МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА МЕРЕЖЕВИХ ТА ІНТЕРНЕТ ТЕХНОЛОГІЙ

Лабораторне заняття №6

МОДЕЛІ ВІДОБРАЖЕННЯ. ВАЛІДАЦІЯ ДАНИХ У ЗАСТОСУНКУ ASP.NET CORE

Виконав студент групи: МІТ-41
Кухарчук Богдан Петрович

Хід роботи

1. Створення моделі відображення (BookViewModel)
У проєкті було створено спеціальний клас BookViewModel у папці Models, який використовується для передачі даних з форми створення книги. До властивостей моделі додано атрибути валідації з простору імен System.ComponentModel.DataAnnotations. Використано такі атрибути:

[Required] — для обов'язкових полів (Назва, Email, Рік, Ціна).
[StringLength] — для обмеження довжини назви (3-100 символів).
[EmailAddress] — для перевірки формату електронної пошти.
[Compare] — для підтвердження email (порівняння двох полів).
[Range] — для обмеження року видання (1900–2025) та ціни.
[Remote] — для перевірки унікальності назви та залежної валідації знижки.

```
namespace WebApplication1.Models
{
    public class BookViewModel
    {
        // [Required] – Обов'язкове поле
        // [StringLength] – Обмеження довжини
        // [Remote] – Валідація через запит на сервер (перевірка унікальності)

        [Required(ErrorMessage = "Назва книги є обов'язковою.")]
        [StringLength(100, MinimumLength = 3, ErrorMessage = "Довжина назви має бути від 3 до 100 символів.")]
        [Remote(action: "VerifyTitle", controller: "Books", ErrorMessage = "Така назва книги вже існує.")]
        [Display(Name = "Назва книги")]
        public string Title { get; set; }

        // [EmailAddress] – Перевірка формату email
        [Required(ErrorMessage = "Email автора обов'язковий.")]
        [EmailAddress(ErrorMessage = "Некоректний формат Email.")]
        [Display(Name = "Email автора")]
        public string AuthorEmail { get; set; }

        // [Compare] – Порівняння з іншим полем
        [Compare("AuthorEmail", ErrorMessage = "Email адреси не співпадають.")]
        [Display(Name = "Підтвердження Email")]
        public string ConfirmAuthorEmail { get; set; }

        // [Range] – Діапазон значень
        [Required]
        [Range(1900, 2025, ErrorMessage = "Рік має бути між 1900 та 2025.")]
        [Display(Name = "Рік видання")]
        public int Year { get; set; }

        [Required]
        [Range(0, 10000, ErrorMessage = "Ціна має бути більше 0.")]
        [Display(Name = "Ціна")]
        public decimal Price { get; set; }

        // Залежна Remote валідація (перевірка залежить від ціни)
        [Remote(action: "VerifyDiscount", controller: "Books", AdditionalFields = "Price", ErrorMessage = "Знижка не може бути більшою за ціну.")]
        [Display(Name = "Знижка")]
        public decimal Discount { get; set; }
    }
}
```

