بسم اللّه الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

بررسی Edit View و متدهای Edit

مدرس : مهندس افشین رفوآ

دوره آموزش MVC

بررسی Edit View و متدهای Edit

در این فصل، view ها و action method های Edit ایجاد شده برای movie controller را بررسی خواهیم کرد. اما اول به بهبود "Release Date" می پردازیم. فایل Models\Movie.cs را باز کرده و خط کدهای رنگی شده ی زیر را به آن اضافه کنید:

```
using System;
using System.ComponentModel.DataAnnotations;
using System.Data.Entity;
Namespace MvcMovie.Models
{
  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 class MovieDBContext : DbContext
  {
    public DbSet<Movie> Movies { get; set; }
  }
```

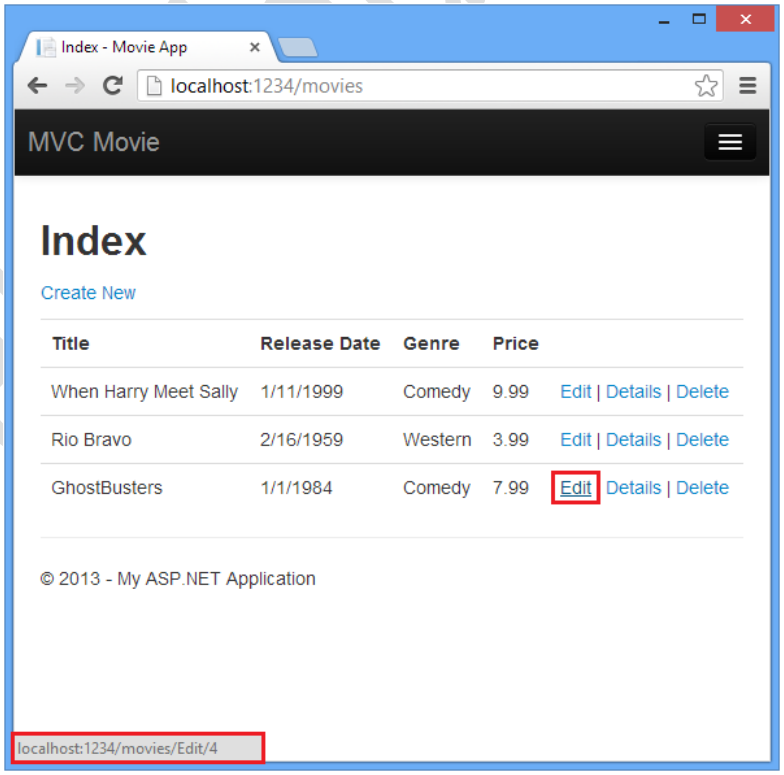
```
}
```

همچنین می توان date culture (تنظیمات تاریخ ویژه ی زبان آن کشور) را اختصاصی (تنظیم) کرد:

```
Display(Name = "Release Date")]
[DataType(DataType.Date)]
[DisplayFormat(DataFormatString = "{0:d}", ApplyFormatInEditMode = true)]
public DateTime ReleaseDate { get; set; }
```

درباره ی DataAnnotation در مبحث بعدی توضیحاتی را ارائه خواهیم داد. خصیصه ی Display مشخص می کند چه چیزی به عنوان اسم فیلد مورد نظر نمایش داده شود (در این نمونه بجای "ReleaseDate" نمایش دهد: "Release Date"). خصیصه ی DataType نوع داده، که در این مورد از نوع date هست، را مشخص می کند تا اطلاعات مربوط به زمان ذخیره شده در فیلد نمایش داده نشود. خصیصه ی DisplayFormat برای رفع خطایی که در مرورگر chrome وجود دارد و فرمت های تاریخ را به صورت نادرست render می کند، مورد نیاز می باشد.

برنامه را اجرا کرده و به Movie controller مراجعه (browse) کنید. حال مکان نمای موس را روی لینک Edit قرار داده تا URL ای که به آن لینک می شود، مشاهده کنید.



لینک Edit توسط متد Html.ActionLink در view Views\Movies\Index.cshtml ایجاد می گردد:

```
@Html.ActionLink("Edit", "Edit", new { id = item.ID })
```

```
<td>
    @Html.ActionLink("Edit Me", "Edit", new { id=item.ID }) |
    @Html.
    @Html.
```

(extension) MvcHtmlString HtmlHelper.ActionLink(string linkText, string actionName, object routeValues)
Returns an anchor element (a element) that contains the virtual path of the specified action.

