Міністерство освіти України

Національний технічний університет "ХПІ"

кафедра "Інформатики та інтелектуальної власності"

**Звіт**

**Лабораторна робота 4**

з дисципліни "Проєктування серверних застосунків"

Виконав: студент групи КН-922В

Шершньов М.О.

Перевірив:

Івашко А.В.

Харків 2025

Зміст

[Завдання 3](#_Toc193668890)

[Шаблони textblocks, images, tables, counter, stringprocessing, databasefilter 6](#_Toc193668891)

[Шаблон TextBlocks 6](#_Toc193668892)

[Шаблон Images 6](#_Toc193668893)

[Шаблон Tables 7](#_Toc193668894)

[Шаблон Counter 7](#_Toc193668895)

[Шаблон StringProcessing 8](#_Toc193668896)

[Шаблон DatabaseFilter 9](#_Toc193668897)

[Висновок 10](#_Toc193668898)

**Застосування технології Razor для реалізації інтерфейсної логіки по-дання (View) MVC-застосунків**

**Мета роботи:** ознайомитися із можливостями технології Razor та придбати практику у використанні засобів Razor для оптимального формування та обробки інформації на view

Завдання

1. Виконати перше завдання минулої роботи згідно свого варіанту, створю-ючи текстові блоки на той самій сторінці (поданні), де знаходиться почат кова форма із відповідними параметрами, тобто у застосунку має бути одне подання (Index) та один контролер.

Мій варіант:

Изображение выглядит как текст, снимок экрана, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

1. Виконати друге завдання минулої роботи згідно свого варіанту, створюючи відповідні топології зображень на той самій сторінці (поданні), де знаходиться початкова форма із відповідними параметрами, тобто у застосунку має бути одне подання (Index) та один контролер.
2. Виконати третє завдання минулої роботи згідно свого варіанту, створюючи відповідну таблицю на той самій сторінці (поданні), де знаходиться початкова форма із відповідними параметрами, тобто у застосунку має бути одне подання (Index) та один контролер.
3. Виконати четверте завдання минулої роботи згідно свого варіанту, реалізуючи переходи між поданнями за допомогою одного контролеру, а формування чергового вмісту на одному поданні, тобто у застосунку має бути одне подання (Index) та один контролер.
4. \*Модифікувати минуле завдання додавши відповідні лічильники відвідувань до кожного подання.
5. \*Модифікувати минуле завдання, використовуючи лише один метод од-ного контролеру.
6. Зробити простий (одне подання та один контролер) MVC-застосунок, який вирішує таку задачу: на поданні (Index) в окремому текстовому полі ввести рядок, що містить не менше п'яти слів, а в інше окреме поле кількість разів для повторення обробленого за власним варіантом рядка; нижче на цьому ж поданні має з'явитися вже оброблений рядок, повторе-ний задану кількість разів

Мій варіант:





1. \*В робочій базі даних створити таблицю за наступним зразком

Изображение выглядит как текст, число, Шрифт, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Зробити MVC-застосунок, який вирішує таку задачу: на основному поданні (Index) вміст усієї таблиці та форма з умовою згідно до власного варіанту до відображеної таблиці; друге (або це ж але трохи нижче) подання має відобразити результат виконання умови щодо таблиці

Мій варіант:





Razor-шаблони для роботи:

1. «Текстові блоки» - **TextBlocks**
2. «Зображення» – **Images**
3. «Таблиці» – **Tables**
4. «Лічильник» – **Counter**
5. «Обробка рядка» – **StringProcessing**
6. «Фільтр бази даних» – **DatabaseFilter**

Шаблони textblocks, images, tables, counter, stringprocessing, databasefilter

Шаблон TextBlocks

Шаблон Text Blocks (@{ ... }) у Razor використовується для виконання коду у представленнях.

1. Зчитує дані з форми через Context.Request.Form.
2. Заповнює поля введеними значеннями.
3. Відображає результати, якщо форма відправлена.