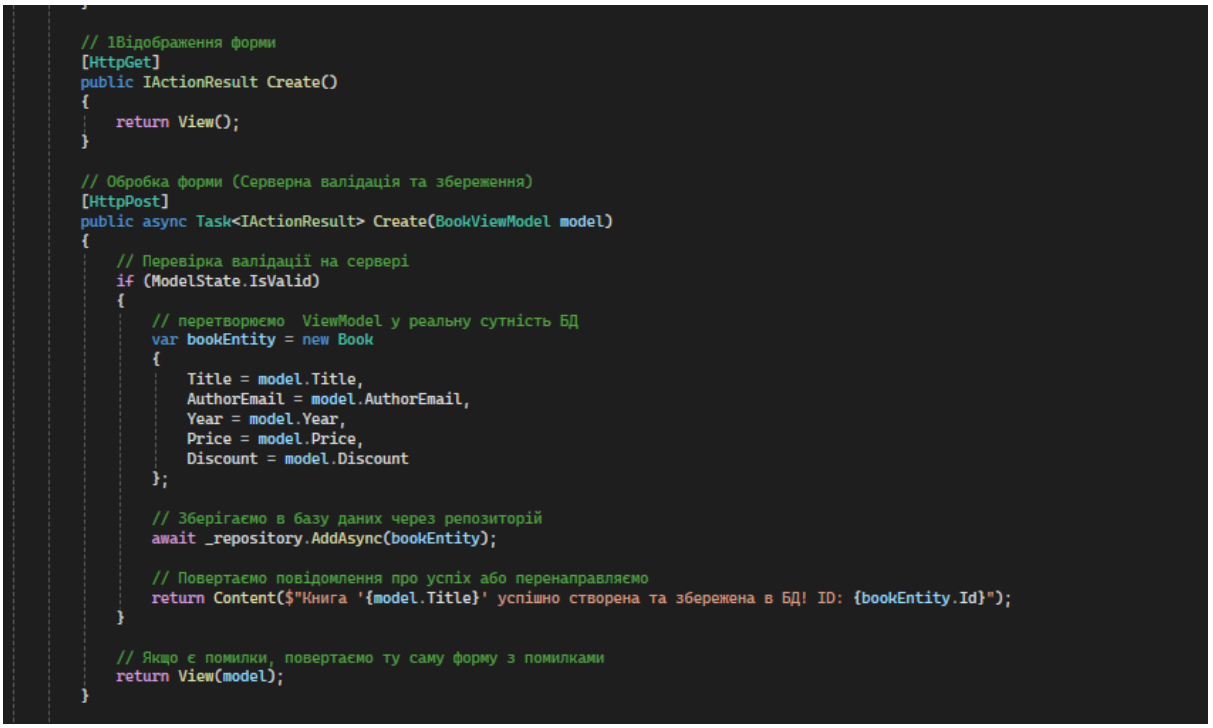
Рисунок 1. Модель BookViewModel з атрибутами валідації.

2. Реалізація контролера (BooksController)
Створено контролер BooksController для обробки запитів, пов'язаних із книгами. Реалізовано методи:

Create (GET): Відображає форму створення.

Create (POST): Приймає дані форми, перевіряє ModelState.IsValid. Якщо дані валідні — створює сутність Book і зберігає її в базу даних через репозиторій. Якщо ні — повертає форму з помилками.

VerifyTitle: Метод для Remote-валідації, який перевіряє в БД, чи існує книга з такою назвою.
VerifyDiscount: Метод для залежної валідації, який перевіряє, чи знижка не перевищує ціну товару.



```
// 1Відображення форми
[HttpGet]
public IActionResult Create()
{
    return View();
}

// Обробка форми (Серверна валідація та збереження)
[HttpPost]
public async Task<IActionResult> Create(BookViewModel model)
{
    // Перевірка валідації на сервері
    if (ModelState.IsValid)
    {
        // перетворюємо  ViewModel у реальну сутність БД
        var bookEntity = new Book
        {
            Title = model.Title,
            AuthorEmail = model.AuthorEmail,
            Year = model.Year,
            Price = model.Price,
            Discount = model.Discount
        };

        // Зберігаємо в базу даних через репозиторій
        await _repository.AddAsync(bookEntity);

        // Повертаємо повідомлення про успіх або перенаправляємо
        return Content($"Книга '{model.Title}' успішно створена та збережена в БД! ID: {bookEntity.Id}");
    }

    // Якщо є помилки, повертаємо ту саму форму з помилками
    return View(model);
}
```

