عنوان: **EF Code First #4**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۱۷ ۱۳:۴۹:۰۰
آدرس: www.dotnettips.info
گروه‌ها: Entity framework

آشنایی با Code first migrations

ویژگی Code first migrations برای اولین بار در EF 4.3 ارائه شد و هدف آن سهولت هماهنگ سازی کلاس‌های مدل برنامه با بانک اطلاعاتی است؛ به صورت خودکار یا با تنظیمات دقیق دستی.

همانطور که در قسمت‌های قبل نیز به آن اشاره شد، تا پیش از EF 4.3، پنج روال جهت آغاز به کار با بانک اطلاعاتی در EF code first وجود داشت و دارد:
1) در اولین بار اجرای برنامه، در صورتیکه بانک اطلاعاتی اشاره شده در رشته اتصالی وجود خارجی نداشته باشد، نسبت به ایجاد خودکار آن اقدام می‌گردد. اینکار پس از وهله سازی اولین DbContext و همچنین صدور یک کوئری به بانک اطلاعاتی انجام خواهد شد.
2) DropCreateDatabaseAlways : همواره پس از شروع برنامه، ابتدا بانک اطلاعاتی را drop کرده و سپس نمونه جدیدی را ایجاد می‌کند.
3) DropCreateDatabaseIfModelChanges : اگر EF Code first تشخیص دهد که تعاریف مدل‌های شما با بانک اطلاعاتی مشخص شده توسط رشته اتصالی، هماهنگ نیست، آن‌را drop کرده و نمونه جدیدی را تولید می‌کند.
4) با مقدار دهی پارامتر متد System.Data.Entity.Database.SetInitializer به نال، می‌توان فرآیند آغاز خودکار بانک اطلاعاتی را غیرفعال کرد. در این حالت شخص می‌تواند تغییرات انجام شده در کلاس‌های مدل برنامه را به صورت دستی به بانک اطلاعاتی اعمال کند.
5) می‌توان با پیاده سازی اینترفیس IDatabaseInitializer، یک آغاز کننده بانک اطلاعاتی سفارشی را نیز تولید کرد.

اکثر این روش‌ها در حین توسعه یک برنامه یا خصوصا جهت سهولت انجام آزمون‌های خودکار بسیار مناسب هستند، اما به درد محیط کاری نمی‌خورند؛ زیرا drop یک بانک اطلاعاتی به معنای از دست دادن تمام اطلاعات ثبت شده در آن است. برای رفع این مشکل مهم، مفهومی به نام «Migrations» در EF 4.3 ارائه شده است تا بتوان بانک اطلاعاتی را بدون تخریب آن، بر اساس اطلاعات تغییر کرده‌ی کلاس‌های مدل برنامه، تغییر داد. البته بدیهی است زمانیکه توسط NuGet نسبت به دریافت و نصب EF اقدام می‌شود، همواره آخرین نگارش پایدار که حاوی اطلاعات و فایل‌های مورد نیاز جهت کار با «Migrations» است را نیز دریافت خواهیم کرد.

تنظیمات ابتدایی Code first migrations

در اینجا قصد داریم همان مثال قسمت قبل را ادامه دهیم. در آن مثال از یک نمونه سفارشی سازی شده DropCreateDatabaseAlways استفاده شد.
نیاز است از منوی Tools در ویژوال استودیو، گزینه Library package manager آن، گزینه package manager console را انتخاب کرد تا کنسول پاورشل NuGet ظاهر شود.
اطلاعات مرتبط با پاورشل EF، به صورت خودکار توسط NuGet نصب می‌شود. برای مثال جهت مشاهده آن‌ها به مسیر packages\EntityFramework.4.3.1\tools در کنار پوشه پروژه خود مراجعه نمائید.
در ادامه در پایین صفحه، زمانیکه کنسول پاورشل NuGet ظاهر می‌شود، ابتدا باید دقت داشت که قرار است فرامین را بر روی چه پروژه‌ای اجرا کنیم. برای مثال اگر تعاریف DbContext را به یک اسمبلی و پروژه class library مجزا انتقال داده‌اید، گزینه Default project را در این قسمت باید به این پروژه مجزا، تغییر دهید.
سپس در خط فرمان پاور شل، دستور enable-migrations را وارد کرده و دکمه enter را فشار دهید.
پس از اجرای این دستور، یک سری اتفاقات رخ خواهد داد:

الف) پوشه‌ای به نام Migrations به پروژه پیش فرض مشخص شده در کنسول پاورشل، اضافه می‌شود.
ب) دو کلاس جدید نیز در آن پوشه تعریف خواهند شد به نام‌های Configuration.cs و یک نام خودکار مانند number_InitialCreate.cs
ج) در کنسول پاور شل، پیغام زیر ظاهر می‌گردد:

```
Detected database created with a database initializer. Scaffolded migration
'201205050805256_InitialCreate'
corresponding to current database schema. To use an automatic migration instead, delete the Migrations
folder and re-run Enable-Migrations specifying the -EnableAutomaticMigrations parameter.
```

با توجه به اینکه در مثال قسمت سوم، از آغاز کننده سفارشی سازی شده DropCreateDatabaseAlways استفاده شده بود، اطلاعات آن در جدول سیستمی dbo.__MigrationHistory در بانک اطلاعاتی برنامه موجود است (تصویری از آن‌را در قسمت اول این سری مشاهده کردید). سپس با توجه به ساختار بانک اطلاعاتی جاری، دو کلاس خودکار زیر را ایجاد کرده است:

```
namespace EF_Sample02.Migrations
{
    using System;
    using System.Data.Entity;
    using System.Data.Entity.Migrations;
    using System.Linq;

    internal sealed class Configuration : DbMigrationsConfiguration<EF_Sample02.Sample2Context>
    {
        public Configuration()
        {
            AutomaticMigrationsEnabled = false;
        }

        protected override void Seed(EF_Sample02.Sample2Context context)
        {
            //  This method will be called after migrating to the latest version.

            //  You can use the DbSet<T>.AddOrUpdate() helper extension method
            //  to avoid creating duplicate seed data. E.g.
            //
            //    context.People.AddOrUpdate(
            //      p => p.FullName,
            //      new Person { FullName = "Andrew Peters" },
            //      new Person { FullName = "Brice Lambson" },
            //      new Person { FullName = "Rowan Miller" }
            //    );
            //
        }
    }
}
```

```
namespace EF_Sample02.Migrations
{
    using System.Data.Entity.Migrations;

    public partial class InitialCreate : DbMigration
    {
        public override void Up()
        {
            CreateTable(
                "Users",
                c => new
                    {
                        Id = c.Int(nullable: false, identity: true),
                        Name = c.String(),
                        LastName = c.String(),
                        Email = c.String(),
                        Description = c.String(),
                        Photo = c.Binary(),
                        RowVersion = c.Binary(nullable: false, fixedLength: true, timestamp: true,
storeType: "rowversion"),
                        Interests_Interest1 = c.String(maxLength: 450),
```

```
                        Interests_Interest2 = c.String(maxLength: 450),
                        AddDate = c.DateTime(nullable: false),
                    })
                .PrimaryKey(t => t.Id);

            CreateTable(
                "Projects",
                c => new
                    {
                        Id = c.Int(nullable: false, identity: true),
                        Title = c.String(maxLength: 50),
                        Description = c.String(),
                        RowVesrion = c.Binary(nullable: false, fixedLength: true, timestamp: true,
storeType: "rowversion"),
                        AddDate = c.DateTime(nullable: false),
                        AdminUser_Id = c.Int(),
                    })
                .PrimaryKey(t => t.Id)
                .ForeignKey("Users", t => t.AdminUser_Id)
                .Index(t => t.AdminUser_Id);

        }

        public override void Down()
        {
            DropIndex("Projects", new[] { "AdminUser_Id" });
            DropForeignKey("Projects", "AdminUser_Id", "Users");
            DropTable("Projects");
            DropTable("Users");
        }
    }
}
```

