بسم الله الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

افزودن یک فیلد جدید

مدرس : مهندس افشین رفوآ

دوره آموزش MVC

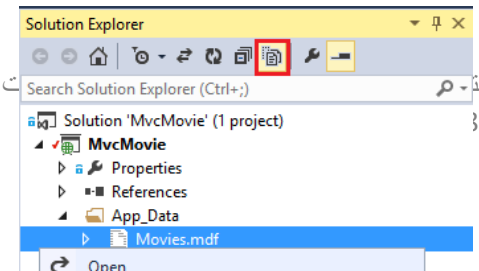
افزودن یک فیلد جدید

در این بخش با بهره گیری از Entity Framework Code First Migrations، بخشی از تغییرات را به کلاس های model انتقال داده تا تغییرات مورد نظر به کل پایگاه داده اعمال شود.

به صورت پیش فرض، هنگامی که از EF برای ایجاد خودکار یک پایگاه داده استفاده می کنید، همان گونه که در فصل قبلی انجام دادید، Code First با اضافه کردن یک جدول به پایگاه داده این قابلیت را فراهم می کند که بتوان بررسی و ردیابی کرد آیا schema ی پایگاه داده و کلاس های model که schema در اصل از این کلاس ها ساخته شده، با هم هماهنگ هستند یا خیر. در صورت هماهنگ نبودن این دو، EF یک پیغام خطا صادر می کند. این امر به یافتن مشکلات برنامه در زمان توسعه کمک شایانی می کند، مشکلاتی که در غیر این صورت ممکن است تنها در زمان اجرا (توسط خطاهای مبهم) از وجود آن ها آگاه شوید.

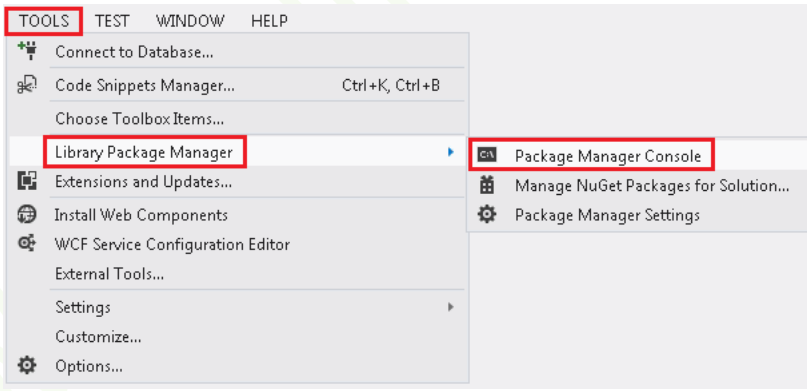
تنظیمات لازم Code First Migration برای انتقال تغییرات به کلاس های model

پنجره ی Solution Explorer را باز کنید. بر روی فایل Movies.mdf راست کلیک کرده و با انتخاب Delete، پایگاه داده ی با نام movies را حذف کنید. در صورت نیافتن فایلی به نام Movies.mdf، روی آیکون Show All Files که در کادر قرمز کوچک تصویر زیر نشان داده شده، کلیک نمایید.

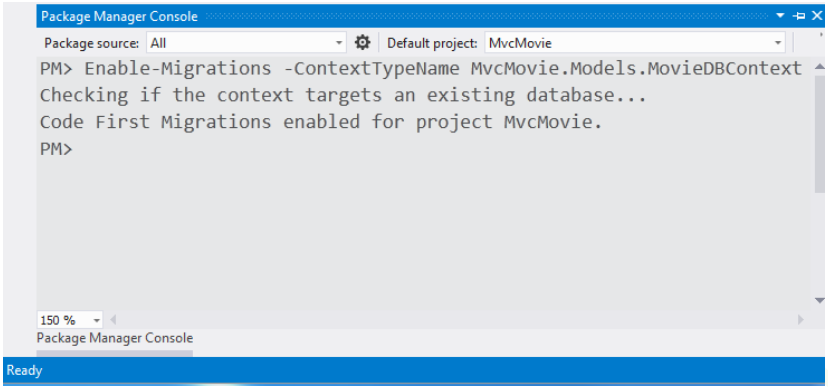


برنامه را کامپایل (build) کرده تا از عدم وجود خطا در آن اطمینان حاصل نمایید.

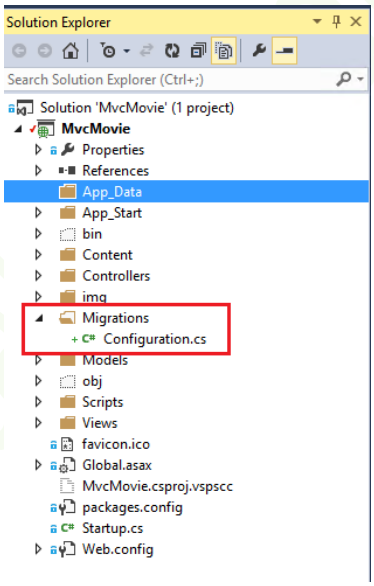
از منو Tools، Library Package Manager را انتخاب نموده، سپس Package Manager Console را کلیک کنید.



