عنوان: EF Code First #5
نویسنده: وحید نصیری
تاریخ: ۰۹:۰۵:۰۰ ۱۳۹۱/۰۲/۱۸
آدرس: www.dotnettips.info
گروه‌ها: Entity framework

در قسمت قبل خاصیت AutomaticMigrationsEnabled را در کلاس Configuration به true تنظیم کردیم. به این ترتیب، عملیات ساده شده، اما یک سری از قابلیت‌های ردیابی تغییرات را از دست خواهیم داد و این عملیات، صرفا یک عملیات رو به جلو خواهد بود.
اگر AutomaticMigrationsEnabled را مجددا به false تنظیم کنیم و هربار به کمک دستوارت Add-Migration و Update-Database تغییرات مدل‌ها را به بانک اطلاعاتی اعمال نمائیم، علاوه بر تشکیل تاریخچه این تغییرات در برنامه، امکان بازگشت به عقب و لغو تغییرات صورت گرفته نیز مهیا می‌گردد.

هدف قرار دادن مرحله‌ای خاص یا لغو آن

به همان پروژه قسمت قبل مراجعه نمائید. در کلاس Configuration آن، خاصیت AutomaticMigrationsEnabled را به false تنظیم کنید. سپس یک خاصیت جدید را به کلاس Project اضافه نموده و برنامه را اجرا نمائید. بلافاصله خطای زیر را دریافت خواهیم کرد:

```
Unable to update database to match the current model because there are pending changes and
automatic migration is disabled. Either write the pending model changes to a code-based migration
or enable automatic migration. Set DbMigrationsConfiguration.AutomaticMigrationsEnabled to true
to enable automatic migration.
```

EF تشخیص داده است که کلاس مدل برنامه، با بانک اطلاعاتی تطابق ندارد و همچنین ویژگی مهاجرت خودکار نیز فعال نیست. بنابراین اعمال code-based migration را توصیه کرده است.
برای این منظور به کنسول پاورشل NuGet مراجعه نمائید (منوی Tools در ویژوال استودیو، گزینه Library package manager آن و سپس انتخاب گزینه package manager console). در ادامه فرمان add-m را نوشته و دکمه tab را فشار دهید. یک منوی Auto Complete ظاهر خواهد شد که از آن می‌توان فرمان add-migration را انتخاب نمود. در اینجا یک نام را هم نیاز است وارد کرد؛ برای مثال:

```
Add-Migration AddSomeProp2ToProject
```

به این ترتیب کلاس زیر را به صورت خودکار تولید خواهد کرد:

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

    public partial class AddSomeProp2ToProject : DbMigration
    {
        public override void Up()
        {
            AddColumn("Projects", "SomeProp", c => c.String());
            AddColumn("Projects", "SomeProp2", c => c.String());
        }

        public override void Down()
```

```
        {
            DropColumn("Projects", "SomeProp2");
            DropColumn("Projects", "SomeProp");
        }
    }
}
```

مدل‌های برنامه را با بانک اطلاعاتی تطابق داده و دریافته است که هنوز دو خاصیت در اینجا به بانک اطلاعاتی اضافه نشده‌اند.
از متد Up برای اعمال تغییرات و از متد Down برای بازگشت به قبل استفاده می‌گردد. نام فایل این کلاس هم طبق معمول چیزی است شبیه به timeStamp_AddSomeProp2ToProject.cs .

در ادامه نیاز است این تغییرات به بانک اطلاعاتی اعمال شوند. به همین منظور دستور زیر را در کنسول پاورشل وارد نمائید:

```
Update-Database -Verbose
```

پارامتر Verbose آن سبب خواهد شد تا جزئیات عملیات به صورت مفصل گزارش داده شود که شامل دستورات ALTER TABLE نیز هست:

```
Using NuGet project 'EF_Sample02'.
Using StartUp project 'EF_Sample02'.
Target database is: 'testdb2012' (DataSource: (local), Provider: System.Data.SqlClient, Origin:
Configuration).
Applying explicit migrations: [201205061835024_AddSomeProp2ToProject].
Applying explicit migration: 201205061835024_AddSomeProp2ToProject.
ALTER TABLE [Projects] ADD [SomeProp] [nvarchar](max)
ALTER TABLE [Projects] ADD [SomeProp2] [nvarchar](max)
[Inserting migration history record]
```

اکنون مجددا یک خاصیت دیگر را مثلا به نام public string SomeProp3، به کلاس Project اضافه نمائید.
سپس همین روال باید مجددا تکرار شود. دستورات زیر را در کنسول پاورشل NuGet اجرا نمائید:

```
Add-Migration AddSomeProp3ToProject
Update-Database -Verbose
```

اینبار نیز یک کلاس جدید به نام AddSomeProp3ToProject به پروژه اضافه خواهد شد و سپس بر اساس آن، امکان به روز رسانی بانک اطلاعاتی میسر می‌گردد.

در ادامه برای مثال به این نتیجه رسیده‌ایم که نیازی به خاصیت public string SomeProp3 اضافه شده، نبوده است. روش متداول، باز هم مانند سابق است. ابتدا خاصیت را از کلاس Project حذف خواهیم کرد و سپس دو دستور Add-Migration و Update-Database را اجرا خواهیم نمود.
اما با توجه به اینکه مهاجرت خودکار را غیرفعال کرده‌ایم و هربار با فراخوانی دستور Add-Migration یک کلاس جدید، با متدهای Up و Down به پروژه، جهت نگهداری سوابق عملیات اضافه می‌شوند، می‌توان دستور Update-Database را جهت فراخوانی متد Down صرفا یک مرحله موجود نیز فراخوانی نمود.

نکته:

اگر علاقمند باشید که راهنمای مفصل پارامترهای دستور Update-Database را مشاهده کنید، تنها کافی است دستور زیر را در کنسول پاورشل اجرا نمائید:

```
get-help update-database -detailed
```

به عنوان نمونه اگر در حین فراخوانی دستور Update-Database احتمال از دست رفتن اطلاعات باشد، عملیات متوقف می‌شود. برای وادار کردن پروسه به انجام تغییرات بر روی بانک اطلاعاتی می‌توان از پارامتر Force در اینجا استفاده کرد.

در ادامه برای اینکه دستور Update-Database تنها یک مرحله مشخص را که سابقه آن در برنامه موجود است، هدف قرار دهد، باید از پارامتر TargetMigration به همراه نام کلاس مرتبط استفاده کرد:

```
Update-Database -TargetMigration:"AddSomeProp2ToProject" -Verbose
```

اگر دقت کرده باشید در اینجا AddSomeProp **2** ToProject بجای AddSomeProp **3** ToProject بکارگرفته شده است. اگر یک مرحله قبل را هدف قرار دهیم، متد Down را اجرا خواهد کرد:

```
Using NuGet project 'EF_Sample02'.
Using StartUp project 'EF_Sample02'.
Target database is: 'testdb2012' (DataSource: (local), Provider: System.Data.SqlClient, Origin:
Configuration).
Reverting migrations: [201205061845485_AddSomeProp3ToProject].
Reverting explicit migration: 201205061845485_AddSomeProp3ToProject.
DECLARE @var0 nvarchar(128)
SELECT @var0 = name
FROM sys.default_constraints
WHERE parent_object_id = object_id(N'Projects')
AND col_name(parent_object_id, parent_column_id) = 'SomeProp3';
IF @var0 IS NOT NULL
    EXECUTE('ALTER TABLE [Projects] DROP CONSTRAINT ' + @var0)
ALTER TABLE [Projects] DROP COLUMN [SomeProp3]
[Deleting migration history record]
```

همانطور که ملاحظه می‌کنید در اینجا عملیات حذف ستون SomeProp3 انجام شده است. البته این خاصیت به صورت خودکار از کدهای برنامه (کلاس Project در این مثال) حذف نمی‌شود و فرض بر این است که پیشتر اینکار را انجام داده‌اید.

سفارشی سازی کلاس‌های مهاجرت

تمام کلاس‌های خودکار مهاجرت تولید شده توسط پاورشل، از کلاس DbMigration ارث بری می‌کنند. در این کلاس امکانات قابل توجهی مانند AddColumn، AddForeignKey، AddPrimaryKey، AlterColumn، CreateIndex و امثال آن وجود دارند که در تمام کلاس‌های مشتق شده از آن، قابل استفاده هستند. حتی متد Sql نیز در آن پیش بینی شده است که در صورت نیاز به اجرای دستوارت خام SQL، می‌توان از آن استفاده کرد.
برای مثال فرض کنید مجددا همان خاصیت public string SomeProp3 را به کلاس Project اضافه کرده‌ایم. اما اینبار نیاز است حین تشکیل این فیلد در بانک اطلاعاتی، یک مقدار پیش فرض نیز برای آن درنظر گرفته شود که در صورت نال بودن مقدار خاصیت آن در برنامه، به صورت خودکار توسط بانک اطلاعاتی مقدار دهی گردد:

```
namespace EF_Sample02.Migrations
```

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

    public partial class AddSomeProp3ToProject : DbMigration
    {
        public override void Up()
        {
            AddColumn("Projects", "SomeProp3", c => c.String(defaultValue: "some data"));
            Sql("Update Projects set SomeProp3=N'some data'");
        }

        public override void Down()
        {
            DropColumn("Projects", "SomeProp3");
        }
    }
}
```

متد String در اینجا چنین امضایی دارد:

```
public ColumnModel String(bool? nullable = null, int? maxLength = null, bool? fixedLength = null,
bool? isMaxLength = null, bool? unicode = null, string defaultValue = null, string defaultValueSql =
null,
string name = null, string storeType = null)
```

که برای نمونه در اینجا پارامتر defaultValue آن‌را در کلاس AddSomeProp3ToProject مقدار دهی کرده‌ایم.
برای اعمال این تغییرات تنها کافی است دستور Update-Database -Verbose اجرا گردد. اینبار خروجی SQL اجرا شده آن به نحو زیر است که شامل مقدار پیش فرض نیز شده است:

```
ALTER TABLE [Projects] ADD [SomeProp3] [nvarchar](max) DEFAULT 'some data'
```

تعیین مقدار پیش فرض، زمانیکه یک فیلد not null تعریف شده‌است نیز می‌تواند مفید باشد. همچنین در اینجا امکان اجرای دستورات مستقیم SQL نیز وجود دارد که نمونه‌ای از آن‌را در متد Up فوق مشاهده می‌کنید.

افزودن رکوردهای پیش فرض در حین به روز رسانی بانک اطلاعاتی

در قسمت‌های قبل با متد Seed که به همراه آغاز کننده‌های بانک اطلاعاتی EF ارائه شده‌اند، جهت افزودن رکوردهای اولیه و پیش فرض به بانک اطلاعاتی آشنا شدید. در اینجا نیز با تحریف متد Seed در کلاس Configuration، چنین امری میسر است:

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

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

        protected override void Seed(EF_Sample02.Sample2Context context)
        {
            context.Users.AddOrUpdate(
                 a => a.Name,
```

```
                new Models.User { Name = "Vahid", AddDate = DateTime.Now },
                new Models.User { Name = "Test", AddDate = DateTime.Now });
        }
    }
}
```

متد AddOrUpdate در EF 4.3 اضافه شده است. این متد ابتدا بررسی می‌کند که آیا رکورد مورد نظر در بانک اطلاعاتی وجود دارد یا خیر. اگر خیر، آن‌را اضافه خواهد کرد در غیراینصورت، نمونه موجود را به روز رسانی می‌کند. اولین پارامتر آن، identifierExpression نام دارد. توسط آن مشخص می‌شود که بر اساس چه خاصیتی باید در مورد update یا add تصمیم‌گیری شود. دراینجا اگر نیاز به ذکر بیش از یک خاصیت وجود داشت، از anonymously type object می‌توان کمک گرفت new { p.Name, p.LastName } .

تولید اسکریپت به روز رسانی بانک اطلاعاتی

بهترین کار و امن‌ترین روش حین انجام این نوع به روز رسانی‌ها، تهیه اسکریپت SQL فرامینی است که باید بر روی بانک اطلاعاتی اجرا شوند. سپس می‌توان این دستورات و اسکریپت نهایی را دستی هم اجرا کرد (که روش متداول‌تری است در محیط کاری).
برای اینکار تنها کافی است دستور زیر را در کنسول پاورشل اجرا نمائیم:

```
Update-Database -Verbose -Script
```

پس از اجرای این دستور، یک فایل اسکریپت با پسوند sql تولید شده و بلافاصله در ویژوال استودیو جهت مرور نیز گشوده خواهد شد. برای نمونه محتوای آن برای افزودن خاصیت جدید SomeProp5 به صورت زیر است:

```
ALTER TABLE [Projects] ADD [SomeProp5] [nvarchar](max)
INSERT INTO [__MigrationHistory] ([MigrationId], [CreatedOn], [Model], [ProductVersion]) VALUES
('201205060852004_AutomaticMigration', '2012-05-06T08:52:00.937Z', 0x1F8B0800000..... '4.3.1')
```

همانطور که ملاحظه می‌کنید، در یک مرحله، جدول پروژه‌ها را به روز خواهد کرد و در مرحله بعد، سابقه آن‌را در جدول __MigrationHistory ثبت می‌کند.

یک نکته:
اگر دستور فوق را بر روی برنامه‌ای که با بانک اطلاعاتی هماهنگ است اجرا کنیم، خروجی را مشاهده نخواهیم کرد. برای این منظور می‌توان مرحله خاصی را توسط پارامتر SourceMigration هدف گیری کرد:

```
Update-Database -Verbose -Script -SourceMigration:"stepName"
```

استفاده از DB Migrations در عمل

البته این یک روش پیشنهادی و امن است:
الف) در ابتدای اجرا برنامه، پارامتر ورودی متد System.Data.Entity.Database.SetInitializer را به نال تنظیم کنید تا برنامه تغییری را بر روی بانک اطلاعاتی اعمال نکند.
ب) توسط دستور enable-migrations، فایل‌های اولیه DB Migration را ایجاد کنید. پیش فرض‌های آن را نیز تغییر ندهید.

ج) هر بار که کلاس‌های مدل برنامه تغییر کردند و پس از آن نیاز به به روز رسانی ساختار بانک اطلاعاتی وجود داشت دو دستور زیر را اجرا کنید:

```
Add-Migration AddSomePropToProject
Update-Database -Verbose -Script
```

به این ترتیب سابقه تغییرات در برنامه نگهداری شده و همچنین بدون اجرای دستورات بر روی بانک اطلاعاتی، اسکریپت نهایی اعمال تغییرات تولید می‌گردد.

د) اسکریپت تولید شده را بررسی کرده و پس از تائید و افزودن به سورس کنترل، به صورت دستی بر روی بانک اطلاعاتی اجرا کنید (مثلا توسط management studio).

نظرات خوانندگان

نویسنده: شهروز جعفری
تاریخ: ۱۳۹۱/۰۴/۰۲ ۱۷:۳۸

آیا در نسخه نهایی باید تنظیمات مربوط به Migrations رو حذف کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۰۲ ۱۷:۴۱

فقط قسمت Database.SetInitializer را نال کنید تا برنامه مستقیما کار به روز رسانی ساختار بانک اطلاعاتی را انجام ندهد.
بعد، از اسکریپت تولیدی مطابق روشی که در انتهای بحث توضیح دادم پس از بررسی‌های لازم استفاده کنید. بنابراین تنظیمی را لازم نیست حذف کنید. فقط باید با احتیاط جلو رفت و بررسی کامل اسکریپت تولیدی و سپس اجرای دستی آن روی بانک اطلاعاتی.

نویسنده: شهروز جعفری
تاریخ: ۱۳۹۱/۰۴/۰۷ ۲۳:۲۶

من همین کاری که گفتید کردم سایتو آپلود کردم
ولی این error میده
Exception Details: System.Data.SqlClient.SqlException: Cannot open database
."DataLayer.Context.MedicallexiconContext" requested by the login. The login failed
.'Login failed for user 'ServerName\medicallexicon_web
تو stackoverflow هم مطرح کرم جوای نگرفتم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۰۷ ۲۳:۳۴

چندتا بحث هست در مورد این خطا:
- EF Code first زمانیکه مشاهده کنه دیتابیس تعریف شده در رشته اتصالی وجود خارجی ندارد، سعی در ایجاد آن خواهد کرد. شما در هاست‌ها عموما دسترسی dbo ندارید. یعنی دسترسی ساخت دیتابیس ندارید. به عبارتی باید یک دیتابیس خالی از پیش تعیین شده داشته باشید. اگر پنل خاصی دارید از آن استفاده کنید برای ساخت دیتابیس. اگر ندارید باید تماس بگیرید تا دیتابیس برای شما ایجاد شود.
- زمانیکه خطای یافت نشدن بانک اطلاعاتی DataLayer.Context.MedicallexiconContex را دریافت می‌کنید یعنی پیش فرض‌های EF Code first رو رعایت نکردید. در اینجا name ایی که در رشته اتصالی تعریف می‌کنید مهم است. به صورت پیش فرض این name باید همان نام کلاس Context شما باشد (صرفنظر از اینکه رشته اتصالی شما به چه بانک اطلاعاتی اشاره می‌کند).

نویسنده: شهروز جعفری
تاریخ: ۱۳۹۱/۰۴/۰۷ ۲۳:۴۴

تو سرور خودم آپلود کردما.در ضمن میدونم اینجا مناسب سوال پرسیدن نیس شرمنده

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۰۷ ۲۳:۴۸

مهم این است که یوزری که قصد اتصال دارد چه دسترسی برای آن تعریف شده. آیا دسترسی ایجاد بانک اطلاعاتی را دارد.
همچنین بحث name رشته اتصالی هم هست.
این مباحث رو من در قسمت‌های قبل، لابلای مطالب و کامنت‌ها توضیح دادم. باید وقت بگذارید مطالعه کنید.

نویسنده: شهروز جعفری
تاریخ: ۱۳۹۱/۰۴/۰۸ ۰:۵

چیزی که برای من جالبه اینکه من تو کانکشن استرینگم User مشخص کردم ولی error که به من میده مربوطه به ServerName\medicallexicon_web میگه با این user نمیتونم login کنم اصلا نمیدون این user باسه چی هست؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۰۸ ۰:۱۶

نه. این اصطلاحا instance نام داره. در sql server شما چندین instance می‌تونید تعریف کنید. اگر نام سرور به تنهایی ذکر شود یعنی وهله پیش فرض. در غیر اینصورت به این معنا است که یک سری تنظیمات امنیتی خاص بر روی وهله‌ای خاص اعمال شده و شما باید از آن استفاده کنید. (البته می‌تونه در خطای ذکر شده نام یوزر هم باشه .... باید رشته اتصالی رو ببینم که از چه حالت اعتبار سنجی استفاده می‌کنه)
در کل این بحث در اینجا ادامه پیدا نکند بهتر است. چون نه مشخص است تنظیمات سرور شما چی هست. نه رشته اتصالی شما رو من دیدم. نه تنظیمات برنامه و Context شما مشخص است و نه تعداد وهله‌های سرور. نه سطح دسترسی یوزر اتصالی شما و نه نوع اعتبار سنجی لاگین یوزر متصل به سرور (ویندوزی است یا مخصوص sql server است).

نویسنده: ایلیا اکبری فرد
تاریخ: ۱۳۹۱/۱۰/۲۶ ۱۶:۴۳

آقای نصیری با سلام. من هر وقت برای بروزرسانی دیتابیس دستور Add-migration را در Package manager console وارد می‌کنیم ، همیشه در اسکریپت ایجاد شده من جدول reportparameter هم وجود دارد ، در حالیکه اصلاً تغییری در آن ایجاد نکردم.

