

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
КІЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. ТАРАСА ШЕВЧЕНКА
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра мережевих та інтернет технологій

СУЧАСНІ ІНТЕРНЕТ ТЕХНОЛОГІЇ

**МОДЕЛІ ВІДОБРАЖЕННЯ. ВАЛІДАЦІЯ ДАНИХ У
ЗАСТОСУНКУ ASP.NET CORE**

Лабораторне заняття № 6

Черкун Єви Сергійвни

Хід виконання роботи:

Завдання 1. Ознайомитися з теоретичними основами валідації ([сайт Microsoft](#)).

Валідація — це процес перевірки правильності та відповідності даних певним правилам перед їх обробкою або збереженням. В ASP.NET Core вона поділяється на клієнтську та серверну:

1. Серверна валідація – перевірка даних на сервері після надсилання форми, гарантує безпеку та правильність обробки.
2. Клієнтська валідація – перевірка даних у браузері до відправки на сервер, підвищує зручність для користувача та зменшує навантаження на сервер.
3. Атрибути валідації – вбудовані правила, які можна застосовувати до властивостей моделей (наприклад, [Required], [StringLength], [Range]).
4. Валідаційний pipeline – послідовність кроків, через які проходять дані моделі: перевірка атрибутів, повідомлення про помилки, відображення у вигляді ValidationSummary або поруч з полем форми (asp-validation-for).

Завдання 2. Створити модель з щонайменше 5 властивостями, використовуючи стандартні атрибути валідації

1. [Required]
2. [StringLength]
3. [EmailAddress]
4. [Range]
5. [Compare]

Приклад наведений нижче на рисунку 6.1:

```

9  {
10     [Required(ErrorMessageResourceType = typeof(ValidationMessages), ErrorMessageResourceName = "FullName_Required")]
11     [StringLength(50, MinimumLength = 2, ErrorMessageResourceType = typeof(ValidationMessages), ErrorMessageResourceName
12     [Display(Name = "Повне ім'я")]
13     3 references
14     public string FullName { get; set; } = string.Empty;
15
16     [Required(ErrorMessageResourceType = typeof(ValidationMessages), ErrorMessageResourceName = "Email_Required")]
17     [EmailAddress(ErrorMessageResourceType = typeof(ValidationMessages), ErrorMessageResourceName = "Email_EmailAddress")]
18     [Remote(action: "VerifyEmail", controller: "Validation", ErrorMessage = "Цей email вже зареєстрований")]
19     [Display(Name = "Email")]
20     4 references
21     public string Email { get; set; } = string.Empty;
22
23     [Required(ErrorMessageResourceType = typeof(ValidationMessages), ErrorMessageResourceName = "Age_Required")]
24     [Range(18, 120, ErrorMessageResourceType = typeof(ValidationMessages), ErrorMessageResourceName = "Age_Range")]
25     [Display(Name = "Вік")]
26     3 references
27     public int? Age { get; set; }
28
29     [Required(ErrorMessageResourceType = typeof(ValidationMessages), ErrorMessageResourceName = "Password_Required")]
30     [StringLength(100, MinimumLength = 8, ErrorMessageResourceType = typeof(ValidationMessages), ErrorMessageResourceName
31     [DataType(DataType.Password)]
32     [Display(Name = "Пароль")]
33     3 references
34     public string Password { get; set; } = string.Empty;
35
36     [Required(ErrorMessageResourceType = typeof(ValidationMessages), ErrorMessageResourceName = "ConfirmPassword_Required")]
37     [DataType(DataType.Password)]
38     [Compare("Password", ErrorMessageResourceType = typeof(ValidationMessages), ErrorMessageResourceName = "ConfirmPasswo
39     [Display(Name = "Підтвердження пароля")]
40     3 references
41     public string ConfirmPassword { get; set; } = string.Empty;