در پنجره ی Package Manager Console و پس از اعلان خطی PM، این عبارت را وارد نمایید: Enable-Migrations -ContextTypeName MvcMovie.Models.MovieDBContext.



فرمان Enable-Migrations (که در تصویر زیر به نمایش گذاشته شده) یک فایل Configuration.cs در پوشه ی Migrations ایجاد می کند.

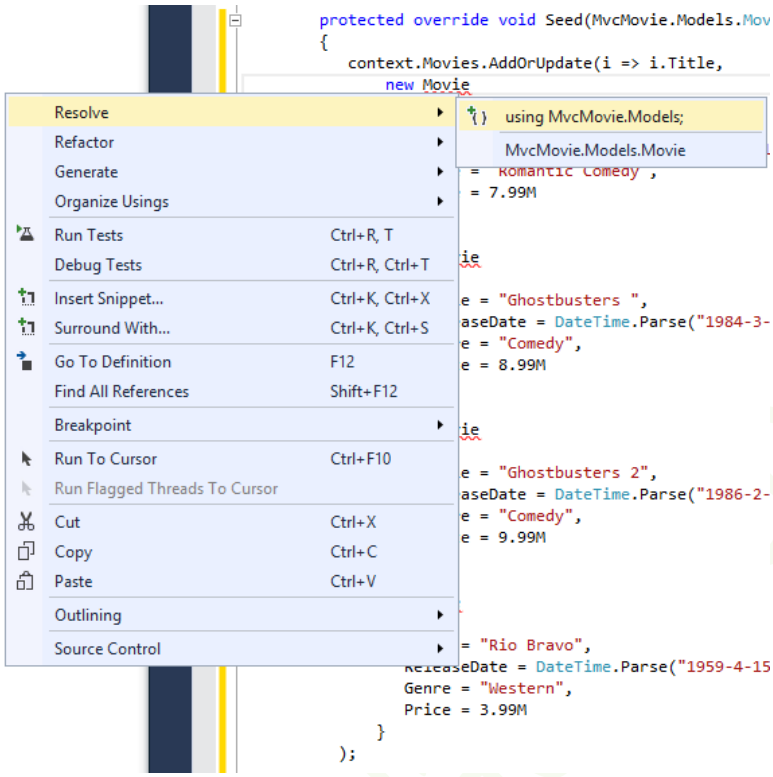


Visual Studio فایل Configuration.cs را باز می کند. متد Seed را در فایل Configuration.cs، با کد زیر جایگزین کنید:

```
protected override void Seed(MvcMovie.Models.MovieDBContext context)
{
  context.Movies.AddOrUpdate( i => i.Title,
    new Movie
    {
      Title = "When Harry Met Sally",
      ReleaseDate = DateTime.Parse("1989-1-11"),
```

```
            Genre = "Romantic Comedy",
            Price = 7.99M
        },

        new Movie
        {
            Title = "Ghostbusters ",
            ReleaseDate = DateTime.Parse("1984-3-13"),
            Genre = "Comedy",
            Price = 8.99M
        },

        new Movie
        {
            Title = "Ghostbusters 2",
            ReleaseDate = DateTime.Parse("1986-2-23"),
            Genre = "Comedy",
            Price = 9.99M
        },

        new Movie
        {
            Title = "Rio Bravo",
            ReleaseDate = DateTime.Parse("1959-4-15"),
            Genre = "Western",
            Price = 3.99M
        }
    );

}
```

روی خط های قرمز در زیر Movie راست کلیک کرده و Resolve را کلیک نمایید، سپس using MvcMovie.Models را انتخاب کنید:



این کار یک دستور using به مانند زیر اضافه می کند:

```
using MvcMovie.Models;
```

Code First Migrations متد Seed را پس از هر بار انتقال (منظور فراخوانی update-database در Package Manager Console می باشد) صدا می زند، متد نام برده سطرهایی که قبلا درج شده باشند را بروز رسانی کرده و یا چنانچه این سطرها هنوز درج نشده باشند، آن ها را درج می کند.