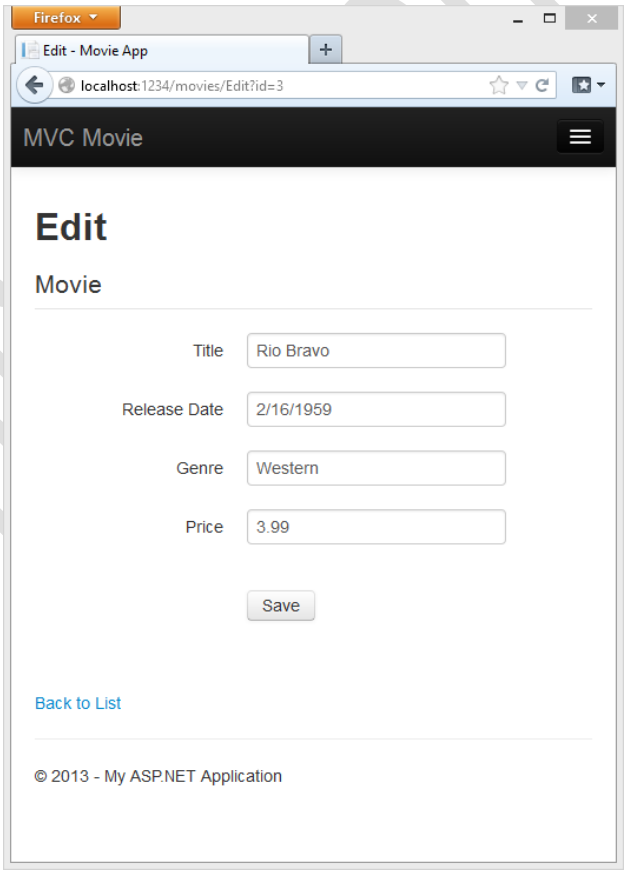
Exceptions:
System.ArgumentException

شی Html یک helper است که با استفاده از یک خاصیت بر روی کلاس پایه ی System.Web.Mvc.WebViewPage، توابع و خاصیت های کلاس نام برده را برای استفاده در دسترس قرار می دهد. متد ActionLink شی Html، فرایند تولید پویا (dynamic) لینک های HTML که به action method های روی controller متصل (لینک) می شوند را سهل می سازد. اولین آرگومان ارسال شده به متد ActionLink، متن لینک را مشخص می کند (برای مثال <a>Edit Me). دومین آرگومان ارسال شده به متد نام برده، اسم action method ای است که می خواهید فراخوانی شود (به عنوان مثال، Edit action). آخرین آرگومان یک شی تعریف نشده (anonymous object) است که route data را ارائه می دهد (در این مثال ID 4).

لینک ایجاد شده که در تصویر قبلی نمایش داده شد، http://localhost:1234/Movies/Edit/4 می باشد. مسیر (route) پیش فرض (تعریف و ایجاد شده در فایل App_Start\RouteConfig.cs) الگوی URL، {controller}/{action}/{id} را دنبال می کند.از این رو ASP.NET، http://localhost:1234/Movies/Edit/4 را به یک درخواست به متد Edit کنترلر movie که پارامتر ID آن برابر با 4 می باشد ترجمه می کند. کد زیر گرفته شده از فایل App_Start\RouteConfig.cs را مورد بررسی قرار دهید. متد MapRoute به منظور هدایت یا مسیردهی درخواست های HTTP به controller و action method مربوطه و نیز ارائه نمودن پارامتر اختیاری ID بکار می رود. متد مذکور همچنین توسط html helper ها همچون ActionLink به منظور ایجاد URL هایی که در آن controller، action method و route data تعریف شده باشد، استفاده می شود.

```
public static void RegisterRoutes(RouteCollection routes)
{
  routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
  routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Home", action = "Index",
      id = UrlParameter.Optional }
  );
}
```

می توان پارامترهای action method را توسط query string نیز ارسال نمود. به عنوان مثال، آدرس URL " http://localhost:1234/Movies/Edit?ID=3 " پارامتر ID که برابر با 3 می باشد را به متد Edit کنترلر Movies ارسال می کند.