```

Рисунок 6.1 – Створена ContactViewModel із атрибутами валідації

Завдання 3. Реалізувати серверну валідацію у контролері через `ModelState.IsValid`. Забезпечити повернення помилок у View.

```

[HttpPost]
[ValidateAntiForgeryToken]
0 references
public IActionResult Index(ContactViewModel model)
{
    if (!ModelState.IsValid)
    {
        _logger.LogWarning("Форма зворотного зв'язку містить помилки валідації");
        return View(model);
    }

    _logger.LogInformation("Форма зворотного зв'язку успішно оброблена для користувача");

    return RedirectToAction(nameof(Success));
}

```

Рисунок 6.2 - Обробка відправки форми зворотного зв'язку з серверною валідацією

```

@model WebAppCore.ViewModels.ContactViewModel

<h1>@ ViewData["Title"]</h1>

<h4>Заповніть форму для зворотного зв'язку</h4>
<hr />

<div class="row">
    <div class="col-md-6">
        <form asp-action="Index" method="post">
            @Html.AntiForgeryToken()

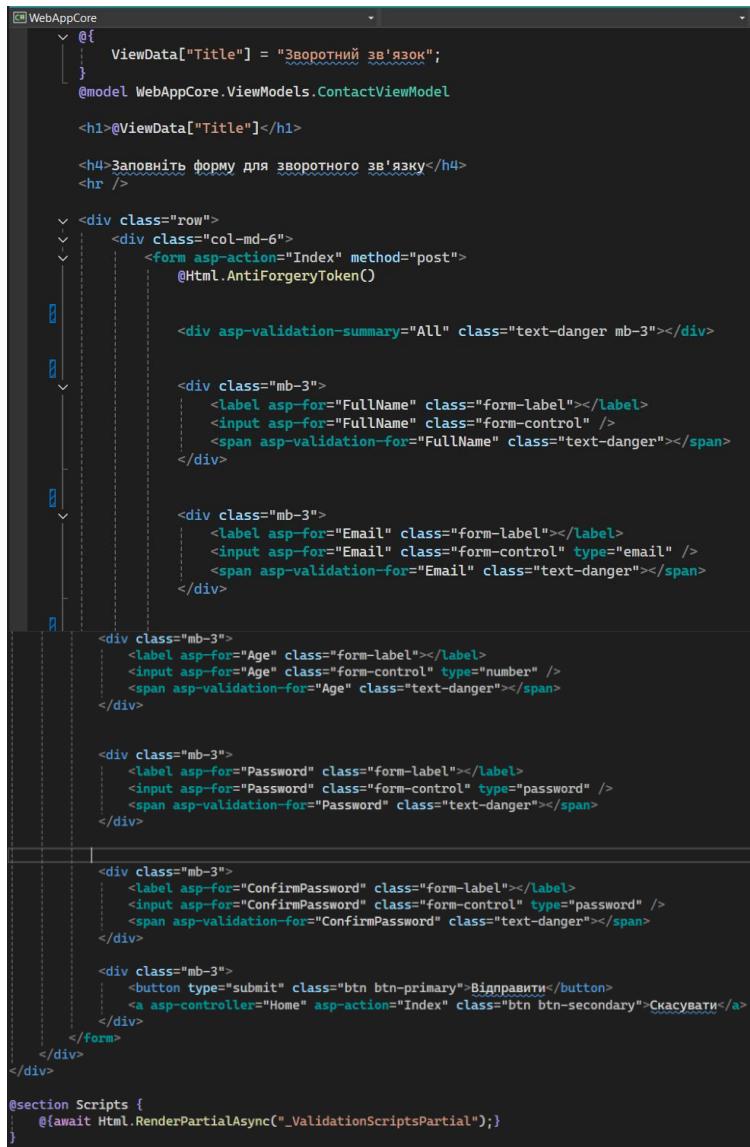
            <div asp-validation-summary="All" class="text-danger mb-3"></div>

```

Рисунок 6.3 - Загальний summary помилок валідації у View

Завдання 4. Налаштувати клієнтську валідацію у Razor View: `asp-validation-for`, `asp-validation-summary`, підключення `_ValidationScriptsPartial`.

