**ЛАБОРАТОРНА РОБОТА № 2**

***Тема*:** Сервіси та Dependency Injection.

***Мета роботи:*** ознайомитися зі структурою ASP.NET Code додатків, навчитися створювати найпростіші та використовувати наявні компоненти middleware у ASP.NET Core.

**Хід роботи:**

1) Зробити обробку звернення до адреси “/services/list”, реалізувавши виведення інформації про всі сервіси, додані у додаток (тип сервісу, тип реалізації сервісу, життєвий цикл сервісу). У звіті до лабораторної роботи навести коротку інформацію про кожний сервіс (достатньо одного речення, що чітко описує сервіс).

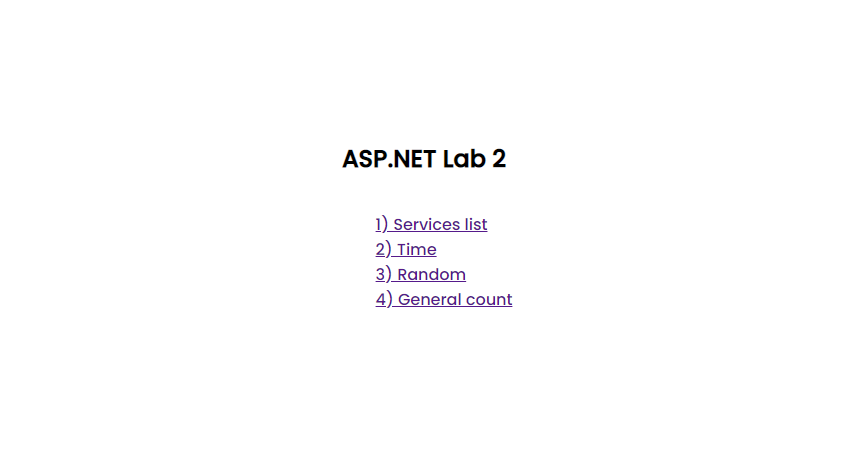


Рис. 1. index.html



Рис. 2. Список всіх сервісів за адресою «/services/list»

Інформація про кожний сервіс:

1. Action Results: Підтримка вихідних результатів дій, таких як JSON-відповіді або перенаправлення.
2. API Explorer: Забезпечує механізм виявлення та документації API-інтерфейсу.
3. API Versioning: Дозволяє керувати версіюванням API-інтерфейсу.
4. Application Insights: Сервіс моніторингу та аналізу продуктивності, що забезпечується Azure.
5. ASP.NET Core Identity: Фреймворк для автентифікації, авторизації та керування ролями.
6. Authentication and Authorization: Дозволяє налаштувати та використовувати різні механізми автентифікації та авторизації.
7. Background Tasks: Дозволяє створювати та виконувати фонові завдання.
8. Caching: Сервіс кешування, що дозволяє зберігати дані у пам'яті, файловій системі або базі даних.
9. Certificates: Дозволяє використовувати SSL-сертифікати для забезпечення безпеки.
10. Configuration: Дозволяє налаштовувати різні параметри додатка, наприклад, рядки підключення до баз даних.
11. CORS: Дозволяє налаштувати міжсайтову політику захисту.
12. Cryptography: Сервіс шифрування, що дозволяє забезпечити безпеку даних.
13. Data Protection: Сервіс захисту даних, що забезпечує їх захист у разі втрати або крадіжки.
14. Dependency Injection: Дозволяє створювати та використовувати залежності між об'єктами.
15. Distributed Caching: Сервіс кешування, що дозволяє зберігати дані у розподіленому середовищі.
16. Distributed Tracing: Сервіс відстеження запитів, що дозвляє знайти проблеми з продуктивністю та відлагодити їх.
17. Email Sender: Сервіс для надсилання електронних листів.
18. Entity Framework Core: ORM-фреймворк для роботи з базами даних.
19. File Providers: Дозволяє управляти файловою системою, наприклад, зчитування та запис файлів
20. Filters: Механізм фільтрації запитів перед їх обробкою.
21. Health Checks: Сервіс перевірки стану додатка.
22. Hosting Environment: Представлення середовища, в якому працює додаток.
23. HTTP Clients: Клієнт для взаємодії з іншими HTTP-сервісами.
24. HTTPS: Дозволяє використовувати протокол HTTPS для забезпечення безпеки під час передачі даних.
25. IIS Integration: Інтеграція з IIS для розгортання додатка на Windows-серверах.
26. JSON Formatters: Форматування даних у форматі JSON.
27. Logging: Сервіс логування для збору та аналізу журналів.
28. MessagePack Formatters: Форматування даних у форматі MessagePack.
29. Middleware: Компоненти, що оброблюють HTTP-запити перед їх обробкою додатком.
30. MVC: Фреймворк для створення веб-додатків з шаблонами.
31. OpenAPI: Сервіс для автоматичної генерації документації API-інтерфейсу.
32. Razor: Фреймворк для створення веб-додатків з використанням Razor-шаблонів.
33. Response Caching: Сервіс кешування вихідних даних відповіді.
34. Response Compression: Сервіс стиснення вихідних даних відповіді.
35. Routing: Сервіс маршрутизації запитів.
36. SignalR: Фреймворк для роботи з веб-сокетами.
37. Static Files: Сервіс для обробки статичних файлів, таких як HTML, CSS та JavaScript.
38. Swagger: Інструмент для автоматичної генерації документації API-інтерфейсу.
39. Telemetry: Сервіс моніторингу та аналізу продуктивності, що забезпечується Azure.
40. Validation: Сервіс валідації даних відповідно до встановлених правил.
41. WebSockets: Механізм для роботи з веб-сокетами, який дозволяє взаємодіяти з клієнтами в реальному часі.
42. Windows Compatibility Pack: Набір компонентів, що дозволяє використовувати функціональність Windows на платформі .NET Core.
43. XML Formatters: Форматування даних у форматі XML.
44. Azure AD Authentication: Сервіс автентифікації користувачів з використанням Azure Active Directory.
45. Azure App Configuration: Сервіс керування конфігурацією додатків на Azure.
46. Azure Blob Storage: Сервіс збереження файлів на Azure Blob Storage.
47. Azure Cache for Redis: Сервіс кешування на Azure Redis Cache.
48. Azure Key Vault Configuration: Сервіс керування конфігурацією додатків з використанням Azure Key Vault.
49. Azure Service Bus: Сервіс для обміну повідомленнями між додатками на Azure.
50. Azure Storage Queues: Сервіс для зберігання повідомлень у черзі на Azure.
51. Azure Table Storage: Сервіс зберігання даних у вигляді таблиць на Azure.
52. Background Tasks: Сервіс для запуску фонових задач в додатку.
53. Configuration: Сервіс для зберігання конфігурації додатку.
54. Distributed Cache: Сервіс кешування даних на різних серверах.
55. Entity Framework 6: ORM-фреймворк для роботи з базами даних.
56. Grpc: Протокол відкритого коду для взаємодії між різними службами.
57. HealthChecks.UI: Інтерфейс користувача для сервісу перевірки стану додатка.
58. Identity: Фреймворк для роботи з автентифікацією та авторизацією користувачів.
59. Microsoft Graph: Сервіс для взаємодії з графом Microsoft.
60. MVC View Features: Функціональність для роботи з відображенням веб-сторінок.
61. Newtonsoft.Json: Бібліотека для роботи з форматом JSON.
62. OAuth: Протокол авторизації та автентифікації користувачів.
63. OpenID
64. Options: Сервіс для зберігання налаштувань додатку.
65. Razor Runtime Compilation: Сервіс для компіляції Razor-сторінок під час роботи додатка.
66. SignalR: Фреймворк для роботи з веб-сокетами.
67. Swagger: Фреймворк для документування та тестування API.
68. System.Net.Http: Бібліотека для роботи з HTTP-запитами.
69. Telemetry: Сервіс для збору та аналізу статистики про додаток.
70. TestServer: Сервіс для тестування веб-додатків.
71. User Secrets Configuration: Сервіс для зберігання конфіденційних даних (таких як паролі) в окремому файлі, що не включається в репозиторій.
72. Web API: Фреймворк для створення API.
73. Web Application: Фреймворк для створення веб-додатків.
74. Web Encoders: Сервіс для кодування та декодування даних у форматі HTML, URL, Base64.
75. Web Host: Сервіс для запуску веб-додатків.
76. Web Infrastructure: Набір компонентів для побудови веб-додатків.
77. Web sockets: Протокол для взаємодії з клієнтами в режимі реального часу.
78. Configuration.Binder: Сервіс для зв'язування значень конфігурації з об'єктами додатку.
79. Https Policy: Сервіс для налаштування політики HTTPS.
80. IIS Integration: Сервіс для інтеграції з IIS.
81. Microsoft.AspNetCore.Mvc.Formatters.Xml: Форматування даних у форматі XML.
82. Microsoft.AspNetCore.ResponseCaching: Сервіс для кешування відповідей веб-додатка.
83. Microsoft.AspNetCore.Rewrite: Сервіс для перенаправлення запитів веб-додатка.
84. Microsoft.AspNetCore.Server.IISIntegration: Сервіс для інтеграції з IIS.
85. Microsoft.AspNetCore.StaticFiles: Сервіс для обробки статичних файлів.
86. Microsoft.Extensions.Caching.Memory: Сервіс для кешування даних у пам'яті.
87. Microsoft.Extensions.Configuration.EnvironmentVariables: Сервіс для зчитування значень конфігурації
88. Microsoft.Extensions.Configuration.FileExtensions: Сервіс для зчитування значень конфігурації з файлів.
89. Microsoft.Extensions.Configuration.Json: Сервіс для зчитування значень конфігурації з файлів у форматі JSON.
90. Microsoft.Extensions.Configuration.UserSecrets: Сервіс для зчитування конфіденційних даних з файлу, що не включається в репозиторій.
91. Microsoft.Extensions.Diagnostics.HealthChecks: Сервіс для перевірки стану додатку (здоров'я) та відображення його у вигляді метрик.
92. Microsoft.Extensions.Logging.Configuration: Сервіс для налаштування логування в додатку.
93. FileServer: обслуговування статичних файлів (зображень, стилів CSS, сценаріїв JavaScript тощо)