کنترلر Movies را باز کنید. دو متد Edit در زیر نمایش داده شده اند:

```
// GET: /Movies/Edit/5
public ActionResult Edit(int? id)
{
  if (id == null)
  {
    return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
  }
  Movie movie = db.Movies.Find(id);
  if (movie == null)
  {
    return HttpNotFound();
  }
  return View(movie);
}
// POST: /Movies/Edit/5
// 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 Edit([Bind(Include = "ID,Title,ReleaseDate,Genre,Price")] Movie movie)
{
  if (ModelState.IsValid)
  {
    db.Entry(movie).State = EntityState.Modified;
    db.SaveChanges();
    return RedirectToAction("Index");
  }
  return View(movie);
}
```

دقت داشته باشید که دومین متد Edit پس از خصیصه ی HttpPost قرار گرفته است. این متد مشخص می کند که overload متد Edit تنها ویژه ی درخواست های POST فراخوانی می شود. می توان خصیصه ی HttpGet را به اولین متد edit اعمال کرد، اما از آنجایی که به صورت پیش فرض اعمال می شود، لزومی به انجام این کار نیست. (به action methods هایی که خصیصه ی HttpGet به آن ها به صورت ضمنی تخصیص می یابد، متدهای HttpGet می گوییم.) خصیصه ی Bind یک سازوکار مهم امنیتی دیگر است که مانع از over-post کردن داده ها توسط هکرها به model شما می شود. شما باید خاصیت های مورد نظر را در خصیصه ی bind ای که می خواهید تغییر دهید، اضافه کنید. در model ساده ای که در این آموزش مورد استفاده قرار می گیرد، تمامی داده ها را در model، bind می کنیم. خصیصه ی

ValidateAntiForgeryToken به منظور جلوگیری از جعل درخواست بکار رفته و با تابع

@Html.AntiForgeryToken() در فایل edit view، (Views\Movies\Edit.cshtml) ست می شود،

بخشی از آن را زیر مشاهده می کنید:

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

@Html.AntiForgeryToken() یک نوع پنهان از token ضد جعل ایجاد می کند که باید در متد Edit کنترلر Movies، match شود.

متد HttpGet Edit پارامتر movie ID را گرفته، سپس با استفاده از متد Find تکنولوژی EF، movie را می یابد، همچنین movie انتخابی را به Edit view برمی گرداند. در صورت یافت نشدن movie، HttpNotFound بازگردانده می شود. به هنگام ایجاد Edit view، سیستم scaffolding کلاس Movie را مورد بررسی قرار داده و کدهایی برای render کردن المان های <label> و <input> هر یک از خاصیت های کلاس مزبور، ایجاد کرد. مثال زیر Edit view ای را به نمایش می گذارد که توسط سیستم scaffolding محیط ویژوال ایجاد شده است:

```
@model MvcMovie.Models.Movie
@{
  ViewBag.Title = "Edit";
```

```
}
<h2>Edit</h2>
@using (Html.BeginForm())
{
  @Html.AntiForgeryToken()
  <div class="form-horizontal">
    <h4>Movie</h4>
    <hr />
    @Html.ValidationSummary(true)
    @Html.HiddenFor(model => model.ID)

    <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>
    <div class="form-group">
      @Html.LabelFor(model => model.ReleaseDate, new { @class = "control-label col-md-2" })
      <div class="col-md-10">
        @Html.EditorFor(model => model.ReleaseDate)
        @Html.ValidationMessageFor(model => model.ReleaseDate)
      </div>
    </div>
    @*Genre and Price removed for brevity.*@
    <div class="form-group">
      <div class="col-md-offset-2 col-md-10">
        <input type="submit" value="Save" class="btn btn-default" />
      </div>
    </div>
  </div>
}
<div>
  @Html.ActionLink("Back to List", "Index")
</div>
@section Scripts {
  @Scripts.Render("~/bundles/jqueryval")
}
```

مشاهده می کنید که دستور @model MvcMovie.Models.Movie در بالای فایل درج گردیده است –

دستور ذکر شده به این حقیقت اشاره دارد که view مورد نظر انتظار دارد مدل view template حتما از نوع

Movie باشد.