```
namespace Dal.Ef.Migrations
{
using System;
using System.Data.Entity.Migrations;
public partial class AddActiveColumnToClassesTable : DbMigration
{
public override void Up()
{
CreateTable(
"dbo.ReportParameters",
c => new
{
Id = c.Int(nullable: false, identity: true),
CenterCode = c.String(maxLength: 10),
CenterTitle = c.String(maxLength: 100),
TermCode = c.String(maxLength: 10),
TermTitle = c.String(maxLength: 100),
MasulBarnamerizi = c.String(maxLength: 100),
ModirAmuzesh = c.String(maxLength: 100),
Term_Id = c.Int(nullable: false),
Center_Id = c.Int(nullable: false),
})
.PrimaryKey(t => t.Id)
.ForeignKey("dbo.Terms", t => t.Term_Id, cascadeDelete: true)
.ForeignKey("dbo.Centers", t => t.Center_Id, cascadeDelete: true)
.Index(t => t.Term_Id)
.Index(t => t.Center_Id);
AddColumn("dbo.Classes", "IsActive", c => c.Boolean(nullable: false));
}
public override void Down()
{
DropIndex("dbo.ReportParameters", new[] { "Center_Id" });
DropIndex("dbo.ReportParameters", new[] { "Term_Id" });
DropForeignKey("dbo.ReportParameters", "Center_Id", "dbo.Centers");
DropForeignKey("dbo.ReportParameters", "Term_Id", "dbo.Terms");
```

;("DropColumn("dbo.Classes", "IsActive

```
DropTable("dbo.ReportParameters");
        }
```

```
    }
}
```

در بالا فقط یک ستون به جدول classes ایجاد کردم ولی همیشه برای جدول reportParameter که در جدول نیز وجود دارد هم اسکریپت ایجاد میکند.
البته من یکبار بعد از آنکه جدول reportParameter را ایجاد کرده بودم ، دستی آن را از دیتابیس حذف کرده بودم.
متشکرم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۲۶ ۱۷:۲۳

یک روش کلی زمانیکه همه چیز را دستی به هم ریخته‌اید وجود دارد:
- جدول سیستمی MigrationHistory را از بانک اطلاعاتی حذف کنید.
- کلاس‌های قبلی migrations موجود را هم کلا حذف کنید (هرچیزی که وجود دارد).
حالا مراجعه کنید به قسمت چهارم «استفاده از Code first migrations بر روی یک بانک اطلاعاتی موجود» تا مجددا بتوانید جدول سیستمی MigrationHistory تازه‌ای را تولید کنید:

```
Add-Migration InitialMigration -IgnoreChanges
Update-Database
```

دستورات فوق به این معنا هستند که فرض می‌شود تطابق کاملی بین بانک اطلاعاتی و کلاس‌های مدل وجود دارند. اکنون جدول سیستمی MigrationHistory متناظری را تولید کن.

نویسنده: ایلیا اکبری فرد
تاریخ: ۱۳۹۱/۱۰/۲۶ ۱۸:۱۳

مهندس جان سلام.
مشکلم کاملاً برطرف شد.
بسیار ممنون از لطف شما. یاعلی.

نویسنده: سارا زرمهر
تاریخ: ۱۳۹۱/۱۱/۰۱ ۱۵:۳

سلام؛

با توجه به این قسمت مطالب **استفاده از DB Migrations در عمل**
من اگر پارامترSystem.Data.Entity.Database.SetInitializer رو null بدم با خطا مواجه میشم!

میشه یه نمونه کد برام مثال بزنید که به چه صورت باید این کد رو بنویسم؟
ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۰۱ ۱۶:۲۶

چه خطایی دریافت می‌کنید؟
ضمنا تمام مثال‌های این سری را از اینجا می‌توانید دریافت کنید.

نویسنده: سارا زرمهر
تاریخ: ۱۳۹۱/۱۱/۰۲ ۱۱:۱۴

فکر کنم من صحیح کدمو ننو شتم.

این لینکی که دادید من چطوری کدهاشو دریافت کنم؟
کلیک که میکنم، کدشده به من نشون داده میشه.

نویسنده: بهروز راد
تاریخ: ۱۳۹۱/۱۱/۰۲ ۱۱:۲۴

کلیک راست کن، گزینه Save Link As رو انتخاب کن (Firefox) یا لینکش رو توی IDM کپی و دانلود کن.

نویسنده: سارا زرمهر
تاریخ: ۱۳۹۱/۱۱/۰۲ ۱۱:۳۸

تشکر استاد راد

نویسنده: میثم خوشقدم
تاریخ: ۱۳۹۲/۰۴/۰۱ ۱۸:۳۳

سلام

من هم مشکل ایشون رو دارم

ولی با کارهایی که شما فرمودین مشکل حل نشد

چندین بار جدول و کلاس‌های میگریشن رو حذف کردم ولی گاهی درست کار می‌کند و بعضی وقت‌ها هم خیر.

مشکل اینجاست که در Add_Migration متدهای Up و Down خالی است یعنی تغییری را تشخیص نداده اما در Update-database می‌خواهد مجددا جدول را ایجاد کند!

نویسنده: سید مهدی فاطمی
تاریخ: ۱۳۹۲/۰۵/۱۱ ۰:۲۲

ضمن تشکر از مطالبتون
آیا میشه این مراحلی رو که گفتید رو خود برنامه انجام بده و کاربر نهایی از اون اطلاعی نداشته باشه ؟
مثلا من یه برنامه دادم تحویل کاربر و دیتابیس اون هم حاوی اطلاعات هست حالا من به این نتیجه رسیدم که تغییری در دیتابیس بدم. طبق گفته شما من باید یه اسکریپت از تغییرات خودم درست کنم و تحویل کاربر بدم تا اونو ایجاد کنه اما من دنبال راهی می‌گردم که در برنامه وقتی کلاس‌ها رو تغییر دادم برنامه‌ی جدید رو تحویل کاربر بدم و کاربر بیاد برنامه قدیمشو حذف و برنامه‌ی جدید و نصب کنه و وقتی که برنامه‌ی جدید برای اولین بار اجرا شد تغییرات رو در دیتابیس قدیمی اعمال کنه بدون اینکه کاربر از پشت پرده اطلاعی داشته باشه

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۵/۱۱ ۰:۵۳

- بله. مراجعه کنید به قسمت چهارم مباحث «فعال سازی گزینه‌های مهاجرت خودکار، آزمودن ویژگی مهاجرت خودکار و عکس العمل ویژگی مهاجرت خودکار در مقابل از دست رفتن اطلاعات» آن.
- امکان ردیابی تغییرات آن (منظور از ردیابی در اینجا، ثبت وقایع و ذخیره سازی خروجی SQL به روز رسانی ساختار بانک اطلاعاتی است) هم در همان قسمت چهارم ذیل مبحث «ساده سازی پروسه مهاجرت خودکار» مطرح شده.

نویسنده: حامد رشنو
تاریخ: ۱۳۹۳/۰۹/۰۳ ۱۶:۲۱

من از چند کانتکست استفاده میکنم و موقع استفاده از دستورات:

```
Add-Migration
```

```
Update-Database
```

به مشکل بر میخورم.
آیا برای استفاده از چند کانتکست باید برای هر کدوم یک DbMigration جداگانه ساخت؟ با قرار دادن T به جای اسم کانتکست در DbMigration هم مشکلم حل نشد

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۰۳ ۱۸:۳۱

« استفاده از چندین Context در EF 6 Code first »