطای Sequence contains no elements را میدهد.

```
using System.Collections.Generic;

namespace EF_Sample04.Models
{
    public class Employee
    {
        public int Id { set; get; }
        public string FirstName { get; set; }
        public string LastName { get; set; }

        public int? ManagerID { get; set; }
        public virtual Employee Manager { get; set; }
    }
}
```

```
using EF_Sample04.Models;
using System.Data.Entity.ModelConfiguration;
namespace EF_Sample04.Mappings
{
    public class EmployeeConfig : EntityTypeConfiguration<Employee>
    {
        public EmployeeConfig()
        {
            this.HasOptional(x => x.Manager)
                .WithMany()
                .HasForeignKey(x => x.ManagerID)
                .WillCascadeOnDelete(false);
        }
    }
}
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۲۲ ۱۰:۴۰

مدل مشکلی نداره (تست شده). اطلاعات بیشتر: « استثنای Sequence contains no elements در حین استفاده از LINQ »
ضمن اینکه مطلب فوق در اینجا کامل شده:
« مباحث تکمیلی مدل‌های خود ارجاع دهنده در EF Code first »

نویسنده: Programer
تاریخ: ۱۳۹۱/۰۷/۲۲ ۱۱:۴۰

علت خطای این قسمت به علت کد زیر بود

```
public ActionResult Index()
{
            NewsContext db = new NewsContext();
            var Query = db.Employee.ToList()
            return View(Query);
}
```

که با تغییر کد به شکل زیر حل شد

```
public ActionResult Index(){
            NewsContext db = new NewsContext();
            var Query = db.Employee.ToList().Where(x=>x.ManagerID==null).ToList();
            return View(Query);
}
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۲۲ ۱۲:۲۴

چقدر خوب که این مطلب رو تکمیل کردید (هرچند View متناظر در اینجا هم باید ذکر می‌شد).
همچنین چقدر خوب می‌شد زمانیکه سؤال اصلی رو اینجا پرسیدید، موارد فوق رو هم ذکر می‌کردید. چون افراد نمی‌تونند از راه دور کار شما را مشاهده یا دیباگ کنند یا دقیقا بدانند که چه کدی را نوشته‌اید که این خطا را داده. به همین جهت سعی می‌کنند حدس بزنند که در مرحله آخر و پس از نگاشت‌ها، چکار کرده‌اید که خطای فوق حاصل شده.

مطلب خوبی در این زمینه: « نحوه صحیح گزارش دادن یک باگ »

نویسنده: Programer
تاریخ: ۱۳۹۱/۰۷/۲۲ ۱۲:۳۷

مرسی از راهنماییتون

نویسنده: رضا بزرگی
تاریخ: ۱۳۹۲/۰۲/۱۴ ۱۸:۱۴

میشه در مورد این کوئری بیشتر توضیح بدین؟

```
db.Employee.ToList().Where(x=>x.ManagerID==null).ToList();
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۱۴ ۱۹:۱

در این مطلب « مباحث تکمیلی مدل‌های خود ارجاع دهنده در EF Code first » بحث شده

نویسنده: احمدعلی شفیعی
تاریخ: ۱۳۹۳/۱۱/۲۷ ۱۸:۵۱

سلام. توی مدل‌های خود ارجاع دهنده، چجوری می‌شه Mapping رو طوری تنظیم کرد که در صورت پاک شدن یک عنصر، عناصر زیرمجموعه‌ی اون هم پاک بشن؟

نویسنده: وحید نصیری

تاریخ: ۱۳۹۳/۱۱/۲۷ ۱۸:۵۸

در نظرات بحث « مباحث تکمیلی مدل‌های خود ارجاع دهنده در EF Code first » به این موضوع پرداخته شده.