در این کلاس خودکار، نحوه ایجاد جداول بانک اطلاعاتی تعریف شده‌اند. در متد تحریف شده Up، کار ایجاد بانک اطلاعاتی و در متد تحریف شده Down، دستورات حذف جداول و قیود ذکر شده‌اند.
به علاوه اینبار متد Seed را در کلاس مشتق شده از DbMigrationsConfiguration، می‌توان تحریف و مقدار دهی کرد.
علاوه بر این‌ها جدول سیستمی dbo.__MigrationHistory نیز با اطلاعات جاری مقدار دهی می‌گردد.

فعال سازی گزینه‌های مهاجرت خودکار

برای استفاده از این کلاس‌ها، ابتدا به فایل Configuration.cs مراجعه کرده و خاصیت AutomaticMigrationsEnabled را true کنید:

```
internal sealed class Configuration : DbMigrationsConfiguration<EF_Sample02.Sample2Context>
{
        public Configuration()
        {
            AutomaticMigrationsEnabled = true;
        }
```

پس از آن EF به صورت خودکار کار استفاده و مدیریت «Migrations» را عهده‌دار خواهد شد. البته برای این منظور باید نوع آغاز کننده بانک اطلاعاتی را از DropCreateDatabaseAlways قبلی به نمونه جدید MigrateDatabaseToLatestVersion نیز تغییر دهیم:

```
//Database.SetInitializer(new Sample2DbInitializer());
Database.SetInitializer(new MigrateDatabaseToLatestVersion<Sample2Context,
Migrations.Configuration>());
```

یک نکته:
کلاس Migrations.Configuration که باید در حین وهله سازی از MigrateDatabaseToLatestVersion قید شود (همانند کدهای فوق)، از نوع internal sealed معرفی شده است. بنابراین اگر این کلاس را در یک اسمبلی جداگانه قرار داده‌اید، نیاز است فایل را ویرایش کرده و internal sealed آن‌را به public تغییر دهید.

روش دیگر معرفی کلاس‌های Context و Migrations.Configuration، حذف متد Database.SetInitializer و استفاده از فایل app.config یا web.config است به نحو زیر (در اینجا حرف ` اصطلاحا back tick نام دارد. فشردن دکمه ~ در حین تایپ انگلیسی):

```
<entityFramework>
    <contexts>
      <context type="EF_Sample02.Sample2Context, EF_Sample02">
        <databaseInitializer
           type="System.Data.Entity.MigrateDatabaseToLatestVersion`2[[EF_Sample02.Sample2Context,
EF_Sample02],
              [EF_Sample02.Migrations.Configuration, EF_Sample02]], EntityFramework"
        />
      </context>
    </contexts>
  </entityFramework>
```

آزمودن ویژگی مهاجرت خودکار

اکنون برای آزمایش این موارد، یک خاصیت دلخواه را به کلاس Project به نام public string SomeProp اضافه کنید. سپس برنامه را اجرا نمائید.
در ادامه به بانک اطلاعاتی مراجعه کرده و فیلدهای جدول Projects را بررسی کنید:

```
CREATE TABLE [dbo].[Projects](
---...
 [SomeProp] [nvarchar](max) NULL,
---...
```

بله. اینبار فیلد SomeProp بدون از دست رفتن اطلاعات و drop بانک اطلاعاتی، به جدول پروژه‌ها اضافه شده است.

عکس العمل ویژگی مهاجرت خودکار در مقابل از دست رفتن اطلاعات

در ادامه، خاصیت public string SomeProp را که در قسمت قبل به کلاس پروژه اضافه کردیم، حذف کنید. اکنون مجددا برنامه را اجرا نمائید. برنامه بلافاصله با استثنای زیر متوقف خواهد شد:

```
Automatic migration was not applied because it would result in data loss.
```

از آنجائیکه حذف یک خاصیت مساوی است با حذف یک ستون در جدول بانک اطلاعاتی، امکان از دست رفتن اطلاعات در این بین بسیار زیاد است. بنابراین ویژگی مهاجرت خودکار دیگر اعمال نخواهد شد و این مورد به نوعی یک محافظت خودکار است که درنظر گرفته شده است.
البته در EF Code first این مساله را نیز می‌توان کنترل نمود. به کلاس Configuration اضافه شده توسط پاورشل مراجعه کرده و خاصیت AutomaticMigrationDataLossAllowed را به true تنظیم کنید:

```
internal sealed class Configuration : DbMigrationsConfiguration<EF_Sample02.Sample2Context>
{
        public Configuration()
        {
            this.AutomaticMigrationsEnabled = true;
            this.AutomaticMigrationDataLossAllowed = true;
        }
```

این تغییر به این معنا است که خودمان صریحا مجوز حذف یک ستون و اطلاعات مرتبط به آن‌را صادر کرده‌ایم.
پس از این تغییر، مجددا برنامه را اجرا کنید. ستون SomeProp به صورت خودکار حذف خواهد شد، اما اطلاعات رکوردهای موجود تغییری نخواهند کرد.

استفاده از Code first migrations بر روی یک بانک اطلاعاتی موجود

تفاوت یک دیتابیس موجود با بانک اطلاعاتی تولید شده توسط EF Code first در نبود جدول سیستمی dbo.__MigrationHistory است.
به این ترتیب زمانیکه فرمان enable-migrations را در یک پروژه EF code first متصل به بانک اطلاعاتی قدیمی موجود اجرا می‌کنیم، پوشه Migration در آن ایجاد خواهد شد اما تنها حاوی فایل Configuration.cs است و نه فایلی شبیه به number_InitialCreate.cs .
بنابراین نیاز است به صورت صریح به EF اعلام کنیم که نیاز است تا جدول سیستمی dbo.__MigrationHistory و فایل number_InitialCreate.cs را نیز تولید کند. برای این منظور کافی است دستور زیر را در خط فرمان پاورشل NuGet پس از فراخوانی enable-migrations اولیه، اجرا کنیم:

```
add-migration Initial -IgnoreChanges
```

با بکارگیری پارامتر IgnoreChanges، متد Up در فایل number_InitialCreate.cs تولید نخواهد شد. به این ترتیب نگران نخواهیم بود که در اولین بار اجرای برنامه، تعاریف دیتابیس موجود ممکن است اندکی تغییر کند.
سپس دستور زیر را جهت به روز رسانی جدول سیستمی dbo.__MigrationHistory اجرا کنید:

```
update-database
```

پس از آن جهت سوئیچ به مهاجرت خودکار، خاصیت AutomaticMigrationsEnabled = true را در فایل Configuration.cs همانند قبل مقدار دهی کنید.

