بسم الله الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

افزودن قابلیت اعتبارسنجی

مدرس : مهندس افشین رفوآ

دوره آموزش MVC

## افزودن قابلیت اعتبارسنجی

در درس حاضر، منطق اعتبار سنجی را به مدل Movie خود اضافه کرده و اطمینان حاصل می کنیم که هر بار کاربر سعی بر ایجاد یا ویرایش یک فیلم در مدل movie می کند، قوانین اعتبارسنجی به درستی و کامل اجرا می شوند.

## اصل عدم تکرار یک رفتار یا قابلیت

یکی از اصول پایه ای چارچوب نرم افزاری تحت وب ASP.NET MVC این است که برنامه نویس نباید یک قابلیت یا رفتار را بیش از یکبار تعریف کند؛ به عبارتی دیگر یک عملکرد را تعریف کرده و آن را طوری پیاده سازی کند که در کل برنامه اعمال شود. این کار باعث می شود مقدار کدی که باید نوشت کاهش یافته و از احتمال رخداد خطا کاسته شود و نیز در صورت برخورد با خطا، مدیریت و نگهداشت آن ها به مراتب آسان تر شود.

امکان پشتیبانی از منطق اعتبارسنجی که چارچوب نرم افزاری ASP.NET MVC و تکنولوژی EF Code First ارائه می کنند، بیانگر این اصل در نوشتن برنامه می باشند. می توانید به صورت اعلانی قوانین اعتبارسنجی را یکجا (در کلاس model ) تعریف کنید تا این قوانین در تمامی بخش های برنامه اعمال شوند.

حال به نحوه ی استفاده ی بهینه از امکان اعتبارسنجی و پیاده سازی آن در برنامه ی movie خواهیم پرداخت.

## پیاده سازی قوانین اعتبار سنجی در مدل Movie

کار خود را با افزودن منطق اعتبار سنجی به کلاس Movie، آغاز می کنیم.

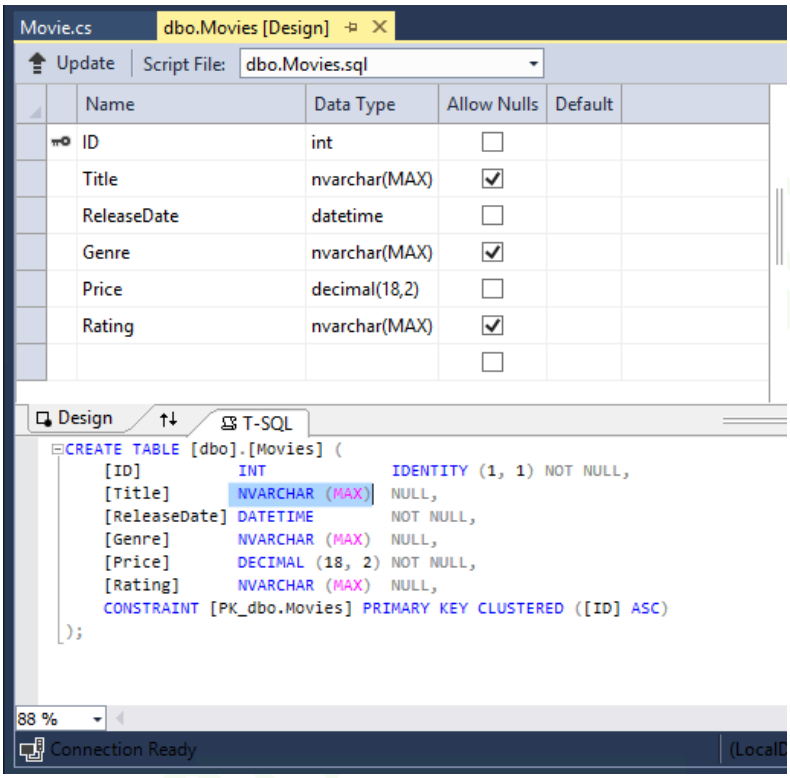
ابتدا فایل Movie.cs را باز کنید. همان طور که در فایل ذکر شده مشاهده می کنید، فضای نامی System.ComponentModel.DataAnnotations شامل System.Web نمی شود. DataAnnotations یک مجموعه ای از خصیصه های اعتبارسنجی توکار فراهم می کند که شما می توانید به صورت اعلانی به هر کلاس یا خاصیتی اعمال کنید. (این مجموعه همچنین دربردارنده ی خصیصه های فرمت دهی مانند DataType است که در قالب دهی کاربرد داشته ولی هیچ اعتبارسنجی ارائه نمی دهد.)

