

COMP 2139

How to Validate Data



Agenda

- Describe the default data validation provided by model binding
- Describe the use of attributes for data validation
- List six data validation attributes
- Describe the use of tag helpers for data validation
- Describe the use of the IsValid property of a controllers ModelState property
- Describe the use of CSS to format validation messages
- Describe the use of a controller's ModelState property for checking the validation state of a control and setting a custom error message
- Describe the use of model-level and property-level validation messages
- Describe the use of unobtrusive client-side validation
- Distinguish between client-side and server-side validation
- Describe the process of implementing remote validation
- Describe the process of customizing client-side data validation

The default data validation provided by model binding

The Movie Class

```
public class Movie
{
    public string Name { get; set; }
    public int Rating { get; set; }
}
```

The default data validation provided by model binding...

A strongly-typed view that posts a Movie object to an action method

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

<form asp-action="Add" method="post" class="form-inline">
  <div class="form-group">
    <label>Name:</label>&nbsp;
    <input asp-for="Name" class="form-control" />
  </div>&nbsp;
  <div class="form-group">
    <label>Rating (1 - 5):</label>&nbsp;
    <input type="text" asp-for="Rating" class="form-control" />
  </div>&nbsp;
  <button type="submit" class="btn btn-primary">Submit</button>
</form>
```

The default data validation provided by model binding...

An action method that receives a Movie object

```
[HttpPost]
public IActionResult Add(Movie movie)
{
    if (ModelState.IsValid) {
        /* code to add movie goes here */
        return RedirectToAction("List", "Movie");
    }
    else {
        return View(movie);    /* error detected, return view back to caller */
    }
}
```

If .NET/MVC fails to bind a view (casting issues etc..) to a model, it adds an error message to the ModelState property of the controller.

How to use data attributes for validation



How to use data attributes for validation

Common data attributes for validation

Attribute	Description
Required	Makes sure the property value is not an empty string or blank space. Numeric data types like int or double must be nullable .
Range(min, max)	Make sure the property value is within the specified range of values. The property must implement the Comparable interface.
Range(type, min, max)	Makes sure a value of a type other than int or double is within a range of values. The min and max arguments must be strings .
StringLength(length)	Makes sure the string property value doesn't exceed a specific length
RegularExpression(ex)	Makes sure a property value matches a regex pattern
Compare(other)	Make sure the property has the same value as another property
Display(Name = "n")	Specifies how the field name should be displayed to the user.

How to use data attributes for validation

The Movie entity class with data attributes

```
using System.ComponentModel.DataAnnotations;
...
public class Movie
{
    [Required]
    [StringLength(30)]
    public string Name { get; set; }

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

The view after the user submit invalid data

- The field Name must be a string with a maximum length of 30.
- The field Rating must be between 1 and 5.

Name:

Rating (1 - 5):

How to use data attributes for validation

The default validation messages

Attribute	Message
Required	The field [Name] is required.
Range	The field [Name] must be between [minimum] and [maximum].
StringLength	The field [Name] must be a string with a maximum length of [length].
RegularExpression	The field [Name] must match the regular expression '[pattern]'.
Compare	'[Name]' and '[OtherName]' do not match.

How to use data attributes for validation

How to replace the default message with a custom message

```
[Required(ErrorMessage = "Please enter a name")]
```

```
[StringLength(30, ErrorMessage = "Name must be 30 characters or less.")]  
public string Name { get; set; }
```

```
[Range(1, 5, ErrorMessage = "Please enter a rating between 1 and 5.")]  
public int Rating { get; set; }
```

```
[Range(typeof(DateTime), "1/1/2021", "4/1/2021",  
        ErrorMessage = "Please enter a DOB between 1/1/2021 and 4/1/2021.")]  
public DateTime? DOB { get; set; }
```

