Міністерство освіти України

Національний технічний університет "ХПІ"

кафедра "Комп’ютерних наук та інформаційних технологій"

**Звіт**

**Лабораторна робота 1**

з дисципліни "Проєктування серверних застосунків"

Виконав: студент групи КН-922В

Мельник Б. В.

Перевірив:

Івашко А. В.

Харків 2025

**Зміст**

[Завдання 1 1](#_Toc193289124)

[Текст програми: 2](#_Toc193289125)

[Результат роботи програми: 16](#_Toc193289126)

[Алгоритм побудови проєкту і кодів 17](#_Toc193289127)

[Функціональність 19](#_Toc193289128)

[Висновок 19](#_Toc193289129)

[Завдання 2\_1 21](#_Toc193289130)

[Текст програми: 21](#_Toc193289131)

[Результат роботи програми: 32](#_Toc193289132)

[Алгоритм побудови проєкту і кодів 33](#_Toc193289133)

[Функціональність 34](#_Toc193289134)

[Висновок 35](#_Toc193289135)

[Завдання 2\_2 36](#_Toc193289136)

[Текст програми: 36](#_Toc193289137)

[Результат роботи програми: 46](#_Toc193289138)

[Алгоритм побудови проєкту і кодів 47](#_Toc193289139)

[Функціональність 48](#_Toc193289140)

[Висновок 49](#_Toc193289141)

[Завдання 3 50](#_Toc193289142)

[Текст програми: 50](#_Toc193289143)

[Результат роботи програми: 59](#_Toc193289144)

[Алгоритм побудови проєкту і кодів 60](#_Toc193289145)

[Функціональність 62](#_Toc193289146)

[Висновок 62](#_Toc193289147)

[Завдання 4 63](#_Toc193289148)

[Текст програми: 63](#_Toc193289149)

[Результат роботи програми: 72](#_Toc193289150)

[Алгоритм побудови проєкту і кодів 74](#_Toc193289151)

[Функціональність 76](#_Toc193289152)

[Висновок 76](#_Toc193289153)

[Завдання 5 77](#_Toc193289154)

[Результати роботи програми: 87](#_Toc193289155)

[Алгоритм побудови проєкту і кодів 89](#_Toc193289156)

[Функціональність 91](#_Toc193289157)

[Висновок 91](#_Toc193289158)

[Загальний висновок 92](#_Toc193289159)

**Тема:** Створення простих MVC WEB-застосунків засобами ASP.NET.

**Мета роботи:** придбати практику у створенні простих MVC WEB-додатків, заснованих на зв'язках Model - View Controller з невеликою кількістю конт--ролерів та елементів подання і без використання моделей.

**Програмні засоби розробки:** середовище програмування (IDE-Integrated De-velopment Environment) MS Visual Studio, програмний фреймворк ASP.NET версії не нижче 6.0 та дуже бажано використання дизайнерського фреймво-рку BOOTSTRAP, або якогось іншого відповідного аналогу, MySQL Server, або якась аналогічна СУБД.

**Індивідуальні завдання**

**Варіант №12**

# **Завдання 1**

Створити МVC WEB-застосунок, головне подання (Index) якого містить нетипізовану (непов'язаною із моделлю) форму (текстові поля, області, списки, радібатони, чекбокси тощо.) для задання групи параметрів, які через контролер, що викликається з цієї ж форми, передаються до іншого подання, на якому створюються три блоки (абзаци) тексту, котрі відрізняються один від одного за розміром, кольором, вмістом, типом вирівнювання та формою тексту; кольором, зображенням (із його параметрами) фону; розмірами та відступами між блоками. Під час виконання завдання треба врахувати те, що за номером власного варіанту частина інформації (вміст та css-параметри абзаців) має розташовуватися в діалогових об'єктах форми головного подання застосунку, а інша частина css-параметрів має бути зчитана контролером із відповідної таблиці чи таблиць бази даних (БД). База даних та таблиця (ї) мають бути створені заздалегідь і мати будь-яку структуру та вміст у будь-якій системі управління базами даних.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ варіанту** | **Вміст абзаців** | **Текстові параметри** | **Фонові параметри** | **Розміри та відступи** |
| 12 | FORM | FORM | FORM | FORM |

## **Текст програми:**

**1. Контекст бази даних**

**ApplicationDbContext.cs** – файл контексту, де визначено DbSet для таблиці з параметрами абзаців.

using Microsoft.EntityFrameworkCore;

using Task\_1.Models;

namespace Task\_1.Data

{

// Контекст БД для Task\_1

public class ApplicationDbContext : DbContext

{

// Конструктор який приймає опції контексту

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

: base(options)

{

}

// Власт для доступу до табл ParagraphStyles (розміри та відступи)

public DbSet<ParagraphStyle> ParagraphStyles { get; set; }

}

}

**2. Контролер**

**HomeController.cs** – файл контролера, який зчитує дані з форми, отримує параметри з бази даних і передає їх у ViewBag для наступного подання.

using Microsoft.AspNetCore.Mvc;

using Microsoft.Data.SqlClient;

namespace Task\_1.Controllers

{

public class HomeController : Controller

{

private readonly IConfiguration \_configuration;

// Конструктор отримує IConfiguration через DI для доступу до рядка підключення

public HomeController(IConfiguration configuration)

{

\_configuration = configuration;

}

// GET: Home/Index

[HttpGet]

public IActionResult Index()

{

// Повертаємо форму для введення даних (нетипізована форма)

return View();

}

// POST: Home/IndexPost

[HttpPost]

public IActionResult IndexPost()

{

// Зчитуємо дані з форми (нетипізовано)

string paragraph1Text = Request.Form["paragraph1Text"];

string paragraph1Color = Request.Form["paragraph1Color"];

string paragraph1Align = Request.Form["paragraph1Align"];

string paragraph1BgColor = Request.Form["paragraph1BgColor"];

string paragraph1BgImage = Request.Form["paragraph1BgImage"];

string paragraph2Text = Request.Form["paragraph2Text"];

string paragraph2Color = Request.Form["paragraph2Color"];

string paragraph2Align = Request.Form["paragraph2Align"];

string paragraph2BgColor = Request.Form["paragraph2BgColor"];

string paragraph2BgImage = Request.Form["paragraph2BgImage"];

string paragraph3Text = Request.Form["paragraph3Text"];

string paragraph3Color = Request.Form["paragraph3Color"];

string paragraph3Align = Request.Form["paragraph3Align"];

string paragraph3BgColor = Request.Form["paragraph3BgColor"];

string paragraph3BgImage = Request.Form["paragraph3BgImage"];

// Ініціалізуємо змінні для зберігання даних розмірів та відступів з БД

string style1FontSize = "16px", style1Margin = "0px", style1Padding = "0px";

string style2FontSize = "16px", style2Margin = "0px", style2Padding = "0px";

string style3FontSize = "16px", style3Margin = "0px", style3Padding = "0px";

// Отримуємо рядок підключення з appsettings.json

string connectionString = \_configuration.GetConnectionString("DefaultConnection");

// Виконуємо запити до БД за допомогою ADO.NET

using (SqlConnection conn = new SqlConnection(connectionString))

{

conn.Open();

// Отримуємо дані для абзацу 1 (Id=1)

using (SqlCommand cmd = new SqlCommand("SELECT FontSize, Margin, Padding FROM ParagraphStyles WHERE Id=1", conn))

{

using (SqlDataReader reader = cmd.ExecuteReader())

{

if (reader.Read())

{

style1FontSize = reader["FontSize"]?.ToString() ?? "16px";

style1Margin = reader["Margin"]?.ToString() ?? "0px";

style1Padding = reader["Padding"]?.ToString() ?? "0px";

}

}

}

// Отримуємо дані для абзацу 2 (Id=2)

using (SqlCommand cmd = new SqlCommand("SELECT FontSize, Margin, Padding FROM ParagraphStyles WHERE Id=2", conn))

{

using (SqlDataReader reader = cmd.ExecuteReader())

{

if (reader.Read())

{

style2FontSize = reader["FontSize"]?.ToString() ?? "16px";

style2Margin = reader["Margin"]?.ToString() ?? "0px";

style2Padding = reader["Padding"]?.ToString() ?? "0px";

}

}

}

// Отримуємо дані для абзацу 3 (Id=3)

using (SqlCommand cmd = new SqlCommand("SELECT FontSize, Margin, Padding FROM ParagraphStyles WHERE Id=3", conn))

{

using (SqlDataReader reader = cmd.ExecuteReader())

{

if (reader.Read())

{

style3FontSize = reader["FontSize"]?.ToString() ?? "16px";

style3Margin = reader["Margin"]?.ToString() ?? "0px";

style3Padding = reader["Padding"]?.ToString() ?? "0px";

}

}

}

}

// Передаємо дані для абзаців через ViewBag

// Абзац 1

ViewBag.Paragraph1Text = paragraph1Text;

ViewBag.Paragraph1Color = paragraph1Color;

ViewBag.Paragraph1Align = paragraph1Align;

ViewBag.Paragraph1BgColor = paragraph1BgColor;

ViewBag.Paragraph1BgImage = paragraph1BgImage;

ViewBag.Paragraph1FontSize = style1FontSize;

ViewBag.Paragraph1Margin = style1Margin;

ViewBag.Paragraph1Padding = style1Padding;

// Абзац 2

ViewBag.Paragraph2Text = paragraph2Text;

ViewBag.Paragraph2Color = paragraph2Color;

ViewBag.Paragraph2Align = paragraph2Align;

ViewBag.Paragraph2BgColor = paragraph2BgColor;

ViewBag.Paragraph2BgImage = paragraph2BgImage;

ViewBag.Paragraph2FontSize = style2FontSize;

ViewBag.Paragraph2Margin = style2Margin;

ViewBag.Paragraph2Padding = style2Padding;

// Абзац 3

ViewBag.Paragraph3Text = paragraph3Text;

ViewBag.Paragraph3Color = paragraph3Color;

ViewBag.Paragraph3Align = paragraph3Align;

ViewBag.Paragraph3BgColor = paragraph3BgColor;

ViewBag.Paragraph3BgImage = paragraph3BgImage;

ViewBag.Paragraph3FontSize = style3FontSize;

ViewBag.Paragraph3Margin = style3Margin;

ViewBag.Paragraph3Padding = style3Padding;

// Повертаємо подання Display для відображення результатів

return View("Display");

}

// Додатковий екшн

public IActionResult Display()

{

return View();

}

}

}

**3. Подання (Views)**

**Index.cshtml** – головне подання з формою введення даних (нетипізована форма).

**Display.cshtml** – подання, яке отримує дані з ViewBag і відображає абзаци з параметрами (шрифти, відступи, фони тощо).

**Layout.cshtml** – шаблон, що забезпечує загальне оформлення застосунку (Navbar, контейнер, футер, фон).

**\_Layout.cshtml.css** – файл, який містить кастомні CSS-стилі для загального оформлення веб-застосунку (Navbar, контейнер, футер, фон тощо).

**Index.cshtml:**

@{

// Заголовок сторінки

ViewData["Title"] = "Головна (Index)";

}

<!-- Заголовок форми -->

<h2 class="mb-4">Форма (нетипізована) для введення параметрів абзаців</h2>

<!-- Форма для введення параметрів (POST-запит до HomeController, метод IndexPost) -->

<form method="post" asp-action="IndexPost" asp-controller="Home">

<div class="row">

<!-- Параграф 1 -->

<div class="col-md-4">

<h4>Параграф 1</h4>

<!-- Поле введення тексту -->

<div class="mb-3">

<label class="form-label">Текст:</label>

<input type="text" name="paragraph1Text" value="FORM" class="form-control" />

</div>

<!-- Поле введення кольору тексту -->

<div class="mb-3">

<label class="form-label">Колір тексту (color):</label>

<input type="text" name="paragraph1Color" placeholder="red, #FF0000" class="form-control" />

</div>

<!-- Випадаючий список вирівнювання тексту -->

<div class="mb-3">

<label class="form-label">Вирівнювання (text-align):</label>

<select name="paragraph1Align" class="form-select">

<option value="left">left</option>

<option value="center">center</option>

<option value="right">right</option>

<option value="justify">justify</option>

</select>

</div>

<!-- Поле введення кольору фону -->

<div class="mb-3">

<label class="form-label">Фон (колір) background-color:</label>

<input type="text" name="paragraph1BgColor" placeholder="white / #FFFFFF / ..." class="form-control" />

</div>

<!-- Поле введення URL фонового зображення -->

<div class="mb-3">

<label class="form-label">Фон (зображення) background-image (URL):</label>

<input type="text" name="paragraph1BgImage" placeholder="https://..." class="form-control" />

</div>

</div>

<!-- Параграф 2 -->

<div class="col-md-4">

<h4>Параграф 2</h4>

<!-- Поле введення тексту -->

<div class="mb-3">

<label class="form-label">Текст:</label>

<input type="text" name="paragraph2Text" value="FORM" class="form-control" />

</div>

<!-- Поле введення кольору тексту -->

<div class="mb-3">

<label class="form-label">Колір тексту:</label>

<input type="text" name="paragraph2Color" placeholder="red, #FF0000" class="form-control" />

</div>

<!-- Випадаючий список вирівнювання -->

<div class="mb-3">

<label class="form-label">Вирівнювання:</label>

<select name="paragraph2Align" class="form-select">

<option value="left">left</option>

<option value="center">center</option>

<option value="right">right</option>

<option value="justify">justify</option>

</select>

</div>

<!-- Поле введення кольору фону -->

<div class="mb-3">

<label class="form-label">Фон (колір):</label>

<input type="text" name="paragraph2BgColor" placeholder="white" class="form-control" />

</div>

<!-- Поле введення URL фонового зображення -->

<div class="mb-3">

<label class="form-label">Фон (зображення) (URL):</label>

<input type="text" name="paragraph2BgImage" placeholder="https://..." class="form-control" />

