Adding Validation

Keeping Things DRY

One of the core design tenets of ASP.NET MVC is  [DRY](http://en.wikipedia.org/wiki/Don't_repeat_yourself) ("Don't Repeat  Yourself"). ASP.NET MVC encourages you to specify functionality or behavior only  once, and then have it be reflected everywhere in an application. This reduces  the amount of code you need to write and makes the code you do write less error  prone and easier to maintain.

The validation support provided by ASP.NET MVC and Entity Framework Code  First is a great example of the DRY principle in action. You can declaratively  specify validation rules in one place (in the model class) and the rules  are enforced everywhere in the application.

Let's look at how you can take advantage of this validation support in the  movie application.

## Step 1: Adding Validation Rules to the Movie Model

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]\*$")]

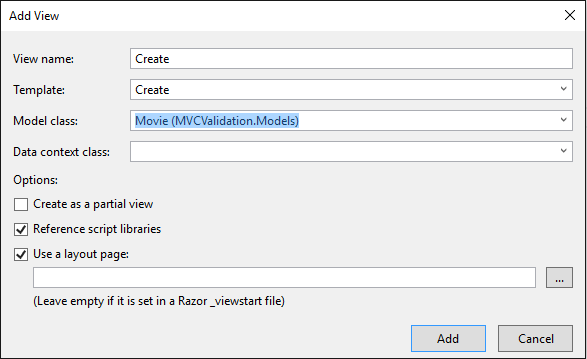
    [StringLength(5)]

    public string Rating { get; set; }

}

Step 2: Add view

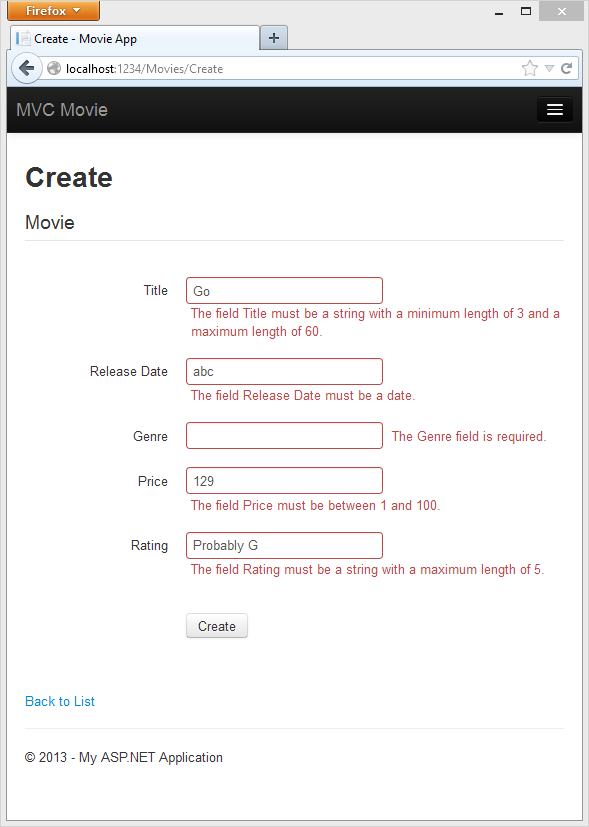
Create a view as given below figure



## Step 3: Validation Error UI in ASP.NET MVC

Run the application and navigate to the /Movies URL.

Click the **Create New** link to add a new movie. Fill out the  form with some invalid values. As soon as jQuery client side  validation detects the error, it displays an error message.



## Step 4: How Validation Occurs in the Create View and Create Action Method

You might wonder how the validation UI was generated without any updates to  the code in the controller or views. The next listing shows what the Create methods in the MovieController class look like.  They're unchanged from how you created them earlier in this tutorial.

