**ASP.Net MVC Client Side Validation**

This article explains how to implement client-side validation in an ASP.NET MVC application. The validation implemented using jQuery and jQuery validation plug-in (jquery.validate.min.js and jquery.validate.unobtrusive.min.js).  
  
In the server-side validation, the page must be submitted via a postback to be validated on the server and if the model data is not valid then the server sends a response back to the client with client-side validation, the input data is checked as soon as they are submitted, so there is no postback to the server and there is no page refresh.  
  
When you are developing an MVC application in Visual Studio 2012 then the client-side becomes enabled by default, but you can easily enable or disable the writing of the following app setting code snippet in the web.config file.

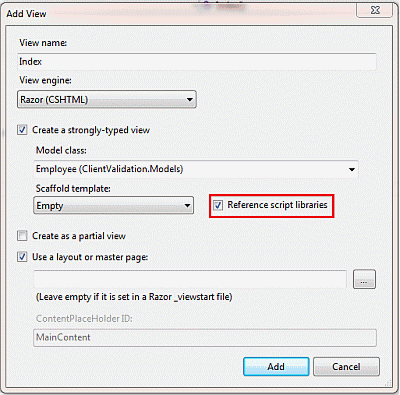
1. <configuration>
2. <appSettings>
3. <add key="ClientValidationEnabled" value="true" />
4. <add key="UnobtrusiveJavaScriptEnabled" value="true" />
5. </appSettings>
6. </configuration>

The jQuery validation plug-in takes advantage of the Data Annotation attributes defined in the model, which means that you need to do very little to start using it. Let's create an Employee model (Employee.cs class file under the Models folder) that has two properties with Data annotation attributes.

1. **using** System.ComponentModel.DataAnnotations;
2. **namespace** ClientValidation.Models
3. {
4. **public** **class** Employee
5. {
6. [Required(ErrorMessage = "Name is Requirde")]
7. **public** **string** Name { **get**; **set**; }
8. [Required(ErrorMessage = "Email is Requirde")]
9. [RegularExpression(@"^([a-zA-Z0-9\_\-\.]+)@((\[[0-9]{1,3}" +
10. @"\.[0-9]{1,3}\.[0-9]{1,3}\.)|(([a-zA-Z0-9\-]+\" +
11. @".)+))([a-zA-Z]{2,4}|[0-9]{1,3})(\]?)$",
12. ErrorMessage = "Email is not valid")]
13. **public** **string** Email { **get**; **set**; }
14. }
15. }

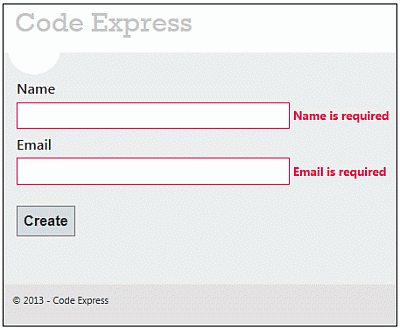
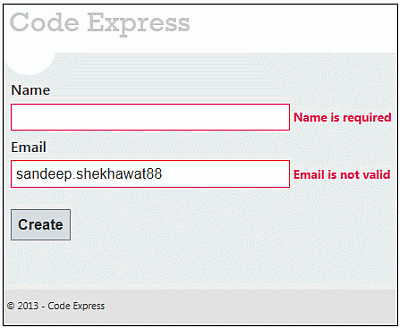
After that you need to create the controller's action methods. These render views on the UI and bind a model with the view. So let's create a controller with two action methods that handle both request types (GET and POST respectively).

1. **using** System.Web.Mvc;
2. **using** ClientValidation.Models;
3. **namespace** ClientValidation.Controllers
4. {
5. **public** **class** EmployeeController : Controller
6. {
7. **public** ActionResult Index()
8. {
9. **return** View();
10. }
11. [AcceptVerbs(HttpVerbs.Post)]
12. **public** ActionResult Index(Employee model)
13. {
14. **if** (ModelState.IsValid)
15. {
16. ViewBag.Name = model.Name;
17. ViewBag.Email = model.Email;
18. }
19. **return** View(model);
20. }
21. }
22. }

Now create a view by right-clicking on the action method then click on "Add View" as:  
  
  
  
**Figure 1.1 :** Add New View Screen  
  
Here I checked "Reference script libraries"which means Visual Studio adds these references automatically. Visual Studio adds the following code snippet to the bottom of the view.

1. @section Scripts {
2. @Scripts.Render("~/bundles/jqueryval")
3. }
5. Now click on the "Add" button and the view is created as **in** the following code snippet:
7. @model ClientValidation.Models.Employee
8. @{
9. ViewBag.Title = "Index";
10. }
11. @**if** (ViewData.ModelState.IsValid)
12. {
13. **if**(@ViewBag.Name != **null**)
14. {
15. <b>
16. Name : @ViewBag.Name<br />
17. Email : @ViewBag.Email
18. </b>
19. }
20. }
21. @using (Html.BeginForm()) {
22. @Html.ValidationSummary(**true**)
23. <fieldset>
24. <legend>Employee</legend>
25. <div **class**="editor-label">
26. @Html.LabelFor(model => model.Name)
27. </div>
28. <div **class**="editor-field">
29. @Html.EditorFor(model => model.Name)
30. @Html.ValidationMessageFor(model => model.Name)
31. </div>
32. <div **class**="editor-label">
33. @Html.LabelFor(model => model.Email)
34. </div>
35. <div **class**="editor-field">
36. @Html.EditorFor(model => model.Email)
37. @Html.ValidationMessageFor(model => model.Email)
38. </div>
39. <p>
40. <input type="submit" value="Create" />
41. </p>
42. </fieldset>
43. }
44. @section Scripts {
45. @Scripts.Render("~/bundles/jqueryval")
46. }

Let's run the application and test the following scenario.