</div>

</div>

<!-- Параграф 3 -->

<div class="col-md-4">

<h4>Параграф 3</h4>

<!-- Поле введення тексту -->

<div class="mb-3">

<label class="form-label">Текст:</label>

<input type="text" name="paragraph3Text" value="FORM" class="form-control" />

</div>

<!-- Поле введення кольору тексту -->

<div class="mb-3">

<label class="form-label">Колір тексту:</label>

<input type="text" name="paragraph3Color" placeholder="red, #FF0000" class="form-control" />

</div>

<!-- Випадаючий список вирівнювання -->

<div class="mb-3">

<label class="form-label">Вирівнювання:</label>

<select name="paragraph3Align" class="form-select">

<option value="left">left</option>

<option value="center">center</option>

<option value="right">right</option>

<option value="justify">justify</option>

</select>

</div>

<!-- Поле введення кольору фону -->

<div class="mb-3">

<label class="form-label">Фон (колір):</label>

<input type="text" name="paragraph3BgColor" placeholder="white" class="form-control" />

</div>

<!-- Поле введення URL фонового зображення -->

<div class="mb-3">

<label class="form-label">Фон (зображення) (URL):</label>

<input type="text" name="paragraph3BgImage" placeholder="https://..." class="form-control" />

</div>

</div>

</div>

<!-- Горизонтальна лінія для розділення -->

<hr />

<!-- Кнопка відправки форми -->

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

</form>

**Display.cshtml:**

@{

// Встановлюємо заголовок сторінки

ViewData["Title"] = "Результат (Display)";

// Дані для 1 абзацу з ViewBag

var p1Text = ViewBag.Paragraph1Text;

var p1Color = ViewBag.Paragraph1Color;

var p1Align = ViewBag.Paragraph1Align;

var p1BgColor = ViewBag.Paragraph1BgColor;

var p1BgImage = ViewBag.Paragraph1BgImage;

var p1FontSize = ViewBag.Paragraph1FontSize;

var p1Margin = ViewBag.Paragraph1Margin;

var p1Padding = ViewBag.Paragraph1Padding;

// Дані для 2 абзацу

var p2Text = ViewBag.Paragraph2Text;

var p2Color = ViewBag.Paragraph2Color;

var p2Align = ViewBag.Paragraph2Align;

var p2BgColor = ViewBag.Paragraph2BgColor;

var p2BgImage = ViewBag.Paragraph2BgImage;

var p2FontSize = ViewBag.Paragraph2FontSize;

var p2Margin = ViewBag.Paragraph2Margin;

var p2Padding = ViewBag.Paragraph2Padding;

// Дані для 3 абзацу

var p3Text = ViewBag.Paragraph3Text;

var p3Color = ViewBag.Paragraph3Color;

var p3Align = ViewBag.Paragraph3Align;

var p3BgColor = ViewBag.Paragraph3BgColor;

var p3BgImage = ViewBag.Paragraph3BgImage;

var p3FontSize = ViewBag.Paragraph3FontSize;

var p3Margin = ViewBag.Paragraph3Margin;

var p3Padding = ViewBag.Paragraph3Padding;

}

<!-- Заголовок вмісту -->

<h2 class="mb-3">Згенеровані абзаци</h2>

<!-- Звідки беруться параметри -->

<p class="mb-4">

Параметри кольору, тексту, фону тощо взято з форми.<br />

Параметри розміру та відступів (FontSize, Margin, Padding) взято з БД (таблиця ParagraphStyles).

</p>

<div class="mb-4">

<!-- Блок для 1 абзацу -->

<div class="border p-3 mb-3">

<p style="

color:@p1Color;

text-align:@p1Align;

background-color:@p1BgColor;

background-image: @(string.IsNullOrEmpty(p1BgImage) ? "none" : "url('" + p1BgImage + "')");

font-size:@p1FontSize;

margin:@p1Margin;

padding:@p1Padding;

">

@p1Text

</p>

</div>

<!-- Блок для 2 абзацу -->

<div class="border p-3 mb-3">

<p style="

color:@p2Color;

text-align:@p2Align;

background-color:@p2BgColor;

background-image: @(string.IsNullOrEmpty(p2BgImage) ? "none" : "url('" + p2BgImage + "')");

font-size:@p2FontSize;

margin:@p2Margin;

padding:@p2Padding;

">

@p2Text

</p>

</div>

<!-- Блок для 3 абзацу -->

<div class="border p-3 mb-3">

<p style="

color:@p3Color;

text-align:@p3Align;

background-color:@p3BgColor;

background-image: @(string.IsNullOrEmpty(p3BgImage) ? "none" : "url('" + p3BgImage + "')");

font-size:@p3FontSize;

margin:@p3Margin;

padding:@p3Padding;

">

@p3Text

</p>

</div>

</div>

<br />

<!-- Кнопка для повернення на головну сторінку -->

<a asp-action="Index" asp-controller="Home" class="btn btn-secondary">Назад</a>

**\_Layout.cshtml:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>@ViewData["Title"] - Task\_1</title>

<!-- Підключення Bootstrap CSS -->

<link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />

<!-- Підключення власних стилів -->

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

<link rel="stylesheet" href="~/Task\_1.styles.css" asp-append-version="true" />

</head>

<body style="background: url('https://coolbackgrounds.io/images/unsplash/josh-bean-medium-9501ba9f.jpg') no-repeat center center fixed; background-size: cover;">

<header>

<!-- Navbar з мін відступом знизу -->

<nav class="navbar navbar-expand-sm navbar-dark bg-dark border-bottom box-shadow mb-1">

<div class="container-fluid">

<!-- Логотип/назва сайту -->

<a class="navbar-brand text-white" asp-area="" asp-controller="Home" asp-action="Index">Task\_1</a>

<!-- Кнопка для адапт меню -->

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse"

aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<!-- Основне меню -->

<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">

<ul class="navbar-nav flex-grow-1">

<li class="nav-item">

<a class="nav-link text-white" asp-area="" asp-controller="Home" asp-action="Index">Home</a>

</li>

<li class="nav-item">

<a class="nav-link text-white" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>

</li>

</ul>

</div>

</div>

</nav>

</header>

<!-- Основний контейнер для вмісту -->

<div class="container my-5" style="background-color: rgba(255, 255, 255, 0.9); border-radius: 10px; padding: 20px;">

<main role="main" class="pb-3">

@RenderBody() <!-- Рендер вміст кожної сторінки -->

</main>

</div>

<!-- Футер з відступом зверху -->

<footer class="border-top footer text-muted mt-4">

<div class="container">

&copy; 2025 - Task\_1 -

<a asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>

</div>

</footer>

<!-- Підключення jQuery, Bootstrap JS та скриптів -->

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

<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>

<script src="~/js/site.js" asp-append-version="true"></script>

@await RenderSectionAsync("Scripts", required: false)

</body>

</html>

**\_Layout.cshtml.css:**

/\* Please see documentation at https://learn.microsoft.com/aspnet/core/client-side/bundling-and-minification

for details on configuring this project to bundle and minify static web assets. \*/

/\* Стиль для логотипу в Navbar \*/

a.navbar-brand {

white-space: normal;

text-align: center;

word-break: break-all;

}

/\* Загальний стиль для всіх посилань \*/

a {

color: #0077cc;

}

/\* Стиль для кнопок primary \*/

.btn-primary {

color: #fff;

background-color: #1b6ec2;

border-color: #1861ac;

}

/\* Стиль для активних елементів в nav-pills \*/

.nav-pills .nav-link.active,

.nav-pills .show > .nav-link {

color: #fff;

background-color: #1b6ec2;

border-color: #1861ac;

}

/\* Стиль для верхньої межі \*/

.border-top {

border-top: 1px solid #e5e5e5;

}

/\* Стиль для нижньої межі \*/

.border-bottom {

border-bottom: 1px solid #e5e5e5;

}

/\* Легка тінь для об'ємного ефекту \*/

.box-shadow {

box-shadow: 0 .25rem .75rem rgba(0, 0, 0, .05);

}

/\* Стиль для кнопки прийняття політики \*/

button.accept-policy {

font-size: 1rem;

line-height: inherit;

}

/\* Стиль для футера \*/

.footer {

position: absolute;

width: 100%;

white-space: nowrap;

line-height: 60px;

}

**4. SQL-скрипт:**

CREATE TABLE [dbo].[ParagraphStyles](

[Id] INT NOT NULL IDENTITY(1,1) PRIMARY KEY,

[FontSize] NVARCHAR(50) NOT NULL,

[Margin] NVARCHAR(50) NOT NULL,

[Padding] NVARCHAR(50) NOT NULL

);

INSERT INTO [dbo].[ParagraphStyles] (FontSize, Margin, Padding)

VALUES

('14px','10px','5px'),

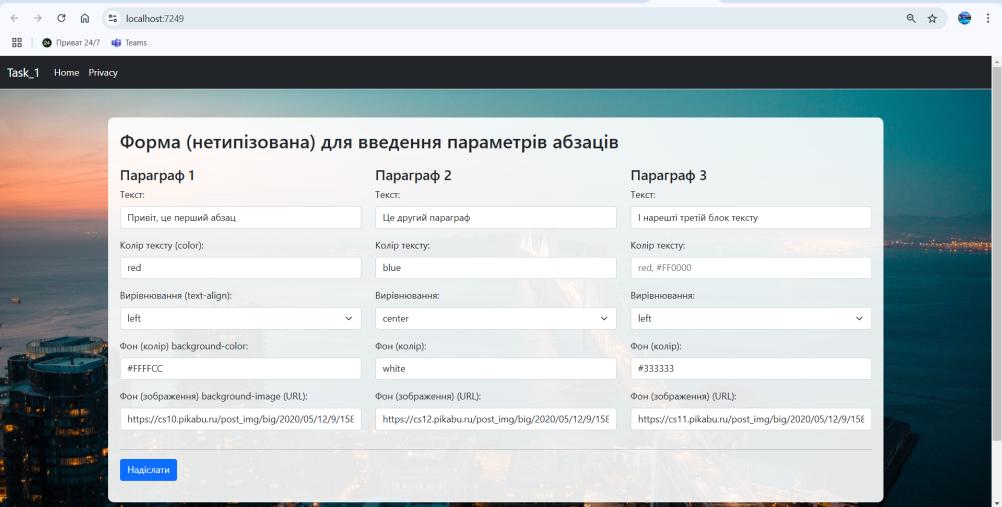
('18px','15px','10px'),

('22px','20px','15px');

## **Результат роботи програми:**

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.



Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

## **Алгоритм побудови проєкту і кодів**

1. **Створення рішення і проекту:**
   * Створив рішення (Lab\_1) у Visual Studio.
   * Додав новий проект типу ASP.NET Core MVC із назвою **Task\_1**.
2. **Налаштування бази даних:**
   * Створив базу даних LocalDB (внутрішню базу Visual Studio).
   * Написав SQL-скрипт для створення таблиці **ParagraphStyles** з полями:
     + **Id** – ціле число, автоінкремент, первинний ключ.
     + **FontSize** – NVARCHAR(50)
     + **Margin** – NVARCHAR(50)
     + **Padding** – NVARCHAR(50)
   * Додав цей SQL-скрипт до проєкту для подальшого використання.
3. **Розробка логіки:**
   * У файлі **Program.cs** видалив реєстрацію EF Core (AddDbContext), оскільки дані з таблиці зчитуватимуться напряму через ADO.NET.
   * Рядок підключення залишається в *appsettings.json*.
4. **Розробка контролера:**
   * Створив **HomeController.cs** у папці *Controllers*.
   * **GET метод Index()** повертає головну сторінку із формою для введення даних (нетипізована форма).
   * **POST метод IndexPost()**:
     + Зчитує дані з форми (вміст абзаців, текстові та фонографічні параметри) через Request.Form.
     + Виконує запити до БД за допомогою ADO.NET (SqlConnection, SqlCommand, SqlDataReader) для отримання розмірів та відступів (FontSize, Margin, Padding) з таблиці **ParagraphStyles**.
     + Дані з форми та отримані параметри передаються через ViewBag до подання **Display.cshtml**.
   * **Метод Display()** повертає подання, де відображаються три абзаци з параметрами, оформленими через inline CSS.
5. **Розробка подань (Views):**
   * **Index.cshtml:**

Містить форму з полями для введення:

* + - Вмісту абзаців (текст).
    - Текстових параметрів (колір, вирівнювання).
    - Фонових параметрів (колір, URL зображення).
  + **Display.cshtml:**

Відображає три абзаци, де дані з форми (FORM) і параметри розмірів та відступів (DB) поєднані за допомогою inline CSS.

* + **\_Layout.cshtml:**

Загальний шаблон застосунку із використанням Bootstrap, що забезпечує оформлення Navbar, контейнера для вмісту та футера з заданим фоном.

1. **Тестування:**
   * Запустив застосунок, заповнив форму параметрами.
   * Переконався, що дані з форми, а також параметри розмірів і відступів, зчитані через ADO.NET, успішно передаються через контролер до подання **Display.cshtml**.
   * Всі елементи оформлені за допомогою Bootstrap і власних CSS, що забезпечує адаптивний інтерфейс.

## **Функціональність**

* **Введення даних:**

Користувач заповнює форму для трьох абзаців, вводячи текст, параметри тексту (колір, вирівнювання) та фон (колір або URL зображення).

* **Обробка даних:**

Контролер зчитує дані з форми через Request.Form. Розміри та відступи абзаців отримуються з бази даних через ADO.NET, без використання моделей.

* **Відображення:**

Подання **Display.cshtml** відображає три абзаци з унікальними CSS-параметрами, де текстові параметри задаються через форму, а розміри та відступи – з бази даних.

## **Висновок**

Завдання **Task\_1** виконано відповідно до варіанту №12 без використання моделей. Дані з форми (вміст, текстові та фонографічні параметри) та параметри розмірів/відступів (зчитані через ADO.NET з бази даних) успішно інтегровані і відображаються у поданні Display. Проєкт демонструє практичне застосування ASP.NET Core MVC, використання ADO.NET для взаємодії з базою даних, а також адаптивне оформлення із застосуванням Bootstrap.