حال کلاس Movie را بروز رسانی کرده تا از validation attribute هایی (خصیصه هایی که اعتبارسنجی انجام می دهند) همچون Required، StringLength، RegularExpression و Range بتوانید بهره بگیرید.

کلاس Movie را با تکه کد زیر جایگزین کنید:

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

    [StringLength(60, MinimumLength = 3)]
    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; }
    [RegularExpression(@"^[A-Z]+[a-zA-Z''-'\s]*$")]
    [Required]
    [StringLength(30)]
    public string Genre { get; set; }
    [Range(1, 100)]
    [DataType(DataType.Currency)]
    public decimal Price { get; set; }
    [RegularExpression(@"^[A-Z]+[a-zA-Z''-'\s]*$")]
    [StringLength(5)]
    public string Rating { get; set; }
}
```

خصیصه ی StringLength حداکثر تعداد کاراکترها یا طول یک رشته را تنظیم می کند و این محدودیت را بر روی پایگاه داده تعریف می کند، به این خاطر هم schema ی پایگاه داده تغییر می کند. بر روی جدول Movies در پنجره ی Server Explorer راست کلیک کرده و Open Table Definition را انتخاب نمایید:




در تصویر فوق، می بینید که تمامی فیلدهای رشته ای بر روی (MAX) NVARCHAR تنظیم شده اند. برای بروز آوری schema از migrations استفاده می کنیم. solution را بازسازی (build) کرده، سپس پنجره ی Package Manager Console را باز کنید و در آن فرمان های زیر را وارد نمایید:

```
add-migration DataAnnotations
update-database
```

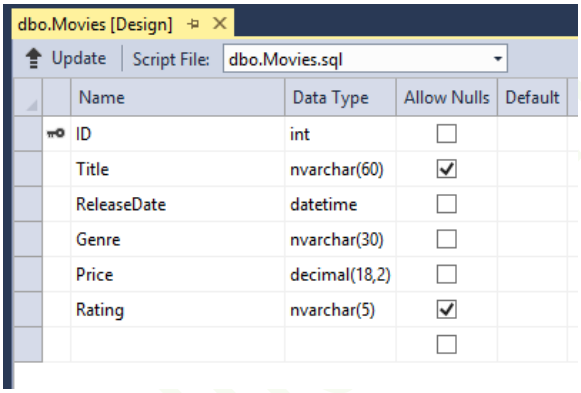
پس از این که اجرای این فرمان به پایان رسید، Visual Studio فایل کلاسی که کلاس مشتق جدید DbMIgration را با اسم (DataAnnotations) تعریف می کند، باز می کند. اگر به متد Up دقت کنید، می توانید کدی که محدودیت های schema (schema constraints) را برزو رسانی می کند، مشاهده کنید:

```
public override void Up()
    {
      AlterColumn("dbo.Movies", "Title", c => c.String(maxLength: 60));
```

```
        AlterColumn("dbo.Movies", "Genre", c => c.String(nullable: false, maxLength: 30));
        AlterColumn("dbo.Movies", "Rating", c => c.String(maxLength: 5));

