**ЛАБОРАТОРНА РОБОТА №1**

**Тема:** Початок роботи з ASP.NET Core 5. Middleware

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

**Хід роботи:**

**3. Реалізація статичного сервера.**

Статичний сервер дозволяє передавати у браузер файли, які зберігаються у відповідній папці на сервері. Створюємо папку з назвою “wwwroot”.

Розміщуємо у папці wwwroot декілька статичних файлів з довільним контентом, наприклад файл index.html та таблицю стилів, яка у ньому підключена:

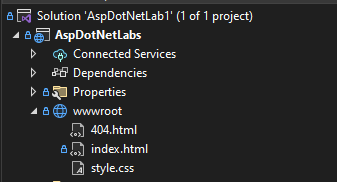


Рис. 1. Створено папку wwwroot і статичні файли

Після рядка app.UseRouting(); додаємо виклик метода: app.UseFileServer();

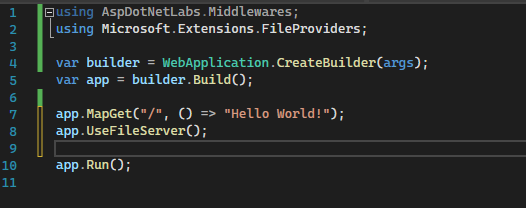


Рис. 2. Додано UseFileServer middleware

Запускаємо проект і дивимось, що виводиться при зверненні до кореня сайту “/” та до сторінки “/index.html”:

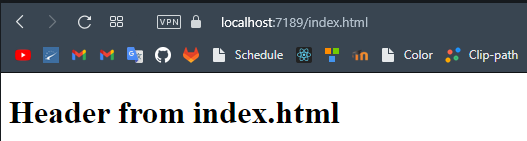


Рис. 3. Результат за адресою /index.html

Міняємо рядки місцями: app.UseFileServer(); app.UseRouting(); Знову запускаємо проект і дивимось, що виводиться при зверненні до кореня сайту “/” та до сторінки “/index.html”. Відмічаємо відмінності.

*Ніяких відмінностей не помічено. Так як якщо одна компонента middleware не відправить контент то керування передається наступній. І наступна компонента вже відправить контент(при умові що адреса передана правильно).*

Додаємо ще один статичний сервер до проекту. Для цього створимо ще одну папку з назвою “static”, у ній - підкаталог “files”, у ньому - файл page.html:

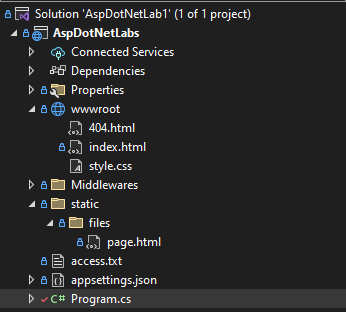


Рис. 4. Створено папка static зі статичними файлами

Додаємо ще один статичний сервер у програмному коді:

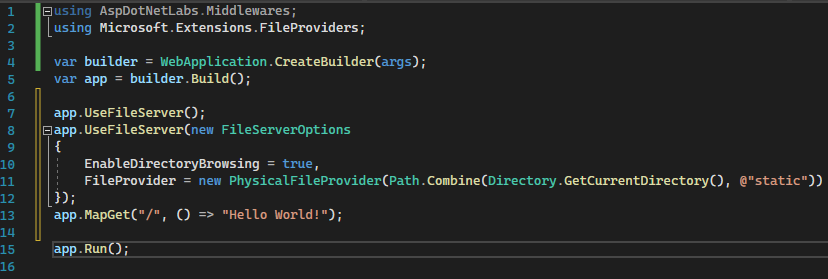


Рис. 5. Додано ще один UseFileServer middleware

Перевіряємо його роботу зайшовши на URL:

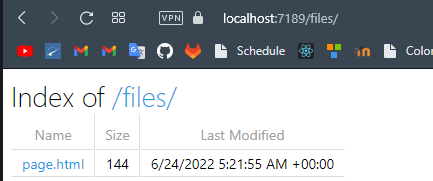


Рис. 6. Результат за адресою /files

Також перевіряємо, щоб працював попередній статичний сервер та відображалася дефолтна сторінка, що формується ASP.NET додатком:

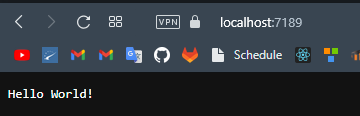


Рис. 7. Результат за адресою /

**4. Реалізація обробки помилок.**

Відповідно до слайдів 68-74 до лекції №1 реалізуйте виведення повідомлення про виключення, які потенційно можуть статися у додатку

Відповідно до слайдів 75-78 до лекції №1 реалізуйте виведення повідомлення про помилку 404 з програмного коду

