FORM VALIDATION

In .NET, you can perform form validation in HTML using various approaches. The main goal of form validation is to ensure that the data submitted by users meets certain criteria before it is processed on the server-side. There are several ways to achieve this, but the most common approaches include:

Server-side validation: This involves validating the form data on the server-side after it has been submitted. Server-side validation is essential to ensure the integrity and security of your data, as client-side validation can be bypassed by malicious users.

Client-side validation: This involves validating the form data on the client-side using JavaScript before it is submitted to the server. Client-side validation helps to provide immediate feedback to users and reduces the number of unnecessary server requests.

Client-side validation (using JavaScript):

<form asp-action="index" method="post" onsubmit="return validateForm();">

<label for="name">Name:</label>

<input type="text" id="name" name="name">

<br>

<label for="email">Email:</label>

<input type="email" id="email" name="email">

<br>

<input type="submit" value="Submit">

</form>

<h2>@ViewBag.Name</h2>

<h2>@ViewBag.Email</h2>

<script>

function validateForm() {

const name = document.getElementById('name').value;

const email = document.getElementById('email').value;

if (name.trim() === "") {

alert('Name is required.');

return false;

}

if (email.trim() === "") {

alert('Email is required.');

return false;

}

// You can add more complex validation rules for the email, etc.

return true;

}

</script>

Server-side validation (Using Models):

We can use model validation attributes to perform server-side validation.

Student.cs

public class Student

{

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

[StringLength(100)]

public string Name { get; set; }

[Required(ErrorMessage = "Please choose gender")]

public string Gender { get; set; }

[Required(ErrorMessage = "Please enter date of birth")]

[Display(Name = "Date of Birth")]

[DataType(DataType.Date)]

public DateTime DateofBirth { get; set; }

[Required(ErrorMessage = "Choose batch time")]

[Display(Name = "Batch Time")]

[DataType(DataType.Time)]

public DateTime BatchTime { get; set; }

[Required(ErrorMessage = "Please enter phone number")]

[Display(Name = "Phone Number")]

[Phone]

public string PhoneNumber { get; set; }

[Required(ErrorMessage = "Please enter email address")]

[Display(Name = "Email Address")]

[EmailAddress]

public string Email { get; set; }

[Required(ErrorMessage = "Please enter website url")]

[Display(Name = "Website Url")]

[Url]

public string WebSite { get; set; }

[Required(ErrorMessage = "Please enter password")]

[DataType(DataType.Password)]

public string Password { get; set; }

[Required(ErrorMessage = "Please enter confirm password")]

[Display(Name = "Confirm Password")]

[Compare("Password", ErrorMessage = "Password and confirm password does not

match")]

public string ConfirmPassword { get; set; }

}

Index.cshtml

@model MyfirstMVCProject1.Models.Student

@{

ViewData["Title"] = "Registration";

}

<div class="card">

<div class="card-header bg-primary text-white text-uppercase">

<h4>Student Information</h4>

</div>

<div class="card-body">

<form asp-action="Index">

<div class="row">

<div class="col-md-6">

<div class="form-group">

<label asp-for="Name" class="label-control"></label>

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

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

</div>

</div>

<div class="col-md-6">

<div class="form-group">

<label asp-for="Gender" class="label-control"></label>

<input type="radio" asp-for="Gender" value="Male" class="label-control" /> Male

<input type="radio" asp-for="Gender" value="Female" class="label-control" /> Female