    }
```

فیلد Genre دیگر nullable نیست (به این معنا که بایستی مقداری را در آن وارد کنید). فیلد Rating، بیش از 5 کاراکتر نمی پذیرد و فیلد Title نیز حداکثر تا 60 کاراکتر می پذیرد. حداقل تعداد کاراکتر 3 در فیلد Title و محدوده ی (range) تعیین شده بر روی فیلد Price، تغییراتی را در schema ی پایگاه داده ایجاد نکرد.

Schema ی پایگاه داده ی Movie را مشاهده کنید:




همان طور که در تصویر بالا مشاهده می کنید، فیلدهای رشته ای، محدودیت های طولی جدید برای تعداد کاراکترهای ورودی را نمایش می دهد و فیلد Genre دیگر nullable نیست.

Validation attribute ها رفتاری را که می خواهید خواص model داشته باشند را مشخص می کند. خصیصه های Required و MinimumLength نشانگر این است که خاصیت بایستی یک مقدار داشته باشد؛ با این وجود هیچ چیز نمی تواند مانع از آن شود که کاربر با وارد کردن یک فاصله یا فضای خالی، شرط این اعتبارسنجی را برآورده سازد. خصیصه ی RegularExpression به منظور مشخص کردن کاراکترهای ورودی بکار می رود. ( regular expression درباره ی الگوی کاراکترها توضیح می دهد.) در کد بالا، Genre و Rating باید فقط از حروف استفاده کنند (فضای خالی، اعداد و کاراکترهای ویژه مجاز نمی باشند). خصیصه ی Range محدوده ی یک مقدار را مشخص می کند (از چه مقداری تا چه مقداری را بپذیرد). خصیصه ی StringLength به شما امکان می دهد حداکثر/حداقل تعداد کاراکترهای ورودی (حداکثر طول خاصیت string) را مشخص نمایید. انواع

مقداری (value type) مانند decimal، int، float و DateTime به صورت ذاتی الزامی بوده و نیازی به خصیصه ی Required ندارد.

Code First اطمینان حاصل می کند که قوانین اعتبارسنجی تعریف شده در کلاس model پیش از اینکه برنامه تغییرات را در پایگاه داده ذخیره کند، اعمال می شوند. در مثال زیر، به محض فراخوانی متد SaveChanges یک خطای (exception) DbEntityValidationException رخ می دهد، زیرا که چندین Movie property values الزامی، تنظیم یا ارائه نشده است:

```
MovieDBContext db = new MovieDBContext();
Movie movie = new Movie();
movie.Title = "Gone with the Wind";
db.Movies.Add(movie);
db.SaveChanges();      // <= Will throw server side validation exception
```

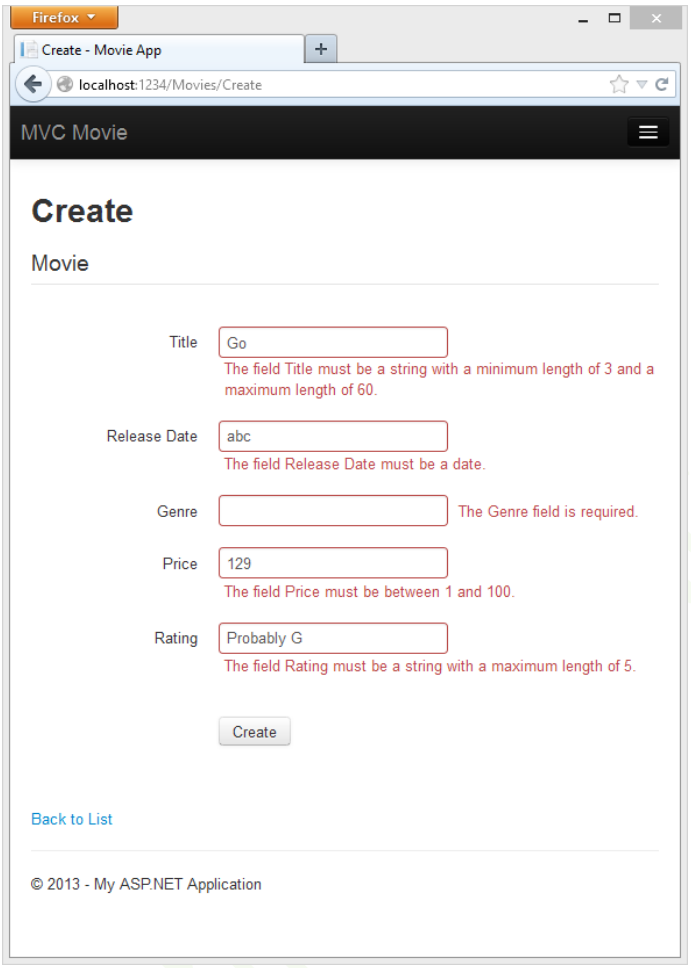
کدی که در نمونه ی فوق مشاهده می کنید، خطای زیر را صادر می کند:

" Validation failed for one or more entities. See 'EntityValidationErrors' property for more details."

.NET Framework به صورت خوکار نیز قوانین اعتبارسنجی را اعمال می کند که در قدرتمندتر ساختن برنامه بسیار موثر می باشد. همچنین با بهره گیری از این قابلیت .NET، احتمال راه یافتن داده های غیرمجاز به پایگاه داده که با عدم اعتبارسنجی ممکن است رخ دهد از بین می رود.

## طراحی رابط کاربری ویژه ی خطاهای اعتبار سنجی (Validation Error UI) در ASP.NET MVC

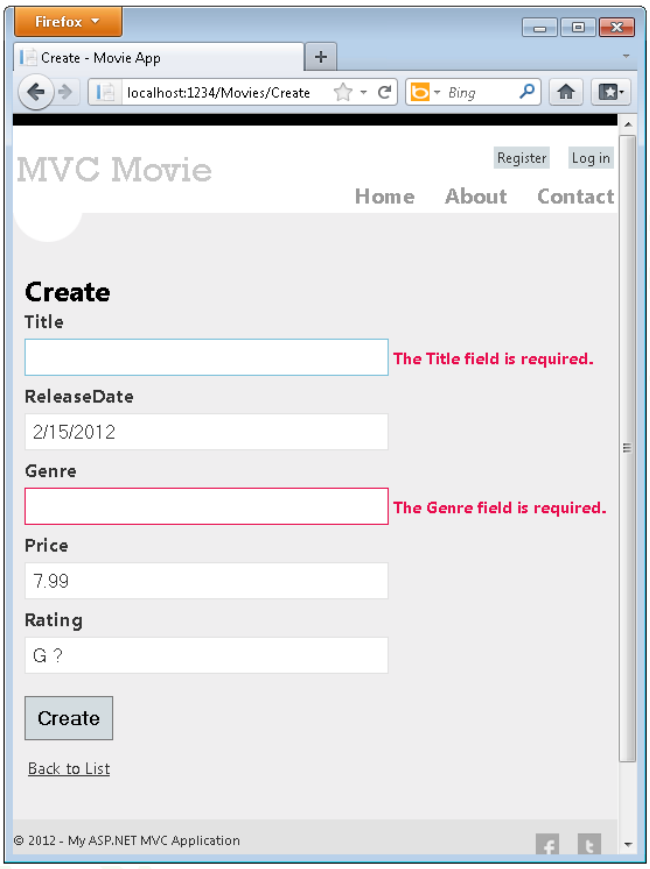
برنامه را اجرا کرده و به آدرس /Movies بروید. با کلیک بر روی لینک Create New یک فیلم جدید اضافه نمایید. فرم حاضر را با چندین مقدار غیرمجاز پر کنید. به محض اینکه اعتبارسنجی jQuery در سمت سرویس گیرنده به وجود خطا پی می برد، یک پیغام خطا صادر می کند.




همان طور که مشاهده می کنید، فرم حاضر با استفاده از خط حاشیه ی قرمز رنگ، آن کادرهای متنی که مقادیر ورودی آن ها غیرمجاز می باشد را برجسته ساخته و پیغام خطای اعتبارسنجی مرتبط را در زیر هر کدام به نمایش گذاشته است. خطاها هم در سمت سرویس گیرنده (با استفاده از JavaScript و jQuery) و هم در سمت سرویس دهنده (در صورتی که کاربر JavaScript را در مرورگر خود غیرفعال کرده باشد) enforce می شوند.

