

# .NET Framework: Developing Modern Web Apps with ASP.NET MVC – Workshop*PLUS*

Wael Kdoun

Senior Consultant

v1.0

# Module 7: Validation

## Module Overview

## Module 7: Validation

### Section 1: Validation Fundamentals

#### Lesson: Overview

# What is Validation?

# Validation

- Validating user inputs and enforcing business rules/logic is a core requirement of most web applications
- Server-side validation
  - Should be done with or without client-side validation
  - Model Validation with Data Annotations
- Client-side validation
  - Unobtrusive validation
  - Extending
  - Remote
- “Don’t Repeat Yourself”

# Data Annotations

- The attribute is declared on the server-side property via metadata
- Built-in validation attributes

Attribute	Description
CompareAttribute	Compares the value of two model properties. Validation succeeds if they are equal
RemoteAttribute	Leverages jQuery Validate to call an action on the server to perform server-side validation with AJAX
RequiredAttribute	Indicates that a value is required
RangeAttribute	Indicates the numeric range constraints for the field value
RegularExpressionAttribute	A data field value must match the specified
StringLengthAttribute	Specifies the maximum string length

# Range Attribute

Example: Range Attribute in Model Metadata

```
[Range(1, 5)]  
public int Rating { get; set; }
```

# Range Attribute Rendered Output

- Example: HTML rendered in View with jQuery unobtrusive validation attributes

```
<input class="text-box single-line" data-val="true" data-val-  
number="The field Rating must be a number." data-val-range="The  
field Rating must be between 1 and 5." data-val-range-max="5"  
data-val-range-min="1" data-val-required="The Rating field is  
required." id="Rating" name="Rating" type="number" value="" />
```

- Example: HTML rendered script references



# Data Annotations & ModelState

```
[HttpPost]
public ActionResult Edit(Game game)
{
    if (ModelState.IsValid)
    {
        db.Entry(game).State = EntityState.Modified;
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    return View(game);
}
```

```
[HttpPost]
public ActionResult Create(Game game)
{
    if (ModelState.IsValid)
    {
        try
        {
            db.Games.Add(game);
            db.SaveChanges();
            return RedirectToAction("Index");
        }
        catch (DbUpdateException ex)
        {
            ModelState.AddModelError("", ex.Message);
        }
    }

    return View(game);
}
```

```
[HttpPost]
public ActionResult Create(Game game)
{
    if (ModelState.IsValid)
    {
        db.Games.Add(game);
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    return View(game);
}
```

# Validation

- ValidationMessage
- ValidationSummary

```
@Html.ValidationMessageFor(model => model.Rating)
```

```
@Html.ValidationMessage("GameName", "some message")
```

```
@Html.ValidationSummary()
```

# Remote Attribute

```
[Remote("IsGameNameUnique", "Games", AdditionalFields = "GameId", ErrorMessage = "Game Name  
must be a unique name!")]  
    public string GameName { get; set; }
```

```
public ActionResult IsGameNameUnique(string gameName, int? gameId)  
{  
    var game = db.Games.FirstOrDefault(o => o.GameName == gameName);  
    if (game == null)  
        return Json(true, JsonRequestBehavior.AllowGet);  
  
    return Json(game.GameId == gameId, JsonRequestBehavior.AllowGet);  
}
```

# Validation != Security

- Include List

```
// include list
[HttpPost]
public ViewResult Edit([Bind(Include = "GameName")]
Game game)
{
    // ...
}
```

- Bind against interface

```
[HttpPost]
public ActionResult Create(Game game)
{
    if (TryUpdateModel<IGameModel>(game))
    {
```

- Use ViewModel (Model-View-ViewModel (MVVM))

```
[HttpPost]
public ActionResult Create(GameViewModel game)
{
```

# Custom Attributes

- Custom Attribute with Client-Side Validation

```
public class UrlValidAttribute : ValidationAttribute, IClientValidatable
{
    public override bool IsValid(object value)
    {
        if (value == null || ((string)value).ToLowerInvariant().Contains("microsoft"))
            return false;
        return true;
    }

    public IEnumerable<ModelClientValidationRule>
        GetClientValidationRules(ModelMetadata metadata, ControllerContext context)
    {
        yield return new ModelClientValidationRule
        {
            ErrorMessage = this.ErrorMessage,
            ValidationType = "urlvalid"
        };
    }
}
```