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

</div>

</div>

</div>

<div class="row">

<div class="col-md-6">

<div class="form-group">

<label asp-for="DateofBirth" class="label-control"></label>

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

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

</div>

</div>

<div class="col-md-6">

<div class="form-group">

<label asp-for="BatchTime" class="label-control"></label>

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

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

</div>

</div>

</div>

<div class="row">

<div class="col-md-4">

<div class="form-group">

<label asp-for="PhoneNumber" class="label-control"></label>

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

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

</div>

</div>

<div class="col-md-4">

<div class="form-group">

<label asp-for="Email" class="label-control"></label>

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

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

</div>

</div>

<div class="col-md-4">

<div class="form-group">

<label asp-for="WebSite" class="label-control"></label>

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

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

</div>

</div>

</div>

<div class="row">

<div class="col-md-6">

<div class="form-group">

<label asp-for="Password" class="label-control"></label>

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

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

</div>

</div>

<div class="col-md-6">

<div class="form-group">

<label asp-for="ConfirmPassword" class="label-control"></label>

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

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

</div>

</div>

</div>

<div class="form-group">

<button type="submit" class="btn btn-primary rounded-0">Submit</button>

</div>

</form>

</div>

</div>

<!-- Activate JQuery Validation -->

@section Scripts{

@await Html.PartialAsync("/Views/Shared/\_ValidationScriptsPartial.cshtml")

}

Student Controller:

public IActionResult Index()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public IActionResult Index(Student student)

{

var return\_page = "Index";

if (ModelState.IsValid)

{

ViewBag.finalcontent = student.Name;

ViewBag.Success = "Successfully Registered";

return\_page = "Success";

}

return View(return\_page);

}

Success.cshtml

@if (ViewBag.Success != null)

{

<h1>@ViewBag.Success</h1>

}

Built-in validation attributes provided by Dotnet Core MVC:

[Display(Name = "Label to display in the form")] - it works directly with the <label asp-for=”property\_name”>

[Compare] - validates two properties of a model match.

[Compare("Password", ErrorMessage = "Password and Confirm Password do not match"]

Password - is the property to match with the current property

Email Validation

[EmailAddress] - validates that the property has a valid email format.

[EmailAddress(ErrorMessage=”Custom error message”)]

Custom email validation using regX:

[RegularExpression(@"^[a-zA-Z0-9.!#$%&'\*+/=?^\_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)\*$",

ErrorMessage = "Invalid Email Address.")]

[Phone] - validates that the property has a valid phone number format.

[Range] - validates that the property value has fallen within the specified range values.

[Range(10, 1000, ErrorMessage = "Value for {0} must be between {1} and {2}.")]

[Range(300, 3000)]

[Range(typeof(DateTime), "1/2/2004", "3/4/2004", ErrorMessage = "Value for {0} must be

between {1} and {2}")]

[RegularExpression] - validates that the property has satisfied the given regular expression.

[RegularExpression(@"^[a-zA-Z0-9]\*$", ErrorMessage = "Enter Alphabets and Numbers")]

[RegularExpression(@"^(([A-za-z]+[\s]{1}[A-za-z]+)|([A-Za-z]+))$")]

public string UserName { get; set; }

[Required] - validate that the property has some value, can not be null or empty.

[StringLength] - validate that the property value length should not exceed the length specified.

[StringLength(4)]

[StringLength(4, ErrorMessage = “Custom error message”)]

[Remote]- validates the input on the client, by calling the action method on the server through

AJAX.

[Remote("EmailExists", "Customer",HttpMethod ="POST", ErrorMessage = "Email Id already exists. Try another one!")]

EmailExists - A custom method in the controller returns JsonResult as true or false.

Customer - The MVC Controller where EmailExists method is defined.

HttpMethod - An optional parameter defines how to send AJAX data in POST or GET

ErrorMessage - The custom message on email already exists.

EmailExists method model:

[HttpPost]

public JsonResult EmailExists(string EmailId)

{

return Json(true);

}

OR

public IActionResult EmailExists(string EmailId)

{

return Json(true);

}

Define DataTypes in Validation

(17 types are available)

Syntax: [DataType(DataType.data\_type, ErrorMessage = "Custom message")]

Examples:

[DataType(DataType.Url, ErrorMessage = "Please Enter a valid URL")]

[DataType(DataType.Password)]

[DataType(DataType.PhoneNumber, ErrorMessage = "Please Enter a valid Phone Number")]

[DataType(DataType.PostalCode, ErrorMessage = "Please Enter a valid PIN/ZIP Code")]

[DataType(DataType.EmailAddress, ErrorMessage = "Please Enter a valid Email")]

[DataType(DataType.DateTime, ErrorMessage = "Please Enter a valid Date")]

[DataType(DataType.Time, ErrorMessage = "Please Enter a valid Time")]