Рисунок 2. Методи Create у контролері BooksController.

```
        // Remote Validation: Перевірка унікальності назви
        [AcceptVerbs("GET", "POST")]
        public async Task<IActionResult> VerifyTitle(string title)
        {
            // Перевіряємо в реальній БД, чи є така книга
            // Використовуємо ExistsAsync, який ми додали в лабораторній №2
            var exists = await _repository.ExistsAsync<Book>(b => b.Title == title);

            if (exists)
            {
                return Json($"Книга з назвою '{title}' вже існує.");
            }

            return Json(true);
        }

        // Dependent Remote Validation: Перевірка знижки залежно від ціни
        [AcceptVerbs("GET", "POST")]
        public IActionResult VerifyDiscount(decimal discount, decimal price)
        {
            if (discount > price)
            {
                return Json("Знижка не може бути більшою за ціну товару.");
            }

            return Json(true);
        }
    }
}
```

Рисунок 3. Методи для Remote Validation у контролері.

3. Налаштування бази даних та міграція

Для збереження даних було створено сутність Book у проєкті даних (WebApplicationData), додано її до контексту WebApplicationDbContext та виконано міграцію бази даних.

```
1   using System.ComponentModel.DataAnnotations;
2   using System.ComponentModel.DataAnnotations.Schema;
3
4   namespace WebApplicationData.Data
5   {
6       public class Book
7       {
8           public int Id { get; set; }
9           public string Title { get; set; }
10          public string AuthorEmail { get; set; }
11          public int Year { get; set; }
12
13          [Column(TypeName = "decimal(18,2)")]
14          public decimal Price { get; set; }
15
16          [Column(TypeName = "decimal(18,2)")]
17          public decimal Discount { get; set; }
18      }
19  }
```

Рисунок 4. Сутність Book для бази даних.

4. Створення представлення з формою (Create.cshtml)
Створено представлення Create.cshtml, яке містить HTML-форму для введення даних. Для відображення повідомлень про помилки використано Tag Helpers:

asp-validation-summary="ModelOnly" — для загальних помилок.
asp-validation-for="PropertyName" — для помилок конкретного поля.

Для забезпечення клієнтської валідації (без перезавантаження сторінки) підключено скрипти _ValidationScriptsPartial (jQuery Validate).

```
<h2>Створити нову книгу</h2>
<hr />

<div class="row">
    <div class="col-md-6">
        <form asp-action="Create">
            <!-- Загальний список помилок -->
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>

            <!-- Назва (Remote Validation) -->
            <div class="mb-3">
                <label asp-for="Title" class="form-label"></label>
                <input asp-for="Title" class="form-control" />
                <span asp-validation-for="Title" class="text-danger"></span>
            </div>

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

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

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

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

            <!-- Discount (Dependent Validation) -->
            <div class="mb-3">
                <label asp-for="Discount" class="form-label"></label>
                <input asp-for="Discount" class="form-control" />
                <span asp-validation-for="Discount" class="text-danger"></span>
            </div>

            <div class="mb-3">
                <input type="submit" value="Створити" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<!-- Підключення скриптів для клієнтської валідації -->
@section Scripts {
    @{
        await Html.RenderPartialAsync("_ValidationScriptsPartial");
    }
}
```

Рисунок 5. Форма створення книги з налаштуваннями клієнтської валідації.

5. Локалізація повідомлень валідації

Для забезпечення багатомовності повідомлень про помилки було створено файли ресурсів. Ключі ресурсів відповідають повідомленням в атрибутах ErrorMessage. Створено файли:

Resources/Models/BookViewModel.uk-UA.resx (українська).
Resources/Models/BookViewModel.en-US.resx (англійська).

Це дозволяє автоматично змінювати мову помилок (наприклад, "The field is required" -> "Поле є обов'язковим") при перемиканні мови сайту.

Файл: Resources/Models/BookViewModel.uk-UA.resx.