# **Завдання 2\_1**

Створити MVC WEB-застосунок, головне подання якого містить нетипізовану форму для задання групи параметрів, які через контролер, що викликається з цієї ж форми, передаються до іншого подання, на якому формуються зображення згідно із топологією власного варіанту.

|  |  |
| --- | --- |
| **№ Варіанту** | **Форма топології** |
| 1 | 2 |
| 12 |  |

Посилання на зображення задаються на формі, розміри зображень з таблиці відповідної БД.

## **Текст програми:**

**1. Контекст БД**

**ApplicationDbContext.cs** – файл контексту, де визначено DbSet для таблиці з розмірами зображень (ImageDimensions). Він забезпечує зв’язок між застосунком і БД (LocalDB).

using Microsoft.EntityFrameworkCore;

using Task\_2\_1.Models;

namespace Task\_2\_1.Data

{

// Контекст БД для Task\_2\_1

public class ApplicationDbContext : DbContext

{

// Конст отримує опції через DI і --> їх у базовий констр

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

: base(options)

{

}

// Властивість для роботи з табл ImageDimensions у БД

public DbSet<ImageDimension> ImageDimensions { get; set; }

}

}

**2. Контролер**

**HomeController.cs** – файл контролера, який зчитує дані з форми (URL зображень) та отримує розміри зображень із БД. Дані передаються через ViewBag до подання Display для відображення зображень за заданою топологією.

using Microsoft.AspNetCore.Mvc;

using Microsoft.Data.SqlClient; // для ADO.NET

using Microsoft.Extensions.Configuration;

namespace Task\_2\_1.Controllers

{

public class HomeController : Controller

{

private readonly IConfiguration \_configuration;

// Конструктор приймає IConfiguration для отримання рядка підключення

public HomeController(IConfiguration configuration)

{

\_configuration = configuration;

}

// GET: Home/Index

[HttpGet]

public IActionResult Index()

{

// Повертаємо форму для введення даних

return View();

}

// POST: Home/IndexPost

[HttpPost]

public IActionResult IndexPost()

{

// Зчитуємо URL зображень з форми

string image1Url = Request.Form["image1Url"];

string image2Url = Request.Form["image2Url"];

string image3Url = Request.Form["image3Url"];

string image4Url = Request.Form["image4Url"];

string image5Url = Request.Form["image5Url"];

// Масиви для зберігання розмірів зображень

string[] widths = new string[5];

string[] heights = new string[5];

// Отримуємо рядок підключення з appsettings.json

string connectionString = \_configuration.GetConnectionString("DefaultConnection");

// Виконуємо запити до БД через ADO.NET

using (SqlConnection conn = new SqlConnection(connectionString))

{

conn.Open();

for (int i = 1; i <= 5; i++)

{

string query = "SELECT Width, Height FROM ImageDimensions WHERE Id = @Id";

using (SqlCommand cmd = new SqlCommand(query, conn))

{

cmd.Parameters.AddWithValue("@Id", i);

using (SqlDataReader reader = cmd.ExecuteReader())

{

if (reader.Read())

{

widths[i - 1] = reader["Width"]?.ToString() ?? "100px";

heights[i - 1] = reader["Height"]?.ToString() ?? "100px";

}

else

{

// Якщо запис відсутній, задаємо значення за замовчуванням

widths[i - 1] = "100px";

heights[i - 1] = "100px";

}

}

}

}

}

// Передаємо URL зображень через ViewBag

ViewBag.Image1Url = image1Url;

ViewBag.Image2Url = image2Url;

ViewBag.Image3Url = image3Url;

ViewBag.Image4Url = image4Url;

ViewBag.Image5Url = image5Url;

// Передаємо розміри зображень через ViewBag

ViewBag.Image1Width = widths[0];

ViewBag.Image1Height = heights[0];

ViewBag.Image2Width = widths[1];

ViewBag.Image2Height = heights[1];

ViewBag.Image3Width = widths[2];

ViewBag.Image3Height = heights[2];

ViewBag.Image4Width = widths[3];

ViewBag.Image4Height = heights[3];

ViewBag.Image5Width = widths[4];

ViewBag.Image5Height = heights[4];

// Перенаправляємо дані до подання Display для відображення зображень

return View("Display");

}

// Метод для відображення подання Display

public IActionResult Display()

{

return View();

}

}

}

**3. Подання (Views)**

**Index.cshtml** – головне подання з формою введення даних, де користувач вводить URL зображень (форма нетипізована).

**Display.cshtml** – подання, яке отримує дані з ViewBag (URL та розміри зображень) і відображає зображення з використанням заданих параметрів (ширина, висота) та CSS-стилів.

**\_Layout.cshtml** – загальний шаблон застосунку, що забезпечує оформлення (Navbar, основний контейнер, футер, фон). Шаблон використовує Bootstrap для адаптивного оформлення.

**\_Layout.cshtml.css** – файл, який містить кастомні CSS-стилі для загального оформлення веб-застосунку (Navbar, контейнер, футер, фон тощо).

**Index.cshtml:**

@{

ViewData["Title"] = "Головна";

}

<!-- Обгортка для форми із відступами -->

<div class="container my-4">

<h2 class="mb-4">Форма для посилань на п'ять зображень</h2>

<!-- Форма що відправляє дані методом POST на IndexPost у HomeController -->

<form method="post" asp-action="IndexPost" asp-controller="Home">

<!-- Використовується відступ gutter -->

<div class="row g-3">

<!-- Поле URL 1 -->

<div class="col-md-2">

<label class="form-label">Зобр1 (URL):</label>

<input type="text" name="image1Url" class="form-control" placeholder="https://site.com/img1.jpg" />

</div>

<!-- Поле URL 2 -->

<div class="col-md-2">

<label class="form-label">Зобр2 (URL):</label>

<input type="text" name="image2Url" class="form-control" placeholder="https://site.com/img2.jpg" />

</div>

<!-- Поле URL 3 -->

<div class="col-md-2">

<label class="form-label">Зобр3 (URL):</label>

<input type="text" name="image3Url" class="form-control" placeholder="https://site.com/img3.jpg" />

</div>

<!-- Поле URL 4 -->

<div class="col-md-2">

<label class="form-label">Зобр4 (URL):</label>

<input type="text" name="image4Url" class="form-control" placeholder="https://site.com/img4.jpg" />

</div>

<!-- Поле URL 5 -->

<div class="col-md-2">

<label class="form-label">Зобр5 (URL):</label>

<input type="text" name="image5Url" class="form-control" placeholder="https://site.com/img5.jpg" />

</div>

</div>

<!-- Блок з кнопкою відправки форми з відступом зверху -->

<div class="mt-4">

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

</div>

</form>

</div>

**Display.cshtml:**

@{

ViewData["Title"] = "Відображення п'яти зображень (Топологія варіант №12)";

// Отримуємо URL зображень з ViewBag

var url1 = ViewBag.Image1Url;

var url2 = ViewBag.Image2Url;

var url3 = ViewBag.Image3Url;

var url4 = ViewBag.Image4Url;

var url5 = ViewBag.Image5Url;

// Отримуємо розміри з БД

var w1 = ViewBag.Image1Width;

var h1 = ViewBag.Image1Height;

var w2 = ViewBag.Image2Width;

var h2 = ViewBag.Image2Height;

var w3 = ViewBag.Image3Width;

var h3 = ViewBag.Image3Height;

var w4 = ViewBag.Image4Width;

var h4 = ViewBag.Image4Height;

var w5 = ViewBag.Image5Width;

var h5 = ViewBag.Image5Height;

}

<!-- Обгортка контейнера з відступами -->

<div class="container my-4">

<h2 class="mb-3">Відображення п'яти зображень (Топологія варіант №12)</h2>

<!-- Опис -->

<p class="mb-4">

Посилання (URL) отримуємо з форми,<br />

розміри (width, height) зчитуються з БД (таблиця ImageDimensions).

</p>

<!-- Контейнер для зображень з nowrap для відображення в одному рядку -->

<div class="border p-3" style="white-space: nowrap; text-align: left;">

<!-- 1 зображення -->

<img src="@url1" alt="Зобр1" style="width:@w1; height:@h1; vertical-align: bottom; margin-right:5px; border:1px solid #333;" />

<!-- 2 зображення -->

<img src="@url2" alt="Зобр2" style="width:@w2; height:@h2; vertical-align: bottom; margin-right:5px; border:1px solid #333;" />

<!-- 3 зображення -->

<img src="@url3" alt="Зобр3" style="width:@w3; height:@h3; vertical-align: bottom; margin-right:5px; border:1px solid #333;" />

<!-- 4 зображення -->

<img src="@url4" alt="Зобр4" style="width:@w4; height:@h4; vertical-align: bottom; margin-right:5px; border:1px solid #333;" />

<!-- 5 зображення -->

<img src="@url5" alt="Зобр5" style="width:@w5; height:@h5; vertical-align: bottom; margin-right:5px; border:1px solid #333;" />

</div>

<br />

<!-- Кнопка повернення на головну сторінку -->

<a asp-action="Index" asp-controller="Home" class="btn btn-secondary">Назад</a>

</div>

**\_Layout.cshtml:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>@ViewData["Title"] - Task\_2\_1</title>

<!-- Підключення Bootstrap CSS -->

<link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />

<!-- Підключення власних стилів -->

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

<link rel="stylesheet" href="~/Task\_2\_1.styles.css" asp-append-version="true" />

</head>

<body style="background: url('https://wallpaper.forfun.com/fetch/98/98cee4756de1ddb07d58638a36e45c22.jpeg') no-repeat center center fixed; background-size: cover;">

<header>

<!-- Навігаційна панель (Navbar) -->

<nav class="navbar navbar-expand-sm navbar-light bg-white border-bottom box-shadow mb-3">

<div class="container-fluid">

<!-- Назва сайту -->

<a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">Task\_2\_1</a>

<!-- Кнопка для мобільного меню -->

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse"

aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<!-- Меню навігації -->

<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">

<ul class="navbar-nav flex-grow-1">

<li class="nav-item">

<a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>

</li>

</ul>

</div>

</div>

</nav>

</header>

<!-- Основний контейнер для вмісту -->

<div class="container my-5" style="background-color: rgba(255,255,255,0.9); border-radius: 10px; padding: 20px;">

<main role="main" class="pb-3">

@RenderBody() <!-- Рендер основний вміст сторінки -->

</main>

</div>

<!-- Футер -->

<footer class="border-top footer text-white mt-4" style="background-color: rgba(0, 0, 0, 0.5);">

<div class="container">

&copy; 2025 - Task\_2\_1 -

<a asp-area="" asp-controller="Home" asp-action="Privacy" class="text-white">Privacy</a>

</div>

</footer>

<!-- Підключення скриптів -->

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

<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>

<script src="~/js/site.js" asp-append-version="true"></script>

@await RenderSectionAsync("Scripts", required: false)

</body>

</html>

**\_Layout.cshtml.css:**

/\* Please see documentation at https://learn.microsoft.com/aspnet/core/client-side/bundling-and-minification

for details on configuring this project to bundle and minify static web assets. \*/

a.navbar-brand {

white-space: normal;

text-align: center;

word-break: break-all;

}

/\* Загальний стиль для посилань \*/

a {

color: #0077cc;

transition: color 0.3s ease; /\* Плавна зміна \*/

}

/\* Стиль для кнопок primary \*/

.btn-primary {

color: #fff;

background-color: #1b6ec2;

border-color: #1861ac;

transition: background-color 0.3s, border-color 0.3s;

}

/\* Стиль для активних посилань у nav-pills \*/

.nav-pills .nav-link.active, .nav-pills .show > .nav-link {

color: #fff;

background-color: #1b6ec2;

border-color: #1861ac;

}

/\* Стиль для верхньої межі елементів \*/

.border-top {

border-top: 1px solid #e5e5e5;

}

/\* Стиль для нижньої межі елементів \*/

.border-bottom {

border-bottom: 1px solid #e5e5e5;

}

/\* Легка тінь для створення об'ємного ефекту \*/

.box-shadow {

box-shadow: 0 .25rem .75rem rgba(0, 0, 0, 0.05);

}

/\* Стиль для кнопки прийняття політики \*/

button.accept-policy {

font-size: 1rem;

line-height: inherit;

cursor: pointer;

}

/\* Стиль для футера \*/

.footer {

position: absolute;

bottom: 0;

width: 100%;

white-space: nowrap;

line-height: 60px;

}

**4. SQL-скрипт:**

-- Створення таблиці:

IF NOT EXISTS (SELECT 1 FROM sys.tables WHERE [name] = 'ImageDimensions')

BEGIN

CREATE TABLE [dbo].[ImageDimensions](

[Id] INT NOT NULL IDENTITY(1,1) PRIMARY KEY,

[Width] NVARCHAR(50) NOT NULL,

[Height] NVARCHAR(50) NOT NULL

);

END

-- Вставлення кількох записів:

INSERT INTO [dbo].[ImageDimensions] (Width, Height)

VALUES

('80px','60px'),

('100px','70px'),

('120px','120px'),

('140px','150px'),

('160px','220px');

## **Результат роботи програми:**

Изображение выглядит как текст, снимок экрана, пляж, на открытом воздухе

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

## **Алгоритм побудови проєкту і кодів**

1. **Створення проєкту:**
   * Додав новий проект типу ASP.NET Core MVC із назвою **Task\_2\_1**.
2. **Налаштування бази даних (LocalDB):**
   * Створив базу даних LocalDB (внутрішню базу Visual Studio).
   * Створив таблицю **ImageDimensions** для зберігання розмірів зображень із наступними полями:
     + **Id** – ціле число, автоінкремент, первинний ключ.
     + **Width** – NVARCHAR(50)
     + **Height** – NVARCHAR(50)
   * Додав SQL-скрипт для створення цієї таблиці до проєкту.
3. **Розробка логіки:**
   * Видалив реєстрацію EF Core (AddDbContext) із **Program.cs**, оскільки для доступу до БД тепер використовується ADO.NET.
   * Рядок підключення залишається в *appsettings.json* для роботи з ADO.NET.
4. **Розробка контролера:**
   * **HomeController.cs** (у папці *Controllers*):
     + **Index() [GET]:** Повертає головну сторінку з формою для введення URL п’яти зображень (нетипізована форма).
     + **IndexPost() [POST]:**
       - Зчитує URL-адреси з форми через Request.Form.
       - За допомогою ADO.NET (SqlConnection, SqlCommand, SqlDataReader) виконує запити до таблиці **ImageDimensions** для отримання розмірів (Width, Height) для кожного зображення (припускаючи, що записи мають Id від 1 до 5).
       - Отримані дані (URL та розміри) передає через **ViewBag** до подання **Display.cshtml**.
     + **Display():** Повертає подання, яке відображає зображення згідно з топологією «сходинок», використовуючи дані з ViewBag (URL, Width, Height).
5. **Розробка подань (Views):**
   * **Index.cshtml:**

Містить форму, де користувач вводить URL 5 зображень. Дані надсилаються методом POST до методу IndexPost() контролера Home.

