Шейко Ростислав. КІ-21-2. Модуль 1

**Запитання 1.**

9. Сервіси MVC. Структура проекту шаблону ASP.NET Core Web App (Model-View-Controller).

ASP.NET Core Web App (Model-View-Controller) є популярним шаблоном для створення веб-додатків з використанням фреймворку ASP.NET Core. Цей шаблон розділяє логіку програми на три основні частини: модель (Model), представлення (View) та контролер (Controller). Нижче наведено загальну структуру проекту шаблону ASP.NET Core Web App (MVC):

1. **Models**: В цій папці зазвичай знаходяться класи, які представляють дані і бізнес-логіку вашого додатку. Ці класи можуть відображати таблиці бази даних, об'єкти JSON, сутності або будь-яку іншу інформацію, яку додаток обробляє.

2. **Views**: Папка "Views" містить представлення, які відображають дані користувачеві. Ці представлення можуть бути створені з використанням HTML, CSS, та C# (зазвичай з використанням технології Razor). Вони розділяються на підпапки для кожного контролера.

3. **Controllers**: Контролери відповідають за обробку HTTP-запитів і взаємодію з моделями та представленнями. Кожен контролер відповідає за конкретний набір операцій і має методи (дії), які обробляють запити від користувачів.

4. **wwwroot**: В цій папці можна зберігати статичні ресурси, такі як зображення, CSS-файли та JavaScript-файли, доступні для веб-клієнтів.

5. **appsettings.json:** Файл конфігурації додатку, де можна зберігати різні налаштування, такі як рядки підключення до бази даних, ключі API, тощо.

7. **Program.cs**: Цей файл містить точку входу до додатку та встановлює, як додаток буде запускатися. Також цей файл визначає налаштування додатку, включаючи маршрутизацію, сервіси та інше.

8. **wwwroot**: В цій папці розташовуються статичні ресурси, такі як CSS-файли, зображення та JavaScript-файли, які можуть бути використані в представленнях.

Ця загальна структура дозволяє розділити логіку додатку на легко керовані компоненти та дозволяє вам створювати додаток з дотриманням принципів розділення обов'язків. Ви можете додавати додаткові файли та папки, які вам потрібні для конкретного проекту, але це основна структура проекту ASP.NET Core Web App (MVC).

Приклад 1.

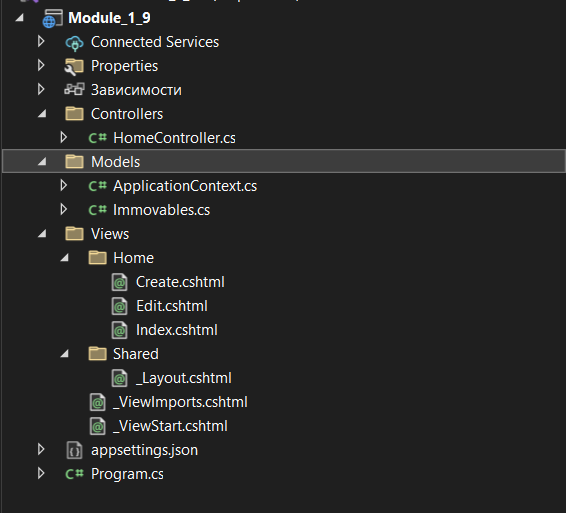


Рис 1. Приблизно ось так виглядає структура проєкту

**Запитання 2.**

19. Передача даних у подання. ViewData.

У шаблоні ASP.NET Core Web App (MVC), дані передаються з контролера до представлення (Views) за допомогою моделей. Це робиться для того, щоб відобразити дані користувачеві на веб-сторінці. Існують кілька способів передачі даних у представлення:

1. Передача даних через модель (Model):

Контролер створює об'єкт моделі, який містить дані, які потрібно відобразити на сторінці. Далі контролер передає цей об'єкт моделі в метод View() як параметр. У представленні ви можете використовувати дані цієї моделі, використовуючи технологію Razor для відображення даних у HTML-розмітці.

1. ViewBag та ViewData:

Viewbag та ViewData є об'єктами, які дозволяють передавати дані з контролера в представлення без необхідності створення окремої моделі. Вони дозволяють передавати дані як ключ-значення і використовувати ці дані у представленні.

1. Передача даних через параметри представлення:

Ви можете передати дані у представлення, вказуючи їх у параметрах методу View(). Це дозволяє передати будь-які дані, які можуть бути використані у представленні.

Зробимо приклад на ViewData:

Лістинг 1. HomeController

using Microsoft.AspNetCore.Mvc;

namespace Module\_1\_19.Controllers

{

public class HomeController : Controller

{

public IActionResult Index()

{

ViewData["Message"] = "Це приклад ViewData!";

return View();

}

}

}

У цьому лістингу ми встановлюємо значення "Message" у ViewData, яке ми хочемо передати в представлення.

В представленні можна отримати доступ до даних, збережених у ViewData, за допомогою ключа, який був використаний у контролері:

Лістинг 2. Index.cshtml

<!DOCTYPE html>

<html>

<head>

<title>Приклад ViewData</title>

</head>

<body>

<h1>Дані з ViewData</h1>

<p>@ViewData["Message"]</p>

</body>

</html>

У цьому прикладі ми виводимо дані, які були встановлені в ViewData у контролері. Ми використовуємо ключ "Message" для доступу до даних та відображення їх у представленні.

Лістинг 3. Program.cs

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllersWithViews();

var app = builder.Build();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();

А тут додаємо міграцію.

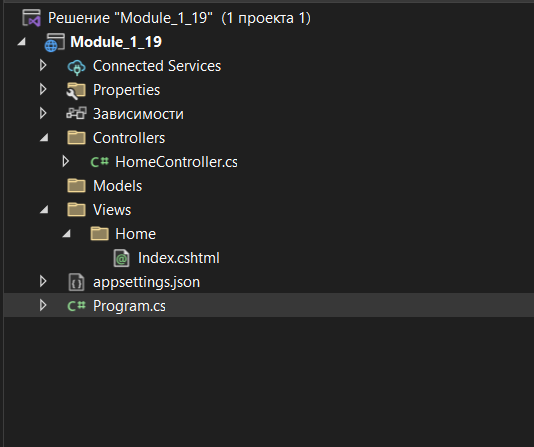


Рис 2. Структура проекту



Рис 3. Вигляд веб сторінки