ASP.NET Core читає атрибути валідації з ContactViewModel та генерує HTML з data-атрибутами (`data-val`, `data-val-required`, `data-val-length` тощо). jQuery Validation автоматично читає ці атрибути та налаштовує клієнтську валідацію без додаткового JavaScript коду. Валідація виконується при введенні даних та при відправці форми.



The screenshot shows the Contact.cshtml file in Visual Studio. The code defines a form for a contact submission. It includes validation for the FullName, Email, Age, and Password fields using the `asp-validation-for` attribute. The `asp-validation-summary="All"` attribute is used to display validation errors at the top of the form. The `_ValidationScriptsPartial` section at the bottom indicates the use of a partial view for client-side validation scripts.

```
WebAppCore
  @{
    ViewData["Title"] = "Зворотний зв'язок";
  }
  @model WebAppCore.ViewModels.ContactViewModel

  <h1>@ViewData["Title"]</h1>
  <h4>Заповніть форму для зворотного зв'язку</h4>
  <hr />

  <div class="row">
    <div class="col-md-6">
      <form asp-action="Index" method="post">
        @Html.AntiForgeryToken()

        <div asp-validation-summary="All" class="text-danger mb-3"></div>

        <div class="mb-3">
          <label asp-for="FullName" class="form-label"></label>
          <input asp-for="FullName" class="form-control" />
          <span asp-validation-for="FullName" class="text-danger"></span>
        </div>

        <div class="mb-3">
          <label asp-for="Email" class="form-label"></label>
          <input asp-for="Email" class="form-control" type="email" />
          <span asp-validation-for="Email" class="text-danger"></span>
        </div>

        <div class="mb-3">
          <label asp-for="Age" class="form-label"></label>
          <input asp-for="Age" class="form-control" type="number" />
          <span asp-validation-for="Age" class="text-danger"></span>
        </div>

        <div class="mb-3">
          <label asp-for="Password" class="form-label"></label>
          <input asp-for="Password" class="form-control" type="password" />
          <span asp-validation-for="Password" class="text-danger"></span>
        </div>

        <div class="mb-3">
          <label asp-for="ConfirmPassword" class="form-label"></label>
          <input asp-for="ConfirmPassword" class="form-control" type="password" />
          <span asp-validation-for="ConfirmPassword" class="text-danger"></span>
        </div>

        <div class="mb-3">
          <button type="submit" class="btn btn-primary">Відправити</button>
          <a asp-controller="Home" asp-action="Index" class="btn btn-secondary">Скасувати</a>
        </div>
      </form>
    </div>
  </div>

  @section Scripts {
    @await Html.RenderPartialAsync("_ValidationScriptsPartial");
}
```

Рисунок 6.4 - Налаштування клієнтської валідації у Razor View

Завдання 5. Реалізувати Remote Validation для перевірки унікальності email або логіна. Створити відповідний метод у контролері.

Remote Validation - це тип валідації, при якому перевірка виконується на сервері через асинхронний HTTP-запит, але інтегрується з клієнтською валідацією.

Метод контролера для Remote Validation:

```
[AcceptVerbs("GET", "POST")]
0 references
public async Task<IActionResult> VerifyEmail(string email)
{
    if (string.IsNullOrEmpty(email))
    {
        return Json("Email є обов'язковим полем");
    }

    var user = await _userManager.FindByEmailAsync(email);
    if (user != null)
    {
        return Json("Цей email вже зареєстрований");
    }

    return Json(true);
}
```

Рисунок 6.5 - Remote Validation для перевірки унікальності

Валідація при введенні:

Зворотний зв'язок

Заповніть форму для зворотного зв'язку

Повне ім'я

Поле 'Повне ім'я' повинно містити від 2 до 50 символів

Email

Цей email вже зареєстрований

Вік

Вік повинен бути від 18 до 120 років

Пароль

Пароль повинен містити від 8 до 100 символів

Підтвердження пароля

Пароль та підтвердження пароля не співпадають

Рисунок 6.6 – Виконання валідації при заповненні форми (asp-validation-for)