A Registration page with data validation

The Customer class

```
public class Customer
{
    0 references
    public int ID { get; set; } // will be autogenerated by database

    [Required(ErrorMessage = "Please enter a username.")]
    [RegularExpression("(?i)^[a-z0-9 ]+$", ErrorMessage = "Username may not contain special characters.")]
    2 references
    public string Username { get; set; }

    [Required(ErrorMessage = "Please enter an email address.")]
    [Remote("CheckEmail", "Validation")]
    5 references
    public string EmailAddress { get; set; }

    [Required(ErrorMessage = "Please enter a date of birth.")]
    [MinimumAge(13, ErrorMessage = "You must be at least 13 years old.")]
    2 references
    public DateTime? DOB { get; set; }

    [Required(ErrorMessage = "Please enter a password.")]
    [Compare("ConfirmPassword")]
    [StringLength(25, ErrorMessage = "Please limit your password to 25 characters.")]
    2 references
    public string Password { get; set; }

    [Required(ErrorMessage = "Please confirm your password.")]
    [Display(Name = "Confirm Password")]
    [NotMapped]
    2 references
    public string ConfirmPassword { get; set; }
}
```

How to use data attributes for validation

A validation tag in a strongly-typed view that posts a Customer object

```
@model Customer
```

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

```
...
```

```
<input asp-for="Username" class="form-control" />
```

```
<input type="text" asp-for="DOB" class="form-control" />
```

```
<input type="password" asp-for="Password" class="form-control" />
```

```
<input type="password" asp-for="ConfirmPassword" class="form-control" />
```

How to use data attributes for validation

An action method that receives a Customer object from the view

```
public IActionResult Index(Customer customer) {  
    if (ModelState.IsValid) {  
        // code that adds customer to database  
        return RedirectToAction("Welcome");  
    } else {  
        return View(customer);  
    }  
}
```

How to use data attributes for validation

The Registration Page after the user submits invalid data

- Name may not contain special characters.
- Date of birth must be after 1/1/1900.
- 'Password' and 'Confirm Password' do not match.
- Please confirm your password.

Name:

Mary D!

DOB:

1/1/1872

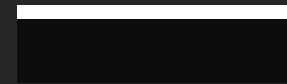
Password:

● ●

Confirm Password:

Submit

How to control user experience



How to control the user experience

The HTML emitted/produced for an `<input>` tag bound to the `Name` property

```
<input type="text" class="form-control"
      data-val="true"
      data-val-regex="Name may not contain special characters."
      data-val-regex-pattern="^[a-zA-Z0-9 ]&#x2B;$"
      data-val-required="Please enter a name."
      id="Name" name="Name" value="" />
```

The HTML emitted for that `<input>` when data validation fails

```
<input type="text" class="form-control" input-validation-error="
      data-val="true"
      data-val-regex="Name may not contain special characters."
      data-val-regex-pattern="^[a-zA-Z0-9 ]&#x2B;$"
      data-val-required="Please enter a name."
      id="Name" name="Name" value="" />
```

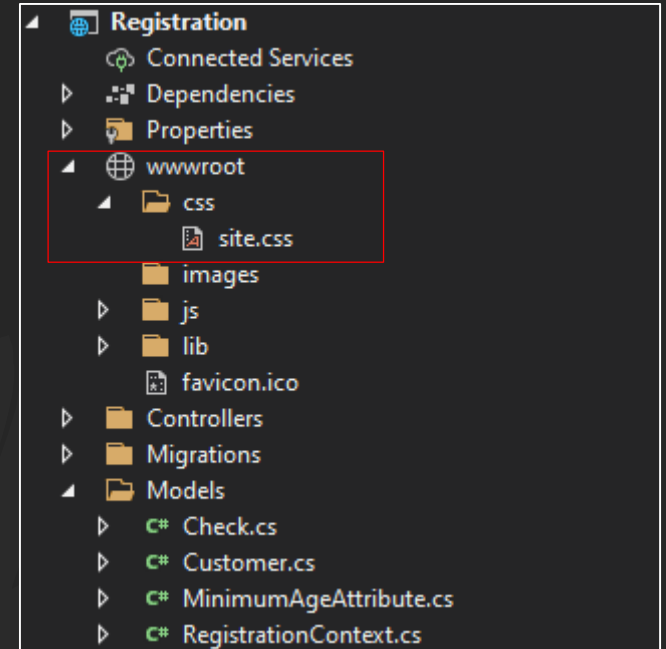

How to control the user experience...