کدهایی که توسط scaffolding ایجاد شده با بهره گیری از چندین helper method، نشانه گذاری های (markup) HTML را ساده و در عین حال کارآمد تر می سازد. Html.LabelFor، اسم فیلد را نمایش می دهد ("Title"، "Release"، "Genre" یا "Price"). خاصیت Html.EditorFor برابر با همان تگ <input> html می باشد. Html.ValidationMessageFor نیز تمامی پیام های اعتبارسنجی مربوط به آن خاصیت را نمایش می دهد.

برنامه را اجرا کرده و به آدرس /Movies پیمایش کنید. بر روی لینک Edit کلیک نمایید، سپس source صفحه ی مورد نظر را در پنجره ی مرورگر مشاهده کنید. HTML المان form در زیر نمایش داده شده:

```
<form action="/movies/Edit/4" method="post">
  <input name="__RequestVerificationToken" type="hidden" value="UxY6bkQyJCXO3Kn5AXg-
6TXxOj6yVBi9tghHaQ5Lq_qwKvcojNXEEfcbn-FGh_0vuw4tS_BRk7QQQHlJp8AP4_X4orVNoQnp2cd8kXhykS01" />
<fieldset class="form-horizontal">
    <legend>Movie</legend>
    <input data-val="true" data-val-number="The field ID must be a number." data-val-required="The ID field is
required." id="ID" name="ID" type="hidden" value="4" />
    <div class="control-group">
      <label class="control-label" for="Title">Title</label>
      <div class="controls">
        <input class="text-box single-line" id="Title" name="Title" type="text" value="GhostBusters" />
        <span class="field-validation-valid help-inline" data-valmsg-for="Title" data-valmsg-replace="true"></span>
      </div>
    </div>
    <div class="control-group">
      <label class="control-label" for="ReleaseDate">Release Date</label>
      <div class="controls">
        <input class="text-box single-line" data-val="true" data-val-date="The field Release Date must be a date." data-
val-required="The Release Date field is required." id="ReleaseDate" name="ReleaseDate" type="date"
value="1/1/1984" />
        <span class="field-validation-valid help-inline" data-valmsg-for="ReleaseDate" data-valmsg-
replace="true"></span>
      </div>
    </div>
    <div class="control-group">
      <label class="control-label" for="Genre">Genre</label>
      <div class="controls">
        <input class="text-box single-line" id="Genre" name="Genre" type="text" value="Comedy" />
        <span class="field-validation-valid help-inline" data-valmsg-for="Genre" data-valmsg-replace="true"></span>
      </div>
    </div>
    <div class="control-group">
      <label class="control-label" for="Price">Price</label>
      <div class="controls">
```

```
        <input class="text-box single-line" data-val="true" data-val-number="The field Price must be a number." data-
val-required="The Price field is required." id="Price" name="Price" type="text" value="7.99" />
        <span class="field-validation-valid help-inline" data-valmsg-for="Price" data-valmsg-replace="true"></span>
      </div>
    </div>
    <div class="form-actions no-color">
      <input type="submit" value="Save" class="btn" />
    </div>
  </fieldset>
</form>
```

تگ های <input> داخل یک المان <form> قرار دارند که خصیصه ی action آن بر روی مقدار /Movies/Edit تنظیم شده و اطلاعات آن را به آدرس مزبور ارسال (post) می کند. به مجرد کلیک بر روی دکمه ی Save، داده های form به سرور ارسال (post) می شود. خط دوم token مخفی XSRF که با فراخوانی متد @Html.AntiForgeryToken() ایجاد شده را نشان می دهد.

پردازش درخواست POST

تکه کد زیر نسخه ی HttpPost متد Edit را نمایش می دهد.

```
    [HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit([Bind(Include="ID,Title,ReleaseDate,Genre,Price")] Movie movie)
{
  if (ModelState.IsValid)
  {
    db.Entry(movie).State = EntityState.Modified;
    db.SaveChanges();
    return RedirectToAction("Index");
  }
  return View(movie);
}
```

