Curs 6

Validarea datelor
Autentificare si Autorizare

DRY

- Un principiu cheie al dezvoltării software se numește DRY (Don’t repeat yourself).
- Razor Pages încurajează dezvoltarea în care funcționalitatea este specificată o singură dată și se reflectă în întreaga aplicație.
- DRY poate ajuta:
  - Reduceți cantitatea de cod într-o aplicație.
  - Faceți codul mai puțin predispus la erori și mai ușor de testat și întreținut.
- Suportul de validare oferit de Razor Pages și Entity Framework este un bun exemplu al principiului DRY: Regulile de validare sunt specificate declarativ într-un singur loc, în clasa modelului. Regulile sunt aplicate peste tot în aplicatie

DataAnnotations

- Namespace-ul System.ComponentModel.DataAnnotations furnizează:
  - Un set de atribute de validare încorporate care sunt aplicate declarativ unei clase sau proprietăți.
  - Atribute de formatare precum
    - [DataType] care ajută la formatare și nu oferă nicio validare;
    - [Column ("Pret", TypeName = "decimal(6, 2)")]

Atribute de validare

```
public class Book {
public int Id { get; set; }
[StringLength(60, MinimumLength = 3)]
[Required]
[RegularExpression(@"^[A-Z]+[a-zA-Z\s]*$")]
public string Title { get; set; }
[DataType(DataType.Date)]
public DateTime PublishingDate { get; set; }
[Range(1, 100)]
[Column(TypeName = "decimal(6, 2)")]
 public decimal Price { get; set; }
}
```

- [Required] și [MinimumLength] indică faptul că o proprietate trebuie să aibă o valoare. Nimic nu împiedică un utilizator să introducă spațiu alb pentru a satisface această validare.
- [RegularExpression] este folosit pentru a limita caracterele care pot fi introduse. Title: Trebuie să folosească numai litere. Prima literă trebuie să fie majusculă. Spațiile albe sunt permise în timp ce numerele și caracterele speciale nu sunt permise.
- Atributul [Range] constrânge introducerea unei valori într-un interval specificat.
- Atributul [StringLength] poate seta o lungime maximă a unei proprietăți și opțional lungimea minimă a acesteia.
- Tipurile de valori, cum ar fi decimal, int, float, DateTime, sunt obligatorii în mod inerent și nu au nevoie de atributul [Required].

[DataType]

- Ofera indicii pentru ca motorul de vizualizare să formateze datele.
- Utilizați atributul [RegularExpression] pentru a valida formatul datelor.
- Atributul [DataType] este utilizat pentru a specifica un tip de date care este mai specific decât tipul intrinsec al bazei de date.
- Atributele [DataType] nu sunt atribute de validare
- Poate furniza un selector de dată DataType.Date în browserele care acceptă HTML5.
- DataType.Date nu specifică formatul datei care este afișată. În mod implicit, câmpul de date este afișat conform formatelor implicite bazate pe CultureInfo a serverului.

Mesaje de eroare

- Atributele de validare permit specificarea unui mesaj de eroare care va fi afisat pentru input invalid

```
[StringLength(8, ErrorMessage = "Name length can't be more than 8.")]
```

```
[StringLength(8, ErrorMessage = "{0} length must be between {2} and {1}.", MinimumLength = 6)
public string Name { get; set; }
```

"Name length must be between 6 and 8.".

Validare Client - side

- Previne trimiterea formularului daca formularul nu este valid
- Previne round-trip care nu este necesar la server in cazul in care exista erori de input la un formular
- Scripturile specificate in *_ValidationScriptsPartial.cshtml* asigura support pentru validarea de tip client-side

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

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

jQuery Unobtrusive Validation

- Scriptul jQuery Unobtrusive Validation este o librarie de front-end custom Microsoft bazat pe pluginul jQuery Validation
- Fara aceasta librarie ar trebui sa realizam logica de validare atat server-side cat si client-side
- TagHelpers utilizeaza atribute de validare ale proprietatilor modelului si afiseaza attribute data- HTML5 pentru elementele din formular care necesita validare
- jQuery Unobtrusive Validation parseaza atributele data- si le trimite la jQueryValidation

