.NET 9 App Dev Hands-On Lab

Razor Pages Lab 7b – Custom Validation

This lab walks you through creating custom validation attributes and the related client-side scripts. Before starting this lab, you must have completed Razor Pages\MVC Lab 7a.

Part 1: Create the Client-Side Validation Scripts

Step 1: Create the Validators

This code attaches the validation attributes to the jQuery validation system.

- Add a new folder named `validations` under the `wwwroot/js` folder in the `AutoLot.Web` project. Add a new JavaScript file named `validators.js` to the new folder. Add the validators and adapters for the two validation attributes:

```
$.validator.addMethod("greaterthanzero", function (value, element, params) {
  return value > 0;
});

$.validator.unobtrusive.adapters.add("greaterthanzero", function (options) {
  options.rules["greaterthanzero"] = true;
  options.messages["greaterthanzero"] = options.message;
});

$.validator.addMethod("notgreaterthan", function (value, element, params) {
  return +value <= +$(params).val();
});

$.validator.unobtrusive.adapters.add("notgreaterthan", ["otherpropertyname","prefix"], function
(options) {
  var rule = "#" + options.params.otherpropertyname;
  if (options.params.prefix !== undefined && options.params.prefix !== null &&
options.params.prefix !== '') {
    rule = "#" + options.params.prefix + "_" + options.params.otherpropertyname;
  }
  options.rules["notgreaterthan"] = rule;
  options.messages["notgreaterthan"] = options.message;
});
```

- Add a new function that will force revalidation of the entire form:

```
function reValidate(ctl) {
  var form = ctl.form;
  $(form).valid();
}
```

Step 2: Create the Error Formatter

This code prettifies errors in the UI.

- Create a new JavaScript file named `errorFormatting.js` in the `validations` folder. Update the code to match the following:

```
$.validator.setDefaults({
    highlight: function (element, errorClass, validClass) {
        if (element.type === "radio") {
            this.findByName(element.name).addClass(errorClass).removeClass(validClass);
        } else {
            $(element).addClass(errorClass).removeClass(validClass);
            $(element).closest('.form-group').addClass('has-error');
        }
    },
    unhighlight: function (element, errorClass, validClass) {
        if (element.type === "radio") {
            this.findByName(element.name).removeClass(errorClass).addClass(validClass);
        } else {
            $(element).removeClass(errorClass).addClass(validClass);
            $(element).closest('.form-group').removeClass('has-error');
        }
    }
});
```

Part 2: Minify and Bundle the JavaScript Files

Add configuration options to the `AddWebOptimizer` method in the `Program.cs` file to minimize and bundle (in prod) the new JS files.

- Use `AddJavaScriptBundle` to bundle the files. The first argument is the bundle name, and the second argument(s) is(are) the files to be bundled:

```
builder.Services.AddWebOptimizer(options =>
{
  //omitted for brevity
  options.AddJavaScriptBundle("js/validations/validationCode.js", "js/validations/**/*.js");
  //This is another format to bundle and minify the files
  //options.AddJavaScriptBundle("js/validations/validationCode.js",
  //  "js/validations/validators.js", "js/validations/errorFormatting.js");
});
```

Part 3: Update the _ValidationScriptsPartial.cshtml

- Open `Pages\Shared\_ValidationScriptsPartial.cshtml`. Add the following at the end of the `<environment include="Development,Local">` block:

```
<script src="~/js/validations/validators.js" asp-append-version="true"></script>
<script src="~/js/validations/errorFormatting.js" asp-append-version="true"></script>
```

- Add the following at the end of the non-development block:

```
<script src="~/js/validations/validationCode.js"></script>
```

Part 4: Use the Validation Attribute

Step 1: Add the Razor Pages specific View Model

- Add a new class named `AddToCartViewModelRp` in the `ViewModels` folder of the `AutoLot.Services` project and update it to the following:

```csharp
namespace AutoLot.Services.ViewModels;
public class AddToCartViewModelRp : AddToCartViewModelBase
{
  [Required]
  [MustBeGreaterThanZero]
  [MustNotBeGreaterThan(nameof(StockQuantity), prefix: "Entity")]
  public int Quantity { get; set; }
}
```

