عنوان: **EF Code First #2**
نویسنده: وحید نصیری
تاریخ: ۰۹:۳۶:۰۰ ۱۳۹۱/۰۲/۱۵
آدرس: www.dotnettips.info
گروه‌ها: Entity framework

در قسمت قبل با تنظیمات و قراردادهای ابتدایی EF Code first آشنا شدیم، هرچند این تنظیمات حجم کدنویسی ابتدایی راه اندازی سیستم را به شدت کاهش می‌دهند، اما کافی نیستند. در این قسمت نگاهی سطحی و مقدماتی خواهیم داشت بر امکانات مهیا جهت تنظیم ویژگی‌های مدل‌های برنامه در EF Code first.

تنظیمات EF Code first توسط اعمال متادیتای خواص

اغلب متادیتای مورد نیاز جهت اعمال تنظیمات EF Code first در اسمبلی System.ComponentModel.DataAnnotations.dll قرار دارند. بنابراین اگر مدل‌های خود را در اسمبلی و پروژه class library جداگانه‌ای تعریف و نگهداری می‌کنید (مثلا به نام DomainClasses)، نیاز است ابتدا ارجاعی را به این اسمبلی به پروژه جاری اضافه نمائیم. همچنین تعدادی دیگر از متادیتای قابل استفاده در خود اسمبلی EntityFramework.dll قرار دارند. بنابراین در صورت نیاز باید ارجاعی را به این اسمبلی نیز اضافه نمود.
همان مثال قبل را در اینجا ادامه می‌دهیم. دو کلاس Blog و Post در آن تعریف شده (به این نوع کلاس‌ها POCO – the Plain Old CLR Objects نیز گفته می‌شود)، به همراه کلاس Context که از کلاس DbContext مشتق شده است. ابتدا دیتابیس قبلی را دستی drop کنید. سپس در کلاس Blog، خاصیت public int Id را مثلا به public int MyTableKey تغییر دهید و پروژه را اجرا کنید.
برنامه بلافاصله با خطای زیر متوقف می‌شود:

```
One or more validation errors were detected during model generation:
\tSystem.Data.Entity.Edm.EdmEntityType: : EntityType 'Blog' has no key defined.
```

زیرا EF Code first در این کلاس خاصیتی به نام Id یا BlogId را نیافته‌است و امکان تشکیل Primary key جدول را ندارد. برای رفع این مشکل تنها کافی است ویژگی Key را به این خاصیت اعمال کنیم:

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace EF_Sample01.Models
{
    public class Blog
    {
        [Key]
        public int MyTableKey { set; get; }
```

همچنین تعدادی ویژگی دیگر مانند MaxLength و Required را نیز می‌توان بر روی خواص کلاس اعمال کرد:

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace EF_Sample01.Models
{
    public class Blog
    {
        [Key]
        public int MyTableKey { set; get; }

        [MaxLength(100)]
```

```
        public string Title { set; get; }

        [Required]
        public string AuthorName { set; get; }

        public IList<Post> Posts { set; get; }
    }
}
```

این ویژگی‌ها دو مقصود مهم را برآورده می‌سازند:
الف) بر روی ساختار بانک اطلاعاتی تشکیل شده تاثیر دارند:

```
CREATE TABLE [dbo].[Blogs](
 [MyTableKey] [int] IDENTITY(1,1) NOT NULL,
 [Title] [nvarchar](100) NULL,
 [AuthorName] [nvarchar](max) NOT NULL,
 CONSTRAINT [PK_Blogs] PRIMARY KEY CLUSTERED
(
 [MyTableKey] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
```

همانطور که ملاحظه می‌کنید در اینجا طول فیلد Title به 100 تنظیم شده است و همچنین فیلد AuthorName اینبار NOT NULL است. به علاوه primary key نیز بر اساس ویژگی Key اعمالی تعیین شده است.
البته برای اجرای کدهای تغییر کرده مدل، فعلا بانک اطلاعاتی قبلی را دستی می‌توان حذف کرد تا بتوان به ساختار جدید رسید. در مورد جزئیات مبحث DB Migration در قسمت‌های بعدی مفصلا بحث خواهد شد.

ب) اعتبار سنجی اطلاعات پیش از ارسال کوئری به بانک اطلاعاتی
برای مثال اگر در حین تعریف وهله‌ای از کلاس Blog، خاصیت AuthorName مقدار دهی نگردد، پیش از اینکه رفت و برگشتی به بانک اطلاعاتی صورت گیرد، یک validation error را دریافت خواهیم کرد. یا برای مثال اگر طول اطلاعات خاصیت Title بیش از 100 حرف باشد نیز مجددا در حین ثبت اطلاعات، یک استثنای اعتبار سنجی را مشاهده خواهیم کرد. البته امکان تعریف پیغام‌های خطای سفارشی نیز وجود دارد. برای این حالت تنها کافی است پارامتر ErrorMessage این ویژگی‌ها را مقدار دهی کرد. برای مثال:

```
[Required(ErrorMessage = "لطفا نام نویسنده را مشخص نمائید")]
public string AuthorName { set; get; }
```

نکته‌ی مهمی که در اینجا وجود دارد، وجود یک اکوسیستم هماهنگ و سازگار است. این نوع اعتبار سنجی هم با EF Code first هماهنگ است و هم برای مثال در ASP.NET MVC به صورت خودکار جهت اعتبار سنجی سمت سرور و کلاینت یک مدل می‌تواند مورد استفاده قرار گیرد و مفاهیم و روش‌های مورد استفاده در آن نیز یکی است.

تنظیمات EF Code first به کمک Fluent API

اگر علاقمند به استفاده از متادیتا، جهت تعریف قیود و ویژگی‌های خواص کلاس‌های مدل خود نیستید، روش دیگری نیز در EF Code first به نام Fluent API تدارک دیده شده است. در اینجا امکان تعریف همان ویژگی‌ها توسط کدنویسی نیز وجود دارد، به علاوه اعمال قیود دیگری که توسط متادیتای مهیا قابل تعریف نیستند.
محل تعریف این قیود، کلاس Context که از کلاس DbContext مشتق شده است، می‌باشد و در اینجا، کار با تحریف متد OnModelCreating شروع می‌شود:

```
using System.Data.Entity;
using EF_Sample01.Models;

namespace EF_Sample01
{
    public class Context : DbContext
    {
        public DbSet<Blog> Blogs { set; get; }
        public DbSet<Post> Posts { set; get; }

        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Blog>().HasKey(x => x.MyTableKey);
            modelBuilder.Entity<Blog>().Property(x => x.Title).HasMaxLength(100);
            modelBuilder.Entity<Blog>().Property(x => x.AuthorName).IsRequired();

            base.OnModelCreating(modelBuilder);
        }
    }
}
```

به کمک پارامتر modelBuilder، امکان دسترسی به متدهای تنظیم کننده ویژگی‌های خواص یک مدل یا موجودیت وجود دارد. در اینجا چون می‌توان متدها را به صورت یک زنجیره به هم متصل کرد و همچنین حاصل نهایی شبیه به جمله بندی انگلیسی است، به آن Fluent API یا API روان نیز گفته می‌شود.
البته در این حالت امکان تعریف ErrorMessage وجود ندارد و برای این منظور باید از همان data annotations استفاده کرد.