* + **Display.cshtml:**

Подання, яке отримує з ViewBag URL зображень і їхні розміри, та відображає зображення в один рядок із застосуванням inline CSS (наприклад, white-space: nowrap;, vertical-align: bottom;, відступи тощо) для створення ефекту «сходинок».

* + **\_Layout.cshtml:**

Загальний шаблон застосунку, який забезпечує оформлення застосунку із використанням Bootstrap (Navbar, основний контейнер, футер, фон тощо).

1. **Тестування:**
   * Запустив застосунок і заповнив форму введення URL зображень.
   * Перевірив, що дані з форми зчитуються, а розміри зображень отримуються з таблиці **ImageDimensions** за допомогою ADO.NET.
   * Переконався, що зображення відображаються у поданні **Display.cshtml** згідно з топологією «сходинок».
   * Змінив значення ширини/висоти в таблиці **ImageDimensions**, щоб перевірити, чи застосунок правильно відображає нові розміри.

## **Функціональність**

1. **Форма введення та обробка даних**

Користувач вводить URL п’яти зображень у нетипізовану форму. Форма надсилає дані методом POST до контролера, який отримує URL-адреси, виконує запити до БД для отримання розмірів зображень та передає ці дані у ViewBag для відображення.

1. **Відображення зображень відповідно до топології**

Подання Display.cshtml отримує URL та розміри з ViewBag і відображає зображення відповідно до заданої топології (сходинки), використовуючи CSS-стилі для правильного розташування елементів.

1. **Робота з базою даних та загальне оформлення**

Дані про розміри зображень зберігаються у таблиці ImageDimensions у LocalDB. Запити до БД виконуються через ADO.NET. Інтерфейс оформлено за допомогою Bootstrap, а \_Layout.cshtml забезпечує загальне оформлення сторінок.

## **Висновок**

Розроблено MVC WEB-застосунок, який дозволяє користувачам вводити URL-адреси зображень, отримувати їхні розміри з бази даних і відображати зображення згідно з визначеною топологією. Використання ASP.NET Core MVC забезпечило структурованість коду, а ADO.NET дозволило ефективно взаємодіяти з базою даних. Завдяки адаптивному дизайну на основі Bootstrap веб-застосунок має зручний інтерфейс. Тестування показало, що система працює коректно, динамічно змінюючи відображення зображень відповідно до отриманих даних.

# **Завдання 2\_2**

Модифікувати застосунок Завдання 2\_1 для реалізації ще однієї топології зображень.

|  |  |
| --- | --- |
| № Варіанту | Форма топології |
| 1 | 2 |
| 12 |  |

Посилання на зображення задаються на формі, розміри зображень з таблиці відповідної БД.

## **Текст програми:**

**1. Контекст бази даних**

**ApplicationDbContext.cs** – Файл контексту, де визначено DbSet<ImageDimension> (таблиця ImageDimensions). Він відповідає за підключення до БД (LocalDB).

using Microsoft.EntityFrameworkCore;

using Task\_2\_2.Models;

namespace Task\_2\_2.Data

{

// Контекст БД для Task\_2\_2

public class ApplicationDbContext : DbContext

{

// Констр який приймає опції БД через DI

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

: base(options)

{

}

// DbSet для таблиці ImageDimensions

// Таблиця зберігає розміри зображень (Width, Height)

public DbSet<ImageDimension> ImageDimensions { get; set; }

}

}

**2. Контролер**

**HomeController.cs** – Файл контролера, який:

* Зчитує URL трьох зображень із форми (нетипізована форма).
* Звертається до бази даних (ImageDimensions) для отримання розмірів кожного зображення.
* Передає всі дані (URL, розміри) через ViewBag у подання Display для відображення.

using Microsoft.AspNetCore.Mvc;

using Microsoft.Data.SqlClient; // для ADO.NET

using Microsoft.Extensions.Configuration;

namespace Task\_2\_2.Controllers

{

public class HomeController : Controller

{

// IConfiguration для отримання рядка підключення

private readonly IConfiguration \_configuration;

public HomeController(IConfiguration configuration)

{

\_configuration = configuration;

}

// GET: Home/Index – повертаємо форму для введення URL зображень

[HttpGet]

public IActionResult Index()

{

return View();

}

// POST: Home/IndexPost – обробка даних форми

[HttpPost]

public IActionResult IndexPost()

{

// Зчитуємо URL для трьох зображень з форми

string image1Url = Request.Form["image1Url"];

string image2Url = Request.Form["image2Url"];

string image3Url = Request.Form["image3Url"];

// Масиви для зберігання розмірів (ширина та висота) з БД

string[] widths = new string[3];

string[] heights = new string[3];

// Отримуємо рядок підключення з appsettings.json

string connectionString = \_configuration.GetConnectionString("DefaultConnection");

// Виконуємо запити до БД через ADO.NET для кожного зображення (Id від 1 до 3)

using (SqlConnection conn = new SqlConnection(connectionString))

{

conn.Open();

for (int i = 1; i <= 3; i++)

{

string query = "SELECT Width, Height FROM ImageDimensions WHERE Id = @Id";

using (SqlCommand cmd = new SqlCommand(query, conn))

{

cmd.Parameters.AddWithValue("@Id", i);

using (SqlDataReader reader = cmd.ExecuteReader())

{

if (reader.Read())

{

widths[i - 1] = reader["Width"]?.ToString() ?? "100px";

heights[i - 1] = reader["Height"]?.ToString() ?? "100px";

}

else

{

// Значення за замовчуванням, якщо запис не знайдено

widths[i - 1] = "100px";

heights[i - 1] = "100px";

}

}

}

}

}

// Передаємо URL зображень та їхні розміри через ViewBag до подання Display

ViewBag.Image1Url = image1Url;

ViewBag.Image2Url = image2Url;

ViewBag.Image3Url = image3Url;

ViewBag.Image1Width = widths[0];

ViewBag.Image1Height = heights[0];

ViewBag.Image2Width = widths[1];

ViewBag.Image2Height = heights[1];

ViewBag.Image3Width = widths[2];

ViewBag.Image3Height = heights[2];

return View("Display");

}

// Метод для відображення подання Display

public IActionResult Display()

{

return View();

}

}

}

**3. Подання (Views)**

**Index.cshtml** – Головне подання з формою, де користувач вводить URL трьох зображень. Дані надсилаються методом POST у метод IndexPost контролера Home.

**Display.cshtml** – Подання, яке приймає з ViewBag URL та розміри зображень і відображає їх за новою топологією (наприклад, «сходинки»). Використовуються inline-стилі (white-space, vertical-align тощо) для розміщення зображень у заданому порядку.

**\_Layout.cshtml** – Загальний шаблон застосунку, що забезпечує оформлення (Navbar, контейнер для вмісту, футер, фон). Підключає Bootstrap та власні стилі для гарного відображення сторінок.

**\_Layout.cshtml.css** – файл, який містить кастомні CSS-стилі для загального оформлення веб-застосунку (Navbar, контейнер, футер, фон тощо).

**Index.cshtml:**

@{

ViewData["Title"] = "Головна (Task\_2\_2)";

}

<!-- Контейнер для форми із відступами -->

<div class="container my-4">

<h2 class="mb-4">Форма для 3 зображень (Варіант із іншою топологією)</h2>

<p class="mb-4">URL зображень задаються у формі, а розміри - з БД.</p>

<!-- Форма що відправляє дані методом POST на IndexPost у HomeController -->

<form method="post" asp-action="IndexPost" asp-controller="Home">

<!-- Ряд із полями форми де використовується gutter -->

<div class="row g-3">

<!-- Зображення 1 -->

<div class="col-md-4">

<label class="form-label">Зобр1 (URL):</label>

<input type="text" name="image1Url" class="form-control" placeholder="https://site.com/img1.jpg" />

</div>

<!-- Зображення 2 -->

<div class="col-md-4">

<label class="form-label">Зобр2 (URL):</label>

<input type="text" name="image2Url" class="form-control" placeholder="https://site.com/img2.jpg" />

</div>

<!-- Зображення 3 -->

<div class="col-md-4">

<label class="form-label">Зобр3 (URL):</label>

<input type="text" name="image3Url" class="form-control" placeholder="https://site.com/img3.jpg" />

</div>

</div>

<!-- Блок з кнопкою відправки форми з відступом зверху -->

<div class="mt-4">

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

</div>

</form>

</div>

**Display.cshtml:**

@{

ViewData["Title"] = "Зображення за новою топологією (Сходинки)";

// Зчитуємо URL зображень ViewBag

var url1 = ViewBag.Image1Url;

var url2 = ViewBag.Image2Url;

var url3 = ViewBag.Image3Url;

// Зчитуємо розміри з БД (ширина та висота)

var w1 = ViewBag.Image1Width;

var h1 = ViewBag.Image1Height;

var w2 = ViewBag.Image2Width;

var h2 = ViewBag.Image2Height;

var w3 = ViewBag.Image3Width;

var h3 = ViewBag.Image3Height;

}

<!-- Обгортка контейнера з відступами -->

<div class="container my-4">

<h2 class="mb-3">Зображення “сходинками” (Варіант 12)</h2>

<p class="mb-4">

Зобр1, Зобр2, Зобр3 із різними розмірами. “Сходинки” виходять завдяки

<code>white-space: nowrap;</code> та <code>vertical-align: bottom;</code>.

</p>

<!-- Контейнер для зображень із nowrap (не переносилися) -->

<div class="border p-3" style="white-space: nowrap; text-align: left;">

<!-- Відображення 1 Зображення -->

<img src="@url1" alt="Зобр1" style="width:@w1; height:@h1; vertical-align: bottom; margin-right: 10px; border:1px solid #333;" />

<!-- Відображення 2 Зображення -->

<img src="@url2" alt="Зобр2" style="width:@w2; height:@h2; vertical-align: bottom; margin-right: 10px; border:1px solid #333;" />

<!-- Відображення 3 Зображення -->

<img src="@url3" alt="Зобр3" style="width:@w3; height:@h3; vertical-align: bottom; margin-right: 10px; border:1px solid #333;" />

</div>

<!-- Кнопка повернення на головну сторінку -->

<div class="mt-3">

<a asp-action="Index" asp-controller="Home" class="btn btn-secondary">Назад</a>

</div>

</div>

**\_Layout.cshtml:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>@ViewData["Title"] - Task\_2\_2</title>

<!-- Підключення Bootstrap CSS для стандартного оформлення -->

<link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />

<!-- Підключення власних стилів -->

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

<link rel="stylesheet" href="~/Task\_2\_2.styles.css" asp-append-version="true" />

</head>

<body style="background: url('https://eskipaper.com/images/cityscape-wallpaper-6.jpg') no-repeat center center fixed; background-size: cover;">

<header>

<!-- Навігаційна панель з використанням Bootstrap -->

<nav class="navbar navbar-expand-sm navbar-light bg-white border-bottom box-shadow mb-3">

<div class="container-fluid">

<!-- Назва сайту де посилання веде на головну сторінку -->

<a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">Task\_2\_2</a>

<!-- Кнопка для мобільного меню -->

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse"

aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<!-- Меню навігації -->

<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">

<ul class="navbar-nav flex-grow-1">

<li class="nav-item">

<a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>

</li>

</ul>

</div>

</div>

</nav>

</header>

<!-- Основний контейнер для відображення вмісту -->

<div class="container my-5" style="background-color: rgba(255,255,255,0.9); border-radius: 10px; padding: 20px;">

<main role="main" class="pb-3">

@RenderBody() <!-- Вміст який задається у поданнях -->

</main>

</div>

<!-- Футер -->

<footer class="border-top footer text-white mt-4" style="background-color: rgba(0, 0, 0, 0.5);">

<div class="container">

&copy; 2025 - Task\_2\_2 -

<a asp-area="" asp-controller="Home" asp-action="Privacy" class="text-white">Privacy</a>

</div>

</footer>

<!-- Підключення jQuery та Bootstrap JS -->

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

<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>

<script src="~/js/site.js" asp-append-version="true"></script>

@await RenderSectionAsync("Scripts", required: false)

</body>

</html>

**\_Layout.cshtml.css:**

/\* Please see documentation at https://learn.microsoft.com/aspnet/core/client-side/bundling-and-minification

for details on configuring this project to bundle and minify static web assets. \*/

/\* Стиль для логотипу в navbar \*/

a.navbar-brand {

white-space: normal;

text-align: center;

word-break: break-all; /

}

/\* Загальний стиль для посилань \*/

a {

color: #0077cc;

transition: color 0.3s ease;

}

/\* Стилізація кнопок типу primary \*/

.btn-primary {

color: #fff;

background-color: #1b6ec2;

border-color: #1861ac;

transition: background-color 0.3s, border-color 0.3s;

}

/\* Активні елементи навігації (nav-pills) \*/

.nav-pills .nav-link.active, .nav-pills .show > .nav-link {

color: #fff;

background-color: #1b6ec2;

border-color: #1861ac;

}

/\* Стиль для верхньої межі \*/

.border-top {

border-top: 1px solid #e5e5e5;

}

/\* Стиль для нижньої межі \*/

.border-bottom {

border-bottom: 1px solid #e5e5e5;

}

/\* Додаємо легку тінь для об'ємного ефекту \*/

.box-shadow {

box-shadow: 0 .25rem .75rem rgba(0, 0, 0, 0.05);

}

/\* Стиль для кнопки прийняття політики \*/

button.accept-policy {

font-size: 1rem;

line-height: inherit;

cursor: pointer;