متد AddOrUpdate در کد زیر، عملیات "upsert" بروز رسانی و درج سطر را انجام می دهد:

```
    context.Movies.AddOrUpdate(i => i.Title,
  new Movie
  {
    Title = "When Harry Met Sally",
    ReleaseDate = DateTime.Parse("1989-1-11"),
    Genre = "Romantic Comedy",
    Rating = "PG",
    Price = 7.99M
  }
```

از آنجایی که متد Seed با هر بار انتقال، اجرا می شود، نمی توان به راحتی اطلاعات وارد یا درج نمود، زیرا آن سطرهایی که قصد افزودن آن ها را دارید، از قبل پس از انتقال نخست که پایگاه داده طی آن ایجاد شده، درج گردیده و موجود می باشد. عملیات "upsert" از بروز خطاهایی که با سعی بر درج سطری که از قبل موجود می باشد، جلوگیری می کند ولی در این میان تغییراتی را که ممکن است حین تست برنامه به داده ها اعمال کرده باشید را بازنویسی (override) می کند. ممکن است نخواهید که این اتفاق در مورد داده های آزمایشی موجود در برخی جداول رخ دهد؛ به عبارتی دیگر مایلید داده هایی را که حین تست برنامه تغییر داده شده اند (تغییراتی که به آن ها اعمال شده اند)، آن تغییرات پس از بروز رسانی پایگاه داده نیز باقی بمانند. برای این منظور بایستی یک عملیات درج شرطی (conditional insert) پیاده کنید، به این معنی که یک سطر فقط به این شرط درج شود که از قبل وجود نداشته باشد.

اولین پارامتر ارسالی به متد AddOrUpdate، خاصیتی را که باید برای بررسی وجود یا عدم وجود سطر مورد نظر بکار رود، مشخص می کند. برای اطلاعاتی آزمایشی که ویژه ی movie فراهم می کنید، به این خاطر که هر یک از title های موجود در لیست منحصر بفرد هستند، خاصیت Title مناسب بوده و برای نیل به این مقصود قابل استفاده می باشد:

```
context.Movies.AddOrUpdate(i => i.Title,
```

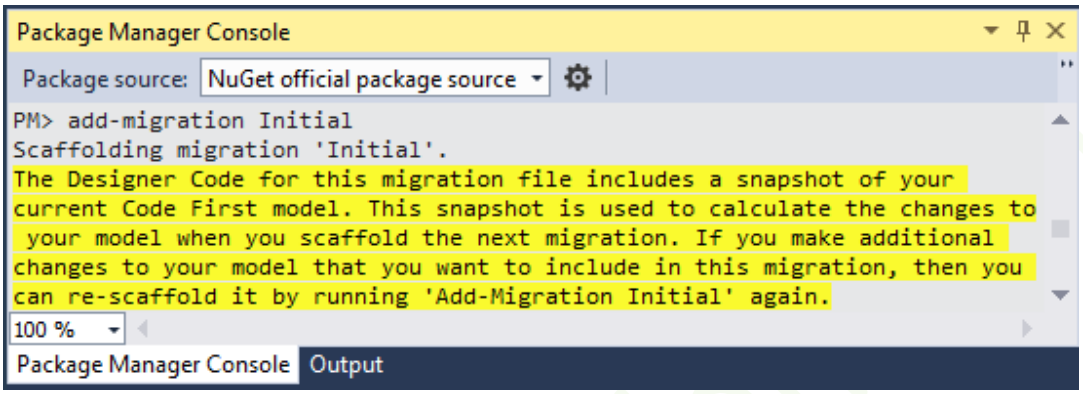
این کد فرض را بر این می گذارد که title ها منحصر بفرد هستند. اگر خودتان به صورت دستی یک title تکراری اضافه نمایید، دفعه ی بعدی که migration اجرا می کنید، خطای زیر (exception) صادر می شود:

Sequence contains more than one element

CTRL-SHIFT-B را زده تا پروژه کامپایل شود. (در صورت build نکردن پروژه در این مرحله، گام های بعدی با مشکل مواجه می شوند.)

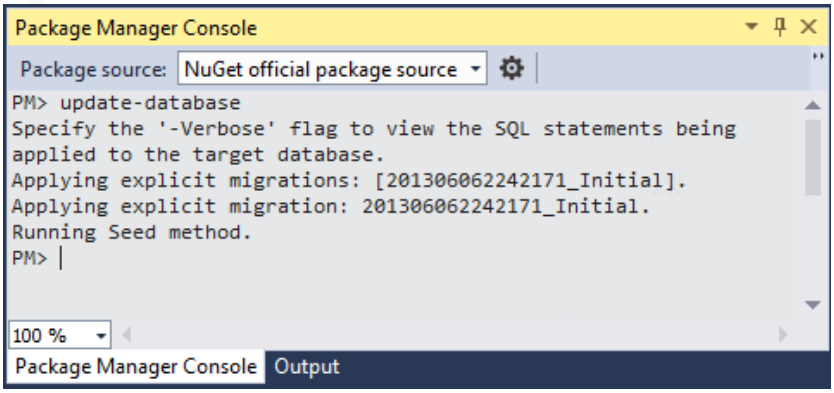
مرحله ی بعدی ایجاد یک کلاس DbMigration برای فایل migration اولیه (initial migration) است. با این انتقال یک پایگاه داده ی جدید ایجاد می شود، به این خاطر هم است که فایل movie.mdf را در گام قبلی حذف کردید.

در پنجره ی Package Manager Console، فرمان add-migration Initial را وارد نموده تا فایل اولیه ی migration ایجاد شود. واژه ی "Initial" صرفا یک اسم برای نام گذاری فایل migration ایجاد شده است و استفاده از آن اختیاری می باشد.