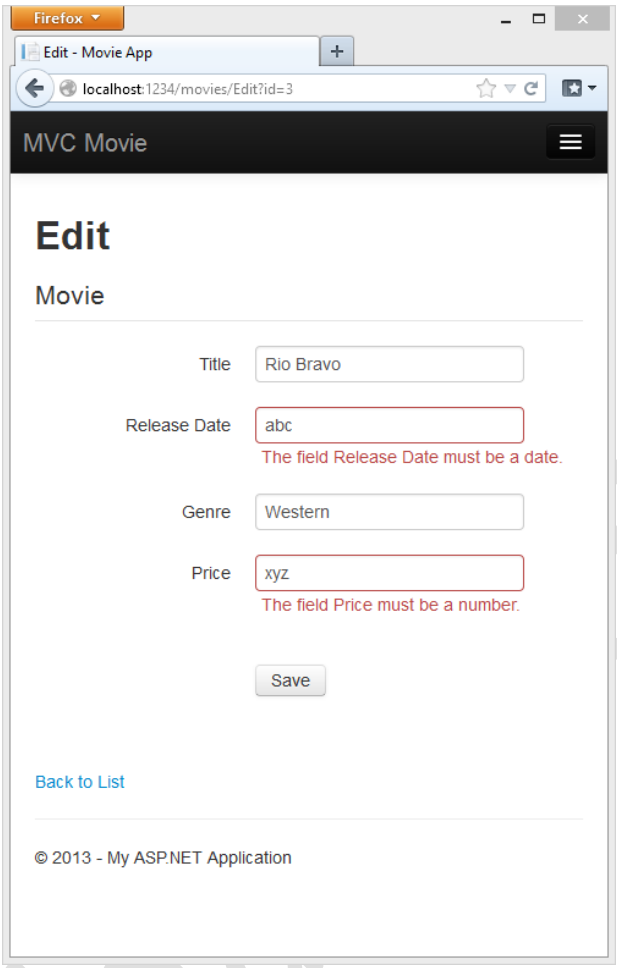
خصیصه ی ValidateAntiForgeryToken، XSRF token که با فراخوانی متد @Html.AntiForgeryToken() در view ایجاد شده را اعتبارسنجی می کند.

ASP.NET MVC model binder مقادیر المان form که ارسال شده اند را دریافت کرده و یک شی Movie ایجاد می کند. این شی به عنوان پارامتر movie ارسال می شود. متد ModelState.IsValid بررسی می کند آیا اطلاعات ارائه شده به فرم می تواند به منظور اصلاح (بروز رسانی یا ویرایش) شی movie بکار رود یا خیر.

در صورت معتبر بودن داده ها، اطلاعات مربوط به movie در مجموعه ی Movies نمونه ی پایگاه داده (MovieDBContext) ذخیره می شود. اطلاعات جدید movie با فراخوانی متد SaveChanges پایگاه داده ی MovieDBContext، در آن (MovieDBContext db) ذخیره می شوند. پس از ذخیره سازی داده ها، کد کاربر را به متد Index کلاس MoviesController هدایت (redirect) می کند. این کد مجموعه ی movie ها را بعلاوه ی تغییراتی که اخیرا اعمال شده، برای کاربر نمایش می دهد.

به محض اینکه اعتبارسنجی سمت سرویس گیرنده (client side) تشخیص دهد که مقادیر فیلد معتبر و مجاز نیستند، یک پیغام خطا نمایش داده می شود. اگر شما JavaScript را غیرفعال کنید، اعتبار سنجی در سمت سرویس گیرنده رخ نمی دهد، با این وجود سرور درمی یابد که مقادیر ارسال شده مجاز نیستند، در پی آن مقادیر فرم همراه با پیام های خطا مجددا نمایش داده می شوند.