Валідація при відправленні форми:

Зворотний зв'язок

Заповніть форму для зворотного зв'язку

- Поле 'Повне ім'я' є обов'язковим для заповнення
- Поле 'Email' є обов'язковим для заповнення
- Поле 'Вік' є обов'язковим для заповнення
- Поле 'Пароль' є обов'язковим для заповнення
- Поле 'Підтвердження пароля' є обов'язковим для заповнення

Повне ім'я

Поле 'Повне ім'я' є обов'язковим для заповнення

Email

Поле 'Email' є обов'язковим для заповнення

Вік

Поле 'Вік' є обов'язковим для заповнення

Пароль

Рисунок 6.7 – Виконання валідації при відправці форми (asp-validation-summary)

Зворотний зв'язок

Заповніть форму для зворотного зв'язку

Повне ім'я

€

Поле 'Повне ім'я' повинно містити від 2 до 50 символів

Email

evacherkun1@gmail.com

Цей email вже зареєстрований

Вік

1

Вік повинен бути від 18 до 120 років

Рисунок 6.8 – Відповідь сервера якщо користувач зареєстрований

Зворотний зв'язок

Заповніть форму для зворотного зв'язку

Поле 'Повне ім'я' є обов'язковим для заповнення

Поле 'Email' є обов'язковим для заповнення

Поле 'Пароль' є обов'язковим для заповнення

Поле 'Підтвердження пароля' є обов'язковим для заповнення

Повне ім'я

Поле 'Повне ім'я' є обов'язковим для заповнення

Email

gleeforever79@gmail.com

Вік

Поле 'Вік' є обов'язковим для заповнення

Пароль

Поле 'Пароль' є обов'язковим для заповнення

Підтвердження пароля

Поле 'Підтвердження пароля' є обов'язковим для заповнення

Відправити Скасувати

Рисунок 6.9 – Відповідь сервера якщо email унікальний

Завдання 6. Реалізувати залежну Remote Validation з використанням AdditionalFields. (адаптувати завдання для своєї предметної області).

```
[AcceptVerbs("GET", "POST")]
public IActionResult VerifyCookingTime(int? cookingTime, string
difficulty)
{
    if (!cookingTime.HasValue)
    {
        return Json(true);
    }

    if (string.IsNullOrWhiteSpace(difficulty))
    {
        return Json(true);
    }

    var difficultyRules = new Dictionary<string, (int min, int
max)>
    {
        { "Easy", (5, 30) },
        { "Medium", (15, 60) },
        { "Hard", (30, 180) }
    };

    var normalizedDifficulty = difficulty.Trim();

    if (difficultyRules.TryGetValue(normalizedDifficulty, out var
rule))
    {
        if (cookingTime.Value >= rule.min && cookingTime.Value <=
rule.max)
        {
            return Json(true);
        }
        else if (cookingTime.Value < rule.min)
        {
            return Json($"Для
{GetDifficultyName(normalizedDifficulty)} рецепту час приготування повинен
бути не менше {rule.min} хвилин");
        }
        else
        {
            return Json($"Для
{GetDifficultyName(normalizedDifficulty)} рецепту час приготування повинен
бути не більше {rule.max} хвилин");
        }
    }
}

return Json("Невідома складність рецепту");
```

```

    }

    private string GetDifficultyName(string difficulty)
    {
        return difficulty switch
        {
            "Easy" => "простого",
            "Medium" => "середнього",
            "Hard" => "складного",
            _ => difficulty
        };
    }
}

```

Додати рецепт

Заповніть форму для додавання нового рецепту

Назва рецепту

Поле 'Назва рецепту' повинно містити від 3 до 200 символів

Опис рецепту



Поле 'Опис рецепту' повинно містити від 10 до 2000 символів

Категорія



Складність



Час приготування буде перевірено залежно від обраної складності

Час приготування (хвилин)

Рисунок 6.10 – Приклад валідації при додаванні рецепту

Завдання 7. Продемонструвати локалізацію повідомлень про помилки валідації через ресурсні файли.