یک مزیت واقعی این است که شما مجبور نبودید حتی یک خط کد را در کلاس MoviesController یا در Create.cshtml view برای فعال سازی validation UI تغییر دهید. Controller و view هایی که شما پیش تر در این آموزش ایجاد کردید، به صورت خودکار قوانین اعتبارسنجی که با استفاده از validation attribute ها را روی خواص (property) کلاس movie model تعریف کردید را اخذ می کنند. Validation را با استفاده از متد Edit تست کنید، خواهید دید که دقیقا همان validation اعمال می شود.

داده های فرم تا زمانی که هیچ خطایی در اعتبارسنجی سمت سرویس گیرنده رخ نداده، به سرویس دهنده ارسال نمی شوند. این کار را می توان با قرار دادن یک نقطه ی انفصال (breakpoint) در متد HTTP POST و از طریق fiddler tool یا F12 developer tools انجام داد.




## چگونه اعتبارسنجی در Create View و متد Create Action انجام می شود

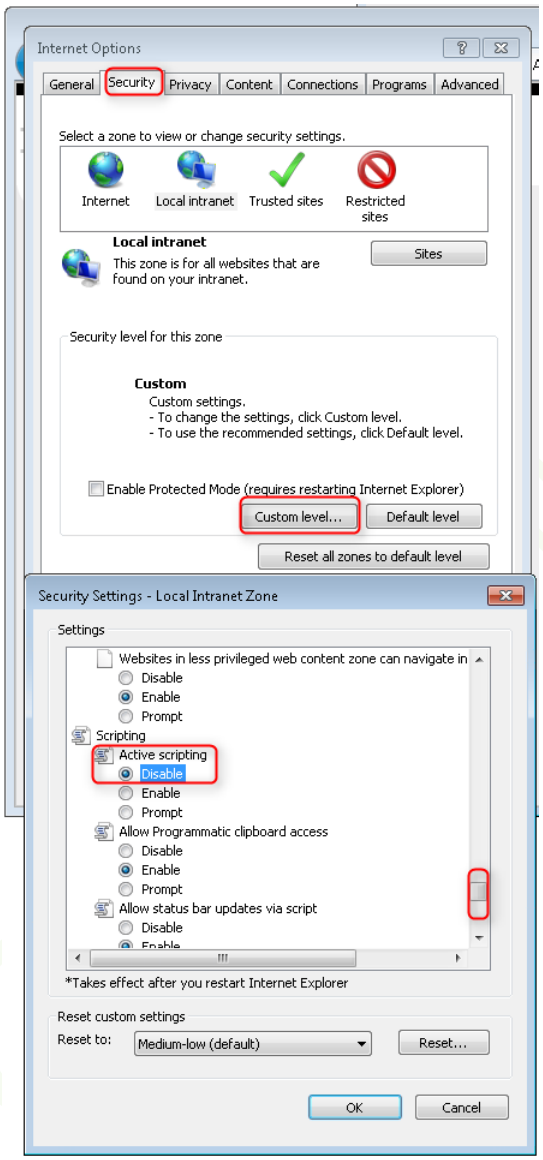
ممکن است این سوال برای شما پیش آید که چگونه validation UI بدون بروز رسانی کد (اضافه کردن کد ویژه ی اعتبارسنجی) در controller ها یا view ها، ایجاد شد. کد زیر متدهای Create در کلاس MovieController را به نمایش می گذارد. می بینید که از زمانی که برای اولین بار این متدها را کدنویسی کردید، تغییر نکرده اند.