Step 2: Create the Validation Page

- Create a new Razor Page named `Validation` in the `Pages` directory of the `AutoLot.Web` project, and update the code behind to the following:

```csharp
namespace AutoLot.Web.Pages;
public class ValidationModel : PageModel
{
  [ViewData]
  public string Title => "Validation Example";
  [BindProperty]
  public AddToCartViewModelRp Entity { get; set; }
  public void OnGet()
  {
    Entity = new AddToCartViewModelRp { Id = 1, ItemId = 1, StockQuantity = 2, Quantity = 0 };
  }
  public async Task<IActionResult> OnPostAsync()
  {
    if (!ModelState.IsValid)
    {
      return Page();
    }
    await Task.Delay(5000);
    return RedirectToPage("Validation");
  }
}
```

- Next, update the markup to the following:

```
@page
@model AutoLot.Web.Pages.ValidationModel
<h1>Validation</h1>
<h4>Add To Cart</h4><hr />
<div class="row">
  <div class="col-md-4">
    <form id="validationForm" asp-page="/Validation">
      <div asp-validation-summary="ModelOnly" class="text-danger"></div>
      <div>
        <label asp-for="Entity.Id" class="col-form-label"></label>
        <input asp-for="Entity.Id" class="form-control" />
        <span asp-validation-for="Entity.Id" class="text-danger"></span>
      </div>
      <div>
        <label asp-for="Entity.StockQuantity" class="col-form-label"></label>
        <input asp-for="Entity.StockQuantity" oninput="reValidate(this)"
            class="form-control" onchange="reValidate(this)" />
        <span asp-validation-for="Entity.StockQuantity" class="text-danger"></span>
      </div>
      <div>
        <label asp-for="Entity.ItemId" class="col-form-label"></label>
        <input asp-for="Entity.ItemId" class="form-control" />
        <span asp-validation-for="Entity.ItemId" class="text-danger"></span>
      </div>
      <div>
        <label asp-for="Entity.Quantity" class="col-form-label"></label>
        <input asp-for="Entity.Quantity" oninput="reValidate(this)" class="form-control" />
        <span asp-validation-for="Entity.Quantity" class="text-danger"></span>
      </div>
      <div style="margin-top:5px">
        <button id="submitEditForm" type="submit" class="btn btn-primary">
          Save <i class="fas fa-save"></i>
        </button>
      </div>
    </form>
  </div>
</div>
```

```
@section Scripts {
  <partial name="_ValidationScriptsPartial" />
  <script type="text/javascript" language="javascript">
    function reValidate(input) {
      const form = $("#validationForm");
      var input1 = $("#Quantity");
      var input2 = $("#StockQuantity");
      form.validate().element(input1);
      form.validate().element(input2);
    }
    $(document).ready(function() {
      $("#validationForm").on("submit",
        function(e) {
          e.preventDefault(); // Prevent the default form submission
          const form = $(this);
          const validator = form.validate();
          if (form.valid()) {
            const submitButton = document.getElementById("submitEditForm");
            submitButton.disabled = true;
            submitButton.innerHTML = 'Submitting...';
            this.submit(); // Submit the form programmatically
          } else {
            validator.resetForm();
          }
        });
    });
  </script>
}
```

Step 3: Update the _Menu Partial

- Add the following menu item to the `_Menu.cshtml` partial:

```
<li class="nav-item">
  <a class="nav-link text-dark" asp-area="" asp-page="/Validation" title="Validation
Example">Validation <i class="fa-solid fa-check"></i></a>
</li>
```

Summary

This lab added validation scripts and demonstrated how to use the validation attributes.

Next Steps

The next lab creates an admin area for `Make` maintenance.