}

/\* Стиль для футера \*/

.footer {

position: absolute;

bottom: 0;

width: 100%;

white-space: nowrap;

line-height: 60px;

}

**4. SQL-скрипт:**

-- Створення таблиці ImageDimensions

IF NOT EXISTS (SELECT \* FROM sys.tables WHERE name = 'ImageDimensions')

BEGIN

CREATE TABLE [dbo].[ImageDimensions](

[Id] INT NOT NULL IDENTITY(1,1) PRIMARY KEY,

[Width] NVARCHAR(50) NOT NULL,

[Height] NVARCHAR(50) NOT NULL

);

END

-- Вставлення 3 записів для нашої топології (зобр1, зобр2, зобр3):

INSERT INTO [dbo].[ImageDimensions] (Width, Height)

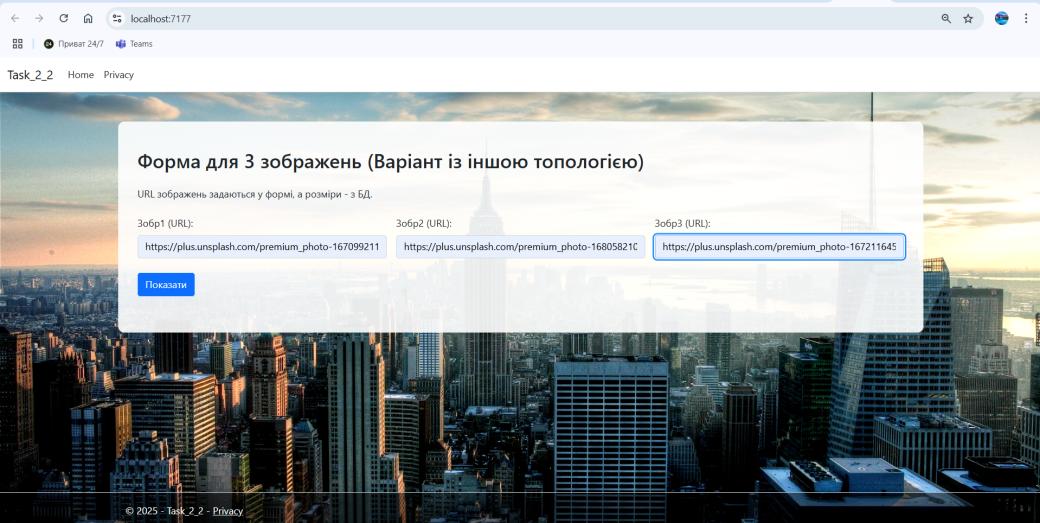
VALUES

('60px', '120px'), -- Зобр1: вертикально-витягнутий

('300px', '100px'), -- Зобр2: широкий

('150px', '50px'); -- Зобр3: горизонтальний прямокутник

## **Результат роботи програми:**



Изображение выглядит как текст, снимок экрана, строительство, город

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

## **Алгоритм побудови проєкту і кодів**

1. **Створення проєкту (на основі Task\_2\_1):**
   * Використав існуюче рішення Lab\_1 і зробив копію проєкту **Task\_2\_1**, перейменувавши його в **Task\_2\_2** для модифікації.
2. **Налаштування бази даних (LocalDB):**
   * Використовується та ж таблиця **ImageDimensions** (поля: Id, Width, Height) для зберігання розмірів зображень.
   * Зберігаються параметри розмірів, які будуть використовуватися для нової топології відображення зображень.
3. **Модифікація контролера:**
   * **HomeController.cs:**
     + **Index() [GET]** – повертає головну сторінку з формою, де користувач вводить URL зображень (наприклад, трьох).
     + **IndexPost() [POST]** – зчитує URL зображень через Request.Form, звертається до таблиці **ImageDimensions** за допомогою ADO.NET (SqlConnection, SqlCommand, SqlDataReader) для отримання розмірів (Width, Height) кожного зображення.
       - Отримані значення зберігаються в масивах і передаються через **ViewBag** до подання **Display**.
     + **Display()** – повертає подання, яке відображає зображення за новою топологією, використовуючи inline CSS (наприклад, white-space: nowrap;, vertical-align: bottom;, відступи тощо).
4. **Модифікація подань (Views):**
   * **Index.cshtml:**
     + Форма залишається аналогічною до попереднього завдання, де користувач вводить URL зображень.
     + Дані надсилаються методом POST до методу IndexPost() контролера Home.
   * **Display.cshtml:**
     + Подання отримує з **ViewBag** URL зображень та їхні розміри і відображає їх згідно з новою топологією (наприклад, зображення розташовуються «по-сходинках» у іншому порядку або вертикально).
     + Для розміщення використовуються inline CSS-правила, що задають white-space: nowrap;, vertical-align: bottom; та відступи.
   * **\_Layout.cshtml:**
     + Загальний шаблон застосунку із використанням Bootstrap (Navbar, контейнер для вмісту, футер, фон тощо).
     + Структура шаблону не змінюється, але може бути оновлено оформлення (наприклад, фон, колір навігації) для нової топології.
5. **Тестування:**
   * Запустив застосунок, ввів URL зображень у формі на сторінці **Index.cshtml**.
   * Перевірив, що розміри зображень зчитуються з таблиці **ImageDimensions** через ADO.NET і правильно передаються у **ViewBag**.
   * Переконався, що подання **Display.cshtml** відображає зображення згідно з новою топологією, що відповідає вимогам завдання.

## **Функціональність**

* **Введення даних:**

Користувач задає URL для кількох зображень через форму на сторінці **Index.cshtml**.

* **Обробка даних:**

Контролер HomeController зчитує URL з форми, виконує запити до бази даних (таблиця **ImageDimensions**) через ADO.NET для отримання розмірів зображень та передає всі значення через **ViewBag**.

* **Відображення:**

Подання **Display.cshtml** динамічно формує розташування зображень (наприклад, «сходинки») за допомогою inline CSS (white-space, vertical-align, margin тощо), використовуючи отримані URL та розміри.

* **Оформлення:**

Загальний шаблон **\_Layout.cshtml** із підключенням Bootstrap забезпечує сучасне та адаптивне оформлення сторінок.

## **Висновок**

Завдання **Task\_2\_2** успішно модифікує попередній застосунок (Task\_2\_1) для реалізації нової топології зображень без використання моделей. Дані з форми (URL зображень) та параметри розмірів (зчитані через ADO.NET) інтегровані через ViewBag та відображаються у поданні Display. Оформлення застосунку здійснено за допомогою Bootstrap і кастомних CSS, що забезпечує адаптивний та привабливий інтерфейс.

# **Завдання 3**

Створити MVC WEB-застосунок, головне подання якого містить нетипізовану форму для задання групи параметрів, які через контролер, що викликається з цієї ж форми, передаються до іншого подання, на якому формуються шахова (без об'єднаних комірок) таблиця із вказаними розмірами. Парні номери варіантів кількість рядків задає на формі, а кількість стовпчиків з БД, непарні номери - навпаки. Комірки таблиці послідовно заповнюються цілими числами, починаючи з 1, тобто 1, 2, 3 і т. д. Під час формування таблиці використовуються рядкові змінні в контролері, a не цикли технології Razor.

## **Текст програми:**

**1. Контекст бази даних**

**ApplicationDbContext.cs** – Файл, у якому визначено DbSet<ColumnCount>, що відповідає таблиці ColumnCounts. Він відповідає за підключення до БД (LocalDB).

using Microsoft.EntityFrameworkCore;

using Task\_3.Models;

namespace Task\_3.Data

{

// Контекст БД для Task\_3

public class ApplicationDbContext : DbContext

{

// Конструктор який приймає опції через DI та передає їх у базовий клас

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

: base(options)

{

}

// DbSet для зберігання кількості стовпчиків (табл ColumnCounts)

public DbSet<ColumnCount> ColumnCounts { get; set; }

}

}

**2. Контролер**

**HomeController.cs** – Файл контролера, що:

* Зчитує кількість рядків із форми (нетипізована форма, Index.cshtml).
* Звертається до БД (ColumnCounts) для отримання кількості стовпчиків.
* Генерує HTML-код шахової таблиці (без Razor-циклів) за допомогою рядкових змінних (StringBuilder) і послідовно заповнює комірки числами від 1, 2, 3 і т.д.
* Передає згенерований HTML-код таблиці у ViewBag, а також кількість рядків і стовпчиків у подання Display.

using Microsoft.AspNetCore.Mvc;

using Microsoft.Data.SqlClient; // для ADO.NET

using Microsoft.Extensions.Configuration;

using System.Text;

namespace Task\_3.Controllers

{

public class HomeController : Controller

{

private readonly IConfiguration \_configuration;

// Конструктор приймає IConfiguration для отримання рядка підключення

public HomeController(IConfiguration configuration)

{

\_configuration = configuration;

}

// GET: Home/Index

[HttpGet]

public IActionResult Index()

{

// Повертаємо форму для введення кількості рядків

return View();

}

// POST: Home/IndexPost

[HttpPost]

public IActionResult IndexPost()

{

// 1. Зчитуємо кількість рядків з форми (нетипізована форма)

string rowStr = Request.Form["rowCount"];

int rowCount = 0;

int.TryParse(rowStr, out rowCount);

// 2. Зчитуємо кількість стовпчиків з БД через ADO.NET

int colCount = 0;

string connectionString = \_configuration.GetConnectionString("DefaultConnection");

using (SqlConnection conn = new SqlConnection(connectionString))

{

conn.Open();

// Припускаємо, що таблиця ColumnCounts містить один запис з Id = 1

string query = "SELECT Columns FROM ColumnCounts WHERE Id = @Id";

using (SqlCommand cmd = new SqlCommand(query, conn))

{

cmd.Parameters.AddWithValue("@Id", 1);

object result = cmd.ExecuteScalar();

if (result != null && int.TryParse(result.ToString(), out int col))

{

colCount = col;

}

else

{

// Значення за замовчуванням, якщо запис не знайдено

colCount = 5;

}

}

}

// 3. Генеруємо HTML-код шахової таблиці за допомогою StringBuilder

StringBuilder sb = new StringBuilder();

sb.Append("<table style='border-collapse: collapse;'>");

int cellNumber = 1;

for (int r = 0; r < rowCount; r++)

{

sb.Append("<tr>");

for (int c = 0; c < colCount; c++)

{

// Шаховий принцип: якщо сума індексів парна – біла клітинка, інакше сіра

string bgColor = ((r + c) % 2 == 0) ? "#ffffff" : "#cccccc";

sb.Append($"<td style='width:50px; height:50px; border:1px solid #000; background-color:{bgColor}; text-align:center;'>");

sb.Append(cellNumber);

sb.Append("</td>");

cellNumber++;

}

sb.Append("</tr>");

}

sb.Append("</table>");

// Передаємо згенерований HTML таблиці та параметри через ViewBag до подання Display

ViewBag.TableHtml = sb.ToString();

ViewBag.Rows = rowCount;

ViewBag.Cols = colCount;

return View("Display");

}

// Метод для відображення подання Display

public IActionResult Display()

{

return View();

}

}

}

**3. Подання (Views)**

**Index.cshtml** – Головне подання з формою, де користувач вводить кількість рядків (для парного варіанту). Дані надсилаються методом POST до IndexPost() контролера Home.

**Display.cshtml** – Подання, що отримує згенерований HTML-код шахової таблиці (через ViewBag) і виводить його за допомогою @Html.Raw(...). Також відображає інформацію про кількість рядків і стовпчиків.

**\_Layout.cshtml** – Загальний шаблон застосунку, що забезпечує оформлення (Navbar, контейнер для основного вмісту, футер, фон). Застосовує Bootstrap для адаптивного дизайну.

**\_Layout.cshtml.css** – файл, який містить кастомні CSS-стилі для загального оформлення веб-застосунку (Navbar, контейнер, футер, фон тощо).

**Index.cshtml:**

@{

ViewData["Title"] = "Головна (Завдання 3)";

}

<!-- Основний контейнер -->

<div class="container my-4">

<!-- Заголовок форми -->

<h2 class="mb-3">Форма: Кількість рядків (парний варіант)</h2>

<p class="mb-4">Кількість стовпчиків буде взята з БД (таблиця ColumnCounts).</p>

<!-- Форма для введення кількості рядків де дані відправляються методом POST -->

<form method="post" asp-action="IndexPost" asp-controller="Home">

<!-- Поле введення кількості рядків -->

<div class="mb-3">

<label class="form-label">Введіть кількість рядків:</label>

<input type="number" name="rowCount" class="form-control" placeholder="наприклад 5" />

</div>

<!-- Кнопка для побудови таблиці -->

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

</form>

</div>

**Display.cshtml:**

@{

ViewData["Title"] = "Результат (Display)";

}

<!-- Контейнер для вмісту -->

<div class="container my-4">

<h2 class="mb-3">Шахова таблиця</h2>

<p class="mb-4">

Кількість рядків (з форми): @ViewBag.Rows<br />

Кількість стовпчиків (з БД): @ViewBag.Cols

</p>

<!-- Виводимо HTML-код таблиці згенерований у контролері -->

<div class="mb-4">

@Html.Raw(ViewBag.TableHtml)

</div>

<!-- Кнопка повернення на головну сторінку -->

<a asp-action="Index" asp-controller="Home" class="btn btn-secondary">Назад</a>

</div>

**\_Layout.cshtml:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>@ViewData["Title"] - Task\_3</title>

<!-- Підключення Bootstrap CSS -->

<link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />

<!-- Підключення власних стилів -->

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

<link rel="stylesheet" href="~/Task\_3.styles.css" asp-append-version="true" />

</head>

<body style="background: url('https://images6.alphacoders.com/126/1261894.jpg') no-repeat center center fixed; background-size: cover;">

<header>

<!-- Навігаційна панель (Navbar) з використанням Bootstrap -->

<nav class="navbar navbar-expand-sm navbar-light bg-white border-bottom box-shadow mb-3">