# Custom Validation

- Custom Client-Side Validation - In Views

```
@section Scripts {  
    @Scripts.Render("~/bundles/jqueryval")  
    <script type="text/javascript">  
        // -- jQuery validation method  
        jQuery.validator.addMethod('urlvalidCheck', function (value, element, params) {  
            return (!/microsoft/.test(value));  
        }, '');  
  
        // add the unobtrusive adapter  
        jQuery.validator.unobtrusive.adapters.add('urlvalid', {}, function (options) {  
            options.rules['urlvalidCheck'] = true;  
            options.messages['urlvalidCheck'] = options.message;  
        });  
    </script>  
}
```

# Client-side Validation

- Built-in jQuery validation methods

Validation Method	Description
minlength( length ) Returns: Boolean	Makes the element require a given minimum length
maxlength( length ) Returns: Boolean	Makes the element require a given maximum length
min( value ) Returns: Boolean	Makes the element require a given minimum
max( value ) Returns: Boolean	Makes the element require a given maximum
email( ) Returns: Boolean	Makes the element require a valid email
url( ) Returns: Boolean	Makes the element require a valid URL
dateISO( ) Returns: Boolean	Makes the element require a ISO date
number( ) Returns: Boolean	Makes the element require a decimal number
digits( ) Returns: Boolean	Makes the element require digits only
creditcard( ) Returns: Boolean	Makes the element require a creditcard number
accept( extension ) Returns: Boolean	Makes the element require a certain file extension
equalTo( other ) Returns: Boolean	Is Equal To

# DataType Attribute

- Use **DataType** Attribute to leverage the existing jQuery validators, or add them to custom client validation rules by name

```
[DataType(DataType.CreditCard)]  
public string CreditCard { get; set; }  
  
[DataType(DataType.EmailAddress)]  
public string Email { get; set; }  
  
[DataType(DataType.Url)]  
public string Url { get; set; }
```



# Handling Validation Errors in Web API

- Web API does not automatically return an error to the client when validation fails
- Use the controller action to check for model state and respond appropriately through HTTP.

```
[HttpPost]
public void CreateTodoItem([FromBody] TodoItem item)
{
    if (!ModelState.IsValid)
    {
        HttpContext.Response.StatusCode = 400;
    }
}
```

# Demo: Validation

## Module 7: Validation

### Section 2: Don't Repeat Yourself Principle

#### Lesson: Example Scenario

Don't Repeat Yourself!

