Validation in Blazor

Basic Validation with DataAnnations

▼ Attributes in (System.ComponentModel.DataAnnotations Namespace)

- **[Required]**: Ensures a property is not null or empty.
- **[StringLength]**: Sets a maximum (and optional minimum) string length.
- **[MaxLength]** / **[MinLength]**: Specifies the maximum/minimum allowable string or array length.
- **[Range]**: Validates numeric or date values fall within a specified range.
- **[RegularExpression]**: Validates against a specified regex pattern.
- **[EmailAddress]**: Validates the format of an email address.
- **[Phone]**: Validates phone number formats.
- **[Url]**: Ensures a valid URL format.
- **[CreditCard]**: Validates credit card numbers.
- **[Compare]**: Compares two properties for equality (e.g., password confirmation).
- **[CustomValidation]**: Enables validation through a custom validation method.
- **[DataType]**: Specifies the type of data (e.g., `DataType.Date`, `DataType.Currency`, `DataType.Password`).
- **[Key]**: Marks a property as the primary key.
- **[Timestamp]**: Identifies a property as a timestamp for concurrency checking.

```
<h3>Basic Validation</h3>

<EditForm Model="person" OnValidSubmit="HandleValidSubmit">

    <DataAnnotationsValidator />
    <ValidationSummary />

    <div class="mb-3">
        <label for="Name" class="form-label">Name:</label>
        <InputText id="Name" @bind-Value="person.Name" class
="form-control" />
        <ValidationMessage For="@(() => person.Name)" />
    </div>

    <div class="mb-3">
        <label for="Age" class="form-label">Age:</label>
        <InputNumber id="Age" @bind-Value="person.Age" class
="form-control" />
        <ValidationMessage For="@(() => person.Age)" />
    </div>

    <div class="mb-3">
        <label for="Email" class="form-label">Email:</label>
        <InputText id="Email" @bind-Value="person.Email" clas
s="form-control" />
        <ValidationMessage For="@(() => person.Email)" />
    </div>

    <button type="submit" class="btn btn-primary">Submit</but
ton>

</EditForm>

@if (IsSubmitted)
{
```

```
    <p class="text-success mt-3 h3">✔ Booking successfully su
bmitted!</p>
}

@code {
    private Person person = new();
    private bool IsSubmitted = false;

    private void HandleValidSubmit()
    {
        IsSubmitted = true;
    }
}
```

```
public class Person
{
    [Required(ErrorMessage = "Name is required")]
    public string Name { get; set; }

    [Range(18, 120, ErrorMessage = "Age must be between 18 an
d 120")]
    public int Age { get; set; }

    [EmailAddress(ErrorMessage = "Invalid email address")]
    public string Email { get; set; }
}
```

Custom Validation

```
public class FutureDateAttribute : ValidationAttribute
{
    protected override ValidationResult IsValid(object value,
```

```
ValidationContext validationContext)
    {
        if (value is DateTime date && date < DateTime.Today)
        {
            return new ValidationResult(ErrorMessage ?? "The
date must be in the future.");
        }

        return ValidationResult.Success;
    }
}
```

```
public class HotelBooking : IValidatableObject
{
    [Required]
    [FutureDate(ErrorMessage = "Booking date cannot be in the
past.")]
    public DateTime BookingDate { get; set; } = DateTime.Now;

    [Required]
    public DateTime CheckInDate { get; set; } = DateTime.Now.
AddDays(1);

    [Required]
    public DateTime CheckOutDate { get; set; } = DateTime.No
w.AddDays(2);

    [ValidateComplexType]
    public Guest GuestDetails { get; set; } = new Guest();

    public IEnumerable<ValidationResult> Validate(ValidationC
ontext validationContext)
    {
        if (CheckOutDate <= CheckInDate)
        {
```

```
            yield return new ValidationResult("Check-out date
must be after the check-in date.", new[] { nameof(CheckOutDat
e) });
        }
    }
}
```

```
public class Guest
{
    [Required]
    [StringLength(100, MinimumLength = 2, ErrorMessage = "Nam
e must be between 2 and 100 characters.")]
    public string Name { get; set; }

    [Required]
    [EmailAddress(ErrorMessage = "Invalid email format.")]
    public string Email { get; set; }
}
```

```
<h3>Custom Validation</h3>

<EditForm Model="Booking" OnValidSubmit="HandleValidSubmit">

    <ObjectGraphDataAnnotationsValidator/>
    <ValidationSummary />

    <div class="mb-3">
        <label class="form-label">Booking Date:</label>
        <InputDate @bind-Value="Booking.BookingDate" class="f
orm-control" />
        <ValidationMessage For="@(() => Booking.BookingDate)"
/>
    </div>
    <div class="mb-3">
```

```
        <label class="form-label">Check-In Date:</label>
        <InputDate @bind-Value="Booking.CheckInDate" class="f
orm-control" />
        <ValidationMessage For="@(() => Booking.CheckInDate)"
/>
    </div>
    <div class="mb-3">
        <label class="form-label">Check-Out Date:</label>
        <InputDate @bind-Value="Booking.CheckOutDate" class
="form-control" />
        <ValidationMessage For="@(() => Booking.CheckOutDat
e)" />
    </div>

    <p class="m-5"></p>

    <div class="mb-3">
        <label class="form-label">Guest Name:</label>
        <InputText @bind-Value="Booking.GuestDetails.Name" cl
ass="form-control" />
        <ValidationMessage For="@(() => Booking.GuestDetails.
Name)" />
    </div>
    <div class="mb-3">
        <label class="form-label">Guest Email:</label>
        <InputText @bind-Value="Booking.GuestDetails.Email" c
lass="form-control" />
        <ValidationMessage For="@(() => Booking.GuestDetails.
Email)" />
    </div>

    <button type="submit" class="btn btn-primary">Submit</but
ton>

    <br /><br />
```

```
</EditForm>

@if (IsSubmitted)
{
    <p class="text-success mt-3 h3">✔ Booking successfully su
bmitted!</p>
}

@code {
    private HotelBooking Booking = new();
    private bool IsSubmitted = false;

    private void HandleValidSubmit()
    {
        IsSubmitted = true;
    }
}
```