<div class="container-fluid">

<!-- Назва сайту де посилання веде на головну сторінку -->

<a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">Task\_3</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse"

aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<!-- Меню навігації -->

<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">

<ul class="navbar-nav flex-grow-1">

<li class="nav-item">

<a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>

</li>

</ul>

</div>

</div>

</nav>

</header>

<!-- Основний контейнер для вмісту -->

<div class="container my-5" style="background-color: rgba(255,255,255,0.9); border-radius: 10px; padding: 20px;">

<main role="main" class="pb-3">

@RenderBody() <!-- Тут рендер основного контенту сторінки -->

</main>

</div>

<!-- Футер -->

<footer class="border-top footer mt-4" style="background-color: rgba(0,0,0,0.5); color: #FFD700;">

<div class="container">

&copy; 2025 - Task\_3 -

<a asp-area="" asp-controller="Home" asp-action="Privacy" style="color: #FFD700;">Privacy</a>

</div>

</footer>

<!-- Підключення jQuery, Bootstrap JS та власних скриптів -->

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

<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>

<script src="~/js/site.js" asp-append-version="true"></script>

@await RenderSectionAsync("Scripts", required: false)

</body>

</html>

**\_Layout.cshtml.css:**

/\* Please see documentation at https://learn.microsoft.com/aspnet/core/client-side/bundling-and-minification

for details on configuring this project to bundle and minify static web assets. \*/

/\* Стиль для логотипу в Navbar \*/

a.navbar-brand {

white-space: normal;

text-align: center;

word-break: break-all;

}

/\* Загальний стиль для всіх посилань \*/

a {

color: #0077cc;

}

/\* Стилізація кнопок типу primary \*/

.btn-primary {

color: #fff;

background-color: #1b6ec2;

border-color: #1861ac;

}

/\* Активні елементи в nav-pills \*/

.nav-pills .nav-link.active, .nav-pills .show > .nav-link {

color: #fff;

background-color: #1b6ec2;

border-color: #1861ac;

}

/\* Стиль для верхньої межі \*/

.border-top {

border-top: 1px solid #e5e5e5;

}

/\* Стиль для нижньої межі \*/

.border-bottom {

border-bottom: 1px solid #e5e5e5;

}

/\* Додаємо легку тінь для створення об'ємного ефекту \*/

.box-shadow {

box-shadow: 0 .25rem .75rem rgba(0, 0, 0, 0.05);

}

/\* Стиль для кнопки прийняття політики \*/

button.accept-policy {

font-size: 1rem;

line-height: inherit;

cursor: pointer;

}

/\* Стиль для футера \*/

.footer {

position: absolute;

bottom: 0;

width: 100%;

white-space: nowrap;

line-height: 40px;

}

**4. SQL-скрипт:**

IF NOT EXISTS (SELECT \* FROM sys.tables WHERE name = 'ColumnCounts')

BEGIN

CREATE TABLE [dbo].[ColumnCounts](

[Id] INT NOT NULL IDENTITY(1,1) PRIMARY KEY,

[Columns] INT NOT NULL

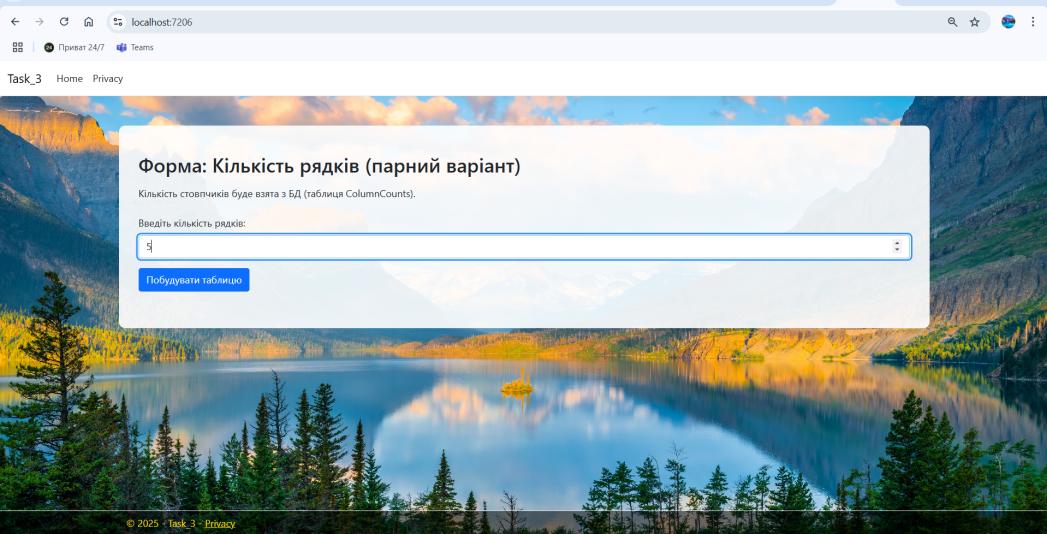
);

END

-- Вставляємо 1 запис, наприклад: Columns=6

INSERT INTO [dbo].[ColumnCounts] ([Columns]) VALUES (6);

## **Результат роботи програми:**



Изображение выглядит как текст, снимок экрана, дерево, гора

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

## **Алгоритм побудови проєкту і кодів**

1. **Створення проєкту:**
   * У Visual Studio додав до існуючого рішення новий проєкт типу ASP.NET Core MVC із назвою **Task\_3**.
2. **Налаштування БД (LocalDB):**
   * Створив таблицю **ColumnCounts** для зберігання кількості стовпчиків із наступними полями:
     + **Id** – ціле число, автоінкремент, первинний ключ.
     + **Columns** – int, що зберігає кількість стовпчиків.
   * Додав SQL-скрипт для створення таблиці до проєкту.
3. **Реалізація:**
   * Видалив реєстрацію EF Core (AddDbContext) з файлу **Program.cs**; рядок підключення залишається в *appsettings.json* для роботи через ADO.NET.
4. **Розробка контролера (HomeController.cs):**
   * **Index() [GET]:** Повертає форму (нетипізована форма), де користувач вводить кількість рядків.
   * **IndexPost() [POST]:**
     + Зчитує значення кількості рядків із форми.
     + За допомогою ADO.NET (SqlConnection, SqlCommand, ExecuteScalar) звертається до таблиці **ColumnCounts** для отримання кількості стовпчиків (наприклад, запис із Id = 1).
     + Використовуючи StringBuilder, генерує HTML-код шахової таблиці (без використання Razor-циклів). Комірки таблиці заповнюються послідовними числами (починаючи з 1).
     + Дані (HTML-код таблиці, кількість рядків і стовпчиків) передаються через **ViewBag** до подання **Display.cshtml**.
   * **Display():** Повертає подання, яке відображає згенерований HTML-код шахової таблиці.
5. **Розробка подань (Views):**
   * **Index.cshtml:**
     + Містить форму з полем введення rowCount (кількість рядків).
     + Дані надсилаються методом POST до методу **IndexPost()** контролера Home.
   * **Display.cshtml:**
     + Подання приймає через ViewBag згенерований HTML-код таблиці (ViewBag.TableHtml) і відображає його за допомогою @Html.Raw().
     + Також відображає кількість рядків і стовпчиків для наочності.
   * **\_Layout.cshtml:**
     + Використовується як загальний шаблон, що включає Navbar, контейнер для вмісту, футер та фон.
     + Підключено Bootstrap для оформлення.
6. **Тестування:**
   * Запустив застосунок і ввів кількість рядків у формі.
   * Переконався, що застосунок правильно зчитує кількість стовпчиків із БД через ADO.NET.
   * Перевірив, що згенерована шахова таблиця (без об’єднаних комірок) відображається правильно, а комірки послідовно заповнюються числами (1, 2, 3, ...).
   * Змінив значення в таблиці **ColumnCounts**, щоб перевірити правильність відображення нових параметрів.

## **Функціональність**

1. **Введення та отримання параметрів:**
   * Користувач вводить кількість рядків у формі, а кількість стовпчиків отримується з бази даних (або навпаки, залежно від варіанту).
2. **Обробка даних у контролері:**
   * Контролер приймає дані з форми, отримує значення з БД та генерує HTML-код шахової таблиці без Razor-циклів, використовуючи StringBuilder.
3. **Відображення результату:**
   * Згенерований HTML-код передається у ViewBag і відображається у Display.cshtml разом із кількістю рядків та стовпчиків.

## **Висновок**

Створений MVC WEB-застосунок успішно реалізує генерацію шахової таблиці із заданими параметрами. Він ефективно взаємодіє з базою даних через ADO.NET та забезпечує відображення таблиці без використання Razor-циклів. Завдяки загальному шаблону \_Layout.cshtml додаток має адаптивний дизайн і зручний інтерфейс. Реалізація відповідає поставленому завданню та забезпечує коректну роботу з параметрами введення.

# **Завдання 4**

Створити подання із переходами на інші подання через відповідні контролери, відповідно до схеми схемою власного варіанту.

|  |  |
| --- | --- |
| № Варіанту | Схема переходів |
| 1 | 2 |
| 12 |  |

## **Текст програми:**

**1. Контролери:**

* **Page1Controller.cs** – контролер для Сторінки 1, який повертає подання, що містить кнопку переходу до Стор3.
* **Page2Controller.cs** – контролер для Сторінки 2, який відповідає за відображення Сторінки 2 та має кнопку для повернення на Стор1 (або інший напрямок, згідно з вашою схемою).
* **Page3Controller.cs** – контролер для Сторінки 3, що повертає подання з кнопкою переходу до Стор4.
* **Page4Controller.cs** – контролер для Сторінки 4, що повертає подання з кнопкою переходу до Стор5.
* **Page5Controller.cs** – контролер для Сторінки 5, що повертає подання з кнопкою переходу на Стор2.

**Page1Controller.cs:**

using Microsoft.AspNetCore.Mvc;

namespace Task\_4.Controllers

{

public class Page1Controller : Controller

{

// GET: Page1/Index

// Метод повертає подання для Сторінки 1

public IActionResult Index()

{

// Повертаємо View для відображ Стор 1

return View();

}

}

}

**Page2Controller.cs:**

using Microsoft.AspNetCore.Mvc;

namespace Task\_4.Controllers

{

public class Page2Controller : Controller

{

// GET: Page2/Index

// Повертаємо подання для Стор 2

public IActionResult Index()

{

// Повертаємо View для Стор 2

return View();

}

}

}

**Page3Controller.cs:**

using Microsoft.AspNetCore.Mvc;

namespace Task\_4.Controllers

{

public class Page3Controller : Controller

{

// GET: Page3/Index

// Повертаємо подання для Стор 3

public IActionResult Index()

{

// Повертаємо подання View для Стор 3

return View();

}

}

}

**Page4Controller.cs:**

using Microsoft.AspNetCore.Mvc;

namespace Task\_4.Controllers

{

public class Page4Controller : Controller

{

// GET: Page4/Index

// Повертаємо View для Сторінки 4

public IActionResult Index()

{

// Повертаємо View Стор 4)

return View();

}

}

}

**Page5Controller.cs:**

using Microsoft.AspNetCore.Mvc;

namespace Task\_4.Controllers

{

public class Page5Controller : Controller

{

// GET: Page5/Index

// Метод повертає подання для Стор 5

public IActionResult Index()

{

// Повертаємо подання для Стор 5

return View();

}

}

}

**2. Подання (Views):**

* **Index.cshtml** для кожної сторінки (Page1, Page2, Page3, Page4, Page5) – ці файли демонструють оформлення кожної окремої сторінки (задання фонового зображення через ViewBag, відображення тексту та кнопок для навігації згідно з заданою схемою переходів).
* **\_Layout.cshtml** – загальний шаблон застосунку, який забезпечує єдине оформлення (Navbar, основний контейнер, футер, фон). Він показує використання Bootstrap та власних стилів.
* **\_Layout.cshtml.css** – файл, який містить кастомні CSS-стилі для загального оформлення веб-застосунку (Navbar, контейнер, футер, фон тощо).

**Page1/Index.cshtml:**

@{

ViewData["Title"] = "Стор1";

// Задаємо фон для Стор1 через ViewBag

ViewBag.BackgroundUrl = "https://travel.home.sndimg.com/content/dam/images/travel/stock/2016/9/9/0/GettyImages-SapnaReddyPhotography-459319033\_Yosemite.jpg.rend.hgtvcom.1280.853.suffix/1491594411873.jpeg";

}

<!-- Контейнер для вмісту сторінки -->

<div class="container my-5">

<h2 class="mb-3">Сторінка 1</h2>

<p>Це Стор1. Перейдіть далі, натиснувши кнопку нижче.</p>

<!-- Кнопка для переходу на сторінку 3 -->

<a asp-controller="Page3" asp-action="Index" class="btn btn-primary">Перейти на Стор3</a>

</div>

**Page2/Index.cshtml:**

@{

ViewData["Title"] = "Стор2";

// Задаємо фон для сторінки 2 через ViewBag

ViewBag.BackgroundUrl = "https://images.unsplash.com/photo-1498429089284-41f8cf3ffd39?fm=jpg&q=60&w=3000&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxzZWFyY2h8OHx8d2FsbHBhcGVyJTIwbmF0dXJlfGVufDB8fDB8fHww";

}

<!-- Контейнер з відступами для розміщення вмісту -->

<div class="container my-5">

<h2 class="mb-3">Сторінка 2</h2>

<p>Це Стор2. Перейдіть далі, натиснувши кнопку нижче.</p>

<!-- Кнопка для повернення на Стор1 -->

<a asp-controller="Page1" asp-action="Index" class="btn btn-primary">Повернутися на Стор1</a>

</div>

**Page3/Index.cshtml:**

@{

ViewData["Title"] = "Стор3";

// Задаємо фон для Стор3 через ViewBag

ViewBag.BackgroundUrl = "https://www.baltana.com/files/wallpapers-17/Yosemite-Valley-California-USA-Wallpaper-44410.jpg";

}

<!-- Основний контейнер для вмісту сторінки -->

<div class="container my-5">

<h2 class="mb-3">Сторінка 3</h2>

<p>Це Стор3. Перейдіть до наступної сторінки.</p>

<!-- Кнопка для переходу на Стор4 -->

<a asp-controller="Page4" asp-action="Index" class="btn btn-primary">Перейти на Стор4</a>