CSHTML

```
<div class="form-group">

    <label asp-for="Book.Title" class="control-label"></label>
    <input asp-for="Book.Title" class="form-control" />
    <span asp-validation-for="Book.Title" class="text-danger"></span>
</div>
```

- HTML

```
<div class="form-group"> <label class="control-label" for="Book_Title">Book Title</label> <input class="form-control" type="text" data-val="true" data-val-regex="The field Book Title must match the regular expression &#x27;^[A-Z]&#x2B;[a-zA-Z0-9\s.]*$&#x27;." data-val-regex-pattern="^[A-Z]&#x2B;[a-zA-Z0-9\s.]*$" data-val-required="The Book Title field is required." id="Book_Title" name="Book.Title" value="" /> <span class="text-danger field-validation-valid" data-valmsg-for="Book.Title" data-valmsg-replace="true"></span> </div>
```

Atribute predefinite

- [CreditCard]: valideaza ca proprietatea respecta formatul unui card de credit. Necesita validari aditionale cu jQuery
- [EmailAddress]: valideaza ca proprietatea respecta formatul unei adrese de email.
- [Phone]: valideaza ca proprietatea respecta formatul unui numar de telefon
- [Url]: valideaza ca proprietatea respecta formatul unui URL

Atribute pentru validare custom

```
public class Book {
public int Id { get; set; }
[StringLength(60, MinimumLength = 3)]
[Required]
[RegularExpression(@"^[A-Z]+[a-zA-Z\s]*$")]
public string Title { get; set; }
[ClassicBook(1960)]
[DataType(DataType.Date)]
public DateTime PublishingDate { get; set; }
[Range(1, 100)]
[Column(TypeName = "decimal(6, 2)")]
 public decimal Price { get; set; }
}
```

Atribute custom

- Mostenim clasa ValidationAttribute
- Suprascriem metoda IsValid
- Argumente ale metodei IsValid:
  - Value – inputul care trebuie validat
  - ValidationContext – obiect care ofera informatii despre instanta modelului
- Ruleaza pe server

Atribute custom - exemplu

```
public class ClassicBookAttribute : ValidationAttribute
 {
        public ClassicBookAttribute(int year)
        { Year = year;  }
        public int Year { get; }
        public string GetErrorMessage() => $"Classic books must have a publishing year no later than {Year}.";
protected override ValidationResult IsValid(object value, ValidationContext validationContext)
{
var book = (Book)validationContext.ObjectInstance;
 var releaseYear = ((DateTime)value).Year;
if (releaseYear > Year)
{ return new ValidationResult(GetErrorMessage()); }
return ValidationResult.Success;
 }}
```

Validation TagHelpers

- Exista doi taghelpers pentru validare:
  - Validation message tag helper – afiseaza un mesaj de validare pentru o singura proprietate din model
  - Validation summary tag helper – specifica daca validarea se va face pe baza modelului (ModelOnly)/tipurile setate in baza de date (All)
- Input Tag Helper adauga client-side validation bazate pe adnotarile de la nivelul modelului
- Validarea se face la nivel de server daca se dezactiveaza JavaScript

Tag Helper pentru validare

- Validation Message Tag Helper
  - afiseaza un mesaj de validare pentru proprietatea aferenta din modelul nostru
  - este utilizat cu atributul asp-validation-for a unui element HTML
  - in general il utilizam dupa un tag helper input pentru acceasi proprietate, pentru a afisa erorile de validare langa input-ul care a cauzat eroarea

```
<input asp-for="Book.Author" />
<span asp-validation-for="Book.Author"></span>
```

- Validation Summary Tag Helper afiseaza un rezumat al erorilor de validare

```
<div asp-validation-summary="ModelOnly"></div>
```

- Validarea se realizeaza pe baza constrangerilor si atributelor adnotarilor in clasele din model