Snippet of the CSS in the site.css file

```
.input-validation-error {  
    border: 2px solid #dc3545;      /* same red as text-danger */  
    background-color: #faebd7;      /* antique white */  
}
```

```
.validation-summary-valid { display: none; }
```

```
.validation-summary-errors ul { list-style: none; }
```



How to control the user experience...

The View with formatted validation



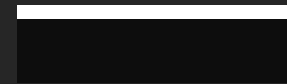
Please enter a name.
Please enter a date of birth.

Name:

DOB:

```
.input-validation-error {  
    border: 2px solid #dc3545;    /* same red as text-danger */  
    background-color: #faebd7;    /* antique white */  
}  
  
.validation-summary-valid { display: none; }  
  
.validation-summary-errors ul { list-style: none; }
```

How to check validation state and use code to set
the validation messages



How to check validation state

Properties of the ModelStateDictionary class

Attribute	Description
Count	The number of key/value pairs sent to the server and stored in the dictionary.
ErrorCount	The number of validation messages stored in the dictionary
IsValid	A Boolean value indicating whether any of the key/value pairs are invalid or not validated.
Keys	A collection of the keys of the form data parameters
Values	A collection of the values of the form data parameters

Methods of the ModelStateDictionary class

Attribute	Description
AddModelError(key, msg)	Adds a validation message to the dictionary and associates it with the property that matches the specified key. To associate a validation message with the overall model, rather than a specific property, specify an empty string as the key.
GetValidationState(key)	Gets the ModelState enum value for the specified property name. Possible values are Valid, Invalid, Unvalidated, and skipped.

How to check validation state...

Code that adds a validation message to the ModelState property

```
using Microsoft.AspNetCore.Mvc.ModelBinding;
...
[HttpPost]
public IActionResult Index(Customer customer)
{
    string key = nameof(Customer.DOB);    //name of DOB property

    if (ModelState.GetValidationState(key) == ModelValidationState.Valid) {
        if (customer.DOB > DateTime.Today) {
            ModelState.AddModelError( key, "Date of birth must not be a future date.");
        }
    }

    if (ModelState.IsValid) {
        return RedirectToAction("Welcome");
    }
    else {
        return View(customer);
    }
}
```

How to display model-level and property-level validation messages



How to display model-level and property-level validation messages

Two tag helpers used with data validation

Attribute	Description
asp-validation-summary	Displays the validation messages in then ModelState property. Accepts a value of the ValidateSummary enum to determine which messages to display Targets the <div> tag
asp-validation-for	Displays the first validation message in the ModelState property for the specified property. Targets the tag.

The values of the ValidateSummary enum

Attribute	Description
All	Displays all the validation messages in the ModelState property
ModelOnly	Displays only those messages in the ModelState property that are associated with the model. In other words, only, those messages whose key is an empty string rather than a property name.
None	Displays no messages

How to display model-level and property-level validation messages...

An action method that adds a model-level validation message

```
[HttpPost]
public IActionResult Index(Customer customer) {
    if (ModelState.IsValid) {
        // code that adds customer to database
        return RedirectToAction("Welcome");
    } else {
        ModelState.AddModelError("", "There are errors in the form.");
        return View(customer);
    }
}
```



Model State Error

How to display model-level and property-level validation messages...

Part of a view that displays both model-level and property-level messages

```
<div asp-validation-summary="ModelOnly" class="text-danger"></div>

<form asp-action="Index" method="post">
  <div class="form-group row">
    <div class="col-sm-2"><label>Name:</label></div>
    <div class="col-sm-4">
      <input asp-for="Name" class="form-control" />
    </div>
    <div class="col-sm-6">
      <span asp-validation-for="Name" class="text-danger"></span>
    </div>
  </div>
  ...
</form>
```

The form in the browser after validation fails

There are errors in the form.