Fluent Validation

```
public class SpaBooking
{
    public string GuestName { get; set; }
    public int NumberOfGuests { get; set; }
    public DateTime CheckInDate { get; set; } = DateTime.Now.
AddDays(1);
    public DateTime CheckOutDate { get; set; } = DateTime.No
w.AddDays(2);
    public string Email { get; set; }
}
```

```
public class SpaBookingValidator : AbstractValidator<SpaBooki
ng>
{
    public SpaBookingValidator()
    {
        RuleFor(booking => booking.GuestName)
            .NotEmpty()
            .WithMessage("Guest name is required.");

        RuleFor(booking => booking.NumberOfGuests)
            .InclusiveBetween(1, 5)
            .WithMessage("Number of guests must be 1 to 5.");

        RuleFor(booking => booking.CheckInDate)
            .GreaterThan(DateTime.Now)
            .WithMessage("Check-in date must be in the futur
e.");

        RuleFor(booking => booking.CheckOutDate)
            .GreaterThan(booking => booking.CheckInDate)
            .WithMessage("Check-out date must be after the ch
eck-in date.");

        RuleFor(booking => booking.Email)
            .NotEmpty()
            .WithMessage("Email address is required.")
            .EmailAddress()
            .WithMessage("Invalid email address.");
    }
}
```

```
<h3>Fluent Validation</h3>

<EditForm Model="booking" OnSubmit="HandleSubmit">
```

```
    <FluentValidationValidator @ref="fluentValidationValidato
r" />
    <ValidationSummary />

    <div class="mb-3">
        <label class="form-label">Guest Name:</label>
        <InputText @bind-Value="booking.GuestName" class="for
m-control" />
        <ValidationMessage For="@(() => booking.GuestName)" /
>
    </div>

    <div class="mb-3">
        <label class="form-label">Number of Guests:</label>
        <InputNumber @bind-Value="booking.NumberOfGuests" cla
ss="form-control" />
        <ValidationMessage For="@(() => booking.NumberOfGuest
s)" />
    </div>

    <div class="mb-3">
        <label class="form-label">Check-In Date:</label>
        <InputDate @bind-Value="booking.CheckInDate" class="f
orm-control" />
        <ValidationMessage For="@(() => booking.CheckInDate)"
/>
    </div>

    <div class="mb-3">
        <label class="form-label">Check-Out Date:</label>
        <InputDate @bind-Value="booking.CheckOutDate" class
="form-control" />
        <ValidationMessage For="@(() => booking.CheckOutDat
e)" />
    </div>
```

```
    <div class="mb-3">
        <label class="form-label">Email:</label>
        <InputText @bind-Value="booking.Email" class="form-co
ntrol" />
        <ValidationMessage For="@(() => booking.Email)" />
    </div>

    <button type="submit" class="btn btn-primary">Submit</but
ton>

</EditForm>

@if (IsSubmitted)
{
    <p class="text-success mt-3 h3">✔ Booking successfully su
bmitted!</p>
}

@code {
    private FluentValidationValidator? fluentValidationValida
tor;
    private SpaBooking booking = new();
    private bool IsSubmitted = false;

    private async Task HandleSubmit()
    {
        if (await fluentValidationValidator!.ValidateAsync())
        {
            IsSubmitted = true;
        }
    }
}
```

Watch the Video Walkthrough