Клас ValidationMessages використовується для централізованого зберігання та отримання локалізованих повідомлень валідації з ресурсних файлів. Він забезпечує підтримку багаторівністі для атрибутів валідації в ASP.NET Core.

```
public class ValidationMessages
{
    private static ResourceManager GetResourceManager()
    {
        var culture = CultureInfo.CurrentCulture;

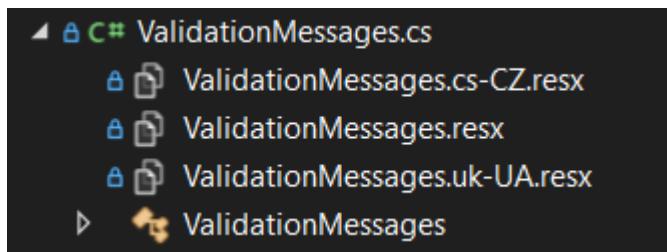
        return new ResourceManager("WebAppCore.Resources.ValidationMessages",
            typeof(ValidationMessages).Assembly);
    }

    public static string FullName_Required =>
        GetResourceManager().GetString("FullName_Required") ?? "FullName_Required";

    public static string FullName_StringLength =>
        GetResourceManager().GetString("FullName_StringLength") ?? "FullName_StringLength";
}
```

Рисунок 6.11 – ValidationMessages клас

У класі використовується ResourceManager, який завантажує ресурсні файли:



6.12 – Ресурсні файли для трьох мов

Приклад локалізації:

The screenshot shows a web application interface for creating a recipe. The URL in the browser bar is <https://localhost:7081/Recipe/Create?culture=en-US>. The page title is "Додати рецепт". Below it, a sub-header says "Заповніть форму для додавання нового рецепту". A list of validation errors is displayed:

- The 'Recipe Name' field must contain between 3 and 200 characters
- The 'Description' field is required
- The 'Category' field is required
- The 'Difficulty' field is required
- The 'Cooking Time' field is required
- The 'Ingredients' field is required

The "Name of the recipe" input field contains "W" and has a red validation message: "The 'Recipe Name' field must contain between 3 and 200 characters". The "Description" input field has a red validation message: "The 'Description' field is required". There are also "Category", "Difficulty", "Cooking Time", and "Ingredients" input fields, each with their own validation messages.

Рисунок 6.13 - Локалізація про помилки валідації англійською

The screenshot shows a web page titled "Додати рецепт" (Add Recipe). The URL in the address bar is <https://localhost:7081/Recipe/Create?culture=UK-ua>. The page includes a navigation bar with links like "Головна", "Пости", "Рецепти", "Premium", "Text Claims", "Зворотний зв'язок", and "lblLanguageSwitcher". A list of validation rules is displayed:

- Поле 'Назва рецепту' повинно містити від 3 до 200 символів
- Поле 'Опис рецепту' є обов'язковим для заповнення
- Поле 'Категорія' є обов'язковим для заповнення
- Поле 'Складність' є обов'язковим для заповнення
- Поле 'Час приготування' є обов'язковим для заповнення
- Поле 'Інгредієнти' є обов'язковим для заповнення

Below the validation rules, there are two input fields: "Назва рецепту" (Name of the recipe) containing the letter "e" and "Опис рецепту" (Description of the recipe). Error messages are shown below each field: "Поле 'Назва рецепту' повинно містити від 3 до 200 символів" and "Поле 'Опис рецепту' є обов'язковим для заповнення".

Рисунок 6.14 - Локалізація про помилки валідації українською

Висновок:

У ході роботи було вивчено теоретичні основи валідації в ASP.NET Core та реалізовано всі основні її механізми: серверну й клієнтську валідацію, перевірку через стандартні атрибути моделі, Remote Validation та залежну Remote Validation. Також було налаштовано повернення помилок у View і впроваджено локалізовані повідомлення через ресурсні файли. Це дозволило комплексно засвоїти принципи перевірки даних у веб-застосунках.