



.NET Core: Developing Cross-Platform Web Apps with ASP.NET Core – Workshop*PLUS*

< Engineer Name >

Customer Engineer

v3.1

Conditions and Terms of Use

Microsoft Confidential

This training package is proprietary and confidential, and is intended only for uses described in the training materials. Content and software is provided to you under a Non-Disclosure Agreement and cannot be distributed. Copying or disclosing all or any portion of the content and/or software included in such packages is strictly prohibited.

The contents of this package are for informational and training purposes only and are provided "as is" without warranty of any kind, whether express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Training package content, including URLs and other Internet Web site references, is subject to change without notice. Because Microsoft must respond to changing market conditions, the content should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

Copyright and Trademarks

© 2016 Microsoft Corporation. All rights reserved.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

For more information, see Use of Microsoft Copyrighted Content at
<http://www.microsoft.com/en-us/legal/intellectualproperty/Permissions/default.aspx>

Internet Explorer, Microsoft, Microsoft Corporate Logo, SQL Server, and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other Microsoft products mentioned herein may be either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are property of their respective owners.

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

```
<environment names="Development">  
  <script src="~/lib/jquery-validation/dist/jquery.validate.js"></script>  
  <script src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.js"></script>  
</environment>  
<environment names="Staging,Production">  
  <script src="https://ajax.aspnetcdn.com/ajax/jquery.validate/1.14.0/jquery.validate.min.js"  
    asp-fallback-src="~/lib/jquery-validation/dist/jquery.validate.min.js"  
    asp-fallback-test="window.jQuery && window.jQuery.validator">  
  </script>
```

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()
```

```
<span asp-validation-for="Rating" class="text-danger"></span>
```

```
<div asp-validation-summary="ValidationSummary.All" class="text-danger"></div>
```

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 ActionResult 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;
    }
}
```

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 'Design' view is active, showing a table with six columns: ID, Genre, Price, ReleaseDate, Title, and Rating. The 'ID' column is marked as the primary key with an identity property. The 'Allow Nulls' column indicates that 'ID' is not nullable, while 'Genre', 'Price', 'ReleaseDate', 'Title', and 'Rating' are nullable.

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

The 'Keys (1)' pane on the right shows a primary key constraint named 'PK_Movie' (Primary Key, Clustered). The 'Check Constraints (0)', 'Indexes (0)', 'Foreign Keys (0)', and 'Triggers (0)' panes are empty.

The 'T-SQL' view at the bottom shows the following script:

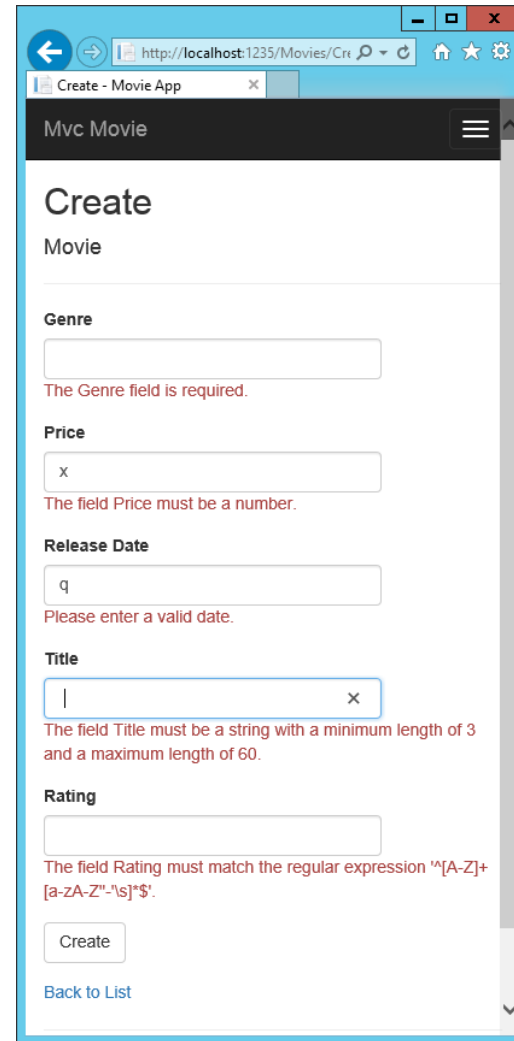
```
1 CREATE TABLE [dbo].[Movie] (  
2     [ID] INT IDENTITY (1, 1) NOT NULL,  
3     [Genre] NVARCHAR (MAX) NOT NULL,  
4     [Price] DECIMAL (18, 2) NULL,  
5     [ReleaseDate] DATETIME2 (7) NULL,  
6     [Title] NVARCHAR (MAX) NULL,  
7     [Rating] NVARCHAR (MAX) NULL,  
8     CONSTRAINT [PK_Movie]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

The status bar at the bottom indicates 'Connection Ready' and shows the server name '(localdb)\MSSQLLocalDB' and the user 'REDMOND\riande'.

3. Scaffolded Views with Validation

```
<form asp-action="Create">
  <div class="form-horizontal">
    <h4>Movie</h4>
    <hr />
    <div asp-validation-summary="ValidationSummary.ModelOnly" class="text-danger"></div>
    <div class="form-group">
      <label asp-for="Genre" class="col-md-2 control-label"></label>
      <div class="col-md-10">
        <input asp-for="Genre" class="form-control" />
        <span asp-validation-for="Genre" class="text-danger" />
      </div>
    </div>
    @*Markup removed for brevity.*@
    <div class="form-group">
      <label asp-for="Rating" class="col-md-2 control-label"></label>
      <div class="col-md-10">
        <input asp-for="Rating" class="form-control" />
        <span asp-validation-for="Rating" class="text-danger" />
      </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>
</form>
```

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 module, you learned about:
 - Validation
 - Data Annotations
 - Client-Side and Server Side Validation
 - Validation != Security



Lab: Validation in ASP.NET MVC