2) Створити три сервіси з різними життєвими циклами і продемонструвати їх роботу:

2.1) Тип “Transient”, назва “TimerService” - повертає поточну дату та час. Результати вивести за адресою “/services/timer”

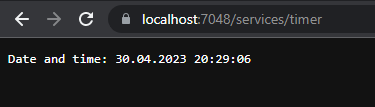


Рис. 3. Transient cервіс TimeService

2.2) Тип “Scoped”, назва “RandomService” - в конструкторі генерує випадкове число. За допомогою методу чи властивості повертає його. Результати роботи вивести за адресою “/services/random”. Має виводитися два числа, які співпадають.

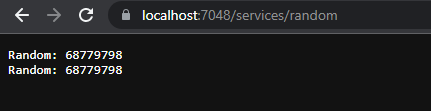


Рис. 4. Scoped сервіс RandomService

2.3) Тип “Singleton”, назва “GeneralCounterService” - веде підрахунок кількості звернень до кожної URL-адреси. Оскільки сервіс матиме тип “Singleton”, то підрахунок кількості буде вестися для усіх запитів. Результати вивести за адресою “/services/general-counter”

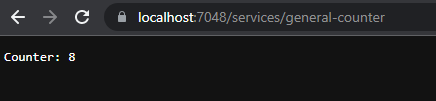


Рис. 5. Singleton сервіс GeneralCounter

3) Для кожного сервісу створити відповідний middleware.

**Лістинг TimeMiddleware:**

using Lab2.Services;

namespace Lab2.Middleware

{

public class TimeMiddleware

{

public readonly RequestDelegate next;

public TimeMiddleware(RequestDelegate next)

{

this.next = next;

}

public async Task InvokeAsync(HttpContext context, ITimeService timeService)

{

await context.Response.WriteAsync($"Date and time: {timeService?.GetDateTime()}");

}

}

}

**Лістинг RandomMiddleware:**

using Lab2.Services;

namespace Lab2.Middleware

{

public class RandomMiddleware

{

public readonly RequestDelegate next;

public RandomMiddleware(RequestDelegate next)

{

this.next = next;

