**Validierung in ASP.NET Core MVC**

* Schutz vor fehlerhaften oder schädlichen Eingaben
* Vermeidung von Abstürzen und Datenbankfehlern
* Bessere Nutzerführung (z. B. Fehlermeldungen anzeigen)

**Validierung mit Data Annotations im Model**

ASP.NET Core unterstützt **Attribute** aus System.ComponentModel.DataAnnotations wie:

|  |  |  |
| --- | --- | --- |
| Attribut | Bedeutung | Beispiel |
| [Required] | Pflichtfeld | public string Name { get; set; } |
| [StringLength(100)] | Maximale Länge |  |
| [Range(1,100)] | Wertebereich | public int Alter { get; set; } |
| [EmailAddress] | Gültige E-Mail |  |
| [RegularExpression] | Regex-Musterprüfung |  |

**Beispiel:**

public class Benutzer

{

public int Id { get; set; }

[Required(ErrorMessage = "Name ist Pflichtfeld.")]

[StringLength(50, ErrorMessage = "Name darf max. 50 Zeichen haben.")]

public string Name { get; set; }

[Required]

[EmailAddress(ErrorMessage = "Ungültige E-Mail.")]

public string Email { get; set; }

[Range(18, 99, ErrorMessage = "Alter muss zwischen 18 und 99 sein.")]

public int Alter { get; set; }

}

**Controller: Validierung prüfen**

Im Controller prüft man mit

if (!ModelState.IsValid)

{

return View(model);

}

Nur wenn das Model valide ist, wird die Weiterverarbeitung (z. B. speichern) durchgeführt.

**View: Validierung anzeigen**

In Razor-Views verwendet man Hilfsmethoden, die Fehlermeldungen und Input-Felder mit Validierung kombinieren.

**Beispiel:**

<form asp-action="Create" method="post">

<div class="form-group">

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

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

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

</div>

<div class="form-group">

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

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

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

</div>

<button type="submit" class="btn btn-primary">Absenden</button>

</form>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

**Client- & Server-Validierung**

* **Clientseitig**: Browser prüft Eingaben **vor dem Absenden** (via **jQuery** Unobtrusive Validation)
* **Serverseitig**: Controller prüft ModelState – wichtig, da Client-Seite manipuliert werden kann

ASP.NET Core generiert automatisch passende HTML5-Attribute und JavaScript für Client-Validation, wenn man jquery.validate & jquery.validate.unobtrusive einbindet (Standard in MVC-Projekten).

**\_ValidationScriptsPartial.cshtml**

Dieses Partial ist standardmäßig in neuen MVC-Projekten enthalten und bindet die JavaScript-Bibliotheken ein.

<script src="~/lib/jquery-validation/dist/jquery.validate.min.js"></script>

<script src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.min.js"></script>

|  |  |  |
| --- | --- | --- |
| Thema | Zweck | Beispiel |
| Data Annotations | Modell-Validierung definieren | [Required], [Range] |
| ModelState | Validierungsstatus im Controller prüfen | if (!ModelState.IsValid) ... |
| asp-validation-for | Fehlermeldungen in View anzeigen | <span asp-validation-for="..."> |
| Client-Validierung | Nutzerfeedback ohne Server-Roundtrip | jQuery Unobtrusive Validation |