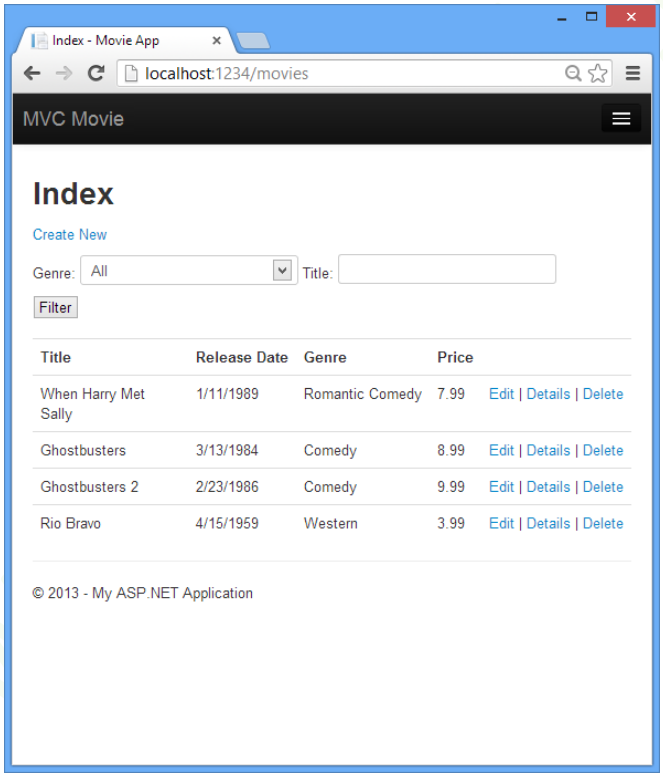
Code First Migrations یک class file (به نام {DateStamp}_Initial.cs) در پوشه ی Migrations ایجاد می کند. این کلاس دربردارنده ی کدی است که schema ی پایگاه داده را ایجاد می کند. اسم فایل migration دارای یک پیشوند timestamp است که در مرتب سازی کمک می کند. اگر فایل {DateStamp}_Initial.cs را بررسی کنید، خواهی دید که حاوی دستورهایی برای ساخت جدول Movies ویژه ی پایگاه داده ی Movie می باشد. اگر پایگاه داده را بر اساس دستورهای زیر بروز رسانی کنید، فایل {DateStamp}_Initial.cs اجرا شده و schema ی پایگاه داده را ایجاد می کند. سپس متد Seed اجرا شده و پایگاه داده ی مربوطه را با داده های آزمایشی پر می کند.

حال به منظور ایجاد پایگاه داده و اجرای متد Seed، فرمان update-database را در Package Manager Console وارد نمایید.



اگر خطایی دریافت کردید که به حاضر (و موجود) بودن جدول مورد نظر اشاره داشت، بدانید که شما برنامه را پس از حذف پایگاه داده و پیش از اینکه دستور update-database را اجرا کنید، راه اندازی کرده اید. در صورت مواجه شدن با چنین شرایطی، بایستی فایل Movies.mdf را مجددا حذف کرده و دستور update-database را بار دیگر اجرا کنید. اگر بازهم با خطا مواجه شدید، پوشه ی migrations و تمام محتویات آن را حذف کرده و دستورات را از بالای صفحه تا پایین دنبال کنید (یعنی ابتدا فایل Movies.mdf را حذف کرده سپس به گام Enable-Migrations بپردازید).

برنامه را اجرا کرده و به آدرس Movies URL/ بروید. داده های متد seed نمایش داده می شوند.



افزودن یک خاصیت به نام Rating به مدل Movie

کار خود را با افزودن یک خاصیت جدید Rating به کلاس Movie آغاز کنید. فایل Models\Movie.cs را باز کرده، سپس خاصیت Rating را مانند مثال زیر اضافه کنید:

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

کد کلاس کامل movie، بدین صورت خواهد بود:

```
public class Movie
{
    public int ID { get; set; }
    public string Title { get; set; }

    [Display(Name = "Release Date")]
    [DataType(DataType.Date)]
    [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]
    public DateTime ReleaseDate { get; set; }
    public string Genre { get; set; }
    public decimal Price { get; set; }
    public string Rating { get; set; }
}
```

اکنون برنامه را کامپایل کنید (Ctrl+Shift+B).

چون که یک فیلد جدید به کلاس Movie اضافه کرده اید، لازم است white list را بروز رسانی کنید تا این خاصیت جدید نیز دربرگرفته شود. خصیصه ی bind را برای متدهای Create و Edit بروز رسانی کرده تا خاصیت Rating اضافه شود:

```
[Bind(Include = "ID,Title,ReleaseDate,Genre,Price,Rating")]
```

کار دیگری که باید انجام دهید، بروز رسانی view template ها برای نمایش، ایجاد و ویرایش خاصیت Rating در browser view می باشد.