Name: Please enter a name.

DOB: Please enter a date of birth.

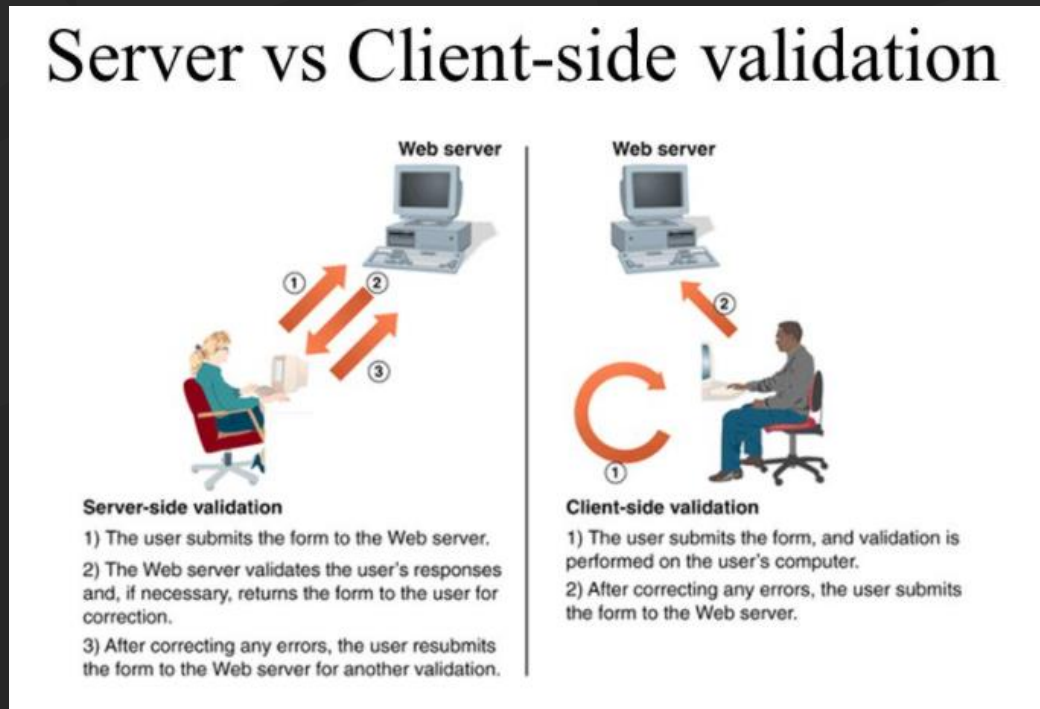
How to enable client-side validation



How to enable client-side validation

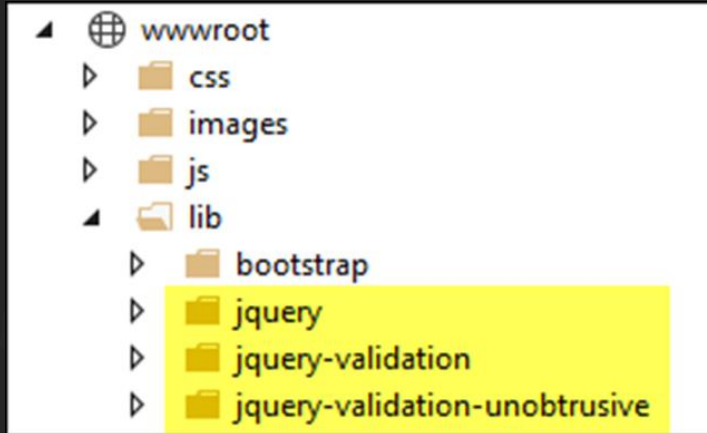
Client-Side Validation

- So far we have only shown how to validate data, server-side, that is, posting it to the server, checking it on the server and returning the validation results to the client.
- However it's generally good practice to validate data on the client before sending it to the server.



How to enable client-side validation

The jQuery libraries that download by default with the MVC Template



- To enable client-side validation, add jQuery validation and jQuery unobtrusive validation libraries to your application in the order shown above
- Client-side validation is optional and should be thought of as a convenience for the user. For security reasons, you should always perform server-side validation as well.