# 1. Define a Model

## 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)]
    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; }
}
```

## 2. Generated DB Schema

The screenshot displays the SQL Server Enterprise Designer interface for a table named `dbo.Movie`. The top pane shows the table's design with columns, data types, and nullability. The bottom pane shows the corresponding T-SQL script.

Name	Data Type	Allow Nulls
ID	int	<input type="checkbox"/>
Genre	nvarchar(MAX)	<input checked="" type="checkbox"/>
Price	decimal(18,2)	<input type="checkbox"/>
ReleaseDate	datetime2(7)	<input type="checkbox"/>
Title	nvarchar(MAX)	<input checked="" type="checkbox"/>
Rating	nvarchar(MAX)	<input checked="" type="checkbox"/>

**Keys (1)**  
PK\_Movie (Primary Key, Clustered)

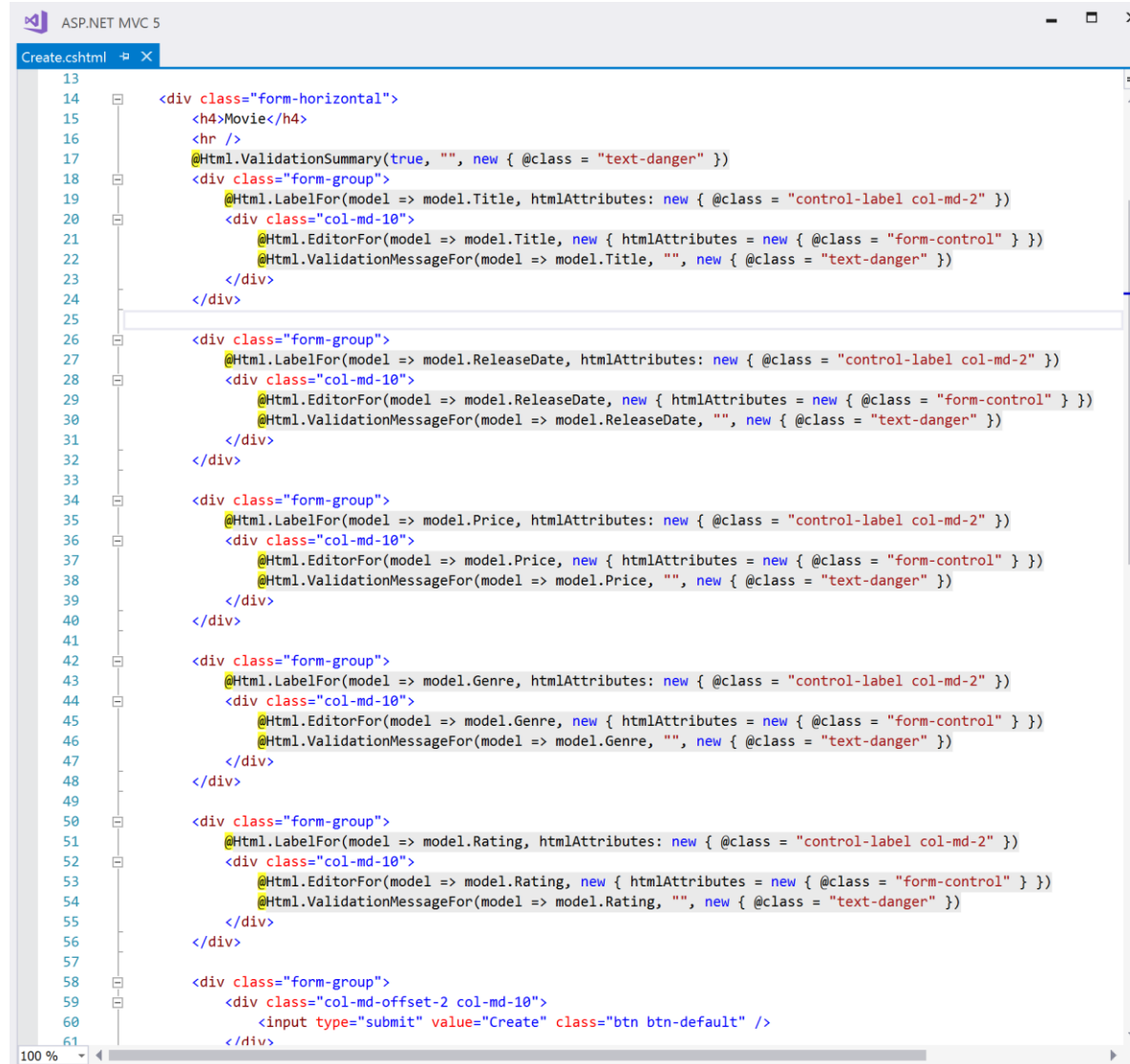
**Check Constraints (0)**  
**Indexes (0)**  
**Foreign Keys (0)**  
**Triggers (0)**

```
1 CREATE TABLE [dbo].[Movie] (  
2     [ID] INT IDENTITY (1, 1) NOT NULL,  
3     [Genre] NVARCHAR (MAX) NULL,  
4     [Price] DECIMAL (18, 2) NOT NULL,  
5     [ReleaseDate] DATETIME2 (7) NOT NULL,  
6     [Title] NVARCHAR (MAX) NULL,  
7     [Rating] NVARCHAR (MAX) NULL,  
8     CONSTRAINT [PK_Movie] PRIMARY KEY CLUSTERED ([ID] ASC)  
9 );  
10  
11
```

100 %

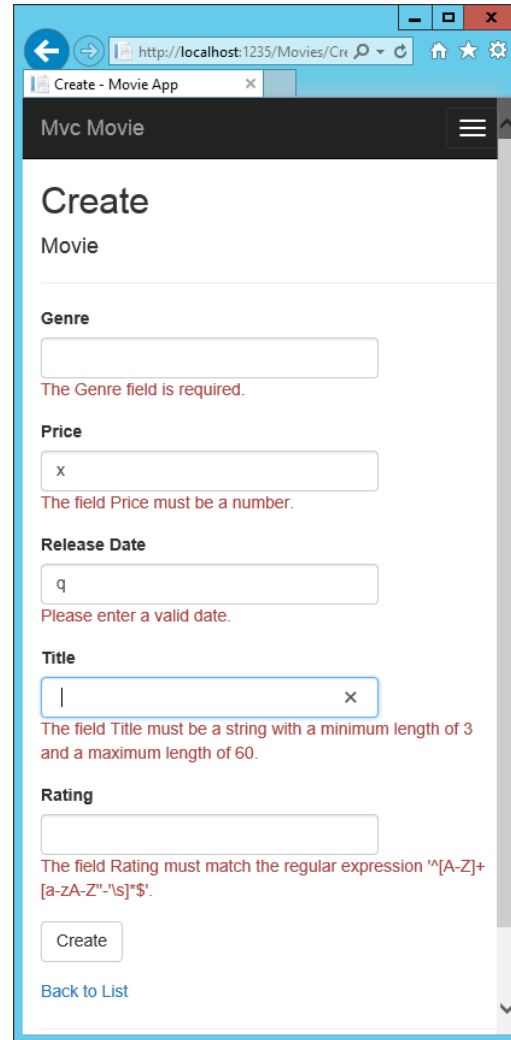
Connection Ready (localdb)\MSSQLLocalDB REDMOND\riande aspnet5-MvcMovie-53e15...

### 3. Scaffolded Views with Validation