فایل \Views\Movies\Index.cshtml را باز کرده و یک عنوان ستون <th>Rating</th> درست بعد از ستون Price در سمت راست اضافه کنید. سپس یک ستون <td> به انتهای template اضافه کرده تا مقدار @item.Rating نمایش داده شود (render). آنچه در زیر مشاهده می کنید، نسخه ی بروز رسانی شده ی Index.cshtml view template است:

```
@model IEnumerable<MvcMovie.Models.Movie>
@{
    ViewBag.Title = "Index";
}
<h2>Index</h2>
<p>
    @Html.ActionLink("Create New", "Create")
    @using (Html.BeginForm("Index", "Movies", FormMethod.Get))
    {
```

```
<p>
  Genre: @Html.DropDownList("movieGenre", "All")
  Title: @Html.TextBox("SearchString")
  <input type="submit" value="Filter" />
</p>
}
</p>
<table class="table">
  <tr>
    <th>
      @Html.DisplayNameFor(model => model.Title)
    </th>
    <th>
      @Html.DisplayNameFor(model => model.ReleaseDate)
    </th>
    <th>
      @Html.DisplayNameFor(model => model.Genre)
    </th>
    <th>
      @Html.DisplayNameFor(model => model.Price)
    </th>
    <th>
      @Html.DisplayNameFor(model => model.Rating)
    </th>
    <th></th>
  </tr>
  @foreach (var item in Model)
  {
    <tr>
      <td>
        @Html.DisplayFor(modelItem => item.Title)
      </td>
      <td>
        @Html.DisplayFor(modelItem => item.ReleaseDate)
      </td>
      <td>
        @Html.DisplayFor(modelItem => item.Genre)
      </td>
      <td>
        @Html.DisplayFor(modelItem => item.Price)
      </td>
      <td>
        @Html.DisplayFor(modelItem => item.Rating)
      </td>
      <td>
        @Html.ActionLink("Edit", "Edit", new { id = item.ID }) |
        @Html.ActionLink("Details", "Details", new { id = item.ID }) |
        @Html.ActionLink("Delete", "Delete", new { id = item.ID })
      </td>
    </tr>
```

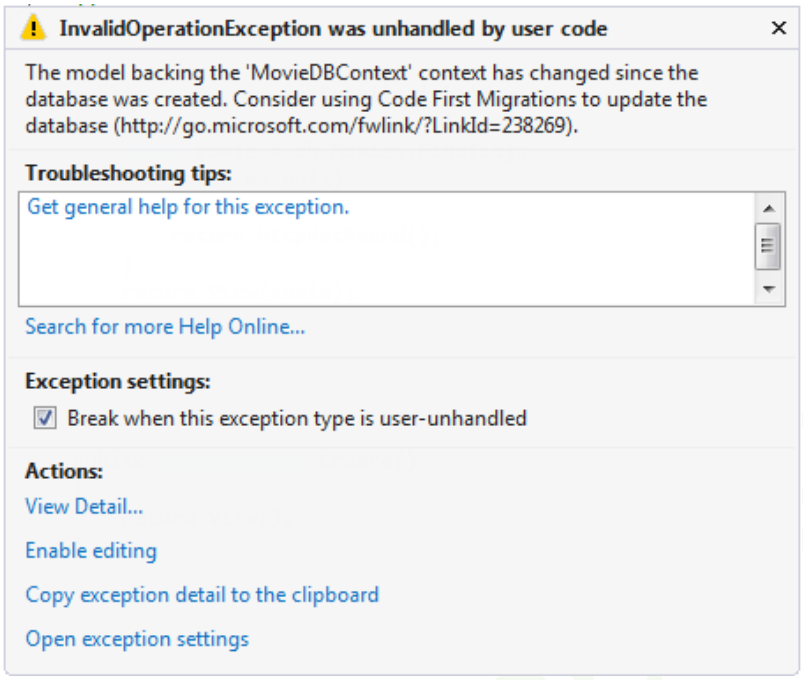
```
    }

  </table>
```

سپس فایل \Views\Movies\Create.cshtml را باز کرده و فیلد Rating را با markup رنگی شده ی زیر اضافه کنید. این کد یک text box نمایش می دهد که تا بتوان زمانی که یک movie جدید ایجاد می شود، rating آن را مشخص کرد.