How to enable client-side validation

The jQuery libraries in a _Layout view

```
<head>
  <meta name="viewport" content="width=device-width" />
  <link href="~/lib/bootstrap/dist/css/bootstrap.min.css" rel="stylesheet" />
  <link href="~/css/site.css" rel="stylesheet" />

  <script src="~/lib/jquery/dist/jquery.min.js"></script>
  <script src="~/lib/jquery-validation/dist/jquery.validate.min.js"></script>
  <script src="~/lib/jquery-validation-unobtrusive/
                                jquery.validate.unobtrusive.min.js">
  </script>
  <title>@ViewBag.Title</title>
</head>
```

How to enable client-side validation

Some caveats to client-side validation

- It only works correctly with property-level validation, NOT model-level validation.
- Any server-side validation doesn't run until all client-side validation passes.
 - This can lead to a 2-step process that may annoy some users.
- Not all data annotations work properly on the client. In the example below, for instance, the Range annotation for the DOB field isn't working as it should.

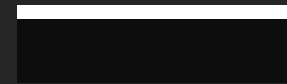
The form in the browser with client-side validation



A screenshot of a web form with two input fields. The first field is labeled 'Name:' and is empty, with a red border and a red error message 'Please enter a name.' to its right. The second field is labeled 'DOB:' and contains the text '1/1/1971', with a red border and a red error message 'Date of birth must be after 1/1/1900.' to its right. The form is set against a white background with a dark, torn-edge border at the bottom.

Name:	<input type="text"/>	Please enter a name.
DOB:	<input type="text" value="1/1/1971"/>	Date of birth must be after 1/1/1900.

How to customize server-side validation



How to customize server-side validation

Classes used to create a custom data attribute

Class	Description
ValidationAttribute	The base class for a custom data structure
ValidationContext	Describes the context for the validation
ValidationResult	Contains data that represents the validation results

A virtual method of the ValidationAttribute class

Virtual method	Description
IsValid(object, context)	Checks whether a value is valid. Accepts the value to check and a ValidationContext object. Returns a ValidationResult object

How to customize server-side validation

A constructor of the ValidationResult class

Virtual method	Description
ValidationResult(string)	Creates an object with validation message.

A field of the ValidationResult class

Static Field	Description
Success	Indicates that validation was successful

How to customize server-side validation

A custom attribute that checks if a date is in the past

```
public class PastDateAttribute : ValidationAttribute
{
    protected override ValidationResult IsValid(object value, ValidationContext ctx)
    {
        if (value is DateTime) {
            DateTime dateToCheck = (DateTime)value;
            if (dateToCheck < DateTime.Today) {
                return ValidationResult.Success;
            }
        }

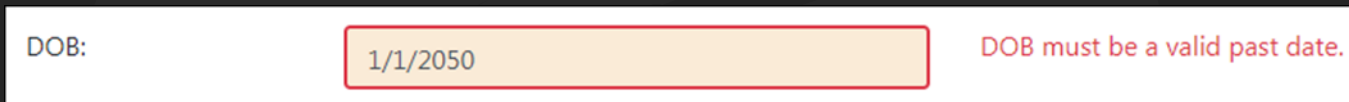
        string msg = base.ErrorMessage ?? $"{ctx.DisplayName} must be a valid past date.";
        return new ValidationResult(msg);
    }
}
```

How to customize server-side validation

Code that uses the PastDate attribute with the default validation message

```
[PastDate]  
public DateTime? DOB { get; set; }
```

How it looks in the browser



A screenshot of a web form with a label "DOB:" followed by a text input field containing "1/1/2050". The input field has a red border, indicating a validation error. To the right of the input field, a red error message reads "DOB must be a valid past date."

Code that uses the PastDate attribute with a custom validation message

```
[PastDate(ErrorMessage = "Please enter a date of birth in the past.")]  
public DateTime? DOB { get; set; }
```

Questions?