مشاهده دستوارت SQL به روز رسانی بانک اطلاعاتی

اگر علاقمند هستید که دستورات T-SQL به روز رسانی بانک اطلاعاتی را نیز مشاهده کنید، دستور Update-Database را با پارامتر Verbose آغاز نمائید:

```
Update-Database -Verbose
```

و اگر تنها نیاز به مشاهده اسکریپت تولیدی بدون اجرای آن‌ها بر روی بانک اطلاعاتی مدنظر است، از پارامتر Script باید استفاده کرد:

```
update-database -Script
```

نکته‌ای در مورد جدول سیستمی dbo.__MigrationHistory

تنها دلیلی که این جدول در SQL Server البته (ونه برای مثال در SQL Server CE) به صورت سیستمی معرفی می‌شود این است که «جلوی چشم نباشد»! به این ترتیب در SQL Server management studio در بین سایر جداول معمولی بانک اطلاعاتی قرار نمی‌گیرد. اما برای EF تفاوتی نمی‌کند که این جدول سیستمی است یا خیر.
همین سیستمی بودن آن ممکن است بر اساس سطح دسترسی کاربر اتصالی به بانک اطلاعاتی مساله ساز شود. برای نمونه ممکن است schema کاربر متصل dbo نباشد. همینجا است که کار به روز رسانی این جدول متوقف خواهد شد.
بنابراین اگر قصد داشتید خواص سیستمی آن‌را لغو کنید، تنها کافی است دستورات T-SQL زیر را در SQL Server اجرا نمائید:

```
SELECT * INTO [TempMigrationHistory]
FROM [__MigrationHistory]
DROP TABLE [__MigrationHistory]
EXEC sp_rename [TempMigrationHistory], [__MigrationHistory]
```

ساده سازی پروسه مهاجرت خودکار

کل پروسه‌ای را که در این قسمت مشاهده کردید، به صورت ذیل نیز می‌توان خلاصه کرد:

```
using System;
using System.Data.Entity;
using System.Data.Entity.Migrations;
using System.Data.Entity.Migrations.Infrastructure;
using System.IO;

namespace EF_Sample02
{
    public class Configuration<T> : DbMigrationsConfiguration<T> where T : DbContext
    {
        public Configuration()
        {
            AutomaticMigrationsEnabled = true;
            AutomaticMigrationDataLossAllowed = true;
        }
    }

    public class SimpleDbMigrations
    {
        public static void UpdateDatabaseSchema<T>(string SQLScriptPath = "script.sql") where T :
DbContext
        {
            var configuration = new Configuration<T>();
            var dbMigrator = new DbMigrator(configuration);
            saveToFile(SQLScriptPath, dbMigrator);
            dbMigrator.Update();
        }

        private static void saveToFile(string SQLScriptPath, DbMigrator dbMigrator)
        {
            if (string.IsNullOrWhiteSpace(SQLScriptPath)) return;

            var scriptor = new MigratorScriptingDecorator(dbMigrator);
            var script = scriptor.ScriptUpdate(sourceMigration: null, targetMigration: null);
            File.WriteAllText(SQLScriptPath, script);
            Console.WriteLine(script);
        }
```

```
    }
}
```

سپس برای استفاده از آن خواهیم داشت:

```
SimpleDbMigrations.UpdateDatabaseSchema<Sample2Context>();
```

در این کلاس ذخیره سازی اسکریپت تولیدی جهت به روز رسانی بانک اطلاعاتی جاری در یک فایل نیز درنظر گرفته شده است.

تا اینجا مهاجرت خودکار را بررسی کردیم. در قسمت بعدی Code-Based Migrations را ادامه خواهیم داد.

نظرات خوانندگان

نویسنده: Naser Tahery
تاریخ: ۱۳۹۱/۰۲/۱۷ ۲۳:۰۲:۳۳

سلام و بسیار ممنون.
مفهومی به نام «Migrations» در EF 4.3 ارائه شده است . آیا این EF 4.3 توسط دات نت 4 پشتیبانی میشود؟
چون در زمینه ی وب ، هاست ها بیشتر از .NET 4 را پشتیبانی نمیکنند.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۱۸ ۰۰:۱۳:۲۱

- بله. این شماره نگارش کتابخانه است، نه خود دات نت. برای اینکه بتوانند به روز رسانی‌ها را سریعتر کنند، آن‌را از پروسه به روز رسانی‌های کل دات نت خارج کرده‌اند.
- هاست‌ها فعلا از دات نت 4.5 پشتیبانی نمی‌کنند. چون نگارش بعدی دات نت دراین تاریخ در مرحله بتا است.

نویسنده: peyman
تاریخ: ۱۳۹۱/۰۴/۰۵ ۱۳:۴۷

سلام آقای نصیری . مشکلی که دارم اینه که فقط Configuration.cs ایجاد میشه و اون یکی فایل ایجاد نمیشه ! مشکل از چی میتونه باشه ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۰۵ ۱۴:۱۸

حتما تغییراتی رو تشخیص نداده. مراحلی که طی شده، کدهای شما، ساختار بانک اطلاعاتی و اطلاعات جدول migration باید بررسی شوند.

نویسنده: میثم
تاریخ: ۱۳۹۱/۰۴/۱۳ ۱۲:۵۹

سلام و بسیار ممنون از مطالب آموزشی زیبا و واضح شما . امیدوارم خداوند در هر دو جهان پاداش این کار خیر شما را بدهد.
در مورد اینکه هاست‌ها از دات نت 4.5 پشتیبانی نمی‌کنند و ما نمی‌توانیم به عنوان مثال از «Migrations» یا برخی امکانت دیگر استفاده کنیم . آیا راه حلی برای این مساله وجود دارد یا فعلا باید این امکانات را در برنامه‌های تحت وب استفاده نکنیم ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۱۳ ۱۳:۲

شماره نگارش EF با شماره نگارش دات نت یکی نیست و مدتی هست که برای انتشار ارائه‌های منظم و در فواصل زمانی کمتر این سیاست رو در پیش گرفتن. EF 4.3 برای مثال مبتنی بر دات نت 4 است و نه دات نت 4 و نیم که در این زمان در نگارش نهایی قرار ندارد.

نویسنده: میثم
تاریخ: ۱۳۹۱/۰۴/۱۳ ۱۴:۱

ممنون از توضیح شما
من درک درستی از کامنت اول این پست و پاسخ شما نداشتم الان همه چیز روشن شد
تشکر فراوان

نویسنده: فربد صالحی
تاریخ: ۱۳۹۱/۰۵/۱۷ ۱۱:۳

اینطور که من متوجه شدم ، تا قبل از EF 4.3 ، جدولی به اسم EdmMetadata ساخته می‌شد که مدل رو به صورت hash نگهدای میکرد و فقط مشخص می‌شد که آیا مدل با بانک اطلاعاتی منطبق هست یا نه. اما چون نیاز به نگهداری اطلاعات بیشتری برای migration بود، الان جدول __MigrationHistory تولید میشه.

