عنوان: EF Code First #13
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۲۷ ۱۰:۰۶:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Entity framework

استفاده مستقیم از عبارات SQL در EF Code first

طراحی اکثر ORMهای موجود به نحوی است که برنامه نهایی شما را مستقل از بانک اطلاعاتی کنند و این پروایدر نهایی است که معادل‌های صحیح بسیاری از توابع توکار بانک اطلاعاتی مورد استفاده را در اختیار EF قرار می‌دهد. برای مثال در یک بانک اطلاعاتی تابعی به نام substr تعریف شده، در بانک اطلاعاتی دیگری همین تابع substring نام دارد. اگر برنامه را به کمک کوئری‌های LINQ تهیه کنیم، نهایتا پروایدر نهایی مخصوص بانک اطلاعاتی مورد استفاده است که این معادل‌ها را در اختیار EF قرار می‌دهد و برنامه بدون مشکل کار خواهد کرد. اما یک سری از موارد شاید معادلی در سایر بانک‌های اطلاعاتی نداشته باشند؛ برای مثال رویه‌های ذخیره شده یا توابع تعریف شده توسط کاربر. امکان استفاده از یک چنین توانایی‌هایی نیز با اجرای مستقیم عبارات SQL در EF Code first پیش بینی شده و بدیهی است در این حالت برنامه به یک بانک اطلاعاتی خاص گره خواهد خورد؛ همچنین مزیت استفاده از کوئری‌های Strongly typed تحت نظر کامپایلر را نیز از دست خواهیم داد. به علاوه باید به یک سری مسایل امنیتی نیز دقت داشت که در ادامه بررسی خواهند شد.

کلاس‌های مدل مثال جاری

در مثال جاری قصد داریم نحوه استفاده از رویه‌های ذخیره شده و توابع تعریف شده توسط کاربر مخصوص SQL Server را بررسی کنیم. در اینجا کلاس‌های پزشک و بیماران او، کلاس‌های مدل برنامه را تشکیل می‌دهند:

```
using System.Collections.Generic;

namespace EF_Sample08.DomainClasses
{
    public class Doctor
    {
        public int Id { set; get; }
        public string Name { set; get; }

        public virtual ICollection<Patient> Patients { set; get; }
    }
}
```

```
namespace EF_Sample08.DomainClasses
{
    public class Patient
    {
        public int Id { set; get; }
        public string Name { set; get; }

        public virtual Doctor Doctor { set; get; }
    }
}
```

کلاس **Context** برنامه به نحو زیر تعریف شده:

```
using System.Data.Entity;
using EF_Sample08.DomainClasses;
```

```
namespace EF_Sample08.DataLayer.Context
{
    public class Sample08Context : DbContext
    {
        public DbSet<Doctor> Doctors { set; get; }
        public DbSet<Patient> Patients { set; get; }
    }
}
```

و اینبار کلاس **DbMigrationsConfiguration** تعریف شده اندکی با مثال‌های قبلی متفاوت است:

```
using System.Data.Entity.Migrations;
using EF_Sample08.DomainClasses;
using System.Collections.Generic;

namespace EF_Sample08.DataLayer.Context
{
    public class Configuration : DbMigrationsConfiguration<Sample08Context>
    {
        public Configuration()
        {
            AutomaticMigrationsEnabled = true;
            AutomaticMigrationDataLossAllowed = true;
        }

        protected override void Seed(Sample08Context context)
        {
            addData(context);
            addSP(context);
            addFn(context);
            base.Seed(context);
        }

        private static void addData(Sample08Context context)
        {
            var patient1 = new Patient { Name = "p1" };
            var patient2 = new Patient { Name = "p2" };
            var doctor1 = new Doctor { Name = "doc1", Patients = new List<Patient> { patient1, patient2
} };
            context.Doctors.Add(doctor1);
        }

        private static void addFn(Sample08Context context)
        {
            context.Database.ExecuteSqlCommand(
                @"IF  EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[FindDoctorPatientsCount]')
                  AND type in (N'FN', N'IF', N'TF', N'FS', N'FT'))
                     DROP FUNCTION [dbo].[FindDoctorPatientsCount]");
            context.Database.ExecuteSqlCommand(
                @"CREATE FUNCTION FindDoctorPatientsCount(@Doctor_Id INT)
                    RETURNS INT
                    BEGIN
                     RETURN
                     (
                         SELECT COUNT(*)
                         FROM   Patients
                         WHERE  Doctor_Id = @Doctor_Id
                     );
                    END");
        }

        private static void addSP(Sample08Context context)
        {
            context.Database.ExecuteSqlCommand(
                @"IF  EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[FindDoctorsStartWith]')
                  AND type in (N'P', N'PC'))
                       DROP PROCEDURE [dbo].[FindDoctorsStartWith]
                ");
            context.Database.ExecuteSqlCommand(
                @"CREATE PROCEDURE FindDoctorsStartWith(@name NVARCHAR(400))
                  AS
                 SELECT *
                 FROM   Doctors
```

```
                WHERE  [Name] LIKE @name + '%'");
        }
    }
}
```