Autorizarea

- Procesul care determina ce actiuni ii sunt premise utilizatorului

- Independenta de autentificare dar necesita un mecansim de autentificare

Identity

- ASP.NET Core Identity:
  - API care ofera functionalitati de tip login
  - Gestioneaza utilizatori, parole, roluri, confirmare email etc.
  - Configurat sa utilizeze o baza de date SQL Server pentru a stoca useri, parole si date de profil
  - Sub forma unei librarii de clase Razor

Areas

- Reprezinta un feature ASP.NET utilizat pentru organizarea unor functionalitati relationate intr-un singur grup ca o structura de foldere
- Ofera o modalitate de partitionare a unei aplicatii web in grupuri functionale mai mici, fiecare cu propriul set de pagini razor, controller, view-uri si modele
- Reprezinta efectiv o structura in interiorul unei aplicatii – pentru aplicatii complexe e avantajos sa partitionam aplicatia in functionalitati (arii) de nivel inalt
- Ex. Aplicatie de e-commerce – checkout, billing si search fiecare cu propria arie care contine clase si paginile aferente

Configurarea Lockout

```
public async Task<IActionResult> OnPostAsync(string returnUrl = null) {
if (ModelState.IsValid) {
 var result = await _signInManager.PasswordSignInAsync(Input.Email, Input.Password, Input.RememberMe, lockoutOnFailure: false);
…
}
```

Optiuni Lockout

- Program.cs

```
builder.Services.Configure<IdentityOptions>(options => {
    options.Lockout.DefaultLockoutTimeSpan = TimeSpan.FromMinutes(5);
    options.Lockout.MaxFailedAccessAttempts = 5;
});
```

- DefaultLockoutTimeSpan – perioada pentru care este blocat un utilizator cand apare lockout
- MaxFailedAccessAttempts – numarul de incercari permise pana la lockout

Configurare Parola

- Implicit – Identity solicita ca parola sa contina un caracter litera mare, un caracter litera mica, un numar, si un carcater non-alfanumeic
- Lungimea trebuie sa fie minim sase caractere – atributul [StringLenght] din InputModel (Areas/Identity/Pages/Account/Register.cshtml.cs si ResetPassword.cshtml.cs)
- PasswordOptions in Program.cs.

Optiuni Parola

Program.cs

```
builder.Services.Configure<IdentityOptions>(options => {
    options.Password.RequireDigit = true;
    options.Password.RequireLowercase = true;
    options.Password.RequireNonAlphanumeric = true;
    options.Password.RequireUppercase = true;
    options.Password.RequiredLength = 6;});
```

Configurare User

```
builder.Services.Configure <IdentityOptions>(options => {
// Default User settings.
options.User.AllowedUserNameCharacters =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123
456789-._@+";
options.User.RequireUniqueEmail = false; });
```

Autorizarea

- Tipuri de autorizare in Pagini Razor:
  - Conventie
  - Atributul [Authorize]

Autorizarea prin conventie

- In fisierul Program.cs

```
builder.Services.AddRazorPages(options =>
{
    options.Conventions.AuthorizeFolder("/Books");
    options.Conventions.AllowAnonymousToPage("/Books/Index");
    options.Conventions.AllowAnonymousToPage("/Books/Details");
    options.Conventions.AuthorizeFolder("/Members", "AdminPolicy");
});
```

- Specificarea unei politici

```
builder.Services.AddAuthorization(options =>
{
    options.AddPolicy("AdminPolicy", policy => policy.RequireRole("Admin"));
});
```

Combinare autorizare cu acces anonim

- Valid

```
.AuthorizeFolder("/Books").AllowAnonymousToPage ("/Books/Index");
```

- Invalid

```
.AllowAnonymousToFolder("/Books").AuthorizePage ("/Books/Index");
```

Autorizarea prin atribut

- Aplicarea atributului Authorize la pagina

```
[Authorize(Roles = “Admin")]
public class EditModel : PageModel
{…
public async Task<IActionResult> OnPostAsync(){ }
…
}
```

- Nu se poate aplica la nivel de metoda, doar la nivel de pagina