@{

ViewData["Title"] = "Анкета відвідувача";

string name = "", phone = "", email = "", visitDate = "", favoriteCuisine = "", desiredDishes = "", reasonForChoosing = "", recommendation = "";

int age = 0;

if (Context.Request.Method == "POST" && Context.Request.HasFormContentType)

{

name = Context.Request.Form["name"].ToString();

phone = Context.Request.Form["phone"].ToString();

email = Context.Request.Form["email"].ToString();

visitDate = Context.Request.Form["visitDate"].ToString();

age = int.TryParse(Context.Request.Form["age"], out int parsedAge) ? parsedAge : 0;

favoriteCuisine = Context.Request.Form["favoriteCuisine"].ToString();

desiredDishes = Context.Request.Form["desiredDishes"].ToString();

reasonForChoosing = Context.Request.Form["reason"].ToString();

recommendation = Context.Request.Form["recommendation"].ToString();

}

}

Шаблон Images

Папка **wwwroot/images** у проєкті ASP.NET Core використовується для зберігання зображень, які можуть відображатися у веб-додатку.

У коді є такі файли:

* **greek-restaurant.jpg** – для грецької кухні
* **italian-restaurant.jpg** – для італійської
* **french-restaurant.jpg** – для французької
* **ukrainian-restaurant.jpg** – для української
* **default-restaurant.jpg** – якщо кухня не обрана

@{

ViewData["Title"] = "Анкета відвідувача";

string name = "", phone = "", email = "", visitDate = "", favoriteCuisine = "", desiredDishes = "", reasonForChoosing = "", recommendation = "";

int age = 0;

if (Context.Request.Method == "POST" && Context.Request.HasFormContentType)

{

name = Context.Request.Form["name"].ToString();

phone = Context.Request.Form["phone"].ToString();

email = Context.Request.Form["email"].ToString();

visitDate = Context.Request.Form["visitDate"].ToString();

age = int.TryParse(Context.Request.Form["age"], out int parsedAge) ? parsedAge : 0;

favoriteCuisine = Context.Request.Form["favoriteCuisine"].ToString();

desiredDishes = Context.Request.Form["desiredDishes"].ToString();

reasonForChoosing = Context.Request.Form["reason"].ToString();

recommendation = Context.Request.Form["recommendation"].ToString();

}

string topologyImage = favoriteCuisine switch

{

"Грецька" => "/images/greek-restaurant.jpg",

"Італійська" => "/images/italian-restaurant.jpg",

"Французька" => "/images/french-restaurant.jpg",

"Українська" => "/images/ukrainian-restaurant.jpg",

\_ => "/images/default-restaurant.jpg"

};

}

Шаблон Tables

У цьому шаблоні використовується елемент <table> для відображення результатів введених користувачем даних.

@if (!string.IsNullOrEmpty(name))

{

<h3 class="mt-4">Результати</h3>

<table class="table table-bordered">

<tr><th>Поле</th><th>Значення</th></tr>

<tr><td><b>Ім'я</b></td><td>@name</td></tr>

<tr><td><b>Телефон</b></td><td>@phone</td></tr>

<tr><td><b>Пошта</b></td><td>@email</td></tr>

<tr><td><b>Дата відвідування</b></td><td>@visitDate</td></tr>

<tr><td><b>Вік</b></td><td>@age</td></tr>

<tr><td><b>Улюблена кухня</b></td><td>@favoriteCuisine</td></tr>

<tr><td><b>Страви в меню</b></td><td>@desiredDishes</td></tr>

<tr><td><b>Причина вибору</b></td><td>@reasonForChoosing</td></tr>

<tr><td><b>Рекомендація</b></td><td>@recommendation</td></tr>

</table>

}

Шаблон Counter

Шаблон counter відповідає за відображення кількості відвідувань сторінки. Саме оновлення лічильника виконується у Controller, використовуючи Session.

@model Task\_5.Models.SurveyModel

@{

ViewData["Title"] = "Анкета відвідувача";

int visitCounter = ViewData["VisitCounter"] as int? ?? 0;

if (Context.Request.Method == "POST" && Context.Request.HasFormContentType)

{

Model.Name = Context.Request.Form["Name"].ToString();

Model.Phone = Context.Request.Form["Phone"].ToString();

Model.Email = Context.Request.Form["Email"].ToString();

Model.VisitDate = Context.Request.Form["VisitDate"].ToString();

Model.Age = int.TryParse(Context.Request.Form["Age"], out int parsedAge) ? parsedAge : 0;

Model.FavoriteCuisine = Context.Request.Form["FavoriteCuisine"].ToString();

Model.DesiredDishes = Context.Request.Form["DesiredDishes"].ToString();

Model.ReasonForChoosing = Context.Request.Form["ReasonForChoosing"].ToString();

Model.Recommendation = Context.Request.Form["Recommendation"].ToString();

}

}

Шаблон StringProcessing

Шаблон stringprocessing обробляє введений рядок, перевіряє, чи містить він щонайменше 5 слів, і повторює його задану кількість разів. Останнє слово виділяється жирним. Якщо слів менше 5, виводиться повідомлення про помилку. Введення даних здійснюється через форму, а результат відображається після обробки.