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");

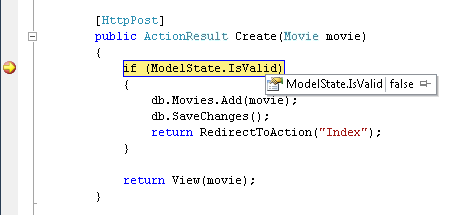
    }

    return View(movie);

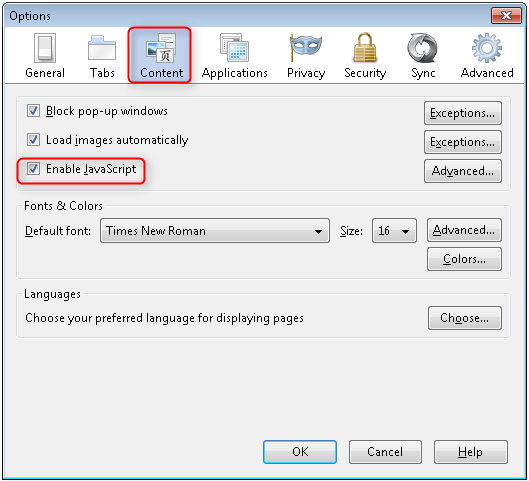
}

The first (HTTP GET) Create action method displays the initial Create form. The second  ([HttpPost]) version handles  the form post. The second Create method (The HttpPost version) calls ModelState.IsValid to check whether the movie has any validation errors.  Calling this method evaluates any validation attributes that have been applied  to the object. If the object has validation errors, the Create  method re-displays the form. If there are no errors, the method saves the new  movie in the database. In our movie example, **the form is not posted  to the server when there are validation errors detected on the client side;  the second Create method is never called**. If you disable JavaScript  in your browser, client validation is disabled and the HTTP POST Create method calls [ModelState.IsValid](http://msdn.microsoft.com/en-us/library/system.web.mvc.modelstatedictionary.isvalid.aspx) to check whether the movie has any validation errors.

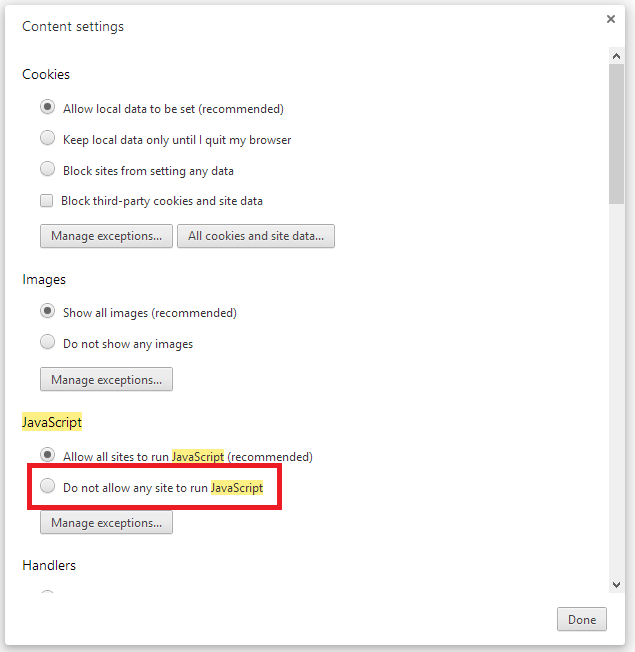
You can set a break point in  the HttpPost Create method  and verify the method is never called, client side validation will not submit  the form data when validation errors are detected. If you disable JavaScript in  your browser, then submit the form with errors, the break point will be hit.  You still get full validation without JavaScript. The following image shows how to disable JavaScript in  Internet Explorer.



The following image shows how to disable JavaScript in the FireFox browser.



The following image shows how to disable JavaScript in the Chrome browser.



Below is the Create.cshtml view template that you scaffolded earlier  in the tutorial. It's used by the action methods shown above both to display the  initial form and to redisplay it in the event of an error.

@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")

}