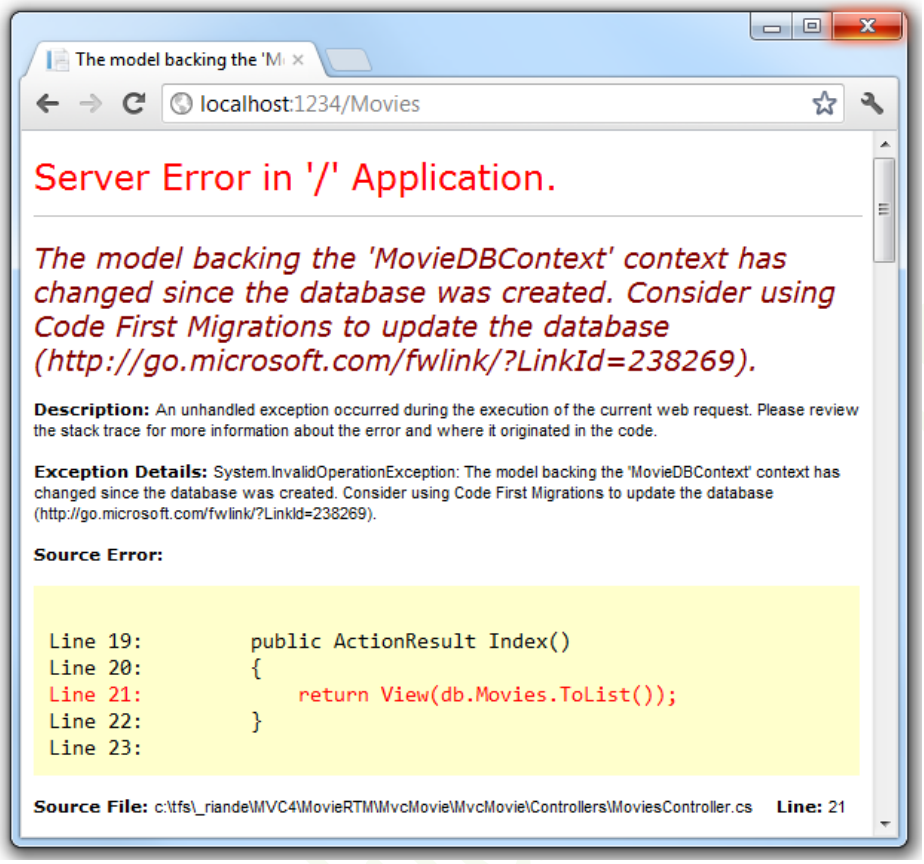
```
<div class="form-group">
        @Html.LabelFor(model => model.Price, new { @class = "control-label col-md-2" })
        <div class="col-md-10">
          @Html.EditorFor(model => model.Price)
          @Html.ValidationMessageFor(model => model.Price)
        </div>
      </div>

      <div class="form-group">
        @Html.LabelFor(model => model.Rating, new { @class = "control-label col-md-2" })
        <div class="col-md-10">
          @Html.EditorFor(model => model.Rating)
          @Html.ValidationMessageFor(model => model.Rating)
        </div>
      </div>

      <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
          <input type="submit" value="Create" class="btn btn-default" />
        </div>
      </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

اکنون کد برنامه بروز رسانی شده و خاصیت Rating را به آن اضافه گردیده.

برنامه را اجرا کرده و به آدرس /Movies بروید. به هنگام انجام این کار، با یکی از خطاهای زیر مواجه می شوید:



Model ای که از بستر اجرا (context) 'MovieDBContext' پشتیبانی می کند، از زمانی که پایگاه داده برای اولین ایجاد شد، تغییر یافته. به همین دلیل توصیه می کنیم با استفاده از Code First Migrations، پایگاه داده را بروز رسانی کنید (http://go.microsoft.com/fwlink/?LinkId=238269).



پیام خطای فوق به این خاطر نمایش داده شده که کلاس Movie model برنامه ی مورد نظر در حال حاضر با schema ی جدول Movie پایگاه داده ی جاری متفاوت است. (هیچ ستونی به نام Rating در جدول پایگاه داده وجود ندارد.)

چندین روش برای برطرف ساختن این خطا وجود دارد:

1. کاری کنید که EF به صورت خودکار پایگاه داده را بر اساس schema ی کلاس model جدید حذف و سپس مجددا ایجاد کند. این روش در مراحل اولیه ی چرخه ی تولید برنامه، زمانی که یک پایگاه داده ی آزمایشی را ایجاد می کنید، بسیار مفید است؛ زیرا به شما امکان می دهد model و schema ی پایگاه داده را همزمان با هم و به سرعت توسعه دهید. جنبه ی منفی آن این است که داده های موجود در پایگاه داده را از دست می دهید – از این رو تحت هیچ شرایطی نباید این روش را برای پایگاه داده ی تولیدی (production database) پیاده کنید! استفاده از یک مقداردهنده ی اولیه (initializer)

برای مقداردهی (seed) خودکار یک پایگاه داده با داده های آزمایشی اغلب یک روش بسیار کارآمد برای توسعه ی برنامه محسوب می شود.

2. Schema ی پایگاه داده ی جاری را به صورت صریح و به گونه ای اصلاح کنید که با کلاس های model منطبق باشد. از مزایای این روش این است که می توانید داده های خود را نگه دارید. می توانید این تغییر را به صورت دستی و یا با ایجاد یک database change script ایجاد کنید.
3. با استفاده از Code First Migrations ، schema ی پایگاه داده را بروز رسانی کنید.

در این آموزش از روش سوم بهره می گیریم.

متد Seed را بروز رسانی کرده تا مقدار ستون جدید را ارائه کند. فایل Migrations\Configuration.cs را باز کرده و یک فیلد Rating به هر شی Movie اضافه کنید.