در اینجا از متد Seed علاوه بر مقدار دهی اولیه جداول، برای تعریف یک رویه ذخیره شده به نام FindDoctorsStartWith و یک تابع سفارشی به نام FindDoctorPatientsCount نیز استفاده شده است. متد context.Database.ExecuteSqlCommand مستقیما یک عبارت SQL را بر روی بانک اطلاعاتی اجرا می‌کند.

در ادامه کدهای کامل برنامه نهایی را ملاحظه می‌کنید:

```
using System;
using System.Data;
using System.Data.Entity;
using System.Data.Objects.SqlClient;
using System.Data.SqlClient;
using System.Linq;
using EF_Sample08.DataLayer.Context;
using EF_Sample08.DomainClasses;

namespace EF_Sample08
{
    class Program
    {
        static void Main(string[] args)
        {
            Database.SetInitializer(new MigrateDatabaseToLatestVersion<Sample08Context,
Configuration>());

            using (var db = new Sample08Context())
            {
                runSp(db);
                runFn(db);
                usingSqlFunctions(db);
            }
        }

        private static void usingSqlFunctions(Sample08Context db)
        {
            var doctorsWithNumericNameList = db.Doctors.Where(x => SqlFunctions.IsNumeric(x.Name) ==
1).ToList();
            if (doctorsWithNumericNameList.Any())
            {
                //do something
            }
        }

        private static void runFn(Sample08Context db)
        {
            var doctorIdParameter = new SqlParameter
            {
                ParameterName = "@doctor_id",
                Value = 1,
                SqlDbType = SqlDbType.Int
            };
            var patientsCount = db.Database.SqlQuery<int>("select
dbo.FindDoctorPatientsCount(@doctor_id)", doctorIdParameter).FirstOrDefault();
            Console.WriteLine(patientsCount);
        }

        private static void runSp(Sample08Context db)
        {
            var nameParameter = new SqlParameter
            {
                ParameterName = "@name",
                Value = "doc",
                Direction = ParameterDirection.Input,
                SqlDbType = SqlDbType.NVarChar
            };
            var doctors = db.Database.SqlQuery<Doctor>("exec FindDoctorsStartWith @name",
nameParameter).ToList();
            if (doctors.Any())
            {
```

```
                foreach (var item in doctors)
                {
                    Console.WriteLine(item.Name);
                }
            }
        }
    }
}
```

توضیحات

همانطور که ملاحظه می‌کنید، برای اجرای مستقیم یک عبارت SQL صرفنظر از اینکه یک رویه ذخیره شده است یا یک تابع و یا یک کوئری معمولی، باید از متد db.Database.SqlQuery استفاده کرد. خروجی این متد از نوع IEnumerable است و این توانایی را دارد که رکوردهای بازگشت داده شده از بانک اطلاعاتی را به خواص یک کلاس به صورت خودکار نگاشت کند.
پارامتر اول متد db.Database.SqlQuery، عبارت SQL مورد نظر است. پارامتر دوم آن باید توسط وهله‌هایی از کلاس SqlParameter مقدار دهی شود. به کمک SqlParameter نام پارامتر مورد استفاده، مقدار و نوع آن مشخص می‌گردد. همچنین Direction آن نیز برای استفاده از رویه‌های ذخیره شده ویژه‌ای که دارای پارامتری از نوع out هستند درنظر گرفته شده است.