</div>

**Page4/Index.cshtml:**

@{

ViewData["Title"] = "Стор4";

// Задаємо фон для Стор4 через ViewBag

ViewBag.BackgroundUrl = "https://wallpapercave.com/wp/wp8975834.jpg";

}

<!-- Основний контейнер для розміщення вмісту -->

<div class="container my-5">

<h2 class="mb-3">Сторінка 4</h2>

<p>Це Стор4. Перейдіть до наступної сторінки.</p>

<!-- Кнопка для переходу на Стор5 -->

<a asp-controller="Page5" asp-action="Index" class="btn btn-primary">Перейти на Стор5</a>

</div>

**Page5/Index.cshtml:**

@{

ViewData["Title"] = "Стор5";

// Задаємо фон для Стор5 через ViewBag

ViewBag.BackgroundUrl = "https://assets.tommackie.com/wp-content/uploads/2021/05/25145056/180540-1-1.jpg";

}

<!-- Основний контейнер для вмісту сторінки -->

<div class="container my-5">

<h2 class="mb-3">Сторінка 5</h2>

<p>Це Стор5. Перейдіть до наступної сторінки.</p>

<!-- Кнопка для переходу на Стор2 -->

<a asp-controller="Page2" asp-action="Index" class="btn btn-primary">Перейти на Стор2</a>

</div>

**\_Layout.cshtml:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>@ViewData["Title"] - Task\_4</title>

<!-- Підключення Bootstrap CSS -->

<link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />

<!-- Підключення загальних стилів сайту -->

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

<!-- Підключення кастомних стилів для Task\_4 -->

<link rel="stylesheet" href="~/Task\_4.styles.css" asp-append-version="true" />

</head>

<body style="background: url('@(ViewBag.BackgroundUrl ?? "https://images4.alphacoders.com/122/1222302.jpg")') no-repeat center center fixed; background-size: cover;">

<header>

<!-- Навігаційна панель з Bootstrap -->

<nav class="navbar navbar-expand-sm navbar-light bg-white border-bottom box-shadow mb-3">

<div class="container-fluid">

<!-- Назва сайту посилання на Home -->

<a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">Task\_4</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse"

aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<!-- Меню навігації -->

<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">

<ul class="navbar-nav flex-grow-1">

<li class="nav-item">

<a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>

</li>

</ul>

</div>

</div>

</nav>

</header>

<!-- Основний контейнер -->

<div class="container my-5" style="background-color: rgba(255,255,255,0.9); border-radius: 10px; padding: 20px;">

<main role="main" class="pb-3">

@RenderBody() <!-- Тут рендер вмісту конкретної сторінки -->

</main>

</div>

<!-- Футер -->

<footer class="border-top footer mt-4" style="background-color: rgba(0,0,0,0.5); color: #FFD700;">

<div class="container">

&copy; 2025 - Task\_4 -

<a asp-area="" asp-controller="Home" asp-action="Privacy" style="color: #FFD700;">Privacy</a>

</div>

</footer>

<!-- Підключення jQuery, Bootstrap JS та власних скриптів -->

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

<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>

<script src="~/js/site.js" asp-append-version="true"></script>

@await RenderSectionAsync("Scripts", required: false)

</body>

</html>

**\_Layout.cshtml.css:**

/\* Please see documentation at https://learn.microsoft.com/aspnet/core/client-side/bundling-and-minification

for details on configuring this project to bundle and minify static web assets. \*/

/\* Стиль для логотипу у navbar \*/

a.navbar-brand {

white-space: normal;

text-align: center;

word-break: break-all;

}

/\* Загальний стиль для посилань \*/

a {

color: #0077cc;

}

/\* Стиль для кнопок primary \*/

.btn-primary {

color: #fff;

background-color: #1b6ec2;

border-color: #1861ac;

}

/\* Стиль для активних елементів у nav-pills \*/

.nav-pills .nav-link.active, .nav-pills .show > .nav-link {

color: #fff;

background-color: #1b6ec2;

border-color: #1861ac;

}

/\* Стиль для верхньої межі \*/

.border-top {

border-top: 1px solid #e5e5e5;

}

/\* Стиль для нижньої межі \*/

.border-bottom {

border-bottom: 1px solid #e5e5e5;

}

/\* Додаємо легку тінь для ефекту об'єму \*/

.box-shadow {

box-shadow: 0 .25rem .75rem rgba(0, 0, 0, .05);

}

/\* Стиль для кнопки прийняття політики \*/

button.accept-policy {

font-size: 1rem;

line-height: inherit;

}

/\* Стиль для футера \*/

.footer {

position: absolute;

bottom: 0;

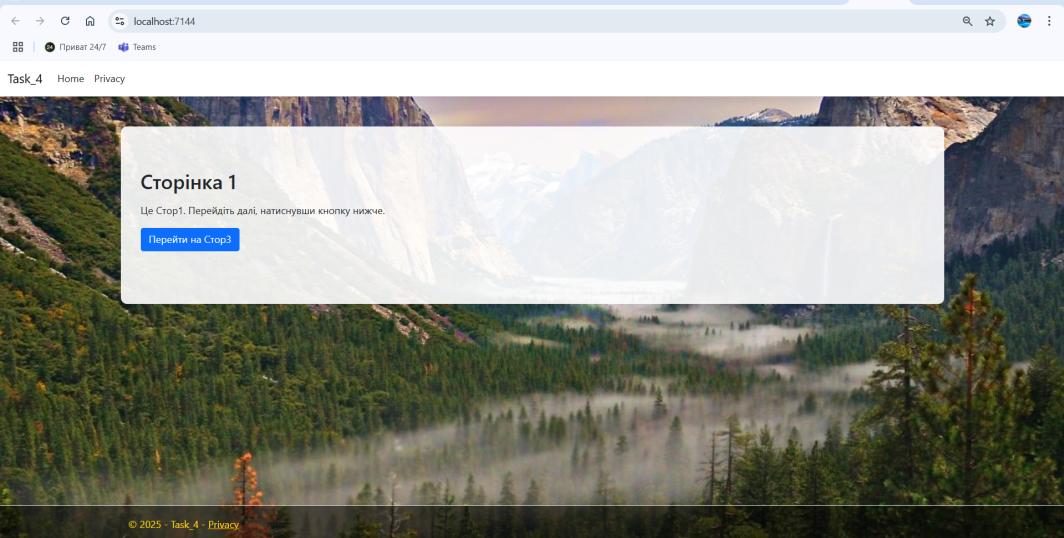
width: 100%;

white-space: nowrap;

line-height: 60px;

}

## **Результат роботи програми:**



Изображение выглядит как текст, снимок экрана, растение, дерево

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, вода, озеро, дерево

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, на открытом воздухе, пейзаж

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, гора, пейзаж

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, озеро, вода

Контент, сгенерированный ИИ, может содержать ошибки.

## **Алгоритм побудови проєкту і кодів**

1. **Створення проєкту:**
   * У Visual Studio додав до існуючого рішення новий проєкт типу ASP.NET Core MVC з назвою *Task\_4*.
2. **Налаштування маршрутів:**
   * У файлі **Program.cs** вказав маршрут за замовчуванням так, щоб відкривалася перша сторінка.
3. **Розробка контролерів (Page1Controller, Page2Controller, Page3Controller, Page4Controller, Page5Controller):**
   * У кожному контролері є метод **Index()**, який повертає відповідне подання (Index.cshtml).
   * Кожен контролер відповідає за свою сторінку (Стор1, Стор2, Стор3, Стор4, Стор5).
   * Схема переходів реалізується через кнопки (або посилання) у відповідному поданні (наприклад, Стор1 має кнопку, що веде до Стор3, Стор3 – до Стор4 тощо).
4. **Розробка подань для кожної сторінки (Page1/Index.cshtml, Page2/Index.cshtml, Page3/Index.cshtml, Page4/Index.cshtml, Page5/Index.cshtml):**
   * Кожне подання відображає свою сторінку з потрібним вмістом (заголовок, текст) і має кнопку для переходу на іншу сторінку за схемою.
   * Для передачі даних про фонове зображення або інші параметри використовуємо ViewBag або прописуємо прямо в коді подання.
5. **Використання \_Layout.cshtml:**
   * Створив загальний шаблон **\_Layout.cshtml** із підключенням Bootstrap для оформлення (Navbar, контейнер, футер).
   * У кожній сторінці (Page1, Page2 тощо) можна задати власне фонове зображення через ViewBag.BackgroundUrl або прямо в поданні.
6. **Тестування:**
   * Запустив застосунок, відкрив Стор1.
   * Перевірив, що кнопка на Стор1 веде до Стор3, Стор3 веде до Стор4, Стор4 – до Стор5, Стор5 – до Стор2, а Стор2 – назад на Стор1.
   * Переконався, що переходи працюють безпомилково та відображаються вірні подання.

## **Функціональність**

* **Переходи між сторінками:** Кожна сторінка має власний контролер і подання, де передбачено кнопку для переходу на іншу сторінку.
* **Схема навігації:** Стор1 → Стор3, Стор3 → Стор4, Стор4 → Стор5, Стор5 → Стор2, Стор2 → Стор1 (згідно з варіантом №12).
* **Оформлення:** Використано Bootstrap для сучасного адаптивного дизайну (\_Layout.cshtml).

## **Висновок**

Завдання **Task\_4** виконано згідно з вимогами варіанта №12. У проєкті створено п’ять сторінок (Стор1…Стор5) із контролерами та поданнями. Кожна сторінка має кнопку для переходу до наступної сторінки за заданою схемою, забезпечуючи циклічну навігацію. Оформлення сторінок реалізовано за допомогою Bootstrap у загальному шаблоні \_Layout.cshtml.

# **Завдання 5**

Використовуючи сесійні атрибути (Session) модифікувати минуле завдання таким чином, щоб на кожному поданні відображався лічильник, який рахував би кількість відвідувань відповідного подання.

**Текст програми:**

**1. Контролери**

**Page1Controller.cs**, **Page2Controller.cs**, **Page3Controller.cs**, **Page4Controller.cs**, **Page5Controller.cs**

Кожен контролер містить метод Index(), де:

* 1. Зчитується поточне значення лічильника зі сесії (наприклад, Page1Count),
  2. Інкрементується (додається 1),
  3. Записується назад у сесію,
  4. Передається у ViewBag.VisitCount для відображення у відповідному поданні.

**Page1Controller.cs:**

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

namespace Task\_5.Controllers

{

public class Page1Controller : Controller

{

// GET: Page1/Index

public IActionResult Index()

{

// Ключ для зберігання ліч Стор 1

const string sessionKey = "Page1Count";

// Зчитуємо поточ значення ліч

int count = (HttpContext.Session.GetInt32(sessionKey) ?? 0) + 1;

// Записуємо оновлене

HttpContext.Session.SetInt32(sessionKey, count);

// Передаємо значення лічильника у ViewBag

ViewBag.VisitCount = count;

return View();

}

}

}

**Page2Controller.cs:**

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

namespace Task\_5.Controllers

{

public class Page2Controller : Controller

{

// GET: Page2/Index

public IActionResult Index()

{

// Визначаємо ключ для збереження ліч відв Стор 2

const string sessionKey = "Page2Count";

// Зчитуємо поточне знач з сесії

int count = (HttpContext.Session.GetInt32(sessionKey) ?? 0) + 1;

// Записуємо оновлене значення

HttpContext.Session.SetInt32(sessionKey, count);

// Передаємо оновлений ліч у ViewBag

ViewBag.VisitCount = count;

return View();

}

}

}

**Page3Controller.cs:**

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

namespace Task\_5.Controllers

{

public class Page3Controller : Controller

{

// GET: Page3/Index

public IActionResult Index()

{

// Визначаємо ключ сесії для збереження ліч відв Стор 3

const string sessionKey = "Page3Count";

// Зчитуємо поточне знач з сесії

int count = (HttpContext.Session.GetInt32(sessionKey) ?? 0) + 1;

// Записуємо оновлене значення

HttpContext.Session.SetInt32(sessionKey, count);

// Передаємо оновлений ліч у ViewBag

ViewBag.VisitCount = count;

return View();

}

}

}

**Page4Controller.cs:**

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

namespace Task\_5.Controllers

{

public class Page4Controller : Controller

{

// GET: Page4/Index

public IActionResult Index()

{

// Визначаємо ключ сесії для збереження ліч відв Стор 4

const string sessionKey = "Page4Count";

// Зчитуємо поточне знач з сесії

int count = (HttpContext.Session.GetInt32(sessionKey) ?? 0) + 1;

// Записуємо оновлене значення ліч

HttpContext.Session.SetInt32(sessionKey, count);

// Передаємо значення ліч у ViewBag

ViewBag.VisitCount = count;

return View();

}

}

}

**Page5Controller.cs:**

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

namespace Task\_5.Controllers

{

public class Page5Controller : Controller

{

// GET: Page5/Index

public IActionResult Index()

{

// Визначаємо ключ для ліч відв Стор 5

const string sessionKey = "Page5Count";

// Зчитуємо поточне знач з сесії

int count = (HttpContext.Session.GetInt32(sessionKey) ?? 0) + 1;

// Записуємо оновлене знач

HttpContext.Session.SetInt32(sessionKey, count);

// Передаємо ліч у ViewBag

ViewBag.VisitCount = count;

return View();

}

}

}

**Подання (Views)**

**Index.cshtml** у кожній папці (Page1, Page2, Page3, Page4, Page5).

* У цих поданнях відображається лічильник відвідувань: @ViewBag.VisitCount.
* Міститься кнопка для переходу до наступної сторінки (згідно з заданою схемою).
* Це найкраще показує, як відображається результат підрахунку відвідувань та організовано навігацію.