```
new Movie
    {
      Title = "When Harry Met Sally",
      ReleaseDate = DateTime.Parse("1989-1-11"),
      Genre = "Romantic Comedy",
      Rating = "PG",
      Price = 7.99M

    },
```

Solution را مورد نظر را دوباره بسازید (build کنید)، سپس فرمان زیر را در پنجره ی Package Manager Console وارد نمایید:

```
add-migration Rating
```

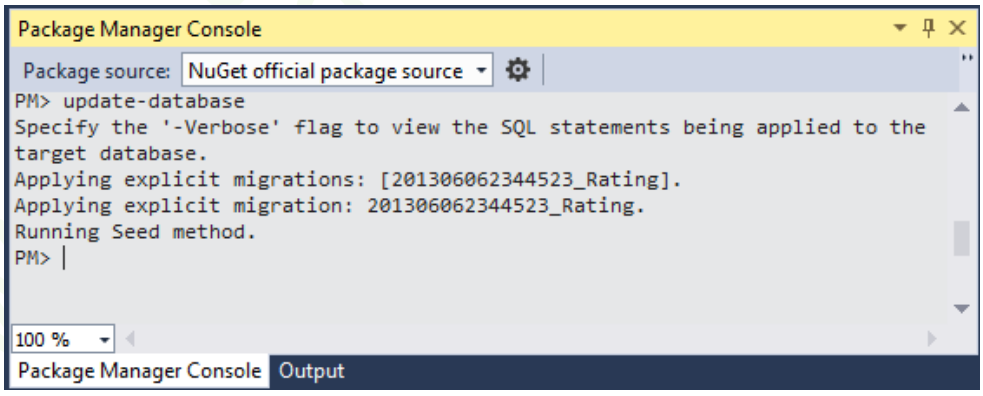
فرمان add-migration به migration framework دستور می دهد که movie model جاری را با schema پایگاه داده ی movie جاری بررسی کرده و کد مورد نیاز برای انتقال (migrate) پایگاه داده به model جدید را تولید کند. واژه ی Rating صرفا یک اسم است و برای نام گذاری فایل migration بکار می رود.

زمانی که این فرمان کاملا اجرا شده و به پایان رسید، محیط ویژوال class file ای که تعریف کننده ی کلاس مشتق DbMIgration می باشد را باز می کند. همچنین می توانید کدی که ستون جدید را ایجاد کرده در متد Up، مشاهده نمایید.

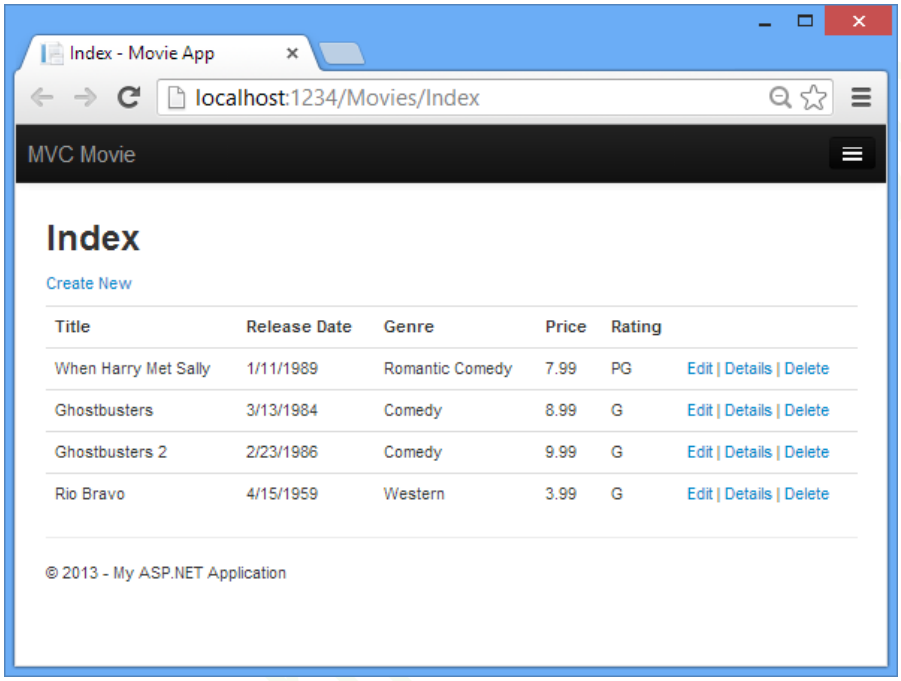
```
public partial class AddRatingMig : DbMigration
{
  public override void Up()
  {
    AddColumn("dbo.Movies", "Rating", c => c.String());
  }