1. When all fields are empty:  
     
     
   **Figure 1.2:**Validation Message when both fields are empty
2. When the Name field is empty but Email is not valid:  
     
     
   **Figure 1.3 :** Validation Message when Email is not valid
3. When both fields are valid:  
     
   A screenshot of a contact form

   Description automatically generated  
   **Figure 1.4**All fields are valid

Unobtrusive Validation

Unobtrusive Validation support for Input controls helps you to validate any C1 input control using client side validation. Unobtrusive validation can implement simple client-side validation without writing a bulk of validation code, and improves the user experience simply by adding the right attributes and including the script files.  
  
In a common validation scenario, when we use a validation to validate any control and use client side validation, JavaScript code is generated and rendered as HTML on the web browser. However, with unobtrusive validation inline JavaScript is not generated for rendering to handle client side validation. Instead, it uses HTML5 data-\* attributes for client side validations.  
  
Unobtrusive Validation support is available in the following Input controls;

* InputColor
* InputDate
* InputDateTime
* InputMask
* InputNumber
* AutoComplete
* ComboBox
* InputTime
* MultiSelect

Before implementing the below steps, you need to create a new MVC application using [ComponentOne template](https://developer.mescius.com/componentone/docs/mvc/online-mvc-core/UsingC1Template.html) or [Visual Studio template](https://developer.mescius.com/componentone/docs/mvc/online-mvc-core/UsingVSTemplate.html).

This topic comprises of four steps:

* [Step 1: Configure your MVC application](https://developer.mescius.com/componentone/docs/mvc/online-mvc-core/InputUnobtrusiveValidation.html#step1)
* [Step 2: Create Validations for Input controls](https://developer.mescius.com/componentone/docs/mvc/online-mvc-core/InputUnobtrusiveValidation.html#step2)
* [Step 3: Add an Input control](https://developer.mescius.com/componentone/docs/mvc/online-mvc-core/InputUnobtrusiveValidation.html#step3)
* [Step 4: Build and Run the Project](https://developer.mescius.com/componentone/docs/mvc/online-mvc-core/InputUnobtrusiveValidation.html#step4)

A white background with black text

Description automatically generated

Step 1: Configure your MVC application

Complete the following steps to add the js file references to your application.

1. From the **Solution Explorer**, open the folders **Views | Shared**.
2. Double click **\_Layout.cshtml** to open it.
3. Add the following code between the <head></head> tags.

|  |  |
| --- | --- |
| **Razor** | **Copy Code** |
| <script src="~/Scripts/jquery-1.10.2.js"></script>  <script src="~/Scripts/jquery.validate.js"></script>  <script src="~/Scripts/jquery.validate.unobtrusive.js"></script> | |

Step 2: Create Validations for Input controls

**Model - Form.cs**

|  |  |
| --- | --- |
| **Razor** | **Copy Code** |
| using System.ComponentModel.DataAnnotations;  namespace UnobtrusiveValidationSample.Models  {  public class Form  {  [Required]  public string Id { get; set; }  [Required]  [DataType(DataType.Text)]  [MinLength(8)]  [MaxLength(30)]  public string Name { get; set; }  [Required]  [DataType(DataType.PhoneNumber)]  [Phone]  [Range(0, 10000000000)]  [StringLength(11)]  [MinLength(7)]  [MaxLength(10)]  public string PhoneNo { get; set; }  [Required]  [DataType(DataType.EmailAddress)]  [EmailAddress]  [MinLength(10)]  [MaxLength(50)]  public string EmailAddress { get; set; }  }  } | |

Step 3: Add an Input control

**View - Index.cshtml**

|  |  |
| --- | --- |
| **Razor** | **Copy Code** |
| @model Form  @section Scripts{  @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}  }  <form>  <br />  <label asp-for="Id"></label>  <br />  <input asp-for="Id" />  <span asp-validation-for="Id"></span>  <br />  <label asp-for="Name"></label>  <br />  <input asp-for="Name" />  <span asp-validation-for="Name"></span>  <br />  <label asp-for="PhoneNo"></label>  <br />  <input asp-for="PhoneNo" />  <span asp-validation-for="PhoneNo"></span>  <br />  <label asp-for="EmailAddress"></label>  <br />  <c1-input-mask for="EmailAddress"></c1-input-mask>  <span asp-validation-for="EmailAddress"></span>  <br />  <input type="submit" />  <input type="reset" />  </form> | |

[Back to Top](https://developer.mescius.com/componentone/docs/mvc/online-mvc-core/InputUnobtrusiveValidation.html#top)

Step 4: Build and Run the Project

1. Click **Build | Build Solution** to build the project.
2. Press **F5** to run the project.