```
13
14 <div class="form-horizontal">
15   <h4>Movie</h4>
16   <hr />
17   @Html.ValidationSummary(true, "", new { @class = "text-danger" })
18   <div class="form-group">
19     @Html.LabelFor(model => model.Title, htmlAttributes: new { @class = "control-label col-md-2" })
20     <div class="col-md-10">
21       @Html.EditorFor(model => model.Title, new { htmlAttributes = new { @class = "form-control" } })
22       @Html.ValidationMessageFor(model => model.Title, "", new { @class = "text-danger" })
23     </div>
24   </div>
25
26   <div class="form-group">
27     @Html.LabelFor(model => model.ReleaseDate, htmlAttributes: new { @class = "control-label col-md-2" })
28     <div class="col-md-10">
29       @Html.EditorFor(model => model.ReleaseDate, new { htmlAttributes = new { @class = "form-control" } })
30       @Html.ValidationMessageFor(model => model.ReleaseDate, "", new { @class = "text-danger" })
31     </div>
32   </div>
33
34   <div class="form-group">
35     @Html.LabelFor(model => model.Price, htmlAttributes: new { @class = "control-label col-md-2" })
36     <div class="col-md-10">
37       @Html.EditorFor(model => model.Price, new { htmlAttributes = new { @class = "form-control" } })
38       @Html.ValidationMessageFor(model => model.Price, "", new { @class = "text-danger" })
39     </div>
40   </div>
41
42   <div class="form-group">
43     @Html.LabelFor(model => model.Genre, htmlAttributes: new { @class = "control-label col-md-2" })
44     <div class="col-md-10">
45       @Html.EditorFor(model => model.Genre, new { htmlAttributes = new { @class = "form-control" } })
46       @Html.ValidationMessageFor(model => model.Genre, "", new { @class = "text-danger" })
47     </div>
48   </div>
49
50   <div class="form-group">
51     @Html.LabelFor(model => model.Rating, htmlAttributes: new { @class = "control-label col-md-2" })
52     <div class="col-md-10">
53       @Html.EditorFor(model => model.Rating, new { htmlAttributes = new { @class = "form-control" } })
54       @Html.ValidationMessageFor(model => model.Rating, "", new { @class = "text-danger" })
55     </div>
56   </div>
57
58   <div class="form-group">
59     <div class="col-md-offset-2 col-md-10">
60       <input type="submit" value="Create" class="btn btn-default" />
61     </div>
62   </div>
```

## 4. Validation Messages on UI



The screenshot shows a web browser window with the address bar displaying `http://localhost:1235/Movies/Crr`. The browser tab is titled 'Create - Movie App'. The page has a dark header with 'Mvc Movie' and a hamburger menu icon. The main content area is titled 'Create Movie' and contains a form with the following fields and validation messages:

- Genre**: A text input field. Below it, a red message reads: 'The Genre field is required.'
- Price**: A text input field containing the value 'x'. Below it, a red message reads: 'The field Price must be a number.'
- Release Date**: A text input field containing the value 'q'. Below it, a red message reads: 'Please enter a valid date.'
- Title**: A text input field with a clear button (x) on the right. Below it, a red message reads: 'The field Title must be a string with a minimum length of 3 and a maximum length of 60.'
- Rating**: A text input field. Below it, a red message reads: 'The field Rating must match the regular expression "[A-Z]+[a-zA-Z"-'\s]\*\$'.

At the bottom of the form, there is a 'Create' button and a 'Back to List' link.



# Module Summary

- In this you learned about:
  - Validation
  - Data Annotations
  - Client-Side and Server Side Validation
  - Validation != Security



# Lab: Validation in ASP.NET MVC