  public override void Down()
  {
    DropColumn("dbo.Movies", "Rating");
  }

}
```

Solution را مجددا ساخته (build)، سپس دستور update-database را در پنجره ی Package Manager Console وارد نمایید.

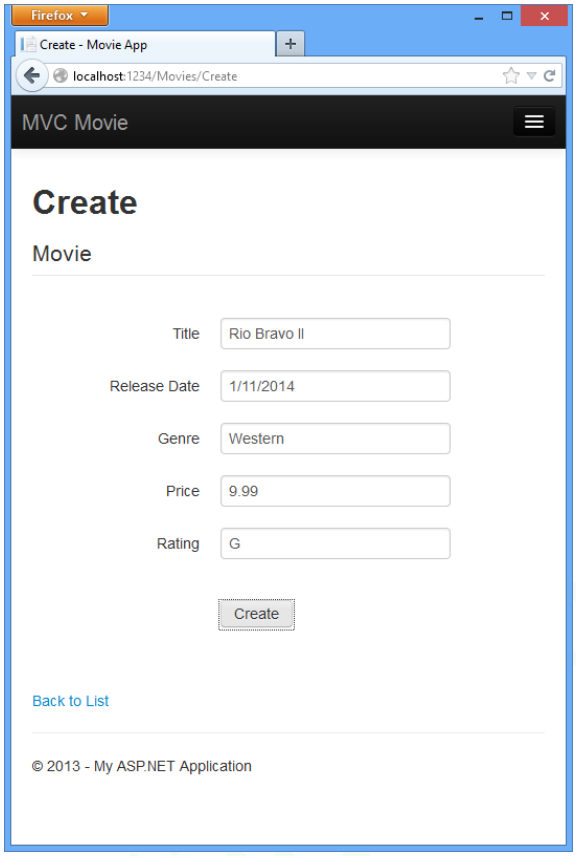
تصویر زیر خروجی را در پنجره ی Package Manager Console به نمایش گذاشته است (date stamp ای که پیشوند واژه ی Rating می باشد، متفاوت خواهد بود).



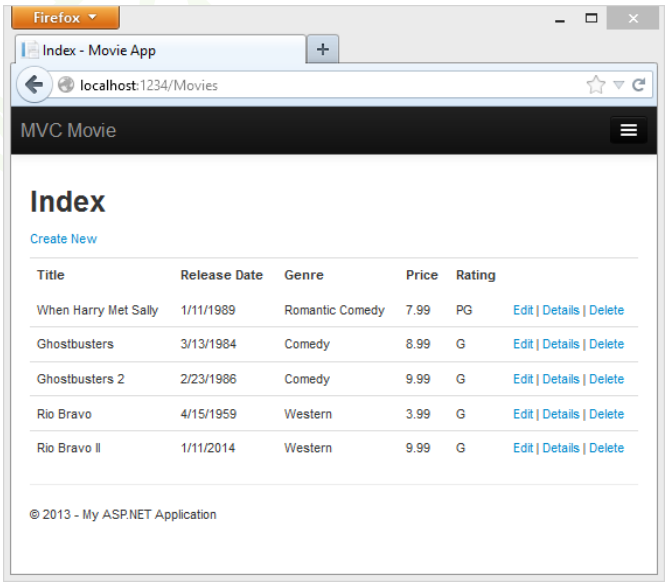
برنامه را مجددا اجرا کرده و به آدرس /Movies بروید. اکنون می توانید فیلد Rating جدید را مشاهده کنید.



بر روی لینک Create New کلیک کرده تا یک فیلم جدید به فیلم های قبلی اضافه شود. همان طور که مشاهده می کنید، می توانید برای فیلم جدید، rating نیز ایجاد کنید.



روی دکمه ی Create کلیک کنید. فیلم جدید، به ضمیمه ی rating آن هم اکنون در فهرست فیلم ها نمایش داده می شود:



اکنون که پروژه از migrations استفاده می کند، دیگر به هنگام افزودن یک فیلد جدید یا بروز رسانی schema نیازی به حذف پایگاه داده نیست.

بایستی فیلد Rating را به view template های Edit، Details و Delete نیز اضافه کنید.

حال اگر بار دیگر فرمان "update-database" را در پنجره ی Package Manager Console وارد کنید، هیچ کد migration ای اجرا نمی‌شود، زیرا که schema با model کاملا منطبق می باشد و نیازی به اجرای کد migration نیست. اما باید دقت داشته باشید که اجرای فرمان "update-database" باعث می‌شود متد Seed مجددا اجرا شود. اگر در این میان داده های متد Seed را تغییر دهید، تغییرات ایجاد شده از دست خواهد رفت زیرا که متد نام برده داده ها را upsert می کند.

در این مبحث با نحوه ی اصلاح اشیا model و هماهنگ نگه داشتن پایگاه داده با تغییرات آشنا شدید. همچنین چگونگی پر کردن یک پایگاه داده ی جدید با داده های نمونه برای تست را آموختید.