JOHN ABBOTT
CEGEP/COLLEGE

(420-PS4-AB)
Developing ASP .NET MVC
Validations

Summer 2018

Outline

- Server-side Validation
- Styling Validations
- Customizing Validation Messages
- Custom Validation
- Refactoring Magic Values
- Validation Summary
- Client-side Validation
- Forgery Attacks

Server Side Validations

Server-side Validation

- Data annotation applied on properties are used to override entity framework conventions.
- APS .NET MVC uses data annotations to validate action parameters.
- To add validation to a form:
  1. Add data annotations.
  2. Use method “ModelState.IsValid” in the http post action
     - If the state is not valid, return same view with the same passed data.
  3. Add validation messages to the form.

Example

- In Customer controller under the “Save” action
  - Parameter is “Customer” object.
  - When the object is populated using request data, MVC will check if the object is valid based on the data annotation applied on the properties.
- ModelState class provides access to validation data.
  - It has a property “IsValid” which can be used to change the application flow.
- A placeholder is added next to each field that requires validation to display the validation messages.

Demo: Validation

- In the Customer controller, under Save action
  - Use ModelState to check if the form is valid
  - Return to the same view (with customer form) if not valid.
  - Pass back the same customer object.

- Add validation message to the CustomerForm view
  - Customer Name requires validation (Required – 255 length)

```
@Html.ValidationMessageFor(m => m.Customer.Name)
```

Styling Validations

- To apply color or other styles to a validation, we need to inspect the classes they are using in the CSS.
- Access CSS and add formatting on specified class.
- Google Chrome provides us with an inspect option that will allow to know each element class.

Demo: Styling Validation

- In Google Chrome, right click on the validation message and click inspect. The following should be selected:

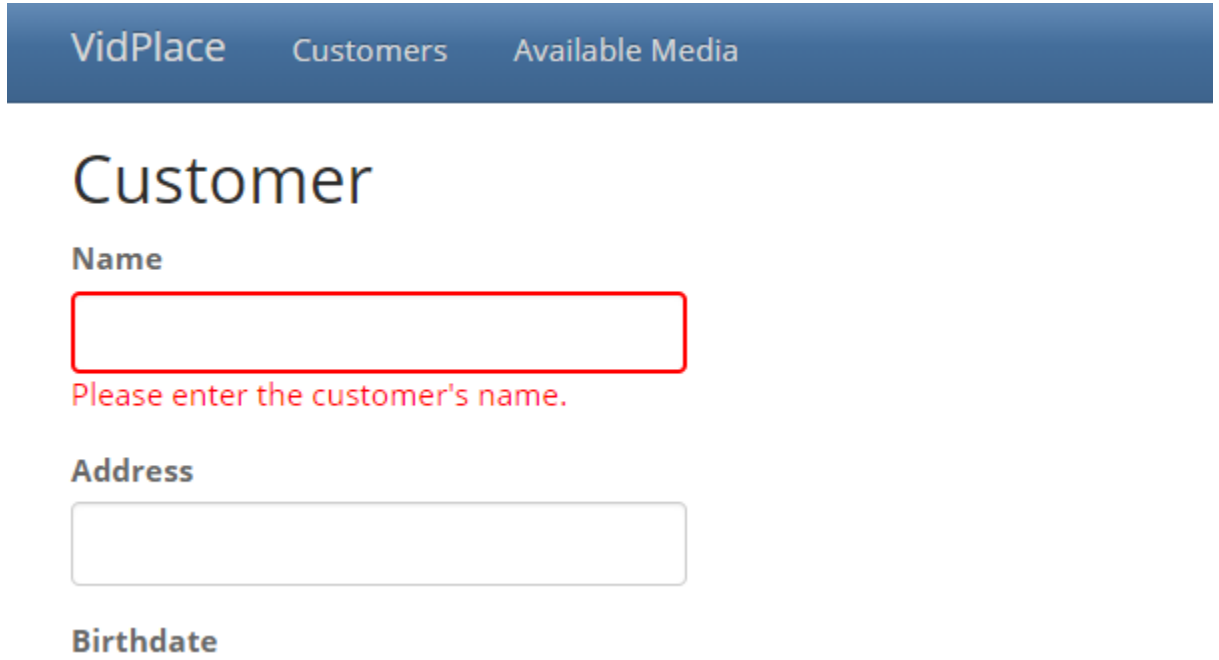
```
<span class="field-validation-error" data-valmsg-
for="Customer.Name" data-valmsg-
replace="true">The Name field is required.</span>
```

- In VS, Solution Explorer, Content, open Site.CSS
  - Define the above class and apply style to it:

```
.field-validation-error { color: red; }
```

  - Apply a styling on the textbox. (Hint: change the border}

Demo: Snapshot



Validation Summary

- Use help class Html.ValidationSummary to add a validation summary.
  - Test on out customer form.
  - Customer ID error? Why? Fix it from the New action by providing an empty Customer object.