**Page1/Index.cshtml:**

@{

ViewData["Title"] = "Сторінка 1";

// Задаємо фон для Стор 1 через ViewBag

ViewBag.BackgroundUrl = "https://static.vecteezy.com/system/resources/previews/049/547/631/non\_2x/stunning-high-resolution-nature-and-landscape-backgrounds-breathtaking-scenery-in-hd-free-photo.jpg";

}

<!-- Контейнер для основного вмісту сторінки -->

<div class="container my-5">

<h2 class="mb-3">Сторінка 1</h2>

<!-- Відображення ліч відвідувань -->

<p class="mb-3">Відвідувань: @ViewBag.VisitCount</p>

<!-- Опис сторінки -->

<p>Це Сторінка 1.</p>

<!-- Кнопка для переходу на Стор 3 -->

<a asp-controller="Page3" asp-action="Index" class="btn btn-primary">Далі</a>

</div>

**Page2/Index.cshtml:**

@{

ViewData["Title"] = "Сторінка 2";

// Задаємо фон для Стор 2 через ViewBag

ViewBag.BackgroundUrl = "https://www.bsr.org/images/heroes/bsr-focus-nature-hero.jpg";

}

<!-- Контейнер для основного вмісту сторінки -->

<div class="container my-5">

<h2 class="mb-3">Сторінка 2</h2>

<!-- Відображення ліч відвідувань -->

<p class="mb-3">Відвідувань: @ViewBag.VisitCount</p>

<p>Це Сторінка 2.</p>

<!-- Кнопка для переходу на Стор 1 -->

<a asp-controller="Page1" asp-action="Index" class="btn btn-primary">Далі</a>

</div>

**Page3/Index.cshtml:**

@{

ViewData["Title"] = "Сторінка 3";

// Задаємо фон для Стор 3 через ViewBag

ViewBag.BackgroundUrl = "https://images.pexels.com/photos/158063/bellingrath-gardens-alabama-landscape-scenic-158063.jpeg?cs=srgb&dl=pexels-pixabay-158063.jpg&fm=jpg";

}

<!-- Контейнер для основного вмісту сторінки -->

<div class="container my-5">

<h2 class="mb-3">Сторінка 3</h2>

<!-- Виводимо ліч відвідувань -->

<p class="mb-3">Відвідувань: @ViewBag.VisitCount</p>

<p>Це Сторінка 3.</p>

<!-- Кнопка для переходу до Стор 4 -->

<a asp-controller="Page4" asp-action="Index" class="btn btn-primary">Далі</a>

</div>

**Page4/Index.cshtml:**

@{

ViewData["Title"] = "Сторінка 4";

// Задаємо фон для Стор 4 через ViewBag

ViewBag.BackgroundUrl = "https://images.squarespace-cdn.com/content/v1/5d777de8109c315fd22faf3a/1693407136044-G90XQURX1GABMYGAS8K1/shutterstock\_1288634614.jpg?format=2500w";

}

<!-- Контейнер для основного вмісту сторінки -->

<div class="container my-5">

<h2 class="mb-3">Сторінка 4</h2>

<!-- Вивід ліч відвідувань -->

<p class="mb-3">Відвідувань: @ViewBag.VisitCount</p>

<p>Це Сторінка 4.</p>

<!-- Кнопка для переходу на Стор 5-->

<a asp-controller="Page5" asp-action="Index" class="btn btn-primary">Далі</a>

</div>

**Page5/Index.cshtml:**

@{

ViewData["Title"] = "Сторінка 5";

// Задаємо фон для Стор 5 через ViewBag

ViewBag.BackgroundUrl = "https://assets.tommackie.com/wp-content/uploads/2021/05/25145056/180540-1-1.jpg";

}

<!-- Контейнер для основного вмісту з відступами -->

<div class="container my-5">

<h2 class="mb-3">Сторінка 5</h2>

<!-- Відображення ліч відвідувань -->

<p class="mb-3">Відвідувань: @ViewBag.VisitCount</p>

<p>Це Сторінка 5.</p>

<!-- Кнопка "Далі" для переходу на Стор 2 -->

<a asp-controller="Page2" asp-action="Index" class="btn btn-primary">Далі</a>

</div>

**2. Подання (Views):**

* **Index.cshtml** для кожної сторінки (Page1, Page2, Page3, Page4, Page5) – ці файли демонструють оформлення кожної окремої сторінки (задання фонового зображення через ViewBag, відображення тексту та кнопок для навігації згідно з заданою схемою переходів).
* **\_Layout.cshtml** – загальний шаблон застосунку, який забезпечує єдине оформлення (Navbar, основний контейнер, футер, фон). Він показує використання Bootstrap та власних стилів.
* **\_Layout.cshtml.css** – файл, який містить кастомні CSS-стилі для загального оформлення веб-застосунку (Navbar, контейнер, футер, фон тощо).

**Index.cshtml:**

@{

ViewData["Title"] = "Home Page";

}

<!-- Контейнер для центрування вмісту на сторінці -->

<div class="text-center">

<h1 class="display-4">Welcome</h1>

<p>

Learn about

<a href="https://learn.microsoft.com/aspnet/core">building Web apps with ASP.NET Core</a>.

</p>

</div>

**\_Layout.cshtml:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>@ViewData["Title"] - Task\_5</title>

<!-- Підключення Bootstrap CSS -->

<link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />

<!-- Підключення основних стилів сайту -->

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

<!-- Підключення кастомних стилів для Task\_5 -->

<link rel="stylesheet" href="~/Task\_5.styles.css" asp-append-version="true" />

</head>

<body style="background: url('@(ViewBag.BackgroundUrl ?? "https://static.vecteezy.com/system/resources/previews/049/547/631/non\_2x/stunning-high-resolution-nature-and-landscape-backgrounds-breathtaking-scenery-in-hd-free-photo.jpg")') no-repeat center center fixed; background-size: cover;">

<header>

<!-- Navbar з Bootstrap -->

<nav class="navbar navbar-expand-sm navbar-light bg-white border-bottom box-shadow mb-3">

<div class="container-fluid">

<!-- Назва сайту де посилання веде на Home -->

<a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">Task\_5</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse"

aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<!-- Меню навігації -->

<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">

<ul class="navbar-nav flex-grow-1">

<li class="nav-item">

<a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>

</li>

</ul>

</div>

</div>

</nav>

</header>

<!-- Основний контейнер для контенту -->

<div class="container my-5" style="background-color: rgba(255,255,255,0.9); border-radius: 10px; padding: 20px;">

<main role="main" class="pb-3">

@RenderBody() <!-- Рендер контенту конкретної стор -->

</main>

</div>

<!-- Футер -->

<footer class="border-top footer mt-4" style="background-color: rgba(0,0,0,0.5); color: #FFD700;">

<div class="container">

&copy; 2025 - Task\_5 -

<a asp-area="" asp-controller="Home" asp-action="Privacy" style="color: #FFD700;">Privacy</a>

</div>

</footer>

<!-- Підключення jQuery та Bootstrap JS -->

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

<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>

<!-- Підключення основних скриптів сайту -->

<script src="~/js/site.js" asp-append-version="true"></script>

@await RenderSectionAsync("Scripts", required: false)

</body>

</html>

**\_Layout.cshtml.css:**

/\* Please see documentation at https://learn.microsoft.com/aspnet/core/client-side/bundling-and-minification

for details on configuring this project to bundle and minify static web assets. \*/

/\* Стиль для логотипу в Navbar \*/

a.navbar-brand {

white-space: normal;

text-align: center;

word-break: break-all;

}

/\* Загальний стиль для посилань \*/

a {

color: #0077cc;

}

/\* Стиль для кнопок з класом btn-primary \*/

.btn-primary {

color: #fff;

background-color: #1b6ec2;

border-color: #1861ac;

}

/\* Стиль для активних елементів у nav-pills \*/

.nav-pills .nav-link.active,

.nav-pills .show > .nav-link {

color: #fff;

background-color: #1b6ec2;

border-color: #1861ac;

}

/\* Стиль для верхньої межі елементів \*/

.border-top {

border-top: 1px solid #e5e5e5;

}

/\* Стиль для нижньої межі елементів \*/

.border-bottom {

border-bottom: 1px solid #e5e5e5;

}

/\* Стиль для додання легкого об'ємного ефекту \*/

.box-shadow {

box-shadow: 0 .25rem .75rem rgba(0, 0, 0, .05);

}

/\* Стиль для кнопки прийняття політики \*/

button.accept-policy {

font-size: 1rem;

line-height: inherit;

}

/\* Стиль для футера \*/

.footer {

position: absolute;

bottom: 0;

width: 100%;

white-space: nowrap;

line-height: 60px;

}

## **Результати роботи програми:**

Изображение выглядит как текст, трава, снимок экрана, растение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, растение, цветок, дерево

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как вода, снимок экрана, текст, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, гора, на открытом воздухе

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, вода, снимок экрана, пейзаж

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, трава, снимок экрана, растение

Контент, сгенерированный ИИ, может содержать ошибки.

## **Алгоритм побудови проєкту і кодів**

1. **Створення проєкту:**
   * У Visual Studio додав до існуючого рішення новий проєкт типу ASP.NET Core MVC з назвою *Task\_5*.
2. **Налаштування підтримки сесій:**
   * У файлі **Program.cs** додано служби:
     + AddDistributedMemoryCache() – для зберігання сесій у пам’яті.
     + AddSession() – для використання сесійних атрибутів.
   * У конвеєрі HTTP-запитів увімкнув використання сесій за допомогою app.UseSession() перед визначенням маршруту.
3. **Розробка контролерів:**
   * Створив 5 контролерів (Page1Controller, Page2Controller, Page3Controller, Page4Controller, Page5Controller), кожен з яких містить метод **Index()**.
   * У кожному методі **Index()** реалізовано:
     + Зчитування поточного значення лічильника із сесії за допомогою HttpContext.Session.GetInt32(...) (використовуючи унікальний ключ для кожної сторінки, наприклад, "Page1Count").
     + Інкрементування значення (додавання 1).
     + Запис оновленого значення назад у сесію за допомогою HttpContext.Session.SetInt32(...).
     + Передача значення лічильника у ViewBag (наприклад, ViewBag.VisitCount) для подальшого відображення у поданні.
4. **Розробка подань (Views):**
   * Створено окремі подання для кожної сторінки (Page1/Index.cshtml, Page2/Index.cshtml, …, Page5/Index.cshtml), у яких:
     + Відображається лічильник відвідувань (наприклад, Відвідувань: @ViewBag.VisitCount).
     + Реалізовано навігацію між сторінками відповідно до схеми (наприклад, Стор1 → Стор3, Стор3 → Стор4, Стор4 → Стор5, Стор5 → Стор2, Стор2 → Стор1).
   * Загальний шаблон **\_Layout.cshtml** забезпечує єдиний вигляд застосунку, підключає Bootstrap та власні стилі.
5. **Тестування:**
   * Запустив застосунок.
   * Перевірив, що при кожному переході на конкретну сторінку лічильник відвідувань оновлюється і відображається в поданні.
   * Переконався, що навігація між сторінками працює відповідно до заданої схеми.

## **Функціональність**

* **Підрахунок відвідувань:**

За допомогою сесійних атрибутів кожна сторінка має свій власний лічильник відвідувань, який збільшується при кожному перегляді.

* **Навігація:**

Кожна сторінка містить кнопку для переходу на наступну сторінку згідно з наступною схемою:

* + Стор1 → Стор3
  + Стор3 → Стор4
  + Стор4 → Стор5
  + Стор5 → Стор2
  + Стор2 → Стор1
* **Оформлення:**

Загальне оформлення застосунку реалізовано за допомогою Bootstrap та власних CSS, що забезпечує адаптивний і сучасний вигляд.

## **Висновок**

Завдання Task\_5 виконано відповідно до вимог варіанта №12. Реалізовано механізм підрахунку відвідувань кожної сторінки за допомогою сесійних атрибутів, що демонструється через оновлення лічильників у кожному контролері і їх відображення у відповідних поданнях. Також реалізовано навігацію між 5 сторінками відповідно до заданої схеми. Оформлення застосунку виконано за допомогою Bootstrap, що забезпечує сучасний та адаптивний інтерфейс.

# **Загальний висновок**

1. **Завдання 1:**

У ході виконання першого завдання було створено MVC-застосунок, який дозволяє передавати параметри через контролер у подання. Використано механізм ViewBag для передачі даних між компонентами. Реалізовано загальний шаблон оформлення (\_Layout.cshtml), що забезпечує сучасний та адаптивний дизайн застосунку.

1. **Завдання 2\_1:**

Реалізовано веб-застосунок із можливістю завантаження зображень, розміри яких отримуються з БД. Використано механізм ViewBag для динамічного відображення зображень відповідно до топології «сходинок». Оформлення виконано з використанням Bootstrap для адаптивного інтерфейсу.

1. **Завдання 2\_2:**

Додано ще одну топологію відображення зображень, що відрізняється від попереднього варіанту. Основна логіка збережена, але змінено CSS-стилі та порядок розташування елементів у поданні Display.cshtml. Завдання виконано з урахуванням вимог варіанта №12.

1. **Завдання 3:**

Реалізовано динамічну генерацію шахової таблиці з введеною користувачем кількістю рядків. Дані про кількість стовпців отримуються з БД. Згенерований HTML-код таблиці передається у ViewBag та виводиться у поданні.

1. **Завдання 4:**

Створено веб-застосунок із 5 сторінками та реалізованою схемою переходів між ними. Навігація між сторінками відбувається за допомогою кнопок, що відповідають заданій у варіанті схемі. Загальний вигляд оформлений за допомогою Bootstrap у шаблоні \_Layout.cshtml.

1. **Завдання 5:**

Додано механізм підрахунку відвідувань кожної сторінки за допомогою сесійних змінних. Підрахунок оновлюється під час кожного перегляду відповідної сторінки. Це дозволяє відстежувати активність користувачів у межах застосунку.

**Загальний висновок**

Лабораторна робота №1 успішно продемонструвала основи використання ASP.NET Core MVC, баз даних та механізмів передачі параметрів між компонентами застосунку. Реалізовані завдання охоплюють ключові концепції, такі як робота з поданнями, контролерами, ViewBag та сесійними змінними. Особливу увагу приділено адаптивному дизайну та зручності використання, що забезпечено завдяки Bootstrap. Усі завдання виконані відповідно до вимог варіанта №12, що дозволяє підтвердити правильність реалізації та ефективність використаних підходів.