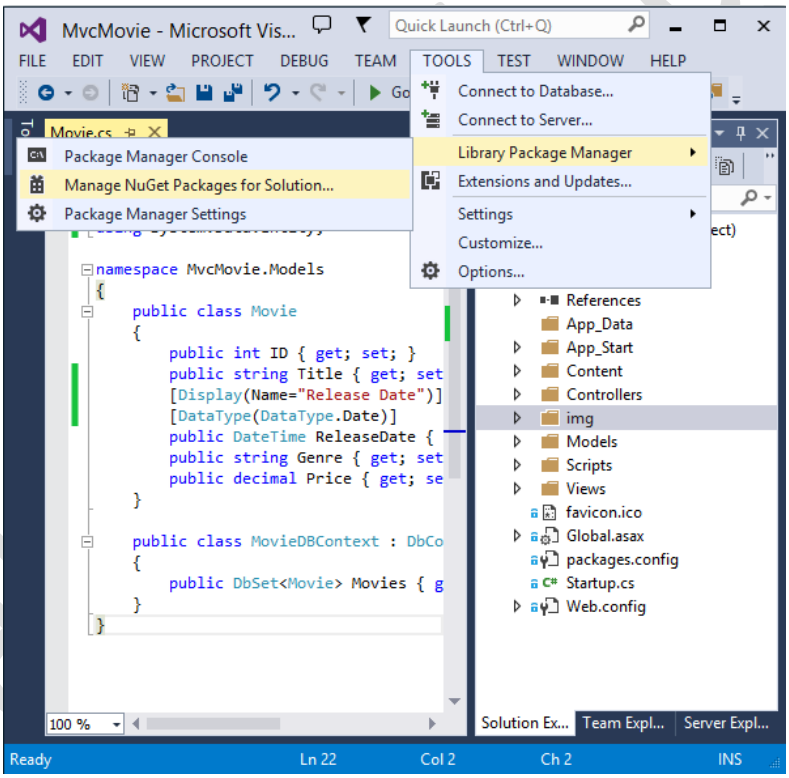
وظیفه ی نمایش پیام های خطای مناسب بر عهده ی helper های Html.ValidationMessageFor موجود در Edit.cshtml view template می باشد.



تمامی متدهای HttpGet، الگویی مشابه را دنبال می کنند. متدهای نام برده یک شی movie (یا در صورت استفاده از متد Index فهرستی از اشیا) دریافت کرده و model را به view ارسال می کند. متد create یک شی خالی movie به Create view پاس می دهد. کلیه ی متدهایی که داده ایجاد، ویرایش، حذف یا به هر شکل دیگری اصلاح می کنند، این کار را در نسخه ی overload شده ی خصیصه ی HttpPost متد انجام می دهند. اصلاح داده ها در متد HTTP GET یک شکاف امنیتی ایجاد می کند. اصلاح داده ها در متد GET همچنین قوانین پروتکل HTTP و الگوی معماری REST (مدل معماری برای طراحی برنامه‌های کاربردی شبکه است) را نقض می کند. HTTP و REST مشخص می کنند که درخواست های GET تحت هیچ شرایطی نباید وضعیت برنامه (application state) را تغییر دهند. به عبارتی دیگر، اجرای عملیات GET باید امن باشد، هیچ اثرات جانبی نداشته باشد و داده های ماندگار شما را تغییر ندهد.

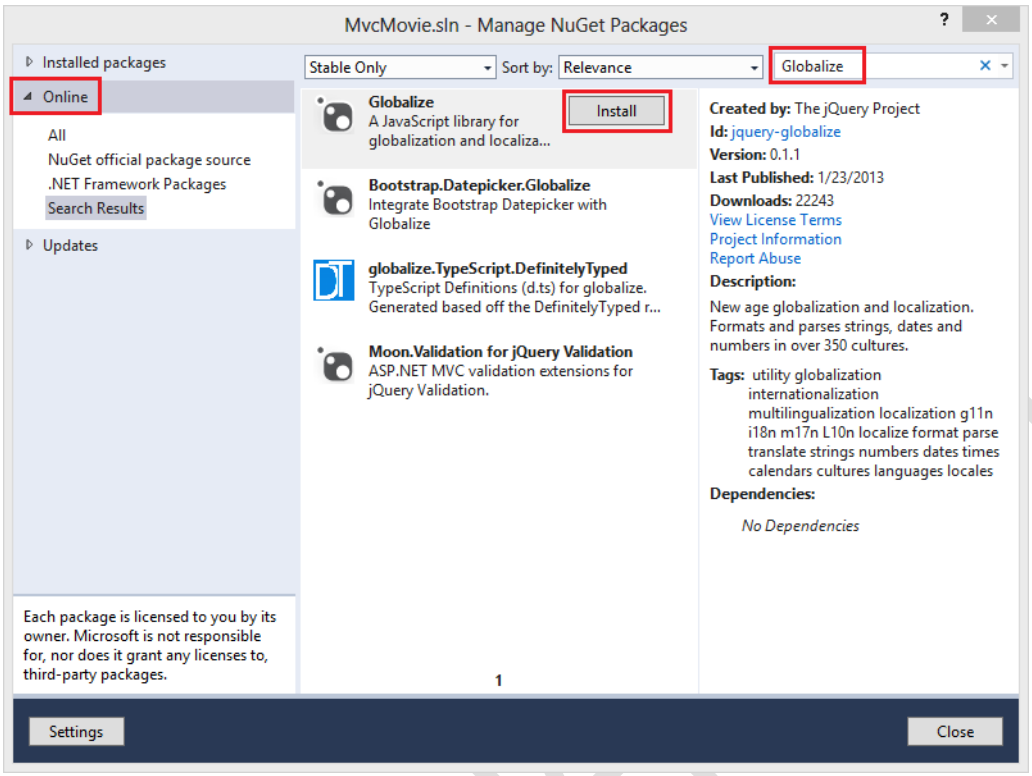
توجه: برای فراهم آوردن امکان پشتیبانی از اعتبارسنجی jQuery ویژه ی زبان های غیر از انگلیسی آمریکایی که از " , " به عنوان نقطه ی ممیز اعشار استفاده می کنند و همچنین فرمت های نمایش تاریخ غیر از انگلیسی آمریکایی، بایستی فایل های globalize.js و cultures/globalize.cultures.js ویژه ی خود را از آدرس https://github.com/jquery/globalize دانلود کنید و نیز JavaScript را برای استفاده از Globalize.parseFloat لحاظ نمایید. می توانید اعتبارسنجی غیر از زبان انگلیسی آمریکایی jQuery را از NuGet دریافت کنید. (در صورت استفاده از زبان انگلیسی، لازم به نصب Globalize نیست.)