نویسنده: davmszd
تاریخ: ۱۳۹۱/۰۷/۱۴ ۱۹:۸

با سلام؛
من بعد از این دستورات رو تو پاور شل زدم به همچین خطایی برخوردم، جریان چیه ؟

```
The term 'enable-migrations' is not recognized as the name of a cmdlet, function, script file, or
operable program. Check the spelling of the name, or if a path was included, veri
fy that the path is correct and try again.
At line:1 char:18
+ enable-migrations <<<<
    + CategoryInfo          : ObjectNotFound: (enable-migrations:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۱۴ ۱۹:۲۱

فایل‌های مرتبط با migrations فقط از طریق NuGet به همراه بسته EF دریافت می‌شوند. بنابراین داشتن فایل‌های DLL مربوط به EF کافی نیست. بعد از آن، این فرامین از طریق پاورشل خود NuGet در vs.net باید اجرا شوند.
در کل اگر می‌خواهید بدانید این بسته درست نصب شده یا نه، دستور زیر را در پاورشل خود NuGet اجرا کنید:

Get-Package

نویسنده: davmszd
تاریخ: ۱۳۹۱/۰۷/۱۶ ۱۶:۴

ممنون از پاسختون با این سرعت :)
فک کنم مشکل از اینجا بود که من DLL های EF رو خودم دستی اضافه کرده بودم و ....
یه بار تمام ارجاع‌ها ش تو پروژه رو حذف کردم و بااستفاده از NUGET دوباره نصبش کردم البته NUGET Manager رو هم با استفاده از Extension Manager به روز رسانی کردم
بازم ممنون از زحماتتون جناب نصیری

نویسنده: حسین
تاریخ: ۱۳۹۱/۰۷/۲۲ ۱۵:۱۹

واقعا عالی بود ممنون

نویسنده: نوید
تاریخ: ۱۳۹۱/۱۲/۰۳ ۱۲:۵۸

سلام
ابتدا از مطالب مفیدتون سپاسگزارم.
من مدل هارو در یک Class Library جداگانه و context رو هم در یک Class Library جداگانه قرار دادم و یک Reference از اون‌ها به پروژه اصلی اضافه کردم.
الان وقتی در Package manager Console دستور enable-migrations رو وارد میکنم خطای
'No context type was found in the assembly 'Online_Store.

رو میده که Online_Store نام پروژه اصلیم است.
ممنون میشم راهنماییم کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۰۳ ۱۳:۲۹

در متن توضیح دادم:
«... ابتدا باید دقت داشت که قرار است فرامین را بر روی چه پروژه‌ای اجرا کنیم. برای مثال اگر تعاریف DbContext را به یک اسمبلی و پروژه class library مجزا انتقال داده‌اید، گزینه *Default project* را در این قسمت (Nuget package manager console) باید به این پروژه مجزا، تغییر دهید . .. »

نویسنده: نوید
تاریخ: ۱۳۹۱/۱۲/۰۴ ۱۱:۳۰

ممنونم از راهنماییتون.
با این کار فایل Configuration ایجاد میشه ولی اون فایل دوم ایجاد نمیشه . وقتی بقیه دستورات رو هم اجرا میکنم(Update-datebase و ...) هم خطای
An error occurred while getting provider information from the database. This can be caused by Entity Framework using an incorrect connection string. Check the inner exceptions for details and ensure that the connection string is correct.

رو میده .
تو فایل app.config ،هم Connection string رو اضافه کردم ولی باز هم همین خطا رو میده!

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۰۴ ۱۱:۴۸

قسمت اول را در مورد نحوه صحیح تعریف رشته اتصالی مجددا مطالعه کنید .

نویسنده: امیر
تاریخ: ۱۳۹۱/۱۲/۲۶ ۱۳:۹

سلام من هر کاری میکنم نمیتونم مشکلم حل کنم این خطا رو میده دوباره نصب کردم باز این خطا رو میده اگه می‌تونید یه کمکی بکنید
((The parameter is incorrect. (Exception from HRESULT: 0x80070057 (E_INVALIDARG

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۲۶ ۱۳:۲۶

اینجا انجمن نیست. من از راه دور نمی‌تونم به شما کمک کنم. نمی‌دونم چکار کردی، تنظیماتت چی هست.
اگر در حین کار با enable-migrations این خطا رو گرفتید، سعی کنید دقیق‌تر کار کنید:

```
enable-migrations -StartUpProjectName "prj name" -ContextTypeName "ctx name"
```

نویسنده: امیر
تاریخ: ۱۳۹۱/۱۲/۲۷ ۱۹:۶

با سلام
اینو خطا میده

```
Database.SetInitializer(new MigrateDatabaseToLatestVersion<Context,Migrations.Configuration>);
```

Migrations.Configuration,
نمیشناسه.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۲۷ ۱۹:۱۳

سورس‌های این سری رو دریافت کنید . کلاس Migrations.Configuration یکی از کلاس‌های سفارشی تعریف شده در sample02 است.

نویسنده: Hamid NCH
تاریخ: ۱۳۹۲/۰۴/۲۳ ۱۲:۵۲

یه مشکلی که من برای بروزرسانی دیتابیسم توسط این روش دارم اینه که وقتی برای بار اول دستور Update-database رو اجرا میکنم دیتابیس بدون هیچ مشکلی ساخته میشه.اما اگه برای بار دوم و بیشتر این دستور اجرا بشه با خطای زیر مواجه میشم:

Sequence contains more than one element

حالا چه کلاسهام رو تغییر بدم چه ندم و این در صورتیه که پیکریندیم به این روش هست:

```
internal sealed class Configuration : DbMigrationsConfiguration<GlucosanContext>
  {
   public Configuration()
   {
    AutomaticMigrationsEnabled = true;
    AutomaticMigrationDataLossAllowed = true;
   }
```

و همچنین تو کلاس Program این دستور رو نوشتم:

```
Database.SetInitializer(new MigrateDatabaseToLatestVersion<GlucosanContext,
Migrations.Configuration>());
```

پروژه بنده ویندوز فرم هست.باتشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۴/۲۳ ۱۲:۵۷

زمانیکه از روش AutomaticMigrationsEnabled به همراه AutomaticMigrationDataLossAllowed استفاده می‌کنید (تنظیم شده به true البته)، نیازی نیست هیچ کار اضافه‌تری انجام بدید؛ همه چیز خودکار است. به روز رسانی ساختار بانک اطلاعاتی، کم و زیاد کردن فیلدها و غیره همگی خودکار است.
بنابراین اصلا نیازی نیست دستورات پاورشل را اجرا کنید و اگر قبلا اینکار انجام شده و یک سری فایل اضافی migration دارید، همه رو حذف کنید تا تداخل ایجاد نکنند.

نویسنده: Hamid NCH
تاریخ: ۱۳۹۲/۰۴/۲۳ ۱۳:۲۲

خیلی ممنون؛ اما بفرض مثال من یه فیلد به یکی از جدولهام اضافه می‌کنم.طبعا باید دیتابیس دوباره بروزرسانی بشه.و این عمل بروزرسانی اگه درست متوجه شده باشم طبق فرمایش شما با ست کردن AutomaticMigrationsEnabled به true باید انجام بشه.که نمیشه.یا اینکه باید دوباره دستور enable-database -force رو تو پاورشل اجرا کنم که این هم منجر به خطایی که عرض کردم میشه.
در کل بنده هربار که تغییری تو دیتابیسم میدم با اینکه دارم از Migration استفاده می‌کنم مجبورم که دیتابیس رو از sql server پاک کنم و دوباره ایجادش کنم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۴/۲۳ ۱۴:۷

- شما نباید دستی تغییری در دیتابیس ایجاد کنید. این روش Code first است. تغییرات باید شامل افزودن خاصیت به کلاس‌ها باشند.
- نباید دستور پاورشلی رو اجرا کنید اگر AutomaticMigrationsEnabled فعال است؛ چون سبب بروز تداخل می‌شود.
- روش Code first، کار به روز رسانی بانک اطلاعاتی رو تا زمان اجرای اولین کوئری به تاخیر می‌اندازد (اینطوری طراحی شده تا آغاز برنامه سریع به نظر برسد). روش دیگری هم وجود داره تا این مساله رو تغییر داد:

« وادار کردن EF Code first به ساخت بانک اطلاعاتی پیش از شروع به کار برنامه »

نویسنده: یوسف
تاریخ: ۱۳۹۲/۰۵/۰۶ ۱۵:۴۶

سلام آقای نصیری؛
منم همین پیام را دریافت می‌کنم، ولی من EF نسخهٔ 5.0 را استفاده می‌کنم و اونو هم با NuGet به پروژه اضافه کرده‌م. ویژوال استدیو نسخهٔ 2012 هست و با دات‌نت 4.5 برنامه را ایجاد کرده‌م. برنامه تا آخر درس قبل کاملاً همونطور که انتظار می‌رفت اجرا شد و مشکلی هم نداشت.
اما به محض باز کردن package manager console از منوی Tools، بعد از معرفی نسخهٔ کنسول Package Manager Console Host Version 2.6.40627.9000 خطوط زیر را با زمینهٔ قرمز می‌نویسه:

```
Test-ModuleManifest : The specified module 'D:\Entity Framework Samples\EF Sample
02\packages\EntityFramework.5.0.0\tools\EntityFramework.psd1' was not loaded because no valid module
file was found in any module directory.
At D:\Entity Framework Samples\EF Sample 02\packages\EntityFramework.5.0.0\tools\init.ps1:14 char:34
+ $thisModule = Test-ModuleManifest <<<<  (Join-Path $toolsPath $thisModuleManifest)
    + CategoryInfo          : ResourceUnavailable: (D:\Entity Framework
Samples\E...yFramework.psd1:String) [Test-ModuleManifest], FileNotFoundException
    + FullyQualifiedErrorId :
Modules_ModuleNotFound,Microsoft.PowerShell.Commands.TestModuleManifestCommand

Import-Module : Cannot bind argument to parameter 'Name' because it is null.
At D:\Entity Framework Samples\EF Sample 02\packages\EntityFramework.5.0.0\tools\init.ps1:31 char:18
+     Import-Module <<<<  $thisModule
    + CategoryInfo          : InvalidData: (:) [Import-Module], ParameterBindingValidationException
    + FullyQualifiedErrorId :
ParameterArgumentValidationErrorNullNotAllowed,Microsoft.PowerShell.Commands.ImportModuleCommand
```

برای دستور enable-migrations هم دقیقاً همون چیزی را می‌نویسه که در کامنت اول **davmszd** بیان شده، یعنی اینو:

```
The term 'enable-migrations' is not recognized as the name of a cmdlet, function, script file, or
operable program. Check the spelling of the name, or if a path was included, verify that the path is
correct and try again.
At line:1 char:18
+ enable-migrations <<<<
    + CategoryInfo          : ObjectNotFound: (enable-migrations:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException
```

و هنگامی هم که دستور Get-Packageرا اجرا می‌کنم، اینو می‌نویسه:

```
Id                              Version            Description/Release Notes
--                              -------            -------------------------
EntityFramework                 5.0.0              Entity Framework is Microsoft's recommended data
access technology for new applications.
```

ضمناً در پوشهٔ packages در کنار پروژه، فولدری بنام EntityFramework.5.0.0 و داخل اون هم فولدر tools با این فایل‌ها وجود داره:

```
about_EntityFramework.help.txt
```

```
EntityFramework.PowerShell.dll
EntityFramework.PowerShell.Utility.dll
EntityFramework.PS3.psd1
EntityFramework.psd1
EntityFramework.psm1
init.ps1
install.ps1
migrate.exe
Redirect.config
Redirect.VS11.config
```

ممنون میشم اگر منو راهنمایی بفرمایید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۵/۰۶ ۱۷:۲۱

powershell ویندوز را اجرا کنید (خارج از ویژوال استودیو) . بعد در خط فرمان آن دستور زیر را وارد نمائید:

```
$psversiontable.psversion
```

اگر Major آن مساوی 2 بود، یعنی از نگارش 2 پاورشل در حال استفاده هستید که باید به روز شود به نگارش 3 .

نویسنده: یوسف
تاریخ: ۱۳۹۲/۰۵/۰۸ ۲۰:۲۹

از توجهتون سپاسگزارم.
مهندس جان من اینکار را انجام دادم و پاورشل را هم ارتقاء دادم، ولی مشکل برطرف نشد.
یه پروژهٔ جدید ایجاد کردم و همهٔ فایل‌های کد پروژهٔ قبلی (بغیر از app.config) را بهش اضافه کردم و بعد با استفاده از NuGet نسخهٔ آخر EF را به پروژه اضافه کردم. کانکشن استرینگ را به پروژه اضافه نکردم (کلاً به app.config دست نزدم) و دستور **enable-migrations** را اجرا کردم. دستور با موفقیت اجرا شد و اون چیزهایی که فرمودین به پروژه اضافه شد.
دیتابیس را به صورت دستی Drop کردم و یه بار برنامه را اجرا کردم و بعد از اون هم تغییراتی که به مدل دادم به دیتابیس اعمال شد و رکوردهای مربوط بهش داخل جدول dbo.__MigrationHistoryثبت شد.

رفتم به درس پنجم، و دوباره به محض بازکردن کنسول همون پیغامی را که توی کامنت اولم نوشتم می‌نویسه و وقتی هم می‌خوام که دستور Add-Migration را اجرا کنم، همون چیزی را می‌نویسه که قبلاً برای enable-migrations می‌نوشت. جالب اینکه برای خود دستور enable-migrations هم همینو می‌نویسه!برای دستور Update-Database هم همینطور، در حالی که برای پروژهٔ شما می‌نوشت که هم اکنون فعال هست.

فکرمی‌کنم مشکل از فایل **init.psd1** باشه که هنگام اضافه کردن EF در فولدر tools ایجاد میشه.
من فایل init.psd1 را که در پوشهٔ tools مربوط به پروژهٔ ایجاد شده توسط شما قرار داشت جایگزین فایل با همین نام که در پروژهٔ خودم بود کردم و مشکل برطرف شد. نمی‌دونم ایراد کار چیه؟ من قدم به قدم همونطور که شما نوشتین پیش رفتم و چیزی را هم تغییر نداده‌م. آیا ممکنه توی پروژه‌هایی که می‌خوام از EF استفاده کنم این مسئله دردسرساز بشه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۵/۰۸ ۲۰:۴۲

این مورد احتمالا یک باگ هست که اگر بهشون گزارش کنید بهتره تا برای همه اعمال شود. عنوان کنید بسته نیوگت دریافتی دات نت 4.5 فایل init.ps1 مشکل داری داره و اگر اون رو با یک نمونه از بسته نیوگت دات نت 4 اندکی قدیمی‌تر جایگزین کنم مشکلی نیست. تمام خطاها رو هم دقیقا گزارش بدید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۵/۰۸ ۲۰:۵۸

ضمنا یک سری تجربه با این خطا در اینجا هم هست:
الف) عنوان شده آخرین نسخه بتا این مشکل رو نداره (Install-Package EntityFramework -IncludePrerelease)
ب) VS.NET را با دسترسی Admin اجرا کنید.
ج) یا یک نفر دیگر در اینجا عنوان کرده که پروژه حتما باید در VS.NET باز شده باشد.
د) یا شخص دیگری عنوان کرده که آدرس فایل‌های پروژه اگر مثلا یک [داشته باشد، کار نمی‌کند.

نویسنده: یوسف
تاریخ: ۱۳۹۲/۰۵/۰۹ ۵:۰

نمی‌دونم چطور تشکر کنم و از وقتی که برای حل شدن مشکل من گذاشته‌اید بی‌اندازه ممنونم.
مشکل حل شد! من همهٔ این چهار مورد را بررسی کردم و اشکال همون وجود کروشه در مسیر پروژه بود. من پروژه‌هام را در فولدری به نام [Projects] نگهداری میکنم و تغییر نام دادن اون حذف کروشه‌ها خطا را برطرف کرد.

ضمناً موارد دیگه را هم امتحان کردم:
مورد الف هم خطا را برطرف می‌کنه. یعنی به نظر میرسه که EF 6.0 این مشکل را نخواهد داشت.
مورد ب هیچ تأثیری نداشت و مورد ج هم که اصلاً مطرح نبود (چون پروژه داخل VS باز می‌شد.)

پاینده و پیروز باشید.

نویسنده: imo0
تاریخ: ۱۳۹۲/۰۶/۰۳ ۱۶:۳۷

سلام آقای نصیری . من یه گیر اساسی کردم تو این مسئله مهاجرت دیتابیس. ببینین من دارم یه ساختاری به شکل ماژولار تو یه وب برای خودم درست می‌کنم . تمام این ماژول هام در نهایت میشن یک دی ال ال .و تمام مسائل مربوط به اونا به صورت خودکار توسط خود ماژول انجام و مدیریت میشه و ...
یکی از این مسائل، ایجاد و بروز رسانی دیتابیس هستش که من با همین Code First Migration ایجاد کردم. مشکل اینجاست که وقتی این ماژول هام اجرا میشن هر کدومشون میخوان دیتا بیسو آپدیت کنند و یا جداولشونو ایجاد کنند اما هر ماژولی میاد اول همه جداول دیتابیسو پاک میکنه بعد ماله خودشو ایجاد میکنه.
این Contexها و migrationهای من هر کدوم جداگانه تو یه دی ال ال هستند. من چطور به اینا حالی کنم که فقط یه سری جدول خاصو چک کنند اگه نبود ایجاد و اگر بود آپدیت کنند و در نهایت متد Seed شون رو فراخوانی کتنند و به بقیه جداولم کار نگیرن... .
من حتی از MigrateDatabaseToLatestVersion هم استفاده میکنم اما فایده نداره . چون گفتم Contextهای من از هم خبر ندارند و هر کدومشون فکر میکنه فقط دیتابیس ماله خودشه . میخوام برای migration تعریف کنم و فقط این مهاجرت رو روی یه سری تیبل خاص بررسی کنه . ممنون . منتظر جوابم ...

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۶/۰۳ ۱۷:۳۴

یک کلاس Context و یک کلاس مهاجرت مرکزی درست کنید که با استفاده از Reflection تمام ماژول‌ها را بارگذاری کرده و تعاریف DbSetها را از روی آن‌ها بر اساس مثلا کلاس پایه‌ای که موجودیت‌های آن‌ها از آن ارث بری می‌کنند، ایجاد کند. در این مورد مطلب داریم در سایت:
خودکار کردن تعاریف DbSetها در EF Code first
افزودن خودکار کلاس‌های تنظیمات نگاشت‌ها در EF Code first

نویسنده: reza110
تاریخ: ۱۳۹۲/۰۸/۰۸ ۱۱:۱۷

بر اساس دیتابیس موجود که دارای اطلاعات است مدل را می‌سازم می‌خواستم migration را فعال کنم. بر اساس دستورات پاور شل قبلا آموزش داده اید می‌خواستم ببینم چگونه می‌توان این کار را بصورت code base انجام داد. ظاهرا دستورات پاورش معادل code base هم دارند.
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۰۸ ۱۱:۵۰

از کدهای کلاس SimpleDbMigrations ذکر شده در انتهای مطلب استفاده کنید یا ایده بگیرید.
ضمنا سورس کامل ابزارهای migration نیز در دسترس است.

نویسنده: reza110
تاریخ: ۱۳۹۲/۰۸/۱۱ ۱۰:۲۸

یعنی راهکار ساده‌تری وجود دارد که معادل سه دستور پاور شل زیر باشد

```
enable-migrations
add-migration Initial -IgnoreChanges
update-database
```

با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۱۱ ۱۱:۱۶

- این‌ها ساده هستند و قابل اجرا بر روی دیتابیس ریموت (فقط باید ConnectionString را صریحا ذکر کنید):

```
Update-Database -StartUpProjectName "...name..." -ConnectionString "...data..." -ConnectionProviderName "System.Data.SqlClient"
```

- و بله. «ویژگی مهاجرت خودکار» را در برنامه فعال کنید و لذت ببرید. نیازی به هیچ دستور پاور شلی ندارد. هر زمان که مدل‌های شما تغییر کرد، به صورت خودکار ساختار بانک اطلاعاتی را در اولین اجرای بعدی برنامه، به روز می‌کند.
- « بازسازی جدول MigrationHistory با کد نویسی در EF Code first »

نویسنده: محبوبه محمدی
تاریخ: ۱۳۹۲/۰۸/۲۹ ۱۴:۳۹

سلام و وقتتون بخیر
ممنون بابت مطلبتون.من دقیقا طبق مطلب شما Migrations رو پیاده کردم.اما یه مشکل دارم.اونم اینه که وقتی دیتابیسم برا اولین بار میخواد ساخته بشه ،خطای لاگین میده:

```
Cannot open database "Test" requested by the login. The login failed.
Login failed for user 'sa'.
```

البته این خطا فقط در صورتی داده میشه که مهاجرت خودکار فعال شده باشه،اگر این خط رو کامنت کنم دیتابیس بدون مشکل ساخته میشه:

```
Database.SetInitializer(new MigrateDatabaseToLatestVersion<CommonContext, Configuration>());
```

با توجه به مطلب شما حدس می‌زدم به خاط نکته ای باشه که در مورد جدول dbo.__MigrationHistory گفتید.اما با استفاده از

این لینک جدول رو از اول به صورت غیر سیستمی میسازم اما بازم مشکل دارم.البته دیتا بیس ایجاد میشه،فقط جدول dbo.__MigrationHistory رو میسازه و بعدش خطایی که گفتم رو میده.
ممنون.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۲۹ ۱۴:۵۶

قسمت connectionStrings در فایل کانفیگ (name و connectionString اضافه شده) را چطور تعریف کردید؟ Cannot open database مشکل اتصال و مشکل سطح دسترسی اکانت مورد استفاده است). ضمنا محتوای inner exception داده شده را هم بررسی کنید.

نویسنده: محبوبه محمدی
تاریخ: ۱۳۹۲/۰۸/۲۹ ۱۵:۷

connectionStrings رو به این دو صورت میگذارم:

```
<add name="TestContext"
          connectionString="Data Source=localhost;Initial Catalog=Test;User
ID=sa;Password=123;MultipleActiveResultSets=True;"
          providerName="System.Data.EntityClient" />
```

و

```
<add name="TestContext"
        connectionString="Data Source=localhost;Initial Catalog=Test;Integrated Security=True"
        providerName="System.Data.EntityClient" />
```

هر دو یه مشکل رو دارند.فکر نمی‌کنم چون اگر مشکل دسترسی اکانت بود منطقا بدون Migration هم باید خطا می‌داد؟!

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۲۹ ۱۵:۱۶

- برای انجام اعمال مختلف در SQL Server، سطوح دسترسی مختلفی وجود دارند. یک کاربر می‌تواند دسترسی درج رکوردها را داشته باشد، اما دسترسی ایجاد یا تغییر ساختار بانک اطلاعاتی را نداشته باشد.
- در EF 6 این جدول MigrationHistory دیگر سیستمی نیست.
- یوزر sa دسترسی مدیریتی دارد (حالت اول). احتمالا در حالت دوم که یکپارچه با ویندوز است، اکانت وارد شده به سیستم نیز admin است؛ وگرنه دسترسی لازم را نخواهد داشت که دیتابیس ایجاد کند.

نویسنده: محبوبه محمدی
تاریخ: ۱۳۹۲/۰۸/۲۹ ۱۵:۴۰

-الان یکبار دیگه یه پروژه کوچیک با EF6 ایجاد کردم و مشکلی نداشت.آیا امکانش هست مربوط به ورژن EFم باشه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۲۹ ۱۷:۱۵

ممکنه در نگارش‌های اولیه EF Code first از این نوع خطاها وجود داشته و بعدا برطرف شده. در مورد ارتقاء به EF 6 به این مطالب مراجعه کنید: « ارتقاء به Entity framework 6 و استفاده از بانک‌های اطلاعاتی غیر از SQL Server » و همچنین « بروز رسانی استفاده از SqlServer Compact در Entityframework 6.0 »

نویسنده: Behnam
تاریخ: ۱۳۹۲/۱۰/۰۳ ۱۷:۳

با سلام و عرض خسته نباشید
من از کد قسمت **ساده سازی پروسه مهاجرت خودکار** استفاده کردم، با EF6، مشکلی که هست اینه که وقتی یک فیلد رو کم یا زیاد میکنم پیغام زیر رو میده:

```
The model backing the 'Sample2Context' context has changed since the database was created. Consider
using Code First Migrations to update the database (http://go.microsoft.com/fwlink/?LinkId=238269).
```

ولی مثال قبلتون با استفاده از