}

public async Task InvokeAsync(HttpContext context, IRandomService randomService)

{

await context.Response.WriteAsync($"Random: {randomService.Number}\nRandom: {randomService.Number}");

}

}

}

**Лістинг GeneralCounterMiddleware:**

using System.Net.Mime;

using Lab2.Services;

namespace Lab2.Middleware

{

public class GeneralCounterMiddleware

{

public readonly RequestDelegate next;

public GeneralCounterMiddleware(RequestDelegate next)

{

this.next = next;

}

public async Task InvokeAsync(HttpContext context, IGeneralCounterService generalCounterService)

{

generalCounterService.Increment();

if (context.Request.Path == "/services/general-counter")

{

await context.Response.WriteAsync($"Counter: {generalCounterService.GetCount()}");

}

else

{

await next.Invoke(context);

}

}

}

}

**Сервіси**

**Лістинг TimeService:**

namespace Lab2.Services

{

public class TimeService : ITimeService

{

public string GetDateTime()

{

return DateTime.Now.ToString();

}

}

}

**Лістинг RandomService:**

namespace Lab2.Services

{

public class RandomService : IRandomService

{

public int Number { get; }

public RandomService()

{

Number = new Random().Next();

}

}

}

**Лістинг GeneralCounterService:**

namespace Lab2.Services

{

public class GeneralCounterService : IGeneralCounterService

{

public int \_count;

public void Increment()

{

\_count++;

}

public int GetCount() => \_count;

}

}

**Лістинг ServicesExtensions:**

namespace Lab2.Services

{

public static class ServicesExtensions

{

public static void AddTimeService(this IServiceCollection services)

{

services.AddTransient<ITimeService, TimeService>();

}

public static void AddRandomService(this IServiceCollection services)

{

services.AddScoped<IRandomService, RandomService>();

}

public static void AddGeneralCounterService(this IServiceCollection services)

{

services.AddSingleton<IGeneralCounterService, GeneralCounterService>();

}

}

}

**Лістинг Program.cs:**

using Lab2.Middleware;

using Lab2.Services;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

var services = builder.Services;

services.AddTimeService();

services.AddRandomService();

services.AddGeneralCounterService();

var app = builder.Build();

app.UseFileServer();

app.UseMiddleware<GeneralCounterMiddleware>();

app.UseWhen(context => context.Request.Path == "/services/timer", appBuilder =>

{

appBuilder.UseMiddleware<TimeMiddleware>();

});

app.UseWhen(context => context.Request.Path == "/services/random", appBuilder =>

{

appBuilder.UseMiddleware<RandomMiddleware>();

});

app.MapGet("/services/list", async (context) =>

{

var sb = new StringBuilder();

sb.Append($"<h1>All services ({services.Count})</h1><table>");

sb.Append("<tr><th>Type</th><th>Lifetime</th><th>Realization</th></tr>");

foreach (var svc in services)

{

sb.Append("<tr>");

sb.Append($"<td>{svc.ServiceType.FullName}</td>");

sb.Append($"<td>{svc.Lifetime}</td>");

sb.Append($"<td>{svc.ImplementationType?.FullName}</td>");

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

}

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

context.Response.ContentType = "text/html;charset=utf-8";

await context.Response.WriteAsync(sb.ToString());

});

app.Run();

**Лістинг index.html:**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title></title>

<link href="style.css" rel="stylesheet" />

</head>

<body>

<h2>ASP.NET Lab 2</h2>

<ul>

<li>

<a href="/services/list">1) Services list</a>

</li>

<li>

<a href="/services/timer">2) Time</a>

</li>

<li>

<a href="/services/random">3) Random</a>

</li>

<li>

<a href="/services/general-counter">4) General count</a>

</li>

</ul>

</body>

</html>

**Лістинг style.css:**

@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@500&display=swap');

\* {

font-family: "Poppins", sans-serif;

}

body {

height: 90vh;

display: flex;

flex-direction:column;

justify-content: center;

align-items: center;

}

li{

list-style-type:none;

}

a{

}

***Висновки:*** під час виконання лабораторної роботи ознайомилися зі структурою ASP.NET Code додатків, навчилися створювати найпростіші та використовувати наявні компоненти middleware у ASP.NET Core.