- Html.ValidationSummary with custom message
  - Use an overloaded version with different parameter (true,"Your Message")

Other Data Annotations

[Required]

[StringLength(255)]

[Range(1,100)]

[Compare("OtherProperty")]

[Phone]

[EmailAddress]

[Url]

[RegularExpression("...")]

Customizing Validation Messages

- To override a validation message, you can set the Error Message property in the field annotation to a custom message.

- Examples:

```
[Required(ErrorMessage ="Please enter customer's name.")]
[MaxLength(255,ErrorMessage = "Address too long")]
[EmailAddress(ErrorMessage = "Email is not valid")]
```

Custom Validation

Custom Validation

- Custom validation usually arise from business logic and are not related to data types.

- We can create our own custom validations and apply them on properties in our domain models using annotations.

Creating Custom Validator

- Create a new class and inherit from ValidationAttribute Class.
  - Available under System.ComponentModel.DataAnnotations
- Override IsValid method
  - Using validationContext.object Instance provide access to the containing class (calling class).
  - Cast the returned object to get access to the properties of that class.
  - ValidationResult.Success: provide a valid input.
  - ValidationResult("Your Message") will return a message one the input is not valid.

Demo: Create Custom Validation

- Add a non required property to Customer class “Birthdate”
  - Apply a display Name “Date of Birth”
  - Add a migration and update database.
  - Add birthdate to CustomerForm and string formatting on the display field.
- New business rule: members with paid plans need to be 18 years old or above. (So, “pay as you go” is not included).
  - Create a custom validator and apply it on the Birthdate.
  - Add a placeholder in the Customer form to display the error.

Refactoring Magic Values

- Using magic values in the code makes maintaining the application difficult.
  - At the time of development, you as the developer will now what they stand for, but the next developer will have hard time trying to figure out where these value coming from.

- There are couple of work arounds.
  - Define static fields in the models include out business logic.

Demo: Refactor Magic Values for Membership Type

- Add static value for each membership type in the MembershipType class
  - Make them public and read only

- Update the code in the age customer validator

Client Side Validations

Client Side Validation

- Server-side validation is absolutely crucial for building secure applications.
- Adding additional validation on the client-side has benefits:
  - Users gets immediate feedback (no round trip to the server)
  - No waste of server-side resources.
- By default client-side validation is not enabled in ASP.NET MVC applications.

Client Side Validation

- Check the layout of your application
  - The Jquery jqueryval is not rendered.
  - The last render statement allow us to add script sections in the views that need it.

- On the page that require jquery validation, we add:
  - @section scripts{ @Scripts.Render("~/bundles/jqueryval")}

Demo: Client-side Validation

- Add the Jquery validation script to the customer form.
- Let us test using the inspect option.

Forgery Attacks

Forgery Attacks

- Imagine that a user who is responsible for creating a customers leaves the web application site without signing out.
  - The user will have an active session on the server (will be still authenticated for a few minutes)

- Now imagine a hacker that could trick this user to visit a malicious page.
  - With a little bit of JavaScript code on the page load an HTTP POST request can be sent to our application.

Cross-site Request Forgery

- What do you think is going to happen?
  - Because there is an active session on the web server, the request will be successfully executed on user's behalf.

- Such attacks are called:

Cross-site request forgery "CSRF"

What is happening?

- The hacker forges the request on a different website.
- Using this technique, a hacker can execute any actions on behalf of the victim.
- In the logs, all the requests appear as legitimate requests coming from the user’s browser.
- There will be no evidence of hacking or IP address traces.

Preventing CSRF Attacks

- Need to insure that that the request only comes from legitimate form not from a different website.

- Two measures to be taken:
  1. In the controller, apply attribute [ValidateAntiForgeryToken] to force validation on the server-side on the action reading for the form.
  2. In the form View: use the helper method in the form body @Html.AntiForgeryToken( )

Preventing CSRF Attacks Back-end

- Adding these elements will:
  - Create a token (like secret code) and place it as a hidden field in the form.
  - Creates a cookie on the user's computer.

- Go to inspect and check the hidden field and cookie.
  - Hidden Field: Called _RequestVerificationTokenI
  - Cookie: under resources, value stored in an encrypted format.

How prevention happens?

- When a post comes from a form, the two values (hidden field and encrypted cookie) are compared
  - If they match → Legitimate request
  - Do NOT match → ATTACK
- If the hacker manages to steal the cookie, he still cannot access to the hidden field.
  - The hidden field only exists when the user is actually visiting the form.
- The complexity of generating, encrypting and validating a token is all done by MVC framework.

Exercise Time

Exercise

- Add validation to the Media form.
  - All fields are required.
  - For Number of Stock: Provide a customer error message to show the range of that field.
  - Enable client-side validation
  - Add anti-forgery token.