```
Database.SetInitializer(new MigrateDatabaseToLatestVersion<Sample2Context,
Migrations.Configuration>());
```

هیچ مشکلی نداره و اجرا میشه، علت خطای قبل از چه چیزی میتونه باشه؟
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۰۳ ۲۲:۴۳

با EF6 هم مشکلی مشاهده نشد:

Sample27.cs

مثال فوق از کلاس DbMigrator استفاده می‌کند. متد Test.RunTests آن‌را اجرا کنید؛ البته بعد از اضافه کردن Connection1 که در سورس ذکر شده. بانک اطلاعاتی جدیدی ساخته می‌شود. سپس به کلاس منو یک فیلد جدید اضافه کنید و مجددا برنامه را اجرا کنید. مشاهده خواهید کرد که این فیلد اضافه شده و خطایی صادر نمی‌شود.

نویسنده: اس ام
تاریخ: ۱۳۹۲/۱۲/۰۶ ۱۵:۵۴

سلام
آیا امکان این وجود داره که Connection string و تنظیمات مربوط به اون مثل نام کاربری دیتابیس و رمز عبور و نام دیتابیس ،رو موقع نصب اولین بار برنامه از کاربر دریافت کنیم؟ مثل دات نت نیوک،که همه عملیات به صورت داینامیک انجام بشه؟
و اینکه این عملیات فقط یک بار در هنگام اولین نصب برنامه انجام بگیره و در ادامه دیگه این کار انجام نشه؟
اگر بله! مکان قرار دادن کد ایجاد دیتابیس از روی مدل رو کجای برنامه قرار بدیم بهتره؟ مثلا تو پروژه MVC

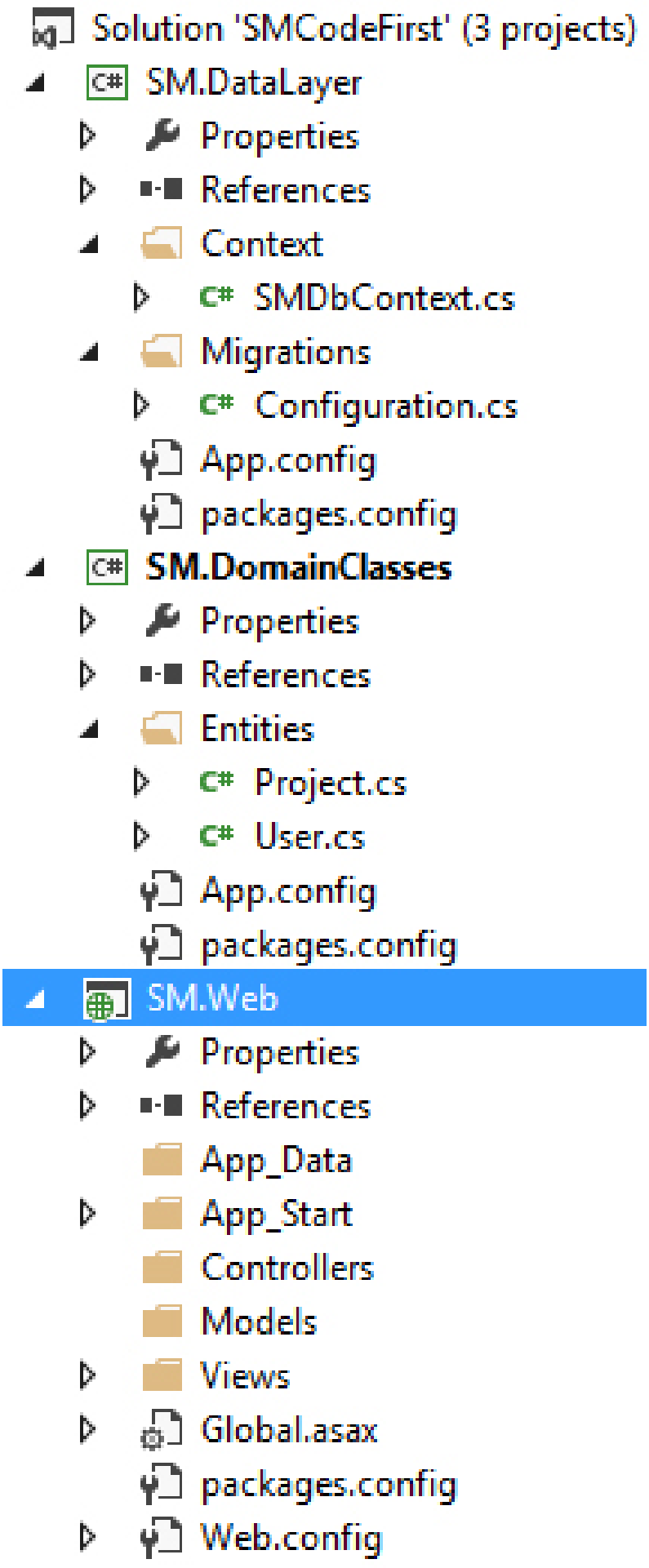
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۰۶ ۱۶:۲۳

روش‌های زیادی برای تعیین رشته اتصالی در EF وجود دارند. این موارد به همراه نظرات و مطلب « نحوه‌ی وادار کردن EF Code first به ساخت بانک اطلاعاتی پیش از شروع به کار برنامه » بحث شدند. کدهای آن اگر قرار است در حین نصب اولیه اجرا شوند، می‌توانند در همان روال مثلا دکمه‌ی نصب یا آغاز به نصب قرار گیرند. ابتدا مثلا ctx.Database.Connection.ConnectionString مقدار دهی می‌شود و بعد نکته‌ی وادار سازی EF به ساخت بانک اطلاعاتی.
پس از انجام اینکار می‌توان این اطلاعات را در فایل کانفیگ برنامه برای استفاده‌های بعدی ذخیره کرد. کلاس WebConfigurationManager امکان ویرایش قسمت‌های مختلف فایل کانفیگ برنامه را می‌دهد.

نویسنده: اس ام
تاریخ: ۱۳۹۳/۰۱/۰۶ ۱۳:۴۶

سلام؛ من این مقاله رو خوندم و این پروژه رو تو MVC5 و vs2013 update1 و EF6 انجام دادم. همه چی درسته ولی کلاس

InitialCreate رو ایجاد نمیکنه. آیا باید این کلاس رو دستی ایجاد کنم؟ من مدل‌ها و context رو تو اسمبلی‌های جدا گانه گذاشتم.


Solution 'SMCodeFirst' (3 projects)
SM.DataLayer
Properties
References
Context
SMDbContext.cs
Migrations
Configuration.cs
App.config
packages.config
SM.DomainClasses
Properties
References
Entities
Project.cs
User.cs
App.config
packages.config
SM.Web
Properties
References
App_Data
App_Start
Controllers
Models
Views
Global.asax
packages.config
Web.config

در ضمن وقتی میخوام Enable-migrations رو انجام بدم اگه تو DataLayer به اسمبلی Web ارجاعی نداشته باشم، میگه Connectionstring درست نیست ولی کلاس مربوط به migration رو میسازه. ممنون میشم راهنمایی کنید. چون در نهایت باید در اسمبلی web به DataLayer ارجاعی داشته باشیم دیگه.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۰۶ ۱۶:۴۳

- InitialCreate زمانی ایجاد خواهد شد که دیتابیسی موجود است و اکنون در مدل‌های شما تغییری حاصل شده‌است. اگر بار اول است، نیازی به آن نیست (ایجاد نخواهد شد) و در حین ایجاد اولیه بانک اطلاعاتی، تمام مراحل لازم طی می‌شوند.
- رشته اتصالی را در فایل کانفیگ پروژه‌ای که مهاجرت روی آن فعال می‌شود نیز قرار دهید.
+ تمام این دستورات پارامتر رشته اتصالی هم دارند:

```
Update-Database -Verbose
 -ConnectionString "CONNECTIONSTRING"
 -ConnectionProviderName "System.Data.SqlClient"
 -StartupProjectName WEBSITE_PROJECT -ProjectName MIGRATION_PROJECT
```

نویسنده: مصطفی
تاریخ: ۱۳۹۳/۰۷/۱۸ ۱۵:۵۷

من مراحل بالا رو رفتم اما مشکلی که هست اینه که دفعه اول که دستور update-databse رو اجرا می‌کنم دیتابیس ایجاد میشه اما دفعه دوم بااجرای این دستور پیغام خطای زیر میاد

```
.Cannot open database "*****" requested by the login. The login failed.
```

```
'*****'  Login failed for user
```

در واقع برنامه به دیتابیسی که خودش ساخته دسترسی نداره .

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۷/۱۸ ۱۸:۵۳

در خطای نهایی، نام کاربر مشخص شده. بررسی کنید آیا دسترسی کافی دارد یا خیر.
همچنین این اطلاعات را صریحا هم می‌شود مشخص کرد:

```
Update-Database -StartUpProjectName "...name..." -ConnectionString "...data..." -ConnectionProviderName
"System.Data.SqlClient"
```

نویسنده: علیرضا م
تاریخ: ۱۳۹۳/۰۷/۲۴ ۱۱:۵۲

سلام
در صورت نیاز به بررسی تطابق مدل با پایگاه داده در نرم افزار :

```
bool isCompatible = Context.Database.CompatibleWithModel(true);
```

نویسنده: زینب
تاریخ: ۱۳۹۳/۰۷/۲۶ ۱۴:۰

سلام و باتشکر

من migration را فعال کردم ولی کلاس دوم که حاوی متدهای dwon, up هست را برام نساخته و زمانی که نوع خاصیت جدولی را

تغییر می‌دم یا جدولی را دستی حذف می‌کنم پیغام خطای زیر را می‌بینم برای اینکه این پیغام را نبینم چه کار کنم

.There is already an object named 'tablename' in the database

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۷/۲۶ ۱۴:۵۹

مورد پنجم « بررسی خطاهای متداول عملیات Migration در حین به روز رسانی پروژه‌های EF Code First »

نویسنده: عثمان رحیمی
تاریخ: ۱۳۹۳/۱۱/۱۹ ۲۰:۶

با سلام؛ زیاد متوجه کاربرد کلاس SimpleDbMigrations نشدم . آیا این کلاس رو فقط به جای متد Seed نوشتید ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۱/۱۹ ۲۰:۲۹

بیشتر هدف آن آشنایی با نحوه‌ی کارکرد این پروسه است. یک نمونه‌ی دیگر: « بازسازی جدول MigrationHistory با کد نویسی در EF Code first »