```
public ActionResult Create()
{
  return View();
```

```
}
// POST: /Movies/Create
// To protect from overposting attacks, please enable the specific properties you want to bind to, for
// more details see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create([Bind(Include = "ID,Title,ReleaseDate,Genre,Price,Rating")] Movie movie)
{
  if (ModelState.IsValid)
  {
    db.Movies.Add(movie);
    db.SaveChanges();
    return RedirectToAction("Index");
  }
  return View(movie);

}
```

اولین متد Create (HTTP GET)، فرم Create اولیه را نمایش می دهد. دومین نسخه ([HttpPost])، ارسال فرم (form post) را مدیریت می کند. دومین متد Create (نسخه ی HttpPost) با فراخوانی ModelState.IsValid بررسی می کند آیا movie هیچ خطای اعتبارسنجی دارد یا خیر. فراخوانی متد مزبور، باعث می شود تمامی validation attribute هایی که به شی مورد نظر اعمال شده اند، ارزیابی شوند. چنانچه شی دارای خطاهای اعتبارسنجی بود، متد Create فرم را مجددا نمایش می دهد. اگر خطایی وجود نداشت، متد نام برده movie جدید را در پایگاه داده ذخیره می کند. در مثال، زمانی که در سمت سرویس گیرنده خطاهای اعتبارسنجی یافت می شوند، فرم به سرویس دهنده ارسال نمی شود؛ دومین متد Create هیچگاه صدا زده نمی شود. اگر JavaScript را در مرورگر غیرفعال کنید، اعتبارسنجی در سمت سرویس گیرنده نیز غیرفعال شده و متد HTTP POST Create، تابع ModelState.IsValid را صدا می زند و از این طریق بررسی می کند، movie خطاهای اعتبارسنجی دارد یا خیر.

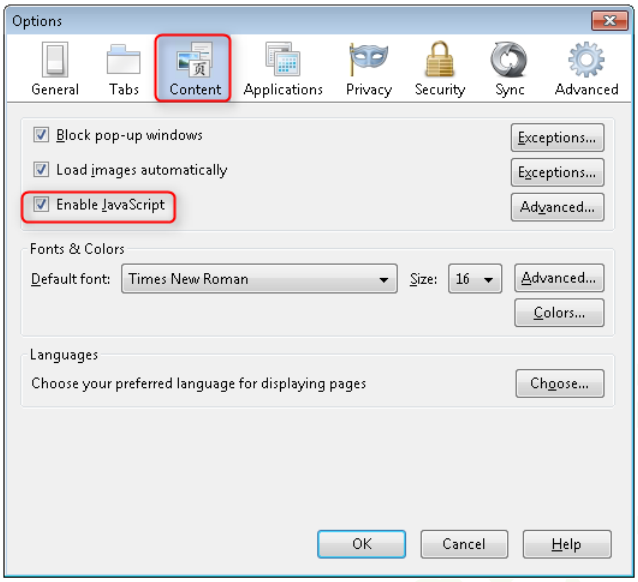
می توانید یک نقطه ی انفصال (breakpoint) در متد HttpPost Create ایجاد کرده و بدین وسیله مطمئن شوید که متد به هیچ وجه فراخوانده نمی شود، اعتبارسنجی سمت سرویس گیرنده داده های فرم را در صورت یافتن خطا، به سرویس دهنده ارسال نمی کند. اگر هم با غیرفعال سازی JavaScript در مرورگر خود، فرم را همراه با خطا ارسال کنید، نقطه ی انفصال فعال می شود، بدون کمک JavaScript اعتبارسنجی به طور کامل اجرا می شود. تصویر زیر نحوه ی غیرفعال ساختن JavaScript در مرورگر IE را نمایش می دهد:



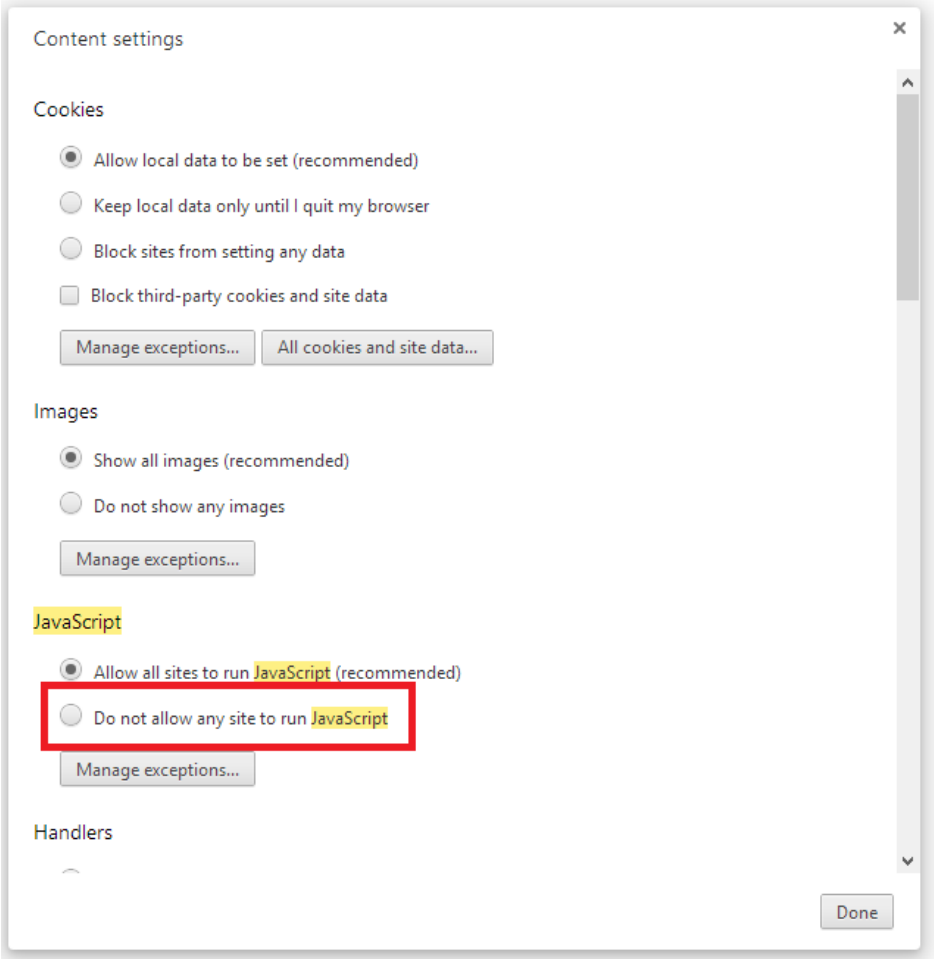