@{

ViewData["Title"] = "Обробка рядка";

var inputText = "";

var repeatCountStr = "";

var repeatCount = 0;

var processedText = "";

var errorMessage = "";

if (Context.Request.Method == "POST")

{

inputText = Context.Request.Form["inputText"].ToString();

repeatCountStr = Context.Request.Form["repeatCount"].ToString();

repeatCount = int.TryParse(repeatCountStr, out int count) ? count : 0;

var words = inputText.Split(new[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

if (words.Length < 5)

{

errorMessage = "Ви ввели менше п'яти слів.";

}

else if (repeatCount > 0)

{

var lastWord = words[words.Length - 1];

processedText = string.Join(' ', words.Take(words.Length - 1)) + " <strong>" + lastWord + "</strong>";

processedText = string.Join("<br>", Enumerable.Repeat(processedText, repeatCount));

}

}

}

Шаблон DatabaseFilter

Шаблон databasefilter відображає список співробітників із можливістю фільтрації за відділом. Фільтр реалізований через текстове поле у формі, яке дозволяє вводити назву відділу та відображати лише відповідних співробітників. Дані виводяться у вигляді таблиці з колонками: ID, ім'я, зарплата, вік, стать і відділ.

@model List<Employee>

@{

ViewData["Title"] = "Список співробітників";

var departmentFilter = ViewData["DepartmentFilter"]?.ToString();

}

<**link** rel="stylesheet" **href**="~/css/site.css" **asp-append-version**="true">

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

<div class="container mt-4">

<h2 class="mb-3">Список співробітників</h2>

<!-- Фільтр -->

<**form** method="get" class="mb-3">

<label>Фільтр за відділом:</label>

<input type="text" name="departmentFilter" class="form-control" value="@departmentFilter">

<button type="submit" class="btn btn-primary mt-2">Застосувати</button>

</**form**>

<table class="table table-bordered">

<thead class="table-dark">

<tr>

<th>ID</th>

<th>Ім'я</th>

<th>Зарплата</th>

<th>Вік</th>

<th>Стать</th>

<th>Відділ</th>

</tr>

</thead>

<tbody>

@foreach (var emp in Model.Where(e => string.IsNullOrEmpty(departmentFilter) || e.e\_dept.Contains(departmentFilter)))

{

<tr>

<td>@emp.e\_id</td>

<td>@emp.e\_name</td>

<td>@emp.e\_salary</td>

<td>@emp.e\_age</td>

<td>@emp.e\_gender</td>

<td>@emp.e\_dept</td>

</tr>

}

</tbody>

</table>

</div>

Висновок

У ході виконання цієї лабораторної роботи я ознайомився із застосуванням технології Razor для реалізації інтерфейсної логіки у поданні (View) MVC-застосунків. Я реалізував обробку введених користувачем даних безпосередньо у поданні, використовуючи шаблони **textblocks**, **images**, **tables**, **counter**, **stringprocessing** та **databasefilter**.

Шаблони дозволили:

* **textblocks** — виконувати логіку (умови, цикли, вирази) у поданні без залучення контролерів.
* **images** — динамічно змінювати зображення в залежності від вибору користувача.
* **tables** — відображати дані у вигляді структурованих таблиць.
* **counter** — реалізувати підрахунок кількості відвідувань сторінки.
* **stringprocessing** — обробляти введені текстові дані та відображати результат.
* **databasefilter** — здійснювати фільтрацію даних безпосередньо у Razor.

Ця робота допомогла мені освоїти технологію Razor, використання шаблонів для створення динамічних подань і реалізацію інтерфейсної логіки без необхідності обробки даних у контролерах.