نحوه مدیریت صحیح تعاریف نگاشت‌ها به کمک Fluent API

OnModelCreating محل مناسبی جهت تعریف حجم انبوهی از تنظیمات کلاس‌های مختلف مدل‌های برنامه نیست. در حد سه چهار سطر مشکلی ندارد اما اگر بیشتر شد بهتر است از روش زیر استفاده شود:

```
using System.Data.Entity;
using EF_Sample01.Models;
using System.Data.Entity.ModelConfiguration;

namespace EF_Sample01
{
    public class BlogConfig : EntityTypeConfiguration<Blog>
    {
        public BlogConfig()
        {
            this.Property(x => x.Id).HasColumnName("MyTableKey");
            this.Property(x => x.RowVersion).HasColumnType("Timestamp");
        }
    }
```

با ارث بری از کلاس EntityTypeConfiguration، می‌توان به ازای هر کلاس مدل، تنظیمات را جداگانه انجام داد. به این ترتیب اصل SRP یا Single responsibility principle نقض نخواهد شد. سپس برای استفاده از این کلاس‌های Config تک مسئولیتی به نحو زیر می‌توان اقدام کرد:

```
protected override void OnModelCreating(DbModelBuilder modelBuilder)
{
     modelBuilder.Configurations.Add(new BlogConfig());
```

نحوه تنظیمات ابتدایی نگاشت کلاس‌ها به بانک اطلاعاتی در EF Code first

الزامی ندارد که EF Code first حتما با یک بانک اطلاعاتی از نو تهیه شده بر اساس پیش فرض‌های آن کار کند. در اینجا می‌توان از بانک‌های اطلاعاتی موجود نیز استفاده کرد. اما در این حالت نیاز خواهد بود تا مثلا نام جدولی خاص با کلاسی مفروض در برنامه، یا نام فیلدی خاص که مطابق استانداردهای نامگذاری خواص در سی شارپ تعریف نشده، با خاصیتی در یک کلاس تطابق داده شوند. برای مثال اینبار تعاریف کلاس Blog را به نحو زیر تغییر دهید:

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace EF_Sample01.Models
{
    [Table("tblBlogs")]
    public class Blog
    {
        [Column("MyTableKey")]
        public int Id { set; get; }

        [MaxLength(100)]
        public string Title { set; get; }

        [Required(ErrorMessage = "لطفا نام نویسنده را مشخص نمائید")]
        public string AuthorName { set; get; }

        public IList<Post> Posts { set; get; }

        [Timestamp]
        public byte[] RowVersion { set; get; }
    }
}
```

در اینجا فرض بر این است که نام جدول متناظر با کلاس Blog در بانک اطلاعاتی مثلا tblBlogs است و نام خاصیت Id در بانک اطلاعاتی مساوی فیلدی است به نام MyTableKey. چون نام خاصیت را مجددا به Id تغییر داده‌ایم، دیگر ضرورتی به ذکر ویژگی Key وجود نداشته است. برای تعریف این دو از ویژگی‌های Table و Column جهت سفارشی سازی نام‌های خواص و کلاس استفاده شده است.

یا اگر در کلاس خود خاصیتی محاسبه شده بر اساس سایر خواص، تعریف شده است و قصد نداریم آن‌را به فیلدی در بانک اطلاعاتی نگاشت کنیم، می‌توان از ویژگی NotMapped برای مزین سازی و تعریف آن کمک گرفت.

به علاوه اگر از نام پیش فرض کلید خارجی تشکیل شده خرسند نیستید می‌توان به کمک ویژگی ForeignKey، نسبت به تعریف مقداری جدید مطابق تعاریف یک بانک اطلاعاتی موجود، اقدام کرد.

همچنین خاصیت دیگری به نام RowVersion در اینجا اضافه شده که با ویژگی TimeStamp مزین گردیده است. از این خاصیت ویژه برای بررسی مسایل همزمانی ثبت اطلاعات در EF استفاده می‌شود. به علاوه بانک اطلاعاتی می‌تواند به صورت خودکار آن‌را در حین ثبت مقدار دهی کند.

تمام این تغییرات را به کمک Fluent API نیز می‌توان انجام داد:

```
modelBuilder.Entity<Blog>().ToTable("tblBlogs");
modelBuilder.Entity<Blog>().Property(x => x.Id).HasColumnName("MyTableKey");
modelBuilder.Entity<Blog>().Property(x => x.RowVersion).HasColumnType("Timestamp");
```

تبدیل پروژه‌های قدیمی EF به کلاس‌های EF Code first به صورت خودکار

روش متداول کار با EF از روز اول آن، مهندسی معکوس خودکار اطلاعات یک بانک اطلاعاتی و تبدیل آن به یک فایل EDMX بوده است. هنوز هم می‌توان از این روش در اینجا نیز بهره جست. برای مثال اگر قصد دارید یک پروژه قدیمی را تبدیل به نمونه جدید

Code first کنید، یا یک بانک اطلاعاتی موجود را مهندسی معکوس کنید، بر روی پروژه در Solution explorer کلیک راست کرده و گزینه Add|New Item را انتخاب کنید. سپس از صفحه ظاهر شده، ADO.NET Entity data model را انتخاب کرده و در ادامه گزینه «Generate from database» را انتخاب کنید. این روال مرسوم کار با EF Database first است.
پس از اتمام کار به entity data model designer مراجعه کرده و بر روی صفحه کلیک راست نمائید. از منوی ظاهر شده گزینه «Add code generation item» را انتخاب کنید. سپس در صفحه باز شده از لیست قالب‌های موجود، گزینه «ADO.NET DbContext Generator» را انتخاب نمائید. این گزینه به صورت خودکار اطلاعات فایل EDMX قدیمی یا موجود شما را تبدیل به کلاس‌های مدل Code first معادل به همراه کلاس DbContext معرف آن‌ها خواهد کرد.

روش دیگری نیز برای انجام اینکار وجود دارد. نیاز است افزونه‌ی به نام Entity Framework Power Tools را دریافت کنید. پس از نصب، از منوی Entity Framework آن گزینه‌ی «Reverse Engineer Code First» را انتخاب نمائید. در اینجا می‌توان مشخصات اتصال به بانک اطلاعاتی را تعریف و سپس نسبت به تولید خودکار کدهای مدل‌ها و DbContext مرتبط اقدام کرد.

استراتژی‌های مقدماتی تشکیل بانک اطلاعاتی در EF Code first

اگر مثال این سری را دنبال کرده باشید، مشاهده کرده‌اید که با اولین بار اجرای برنامه، یک بانک اطلاعاتی پیش فرض نیز تولید خواهد شد. یا اگر تعاریف ویژگی‌های یک فیلد را تغییر دادیم، نیاز است تا بانک اطلاعاتی را دستی drop کرده و اجازه دهیم تا بانک اطلاعاتی جدیدی بر اساس تعاریف جدید مدل‌ها تشکیل شود که ... هیچکدام از این‌ها بهینه نیستند.
در اینجا دو استراتژی مقدماتی را در حین آغاز یک برنامه می‌توان تعریف کرد:

```
System.Data.Entity.Database.SetInitializer(new DropCreateDatabaseIfModelChanges<Context>());
// or
System.Data.Entity.Database.SetInitializer(new DropCreateDatabaseAlways<Context>());
```

می‌توان بانک اطلاعاتی را در صورت تغییر اطلاعات یک مدل به صورت خودکار drop کرده و نسبت به ایجاد نمونه‌ای جدید اقدام کرد (DropCreateDatabaseIfModelChanges)؛ یا در حین آزمایش برنامه همیشه (DropCreateDatabaseAlways) با شروع برنامه، ابتدا باید بانک اطلاعاتی drop شده و سپس نمونه جدیدی تولید گردد.
محل فراخوانی این دستور هم باید در نقطه آغازین برنامه، پیش از وهله سازی اولین DbContext باشد. مثلا در برنامه‌های وب در متد Application_Start فایل global.asax.cs یا در برنامه‌های WPF در متد سازنده کلاس App می‌توان بانک اطلاعاتی را آغاز نمود.
البته الزامی به استفاده از کلاس‌های DropCreateDatabaseIfModelChanges یا DropCreateDatabaseAlways وجود ندارد.
می‌توان با پیاده سازی اینترفیس IDatabaseInitializer از نوع کلاس Context تعریف شده در برنامه، همان عملیات را شبیه سازی کرد یا سفارشی نمود:

```
public class MyInitializer : IDatabaseInitializer<Context>
{
        public void InitializeDatabase(Context context)
        {
            if (context.Database.Exists() ||
                context.Database.CompatibleWithModel(throwIfNoMetadata: false))
                context.Database.Delete();

            context.Database.Create();
        }
}
```

سپس برای استفاده از این کلاس در ابتدای برنامه، خواهیم داشت:

```
System.Data.Entity.Database.SetInitializer(new MyInitializer());
```

نکته:
اگر از یک بانک اطلاعاتی موجود استفاده می‌کنید (محیط کاری) و نیازی به پیش فرض‌های EF Code first ندارید و همچنین این بانک اطلاعاتی نیز نباید drop شود یا تغییر کند، می‌توانید تمام این پیش فرض‌ها را با دستور زیر غیرفعال کنید:

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

بدیهی است این دستور نیز باید پیش از ایجاد اولین وهله از شیء DbContext فراخوانی شود.

همچنین باید درنظر داشت که در آخرین نگارش‌های پایدار EF Code first، این موارد بهبود یافته‌اند و مبحثی تحت عنوان DB Migration ایجاد شده است تا نیازی نباشد هربار بانک اطلاعاتی drop شود و تمام اطلاعات از دست برود. می‌توان صرفا تغییرات کلاس‌ها را به بانک اطلاعاتی اعمال کرد که به صورت جداگانه، در قسمتی مجزا بررسی خواهد شد. به این ترتیب دیگر نیازی به drop بانک اطلاعاتی نخواهد بود. به صورت پیش فرض در صورت از دست رفتن اطلاعات یک استثناء را سبب خواهد شد (که توسط برنامه نویس قابل تنظیم است) و در حالت خودکار یا دستی با تنظیمات ویژه قابل اعمال است.

تنظیم استراتژی‌های آغاز بانک اطلاعاتی در فایل کانفیگ برنامه

الزامی ندارد که حتما متد Database.SetInitializer را دستی فراخوانی کنیم. با اندکی تنظیم فایل‌های app.config و یا web.config نیز می‌توان نوع استراتژی مورد استفاده را تعیین کرد:

```
<appSettings>
 <add key="DatabaseInitializerForType MyNamespace.MyDbContextClass, MyAssembly"
     value="MyNamespace.MyInitializerClass, MyAssembly"  />
</appSettings>

<appSettings>
 <add key="DatabaseInitializerForType MyNamespace.MyDbContextClass, MyAssembly"
     value="Disabled" />
</appSettings>
```

یکی از دو حالت فوق باید در قسمت appSettings فایل کانفیگ برنامه تنظیم شود. حالت دوم برای غیرفعال کردن پروسه آغاز بانک اطلاعاتی و اعمال تغییرات به آن، بکار می‌رود.
برای نمونه در مثال جاری، جهت استفاده از کلاس MyInitializer فوق، می‌توان از تنظیم زیر نیز استفاده کرد:

```
<appSettings>
    <add key="DatabaseInitializerForType EF_Sample01.Context, EF_Sample01"
       value="EF_Sample01.MyInitializer, EF_Sample01" />
  </appSettings>
```

اجرای کدهای ویژه در حین تشکیل یک بانک اطلاعاتی جدید

امکان سفارشی سازی این آغاز کننده‌های پیش فرض نیز وجود دارد. برای مثال:

```
public class MyCustomInitializer : DropCreateDatabaseIfModelChanges<Context>
{
    protected override void Seed(Context context)
    {
        context.Blogs.Add(new Blog { AuthorName = "Vahid", Title = ".NET Tips" });
        context.Database.ExecuteSqlCommand("CREATE INDEX IX_title ON tblBlogs (title)");
        base.Seed(context);
    }
}
```

در اینجا با ارث بری از کلاس DropCreateDatabaseIfModelChanges یک آغاز کننده سفارشی را تعریف کرده‌ایم. سپس با تحریف متد Seed آن می‌توان در حین آغاز یک بانک اطلاعاتی، تعدادی رکورد پیش فرض را به آن افزود. کار ذخیره سازی نهایی در متد base.Seed انجام می‌شود.

برای استفاده از آن اینبار در حین فراخوانی متد System.Data.Entity.Database.SetInitializer، از کلاس MyCustomInitializer استفاده خواهیم کرد.

و یا توسط متد context.Database.ExecuteSqlCommand می‌توان دستورات SQL را مستقیما در اینجا اجرا کرد. عموما دستوراتی در اینجا مدنظر هستند که توسط ORMها پشتیبانی نمی‌شوند. برای مثال تغییر collation یک ستون یا افزودن یک ایندکس و مواردی از این دست.

سطح دسترسی مورد نیاز جهت فراخوانی متد Database.SetInitializer

استفاده از متدهای آغاز کننده بانک اطلاعاتی نیاز به سطح دسترسی بر روی بانک اطلاعاتی master را در SQL Server دارند (زیرا با انجام کوئری بر روی این بانک اطلاعاتی مشخص می‌شود، آیا بانک اطلاعاتی مورد نظر پیشتر تعریف شده است یا خیر). البته این مورد حین کار با SQL Server CE شاید اهمیتی نداشته باشد. بنابراین اگر کاربری که با آن به بانک اطلاعاتی متصل می‌شویم سطح دسترسی پایینی دارد نیاز است Persist Security Info=True را به رشته اتصالی اضافه کرد. البته این مورد را پس از انجام تغییرات بر روی بانک اطلاعاتی جهت امنیت بیشتر حذف کنید (یا به عبارتی در محیط کاری Persist Security Info=False باید باشد).

```
Server=(local);Database=yourDatabase;User
ID=yourDBUser;Password=yourDBPassword;Trusted_Connection=False;Persist Security Info=True
```

تعیین Schema و کاربر فراخوان دستورات SQL

در EF Code first به صورت پیش فرض همه چیز بر مبنای کاربری با دسترسی مدیریتی یا dbo schema در اس کیوال سرور تنظیم شده است. اما اگر کاربر خاصی برای کار با دیتابیس تعریف گردد که در هاست‌های اشتراکی بسیار مرسوم است، دیگر از دسترسی مدیریتی dbo خبری نخواهد بود. اینبار نام جداول ما بجای dbo.tableName مثلا someUser.tableName می‌باشند و عدم دقت به این نکته، اجرای برنامه را غیرممکن می‌سازد.

برای تغییر و تعیین صریح کاربر متصل شده به بانک اطلاعاتی اگر از متادیتا استفاده می‌کنید، روش زیر باید بکارگرفته شود:

```
[Table("tblBlogs", Schema="someUser")]
public class Blog
```

و یا در حالت بکارگیری Fluent API به نحو زیر قابل تنظیم است:

```
modelBuilder.Entity<Blog>().ToTable("tblBlogs", schemaName:"someUser");
```

نظرات خوانندگان

نویسنده: Mohammad
تاریخ: ۱۳۹۱/۰۲/۱۵ ۱۰:۲۰:۲۴

سلام آقای نصیری.
code first اجازه درست کردن trigger رو می‌ده؟
کلا من یه مکانیزمی می‌خوام که اگه کسی (غیر خودم) تو یه جدول خاص insert کرد با یه چیزی مثل event تو برنامه متوجه بشم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۱۵ ۱۰:۳۹:۱۵

لطف کنید سؤالی رو که مطرح می‌کنید در حیطه مطلب جاری عنوان شده باشد و خارج از آن نباشد.
سؤال شما هم بحث کلاینت سروری است و نه بحث کلاینت تنها که EF روی آن مشغول به کار است.
- می‌شود در متد Seed ایی که در بالا توضیح دادم در SQL Server تریگر درست کرد. (که مثلا اگر کاربر دیگری به شرط اینکه این کاربر جزو کاربران تعریف شده در خود SQL Server باشد نه در برنامه شما، اتفاق خاصی رخ دهد. برنامه شما هم بدیهی است باید سرور را مدام چک کند تا از این مساله مطلع شود)- SQL Server مبحثی دارد به نام Service Broker : (^) . توسط آن می‌توان از طریق سرور به کلاینت اطلاع رسانی کرد. بازهم خارج است از بحث یک ORM. یا تمام ORMهای موجود. - EF مبحثی دارد به نام Concurrency check که اگر شخصی در شبکه بر روی رکوردی که همین الان شما مشغول به کار هستید، تغییری را ایجاد کرد، به شما اطلاع رسانی کند. (در قسمت‌های بعدی بحث خواهد شد). البته این هم خودکار نیست. لازم است یک رفت و برگشت به سرور انجام شود.- entity framework auditing هم میسر است. خودکار نیست. در همان کلاس Context فوق که از DbContext مشتق می‌شود می‌توان متد تحریف شده public override int SaveChanges را تعریف کرد. در اینجا می‌توان به تمام تغییراتی که قرار است اعمال شوند دسترسی داشت. مثلا آن‌ها را در یک جدول مجزا ثبت کرد. بدیهی است برنامه بعدا نیاز خواهد داشت از این جدول گزارشگیری کند.

نویسنده: مهمان
تاریخ: ۱۳۹۱/۰۲/۱۵ ۱۸:۰۳:۵۰

با سلام و تشکر

سوال اول:
این قسمت آخری را که فرمودید:
"در EF Code first به صورت پیش فرض همه چیز بر مبنای کاربری با دسترسی مدیریتی یا dbo schema در اس کیوال سرور تنظیم شده است. اما اگر کاربر خاصی برای کار با دیتابیس تعریف گردد که در هاست‌های اشتراکی بسیار مرسوم است، دیگر از دسترسی مدیریتی dbo خبری نخواهد بود."

من متوجه نشدم! ما در هاستهای اشتراکی مثلا از طریق پنل پلسک یک بانک به همراه یک کاربر به نام فرضی user1 ایجاد می کنم و در کانشکن استرینگ هم با همین نام کاربری متصل می شویم. حال منظور شما از کاربر خاص یعنی چه کسی؟
این scheme که نام آنرا someUser گذاشتید، مربوط به چه کسی است و از کجا آمده است؟

سوال دوم:
آیا در مورد بانک Membership پیش فرض مایکروسافت و تلفیق آن با بانک اصلی برنامه در EF راه کاری اندیشیده شده؟
بنده هیچ وقت از این امکان به جهت دو دسته شدن جداول و ساختار بانکم استفاده نکردم ولی با توجه به یکپارچه شدن آن با ASP.NET MVC کم کم دارم متقاعد می شوم که به جای منطق Membership خودم از این امکان استفاده کنم، نظر شما در مورد منطق Membership برنامه ای که با EF و MVC نوشته می شود چیست؟

نویسنده: مهمان
تاریخ: ۱۳۹۱/۰۲/۱۵ ۱۸:۱۶:۵۱

آیا EF امکاناتی دارد که به کمک آن بتوان پس از ران شدن برنامه و ساخت بانک به صورت پویا به آن جدول یا فیلد و چیزهایی مثل Data Annotation اضافه کرد و بعد از هم از POCO آنها در حالت Runtime استفاده کرد؟
ترکیب EF Code First با پروژه Roslyn برای رسیدن به این منظور مناسب است؟ (گرچه فکر کنم راسلین هنوز کلاس های پیچیده زبان را پشتیبانی نمی کند)

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۱۵ ۱۸:۴۹:۱۸

بله؛ این امکان وجود دارد که سیستم را افزونه پذیر کرد و مدل‌ها رو در زمان اجرا به DbContext اضافه کرد.
جزئیات آن خارج است از بحث «قسمت دوم» ما.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۱۵ ۱۸:۵۵:۵۶

- این‌ها مباحث پایه‌ای SQL Server است. در مورد schema ایی که از آن صحبت شد می‌تونید اینجا بیشتر مطالعه کنید: (^)
- شما در ASP.NET MVC می‌تونید این موارد رو سفارشی کنید و از پیاده سازی خودتون استفاده کنید. در یک قسمت مجزا به این مورد پرداختم که در سایت هست (^) . به علاوه پروژه‌هایی مانند این هم وجود دارند: (^)

نویسنده: مهمان
تاریخ: ۱۳۹۱/۰۲/۱۵ ۱۹:۰۷:۰۷

سوال بنده این بود که این someUser که شما به عنوان schema تعریف کرده اید از کجا آمده است؟ متعلق به کدام کاربر است؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۱۵ ۱۹:۲۶:۰۷

در SQL Server برای کار با بانک اطلاعاتی یک سری سطوح امنیتی وجود دارد. عنوان کردید که من یک نام کاربری دارم و پسود و از آن در رشته اتصالی استفاده می‌کنیم. بله. این درسته. اما این فقط ابتدای کار است. زمانیکه کاربری در SQL Server تعریف می‌شود یک سری سطح دسترسی را می‌شود به آن داد یا از آن گرفت. مثلا دسترسی اجرای SPها را نداشته باشد؛ دسترسی drop بانک اطلاعاتی را نداشته باشد؛ یا امکان فراخوانی delete را نداشته باشد. برای اینکه این موارد را بهتر مدیریت کنند یک schema تعریف می‌شود که در حقیقت قالبی است جهت مشخص سازی این سطوح دسترسی. dbo یکی از این قالب‌ها است که جزو مجموعه بالاترین سطوح دسترسی است. در هاست‌های اشتراکی که به مسایل امنیتی اهمیت می‌دهند امکان نداره به شما دسترسی dbo بدن. بله شما نام کاربری و کلمه عبور دارید اما schema سفارشی شما ممکن است دسترسی drop یا create بانک اطلاعاتی رو نداشته باشه (که در هاست‌های خوب ندارید).
این مساله چه مشکلی رو ایجاد می‌کنه؟
اگر کوئری پیش فرض شما select * from dbo.table1 باشد، در یک هاست اشتراکی با سطح دسترسی بالا که به شما دسترسی drop و create یک بانک اطلاعاتی رو نداده، دیگر اجرا نخواهد شد چون dbo نیستید. ضمنا روش صحیح و توصیه شده کار با SQL Server نیز ذکر schema در کوئری‌ها است زیرا سرعت اجرا را بالا می‌برد از این لحاظ که اگر آن‌را ذکر نکنید، SQL Server مجبور خواهد شد دست به سعی و خطا بزند. اما با ذکر آن یک راست از سطح دسترسی صحیح استفاده می‌شود. EF هم از همین روش استفاده می‌کنه. بنابراین لازم است schema رو اینجا در صورت مساوی نبودن با dbo حتما ذکر کرد و گرنه کوئری‌های شما دیگر اجرا نخواهند شد، چون دسترسی dbo رو ندارید.

نویسنده: مرادی
تاریخ: ۱۳۹۱/۰۲/۱۵ ۲۳:۳۹:۲۰

امکانش هست به خاطر عوض کردن اسم یه Property، کل دیتابیس رو Drop نکنه ؟!

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۱۵ ۲۳:۴۳:۵۷

- می‌تونید SetInitializer رو به null مقدار دهی کنید تا کاری به بانک اطلاعاتی نداشته باشد.
و در حالت کلی، بله. مبحثی هست که اخیرا اضافه شده به نام DB Migration که در چند قسمت بعد به آن خواهیم رسید.

نویسنده: Navid
تاریخ: ۱۳۹۱/۰۲/۱۶ ۱۳:۳۹:۰۰

با سلام خدمت جناب نصیری
بنده از روی مثاله شما جلو رفتم و الگوی خودم رو در آوردم به این ترتیب که :
یک کلاس پایه ابسترکت درست کردم که یکسری پراپرتی در آن تعریف نمودم(فرضا کلاس جانداران)
چند کلاس دیگر ایجاد کردم که از کلاس پایه ارث برده بودند و پراپرتی های خود را هم داشتند
زمانیکه برنامه را اجرا کردم و دیتابیس از روی برنامه ایجاد شد ، تنها به ازای کلاس های فرزند جدول درست شده بود ولی تمامی پراپرتی های کلاس پایه در جداول ایجاد شده بود اما جدولی به اسم جاندار (کلاس پایه) ایجاد نشده بود .
سواله بنده اینه که آیا بایستی کاره خاصی کرد که یک جدول جداگانه برای کلاس پایه هم ایجاد شود ؟
من این کار رو توی مدل فرست بدون مشکل انجام میدادم و نتیجه کار هم خوب بود .
با تشکر از زحمات جنابعالی

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۱۶ ۱۴:۲۶:۳۱

من در قسمت «««««««««««««««««««««دوم»»»»»»»»»»»»»»»»»»»» از «ارث بری» و مباحث Table-per-Type Inheritance و Table-per-Hierarchy strategy صحبتی نکردم. این‌ها در قسمت‌های
«««««««««««««««««««««««««««««««««««««بعد»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»» بررسی می‌شوند.

نویسنده: مرادی
تاریخ: ۱۳۹۱/۰۲/۱۶ ۱۴:۳۱:۳۴

ممنون از پاسخ دهی شما، ولی فکر کنم منظورم رو بد رسوندم، من دقیقا می خوام برام Alter Table بزنه، نه این که کاری به دیتابیس نداشته باشه، (مثل NHibernate)

آیا با Migration ها چنین چیزی به سادگی (در حد چند خط کد قابل استفاده برای کل برنامه)، امکانپذیر هست یا خیر ؟
متاسفانه من همچین چیزی ندیدم تو امکاناتش

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۱۶ ۱۴:۳۶:۱۴

بله. بحث DB Migration روی همین مورد تمرکز دارد. جزئیات آن مفصل است. بسیار کاملتر است از نمونه NHibernate و طراحی مهندسی قابل توجهی دارد. شاید توضیحش نیاز به دو جلسه داشته باشد.

نویسنده: Salehi
تاریخ: ۱۳۹۱/۰۲/۱۹ ۰۱:۴۱:۴۸

اگه از خاصیت RowVersion استفاده نشه، همزمانی کنترل نمی شه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۱۹ ۰۹:۱۴:۴۰

نه. همانطوری که کوئری را صادر کردید اجرا می‌شود. یعنی فقط بر اساس primary key کار آپدیت یا delete را انجام می‌دهد.

نویسنده: Salehi
تاریخ: ۱۳۹۱/۰۲/۱۹ ۱۳:۱۹:۵۸

-1آیا منظور از کنترل همزمانی اینه : "من یه رکورد رو گرفتم و ویرایش کردم، و قبل از ثبت ، شخص دیگه ای اون رو ویرایش و ثبت کرد، اجازه ثبت رکورد ویرایش شده به من داده نمی شه"
-2 در حالت database first که این فیلد وجود نداره چطور؟ اونجا که کنترل همزمانی انجام میشه. اونجا از چه روشی استفاده می کنه.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۱۹ ۱۴:۳۲:۰۵

- بله. استثنای زیر را دریافت خواهید کرد:

```
Entities may have been modified or deleted since entities were loaded
```

- اونجا هم وجود داره. در طراح EF در VS.NET یک فیلد رو می‌تونید انتخاب کنید. بعد در برگه خواص، ConcurrencyMode رو تعیین کنید.

نویسنده: peyman
تاریخ: ۱۳۹۱/۰۴/۰۴ ۱۷:۲۵

سلام مهندس . EF 4.3 دسترسی به EntityTypeConfiguration ندارم . چون میخوام بصورت Fluent کار مپینگ رو انجام بدم و یک نفر اشاره کرده بود که از 4.1 به اینور حذف شده آیا درسته ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۰۴ ۱۷:۳۱

خیر. در قسمت ششم توضیح دادم.

نویسنده: torisoft
تاریخ: ۱۳۹۱/۰۴/۱۴ ۱۰:۱۶

سلام جناب نصیری
ببخشید سوالات من در سطح پائینیه و وقت شمارو هم میگیره ولی خوب....
پروژه من بصورت زیر تعریف شده :
1- MVVMLight SL5 بدون هیچ هاستی
2- Wcf service که تو این پروژه اومدم هاست رو تعریف کردم و همچنین پروژه SL رو در Properties این قسمت Add کردم
3- دو پروژه مجزا مطابق با درس شما DataLayer و DomainClasses
پروژه بعد از Run شدن دیتابیس رو تشکیل نمیده ضمن اینکه هیچ خطا یا هشداری هم ندارم.
لطفا در صورت فرصت راهنمائی بفرمائید.
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۱۴ ۱۰:۵۵

در برنامه‌های وب، قسمت Database.SetInitializer باید در روال Application_Start فایل Global.asax.cs اضافه شود.
ضمن اینکه مکان تعریف رشته اتصالی به بانک اطلاعاتی اینبار فایل web.config خواهد بود.

نویسنده: torisoft
تاریخ: ۱۳۹۱/۰۴/۱۴ ۱۹:۲۵

سلام جناب نصیری
ممنون از پاسخگوئیتون
مواردی رو که شما فرمودید انجام دادم(رشته اتصالی به بانک اطلاعاتی در web.config از قبل درست بود) با سه حالت مختلف
-1 فقط از متد Database.SetInitializer در روال Application_Start استفاده کردم که با خطای زیر مواجه شدم

```
GenericArguments[0], 'DataLayer.TestContext', on
'System.Data.Entity.IDatabaseInitializer`1[TContext]' violates the
constraint of type parameter 'TContext'.
```

-2 از متد Database.SetInitializer صرف نظر کردم و موارد زیر رو به web.config اضافه کردم

```
<contexts>
    <context type=" DataLayer.TestContext, DataLayer">
      <databaseInitializer type="System.Data.Entity.DropCreateDatabaseAlways`1[
                          [DataLayer.TestContext, DataLayer]], EntityFramework" />
    </context>
  </contexts>
```

که با خطای زیر مواجه شدم