```
[HttpPost]
public ActionResult Create(Movie movie)
{
    if (ModelState.IsValid)
    {
        db.Movies.Add(movie);
        db.SaveChanges();
        return RedirectToAction("Index");
    }

    return View(movie);
}
```

ModelState.IsValid | false

تصویر زیر نحوه ی غیرفعال سازی JavaScript در مرورگر Firefox را نمایش می دهد:




تصویر زیر نحوه ی غیرفعال سازی JavaScript در مرورگر Chrome را نشان می دهد:




در زیر Create.cshtml view template که قبل تر در این درس ایجاد شد، به نمایش گذاشته شده است.

action method های فوق با استفاده از این view template، فرم اولیه را نمایش می دهند و در صورت رخداد خطا آن را مجددا نمایش می دهند.

```
@model MvcMovie.Models.Movie
@{
  ViewBag.Title = "Create";
}
<h2>Create</h2>
@using (Html.BeginForm())
{
  @Html.AntiForgeryToken()
  <div class="form-horizontal">
    <h4>Movie</h4>
    <hr />
    @Html.ValidationSummary(true)
    <div class="form-group">
```

```
        @Html.LabelFor(model => model.Title, new { @class = "control-label col-md-2" })
        <div class="col-md-10">
          @Html.EditorFor(model => model.Title)
          @Html.ValidationMessageFor(model => model.Title)
        </div>
      </div>
      @*Fields removed for brevity.*@
      <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")
}
```

همان طور که می بینید، کد بالا به وسیله ی Html.EditorFor helper، المان <input> را به ازای هر خاصیت Movie نمایش می دهد. بعد از این متد helper، متد Html.ValidationMessageFor فراخوانی می شود. این دو متد با model object ای که توسط controller به view ارسال می شود، کار می کنند. دو متد مزبور خودکار validation attribute هایی که بر روی model تعریف شده اند، یافته و پیام های خطای مناسب را نمایش می دهند.

مزیت این روش این است که نه controller و نه Create view template، هیچ چیز درباره ی قوانین اعتبارسنجی که اعمال می شوند یا درباره ی پیام های خطایی که نمایش داده می شوند، نمی داند. قوانین اعتبارسنجی و پیام های خطا تنها در کلاس Movie، تعریف می شوند. همین قوانین اعتبارسنجی به صورت خودکار به Edit view و هر view template دیگری که مایل به ایجاد آن هستید و از طریق آن می خواهید model خود را ویرایش کنید، اعمال می شوند.

اگر می خواهید منطق اعتبارسنجی را بعدا عوض کنید، می توانید این کار را یکجا با افزودن یک validation attribute به model، انجام دهید. منطق اعتبارسنجی یکجا تعریف شده و در همه ی بخش های برنامه اعمال می شود. این امر کدهای شما را مرتب نگه داشته، نگهداشت و توسعه ی آن را سهل می سازد، همچنین اصل عدم تکرار تعریف یک قابلیت یا رفتار در MVC را رعایت کرده اید.

## استفاده از خصیصه های DataType

فایل Movie.cs را باز کرده و کلاس Movie را بررسی کنید. فضای نامی System.ComponentModel.DataAnnotations علاوه بر مجموعه validation attribute های توکار، خصیصه های قالب دهی (formatting attributes) نیز ارائه می دهد.

در این مثال، از قبل یک مقدار شمارشی DataType به فیلدهای release date و price اعمال کرده ایم. کد زیر property های ReleaseDate و Price را با DataType attribute مربوطه ی آن نشان می دهد.