چند نکته

- در متد runSp فوق، متد الحاقی ToList را حذف کرده و برنامه را اجرا کنید. بلافاصله پیغام خطای «The SqlParameter is already contained by another SqlParameterCollection.» ظاهر خواهد شد. علت هم این است که با بکارگیری متد ToList، تمام عملیات یکبار انجام شده و نتیجه بازگشت داده می‌شود اما اگر به صورت مستقیم از خروجی IEnumerable آن استفاده کنیم، در حلقه foreach تعریف شده، ممکن است این فراخوانی چندبار انجام شود. به همین جهت ذکر متد ToList در اینجا ضروری است.

- عنوان شد که در اینجا باید به مسایل امنیتی دقت داشت. بدیهی است امکان نوشتن یک چنین کوئری‌هایی نیز وجود دارد:

```
db.Database.SqlQuery<Doctor>("exec FindDoctorsStartWith "+ txtName.Text, nameParameter).ToList()
```

در این حالت به ظاهر مشغول به استفاده از رویه‌های ذخیره شده‌ای هستیم که عنوان می‌شود در برابر حملات تزریق SQL در امان هستند، اما چون در کدهای ما به نحو ناصحیحی با جمع زدن رشته‌ها مقدار دهی شده است، برنامه و بانک اطلاعاتی دیگر در امان نخواهند بود. بنابراین در این حالت استفاده از پارامترها را نباید فراموش کرد.
زمانیکه از کوئری‌های LINQ استفاده می‌شود تمام این مسایل توسط EF مدیریت خواهد شد. اما اگر قصد دارید مستقیما عبارات SQL را فراخوانی کنید، تامین امنیت برنامه به عهده خودتان خواهد بود.

- در متد usingSqlFunctions از SqlFunctions.IsNumeric استفاده شده است. این مورد مختص به SQL Server است و امکان استفاده از توابع توکار ویژه SQL Server را در کوئری‌های LINQ برنامه فراهم می‌سازد. برای مثال متدالحاقی از پیش تعریف شده‌ای به نام IsNumeric به صورت مستقیم در دسترس نیست، اما به کمک کلاس SqlFunctions این تابع و بسیاری از توابع دیگر توکار SQL Server قابل استفاده خواهند بود.
اگر علاقمند هستید که لیست این توابع را مشاهده کنید، در ویژوال استودیو بر روی SqlFunctions کلیک راست کرده و گزینه Go to definition را انتخاب کنید.

نظرات خوانندگان

نویسنده: Hassan
تاریخ: ۱۳۹۱/۰۲/۲۹ ۱۶:۴۸:۱۷

سلام
در صورتی که در query از join و group استفاده کنیم، یعنی در خروجی ResultSet فیلدهای چند جدول و همچنین یکسری فیلدهای جدید که توسط توابع تولید می شوند را داشته باشیم، نحوه نگاشت به کلاس ها چگونه خواهد بود؟ EF خودش آن را مدیریت می کند؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۲۹ ۱۷:۱۱:۲۸

یک کلاس جدید تعریف کنید که شامل فیلدهای متناظر با select شما باشد. کار نگاشت نهایی به این‌ها خودکار خواهد بود.

نویسنده: مهمان
تاریخ: ۱۳۹۱/۰۲/۳۰ ۰۰:۲۱:۰۹

زبان eSql را شما کاربردی برایش می بینید؟
شاید مایکروسافت می خواست یک MSIL برای تکنولوژی Data Access داشته باشد!

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۳۰ ۰۸:۱۷:۲۷

ESQL هم قابل استفاده است: (^)

نویسنده: آرش عظیمی
تاریخ: ۱۳۹۲/۰۵/۱۲ ۲:۲۲

در این روش چطوری می‌شه دستور Where رو به صورت یک رشته اجرا نمود
به طور مثال این دستور

```
DBEntities MyDB = new DBEntities();
        var Query1 = from P in MyDB.Per
        where P.IDRANK == 2
        select P;
```

تبدیل بشه به یه چنین دستوری

```
string strquery = "where P.IDRANK == 2";
    DBEntities MyDB = new DBEntities();
    var Query1 = from P in MyDB.Per
         strquery
    select P;
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۵/۱۲ ۹:۴۲

باید از Dynamic LINQ استفاده کنید.