Name	en-US (English (United States))	uk-UA (Ukrainian (Ukraine))
Email автора	Author's email	Email автора
Email автора обов'язковий.	Author's email is required.	Email автора обов'язковий для за...
Email адреси не співпадають.	The email addresses do not match.	Адреси електронної пошти не сп...
Довжина назви має бути від 3 до...	The name must be between 3 and...	Довжина назви має бути від 3 до...
Знижка	Discount	Знижка
Знижка не може бути більшою з...	The discount exceeds the price of t...	Знижка перевищує ціну товару!
Назва книги	Book name	Назва книги
Назва книги є обов'язковою.	The book title is required.	Назва книги є обов'язковою! (Це...
Некоректний формат Email.	Please enter a valid email.	Будь ласка, введіть коректну еле...
Підтвердження Email	Email confirmation	Підтвердження Email
Рік видання	Year of publication	Рік видання
Рік має бути між 1900 та 2025.	The year of publication must be b...	Рік видання має бути в межах 19...
Така назва книги вже існує.	This book title already exists in the...	Така назва книги вже існує в базі.
Ціна	Price	Ціна
Ціна має бути більше 0.	The price cannot be zero or negati...	Ціна не може бути нульовою аб...

Рисунок 6. Файл ресурсів з перекладами повідомлень про помилки валідації.

Результат роботи
Продемонстровано роботу валідації на сторінці створення книги.
Клієнтська валідація: При спробі відправити порожню форму помилки з'являються миттєво.
Remote Validation: При введенні назви існуючої книги з'являється повідомлення про дублікат.
Залежна валідація: При введенні знижки, більшої за ціну, з'являється відповідна помилка.

Локалізація: Повідомлення відображаються мовою, обраною користувачем.
Успішне збереження: При введенні коректних даних книга зберігається в базу даних.

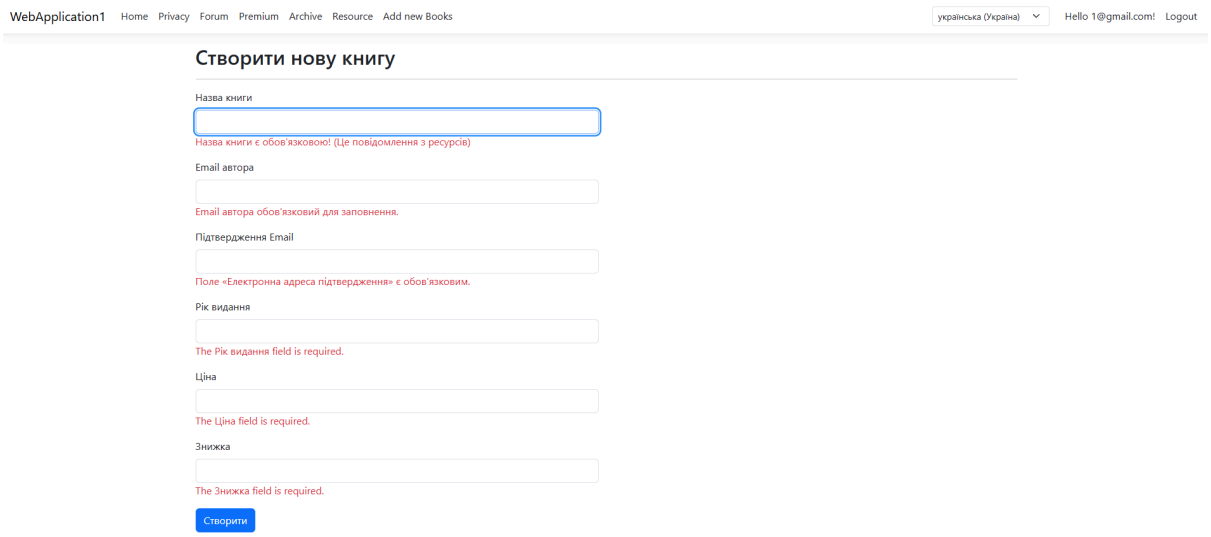


Рисунок 7. Приклад роботи валідації форми (українська локалізація).