```
[DataType(DataType.Date)]
    public DateTime ReleaseDate { get; set; }
    [DataType(DataType.Currency)]
    public decimal Price { get; set; }
```

خصیصه های DataType تنها hint هایی را به view engine برای فرمت بندی داده ها (و خصیصه هایی همچون <a> را ویژه ی URL و <"a href="mailto:EmailAddress.com> را برای ایمیل) ارائه می کند. می توانید با استفاده از خصیصه ی RegularExpression فرمت داده ها را اعتبارسنجی کنید. خصیصه ی DataType به منظور تعریف یک نوع داده که از نوع درونی پایگاه داده خاص تر می باشد، بکار می روند، بنابراین آن ها به هیچ وجه validation attribute محسوب نمی شوند. در این مثال فقط می خواهیم حساب تاریخ را داشته باشیم، هدفمان پیگیری تاریخ و زمان نیست. DataType Enumeration مقدار شمارشی را برای تمامی نوع داده ها از قبیل Date، Time، PhoneNumber، Currency، EmailAddress فراهم می کند. خصیصه ی DataType همچنین به برنامه امکان می دهد امکانات و ویژگی های مختص به هر نوع داده را به صورت خودکار فراهم نماید. به عنوان مثال، می توان یک لینک mailto: ویژه ی DataType.EmailAddress ایجاد کرد و یک انتخاب گر تاریخ (date selector) برای DataType.Date در تمامی مرورگرهایی که از HTML5 پشتیبانی می کنند، فراهم نمود. خصیصه ی DataType، خصیصه های HTML5 data- ارائه می کند که این خصیصه ها توسط مرورگرهایی که قابلیت پشتیبانی از HTML5 را دارند، تفسیر شده و اجرا می شود. DataType هیچ داده ای را اعتبارسنجی نمی کند.

DataType.Date فرمت نمایش تاریخ را تعریف نمی کند. به صورت پیش فرض، فیلد داده ها بر اساس فرمت های پیش فرض و با توجه به CultureInfo سرویس دهنده نمایش داده می شود.

به وسیله ی خصیصه ی DisplayFormat می توان فرمت نمایش تاریخ را به صورت صریح مشخص کرد:

```
[DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]
    public DateTime EnrollmentDate { get; set; }
```

تنظیم ApplyFormatInEditMode تعیین می کند که فرمت بندی تعریف شده باید زمانی که مقدار در یک کادر متن برای ویرایش نمایش داده می شود نیز اعمال شود. (استفاده از ApplyFormatInEditMode برای تمامی فیلدها توصیه نمی شود، به عنوان مثال برای مقادیر ارزی نباید علامت ارز را در کادر متن ویرایش کرد.)

می توان خصیصه ی DisplayFormat را به تنهایی بکار برد، اما در این صورت توصیه می شود خصیصه ی DataType را نیز همراه آن استفاده کنید. خصیصه ی DataType، بجای تعریف نحوه ی ارائه و render کردن داده بر روی صفحه نمایش، semantics و نوع داده را می رساند و همچنین مزایایی فراهم می نماید که در صورت استفاده از خصیصه ی DisplayFormat دریافت نخواهید کرد:

1. مرورگر می تواند امکانات HTML5 را فعال ساخته و مورد بهره وری قرار دهد (برای مثال یک کنترل تقویم، علامت ارز متخص به آن کشور، لینک ایمیل و غیره .. را نمایش دهد).
2. به صورت پیش فرض، مرورگر داده ها را در قالب یا فرمت صحیح و مبتنی بر زبان منطقه ی (کشور و زبان) مورد نظر نمایش می دهد.
3. خصیصه ی DataType به MVC این امکان را می دهد که برای نمایش و ارائه ی داده ها، field template مناسب را برگزیند (اگر خصیصه ی DisplayFormat را به تنهایی بکار ببرید، string template را بکار می برد).

اگر از خصیصه ی DataType همراه با یک date field استفاده کنید، در آن صورت بایستی خصیصه ی DisplayFormat را نیز اضافه کنید تا فیلد مورد نظر به درستی در مرورگرهای Chrome نمایش داده شود.

نکته: اعتبارسنجی jQuery با خصیصه ی Range و DateTime قابل استفاده نمی باشد. به عنوان مثال، کد زیر همیشه یک خطای اعتبارسنجی در سمت سرویس گیرنده نمایش می دهد، حتی اگرهم تاریخ در محدوده ی زمانی مشخص شده باشد:

```
[Range(typeof(DateTime), "1/1/1966", "1/1/2020")]
```

بایستی اعتبارسنجی تاریخ jQuery را غیرفعال سازید تا بتوانید از خصیصه ی Range و DateTime با هم استفاده کنید. استفاده از hard date ها در model روش پسندیده ای نیست، از این رو پیشنهاد می کنیم تا حد امکان از بکار بردن خصیصه های Range و DateTime خودداری کنید.

```
public class Movie
    {
      public int ID { get; set; }
      [Required, StringLength(60, MinimumLength = 3)]
      public string Title { get; set; }
      [Display(Name = "Release Date"), DataType(DataType.Date)]
      public DateTime ReleaseDate { get; set; }
      [Required]
      public string Genre { get; set; }
      [Range(1, 100), DataType(DataType.Currency)]
      public decimal Price { get; set; }
      [Required, StringLength(5)]
      public string Rating { get; set; }
     }
```