```
An error occurred during the processing of a configuration file required to service this request.
Please review the specific error details below and modify your configuration file appropriately
```

-3 فقط از <appSettings> استفاده کردم که بدون هیچ خطا و هشداری بود ولی باز هم دیتابیس تشکیل نشد.
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۱۴ ۱۹:۳۳

به نظر می‌رسه یک آغاز کننده سفارشی رو تهیه کردید و نوع جنریک ارسالی به آن از DbContext مشتق نشده. این مساله بیشتر زمانی رخ می‌ده که در پروژه جاری از چند نگارش مختلف EF در حال استفاده هستید. مثلا لایه سرویس EF A.1 است و لایه دیگر EF A.2 و فایل کانفیگ برنامه به نگارش A.3 اشاره می‌کند. همه رو باید یک دست کنید.

نویسنده: torisoft
تاریخ: ۱۳۹۱/۰۴/۱۵ ۱۷:۵۷

سلام جناب نصیری
یه سوال
نحوه ارتباط wcf با model در mvvm با فرض استفاده از تکنولوژی code first به چه صورت است ؟
آیا از DomainClasses می‌توان به عنوان model در mvvm استفاده کرد ؟
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۱۵ ۱۸:۵۶

بهتر است که model یک view با domain model یکی نباشد. هر view ممکن است در عمل فقط به تعدادی فیلد محدود نیاز داشته باشد. این‌ها با entityهای تعریف شده یکی نیستند و ضرورتی هم ندارد یکی باشند. حتی ممکن است جهت ارائه یک View تعدادی خاصیت جدید را هم تعریف کنید، اما از حاصل محاسباتی خاص بر روی آن‌ها، یک نتیجه را در بانک اطلاعاتی ثبت کنید.
ضمن اینکه تا این سری 15 قسمتی که به عمد با برنامه کنسول جلو رفته رو تموم نکنید، درک صحیحی از اجزای مختلف آن پیدا نخواهید کرد.
هر زمانی هم خواستید مطلبی را در این سطح آموزش دهید با برنامه‌ی کنسول کار کنید چون هدف در اینجا نحوه نمایش آن با

سیلورلایت یا asp.net یا winforms و غیره نیست. هدف آشنایی با زیرساخت‌های اصلی یک فناوری است؛ صرفنظر از نحوه نمایش آن به کاربر.
چگونه باید کلاس‌ها را به بانک اطلاعاتی نگاشت کرد. چگونه باید پس از تغییر کلاس‌ها، بانک اطلاعاتی را با برنامه هماهنگ کرد. چطور باید حالت‌های یک به چند و امثال آن‌را تعریف کرد. چطور باید یک Context را صحیح مدیریت کرد و غیره. هدف این سری، این نوع مباحث پایه‌ای بوده نه فناوری نمایش نهایی آن.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۱۶ ۱۳:۵۵

یک مورد را هم اضافه کنم. تا زمانیکه اولین کوئری، به بانک اطلاعاتی ارسال نشود، کار آغاز دیتابیس انجام نشده و تا آن زمان به تاخیر خواهد افتاد. بنابراین اجرای برنامه به معنای ساخت همزمان بانک اطلاعاتی نخواهد بود.

نویسنده: محمد شهریاری
تاریخ: ۱۳۹۱/۰۴/۲۳ ۱۶:۴

با سلام
در صورتی که بخواهم از یک Initializer استفاده کنم و کلاس رو از DropCreateDatabaseAlways به ارث ببرم . و در متد Seed مانند مثال همین جلسه دیتابیس را مقدار دهی اولیه کنم با خطای Cannot drop database "testdb2012" because it is currently in use روبرو می‌شوم و اصلا متد Seed فراخوانی نمی‌شود . در صورتی که در مثال نمونه که (^) در بخش ارسال نظر گذاشته اید این مورد به درستی اجرا میشود . آیا به نوع پروژه بستگی دارد ؟ پروژه ای که من ایجاد کردم از نوع Consol می‌باشد.

با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۲۳ ۱۶:۳۳

پیغام خطای «it is currently in use» جزو پیغام‌های معروف SQL Server است. زمانیکه در SQL Server بانک اطلاعاتی مورد نظر در حال استفاده باشد، امکان drop آن نیست.
اگر تنها در محیط توسعه خودتان دارید با SQL Server لوکال سیستم کار می‌کنید، این خطا به معنای باز بودن یک کانکشن از management studio به SQL Server است که با بستن management studio مشکل حل می‌شود.
اگر دیتابیس کاری است یا دیتابیس راه دور است، باید بانک اطلاعاتی را به حالت single user دربیارید و بعد می‌تونید اون رو دراپ کنید.

نویسنده: احمد احمدی
تاریخ: ۱۳۹۱/۰۶/۲۶ ۲۳:۴

با سلام و تشکر بخاطر مقالات مفید EF
بنده طبق مثال مقاله پیش رفتم و متادیتاهای Key و Required را اضافه کردم اما با متادیتای MaxLength به مشکل خوردم .
ویژوال همچین پیغامی میده :

```
/*
The type 'System.ComponentModel.DataAnnotations.MaxLengthAttribute' exists in both

'c:\Program Files\Microsoft ADO.NET Entity Framework 4.1\Binaries\EntityFramework.dll'

and

'c:\Program Files\Reference
Assemblies\Microsoft\Framework\.NETFramework\v4.5\System.ComponentModel.DataAnnotations.dll'
*/
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۲۶ ۲۳:۱۷

شما در حال استفاده از EF 4.1 با دات نت 4 و نیم هستید. این دو با هم سازگاری ندارند. از EF 5 با دات نت 4 و نیم استفاده کنید تا مشکل تداخل فضاهای نامی که ذکر شده، برطرف شود.

نویسنده: احمد احمدی
تاریخ: ۱۳۹۱/۰۶/۲۷ ۰:۱

بسیار ممنون - مشکل با نصب این پک برطرف شد :

```
/*
Install-Package EntityFramework -Version 5.0.0-beta2 -Pre
*/
```

فقط :
امکان نصب EF5 بدون Nuget وجود نداره؟
اگر خیر ، پس باید برای هر پروژه یکبار یک دستور رو اجرا کنیم؟
اگر بخوایم پروژه رو روی یک سیستم دیگه که EF5 نداره اجرا کنیم تکلیف چیه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۲۷ ۰:۱۰

- EF 5 نسخه نهایی دارد. نیازی به نسخه بتا نیست.
- بهترین روش استفاده از NuGet است چون دفعه‌ی بعد که به روز رسانی شد به شما اطلاع خواهد داد. مانند به روز رسانی افزونه‌های فایرفاکس. به علاوه سورس آن هم در CodePlex موجود است .
- زمانیکه یکبار دریافت شد در پوشه packages شما قرار می‌گیرد. به این صورت در پروژه‌های دیگر هم قابل ارجاع است.
- توزیع فایل dll آن به همراه برنامه. نیازی به نصب ندارد.

نویسنده: MehRad
تاریخ: ۱۳۹۱/۱۲/۰۴ ۱۳:۵۹

با سلام
اگر برنامه ما به شکلی باشد که ماژول پذیر باشد و در DLL جداگانه همانند این مثال یک ماژول با نام Blog داشته باشیم برای اضافه کردن به Context باید چگونه عمل نماییم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۰۴ ۱۴:۳

« خودکار کردن تعاریف DbSetها در EF Code first »

نویسنده: امیر
تاریخ: ۱۳۹۱/۱۲/۲۴ ۱۰:۲

با سلام (چند سوال)
۱- زمانی که ef کار میکردم اگه میومدیم تیبل درست میکردیم وداخل مثلا۵۰۳ رکورد بود اگه وسط کار این جدوال تغییرات لازم داشت دوباره اسکریپت میگرفتیم تمامی رکوردها حذف و دوبار جدوال ساخته میشد این مشکل در ef حل میشد.ایادر efcf هست که میدونم نیست چطوری حل کرده.مخصوصا در ef
۲- من در efcf امدم جدوال دستی از بانک حذف کردم الان که تغغیران رو میدم میخام دوباره درست کنم این خطا رو میده

.An error occurred while updating the entries. See the inner exception for details
البته قبلا ش یه خطا دیگه میداد

The model backing the 'MyDBContext' context has changed since the database was created. Either manually delete/update the database, or call Database.SetInitializer with an IDatabaseInitializer instance. For example, the DropCreateDatabaseIfModelChanges strategy will automatically delete and recreate the database, and optionally seed it with new data