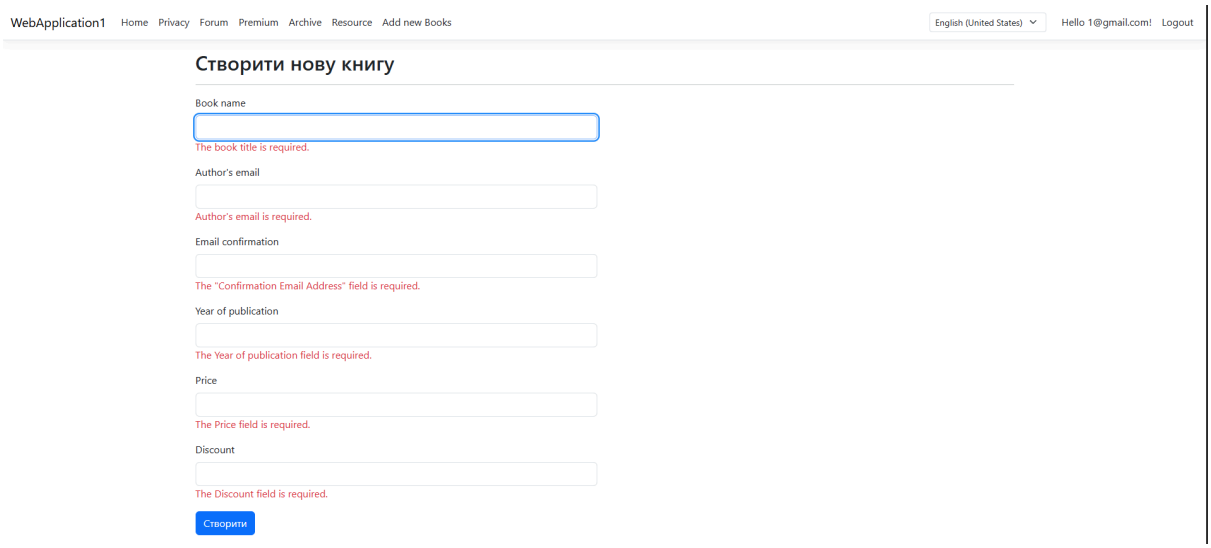


Рисунок 8. Приклад роботи валідації форми (англійська локалізація).

Рисунок 8. Повідомлення про успішне збереження книги.

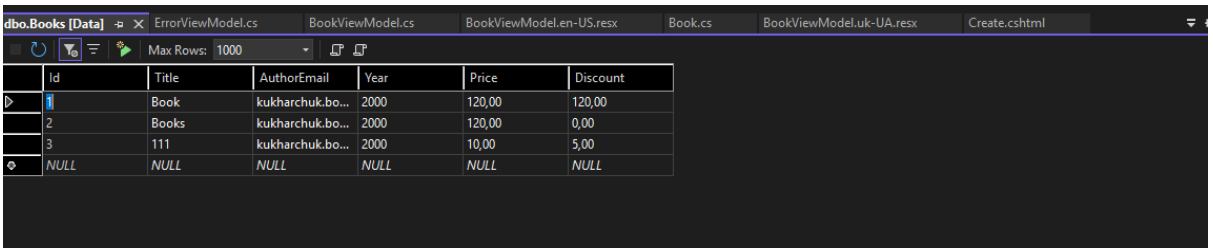


Рисунок 9. Демонстрація БД з збереженими книгами.

Висновок

У ході виконання лабораторної роботи було успішно реалізовано механізми валідації даних у застосунку ASP.NET Core. Створено надійну систему перевірки даних на стороні клієнта та сервера, що забезпечує цілісність інформації та покращує досвід користувача. Застосовано широкий спектр атрибутів валідації, включаючи перевірку типів, діапазонів, форматів та порівняння полів. Особливу увагу приділено реалізації Remote Validation для перевірок, що вимагають звернення до бази даних (унікальність), та залежної валідації для логічного зв'язку між полями. Також було налаштовано локалізацію повідомлень про помилки, що дозволяє адаптувати інтерфейс для користувачів з різних країн. Дані успішно зберігаються в базу даних через репозиторій після проходження всіх перевірок. Усі зміни зафіксовано в репозиторії GitHub.