1. از منو Tools ورودی Library Package Manager را انتخاب کرده، سپس Manage NuGet Packages for Solution را انتخاب نمایید.



2. از کادر سمت چپ، Online را انتخاب کنید.

3. در کادر دریافت ورودی Search Installed packages، واژه ی Globalize را وارد نمایید.



4. حال Install را کلیک کنید. فایل Scripts\jquery.globalize\globalize.js به پروژه ی شما اضافه می شود. پوشه ی Scripts\jquery.globalize\cultures\ دربردارنده ی فایل های JavaScript ویژه ی زبان های(culture) متعددی است. نصب این پکیج ممکن است 5 دقیقه طول بکشد.

کد زیر تغییرات اعمال شده به فایل Views\Movies\Edit.cshtml را نشان می دهد:

```
@section Scripts {
  @Scripts.Render("~/bundles/jqueryval")
  <script src="~/Scripts/globalize/globalize.js"></script>
  <script
src="~/Scripts/globalize/cultures/globalize.culture.@(System.Threading.Thread.CurrentThread.CurrentCulture.Nam
e).js"></script>
  <script>
  $.validator.methods.number = function (value, element) {
    return this.optional(element) ||
      !isNaN(Globalize.parseFloat(value));
  }
  $(document).ready(function () {
    Globalize.culture('@(System.Threading.Thread.CurrentThread.CurrentCulture.Name)');
  });
  </script>
  <script>
```

```
    jQuery.extend(jQuery.validator.methods, {
      range: function (value, element, param) {
        //Use the Globalization plugin to parse the value
        var val = Globalize.parseFloat(value);
        return this.optional(element) || (
          val >= param[0] && val <= param[1]);
      }
    });
    $.validator.methods.date = function (value, element) {
      return this.optional(element) ||
        Globalize.parseDate(value) ||
        Globalize.parseDate(value, "yyyy-MM-dd");
    }
  </script>
}
```

برای اینکه مجبور به تکرار این کد در هر Edit view نشوید، می توانید این کد را به فایل layout انتقال دهید.

به عنوان یک راه حل موقت، چنانچه موفق به فعال سازی اعتبارسنجی در زبان خود نشدید، می توانید از زبان انگلیسی آمریکایی استفاده کنید و یا JavaScript را در مرورگر خود غیرفعال سازید. برای اینکه کامپیوتر خود را مجبور به استفاده از انگلیسی آمریکایی کنید، می توانید المان globalization را به root file های web.config پروژه ی خود اضافه نمایید. نمونه کد زیر المان globalization را نشان می دهد که زبان (culture) آن بر روی United States English تنظیم شده است.

```
<system.web>
  <globalization culture ="en-US" />
  <!--elements removed for clarity-->
 </system.web>
```