که با این دستور حلش کردم

```
System.Data.Entity.Database.SetInitializer<Context>(null);
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۲۴ ۱۰:۱۰

- این هم EF هست. یکی database first، یکی code first و یکی model first. ولی زیر ساخت همشون یکی هست.
- اکثر خطاهای EF به صورت inner exception است. یعنی صفحه نمایش استثناء رو باید باز کنید و کمی درخت نمایش داده شده را پیمایش کنید تا به inner exception برسید.
- ریز مسایل به روز رسانی بانک اطلاعاتی، در قسمت‌های 4 و 5 این سری بررسی شده. عجله نکنید. قدم به قدم ...

نویسنده: میثم خوشقدم
تاریخ: ۱۳۹۲/۰۲/۱۳ ۱:۱۰

سلام

من از ابزار Power tools جهت استخراج Contex از دیتابیس موجود استفاده نکردم بلکه از Ado.net data model خود ویژوال استدیو اما مدلی که به من میده خیلی شلوغ و پیچیده است و قابل استفاده نیست و این درحالی است که من یک جدول ساده بدون هیچ گونه ریلیشن و یا روابط پیچیده را استفاده کردم !

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۲/۱۳ ۹:۱۲

یکی از مزایای Code first همین مسایل است. چون کد رو ابزار تولید نمی‌کنه تمیزتر هست. ضمنا روش Database first که شما رفتید، به ObjectContext ختم میشه اما روش Code first به DbContext. به علاوه DbContext هم می‌تونه سفارشی سازی داشته باشه مثل اضافه شدن تعاریف Fluent API .

نویسنده: پویا امینی
تاریخ: ۱۳۹۲/۰۳/۲۸ ۰:۱

با سلام؛ الان که داشتم این بخش رو می‌خوندم به این قسمت رسیدم

البته در این حالت امکان تعریف ErrorMessage وجود ندارد و برای این منظور باید از همان data annotations استفاده کرد.

حال با توجه به این مطلب آیا بهتره که در MVC از annotations به جای Fluent API استفاده کنیم ؟ (چون با استفاده از Fluent API نمی توانیم متن خطا را ایجاد کنیم)

ممنونم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۲۸ ۰:۳۳

عموما در یک پروژه واقعی، از ترکیبی از هر دو روش استفاده می‌شود.
در قسمت‌های بعد مواردی عنوان شده‌اند که با ویژگی‌ها قابل تنظیم نیست؛ مثلا تعیین نام سفارشی جدول واسط چند به چند یا تنظیم روابط خود ارجاع دهنده و موارد پیشرفته دیگری.

نویسنده: پوریا یک کدنویس
تاریخ: ۱۳۹۲/۱۰/۰۴ ۱۰:۳۳

بنده از Nuget نسخه EF6 رو نصب کردم...
در بخشی گفتید که

```
[Column("MyTableKey")]
public int Id { set; get; }
```

اما فکر کنم در این نسخه خاصیت Column وجود نداره... معادلش وجود داره؟ یا حذف شده این خاصیت؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۰۴ ۱۰:۴۲

- حذف نشده. منتقل شده به فضای نام System.ComponentModel.DataAnnotations.Schema .
- عموما در VS.NET اگر اشاره‌گر را در محل یک کلاس نامشخص قرار دهید، یک منوی drop down ظاهر می‌شود که امکان انتخاب فضای نام ممکن و یافت شده‌ای را برای آن میسر می‌کند.

نویسنده: vici
تاریخ: ۱۳۹۲/۱۱/۰۶ ۹:۱۰

سلام؛ اسم schemaName هر چیزی می‌تونه باشه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۰۶ ۱۰:۱۲

بحث استفاده از چند Context در یک برنامه و نظرات آن‌را مطالعه کنید. بررسی معماری چند مستاجری هم مفید است.

نویسنده: امیر حسین
تاریخ: ۱۳۹۲/۱۱/۱۱ ۲:۲۹

با توجه به اینکه در یک فیلد RowVersion جهت همزمانی ایجاد نمودین ، آیا این فیلد رو باید برای همه جداول در نظر گرفت ؟ به همه جداول این فیلد رو اضافه کنم ؟

```
modelBuilder.Entity<Blog>().ToTable("tblBlogs", schemaName:"someUser");
```

در مورد کد بالا : آیا اگر رشته اتصال با نام user1 بود این قسمت رو به این صورت پر کنم ؟

```
modelBuilder.Entity<Blog>().ToTable("tblBlogs", schemaName:"user1");
```

چطور متوجه بشم در هاست اشتراکی از چه schema استفاده میکنم ؟

نویسنده: وحید نصیری

تاریخ: ۱۳۹۲/۱۱/۱۱ ۱۳:۲۵

- به هر قسمتی که فکر می‌کنید این مورد همزمانی ورود یا ویرایش اطلاعات امکان رخ دادن دارد، بله. بهتر است اضافه شود.
- تا زمانیکه خطایی نگرفتید، هیچ تنظیمی را تغییر ندهید.
برای یافتن schema، با استفاده از management studio به سرور متصل شده و یک جدول ساده را درست کنید. بعد مشاهده کنید که ابتدای نام جدول، به چه صورتی شروع شده. مثلا user1.table1 است؟ یا مورد دیگری.

نویسنده: جوادنبی
تاریخ: ۱۳۹۳/۰۱/۱۲ ۱۲:۵۱

با سلام من کد زیر را در application_Start برنامه ام گذاشته ام

```
System.Data.Entity.Database.SetInitializer(new DropCreateDatabaseAlways<ProductContext>());
```

اما خطای به این صورت می‌دهد.

Cannot drop database "testdb" because it is currently in use.

باید چه کار انجام دهم تا هر دفعه که پروژه‌ام را اجرا می‌کنم دیتابیس از نو ساخته بشه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۱۲ ۱۳:۲

- در SQL Server اگر تنها یک کانکشن باز به دیتابیس مفروضی وجود داشته باشد، امکان drop آن‌را نمی‌دهد. برای مثال اگر همزمان management studio هم باز است، این مورد یعنی یک کانکشن باز. آن‌را ببندید تا SQL Server به این نتیجه برسد که کسی از بانک اطلاعاتی درخواستی در حال استفاده نیست.
- در کل رویه ذخیره شده‌ی سیستمی به نام SP_WHO وجود دارد که مصرف کنندگان را لیست می‌کند. شماره آن‌ها را یافته و سپس توسط رویه ذخیره شده دیگری به نام Kill، حذفشان کنید.
- روش دیگر drop آنی یک بانک اطلاعاتی، تک کاربره کردن و سپس حذف آن است:

```
alter database [MyDatbase] set single_user with rollback immediate
 drop database [MyDatabase]
```