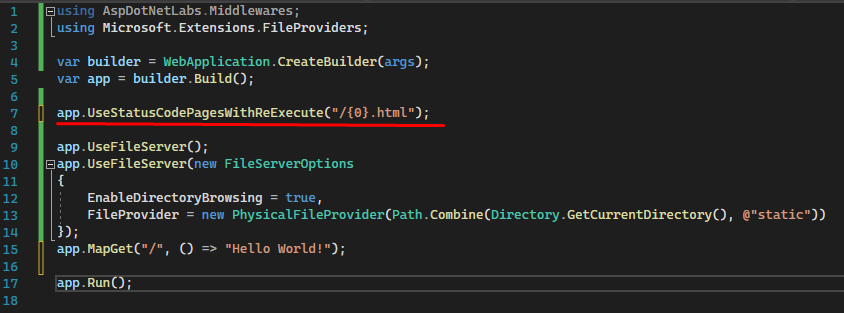


Рис. 8. Додано UseStatusCodePagesWithReExcecute middleware

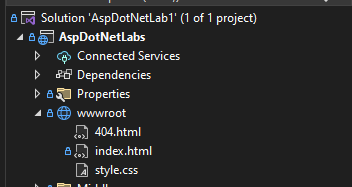


Рис. 9. Створено файл 404.html для обробки помилки

**5. Обробка маршрутів.**

За допомогою методів Map реалізуйте обробники маршрутів /home/index та /home/about відповідно до слайдів 24-25 презентації до лекцій №1. Обробку “Page Not Found” реалізовувати не потрібно, оскільки вже реалізовано обробку сторінок помилок.

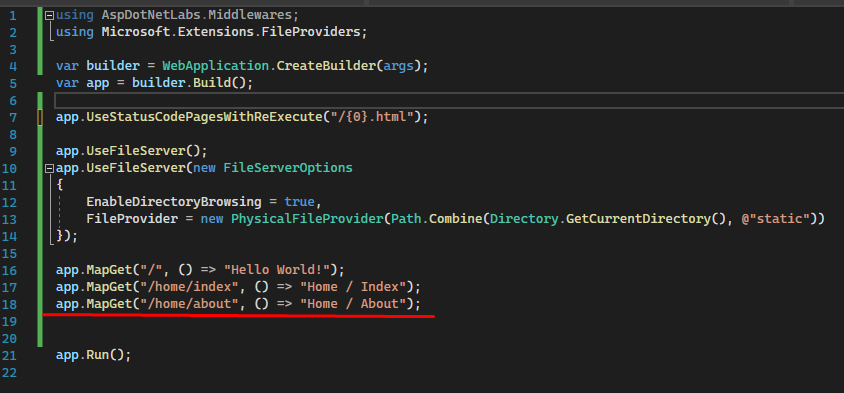


Рис. 10. Додано middleware`и для обробки адрес /home/index та /home/about

**6. Реалізація власних компонентів middleware**

6.1. Створіть та підключіть у проекті власний компонент з назвою LoggerMiddleware, який буде записувати інформацію про усі запити до вашого сайту у текстовий файл з назвою access.txt. Для запису у файл можна використати метод: File.AppendAllText("access.txt", $"{DateTime.Now.ToString()} {context.Request.Path}\n");

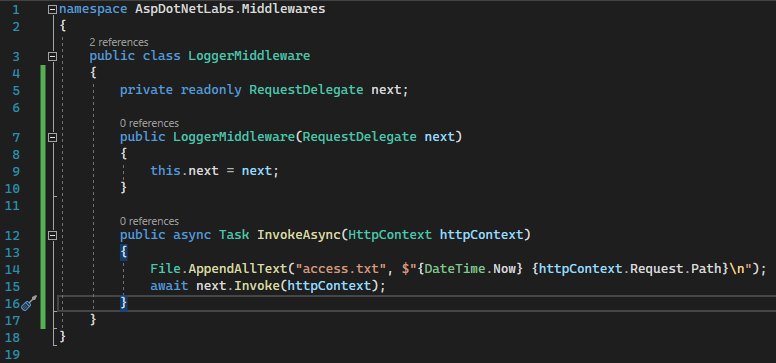


Рис. 11. Код класу LoggerMiddleware

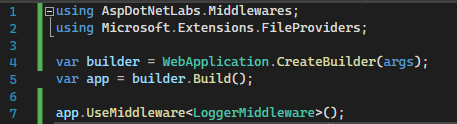


Рис. 12. Зареєстровано LoggerMiddleware

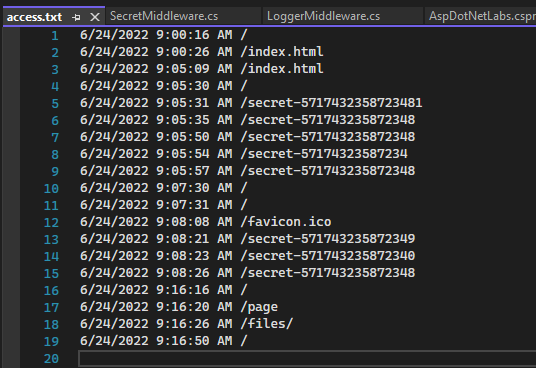


Рис. 13. Результат виконання LoggerMiddleware у файлі access.txt

6.2. Створіть та підключіть у проекті власний компонент з назвою SecretMiddleware, який буде відловлювати декілька маршрутів, наприклад “/secret-571743235872348” і виводити певне повідомлення.

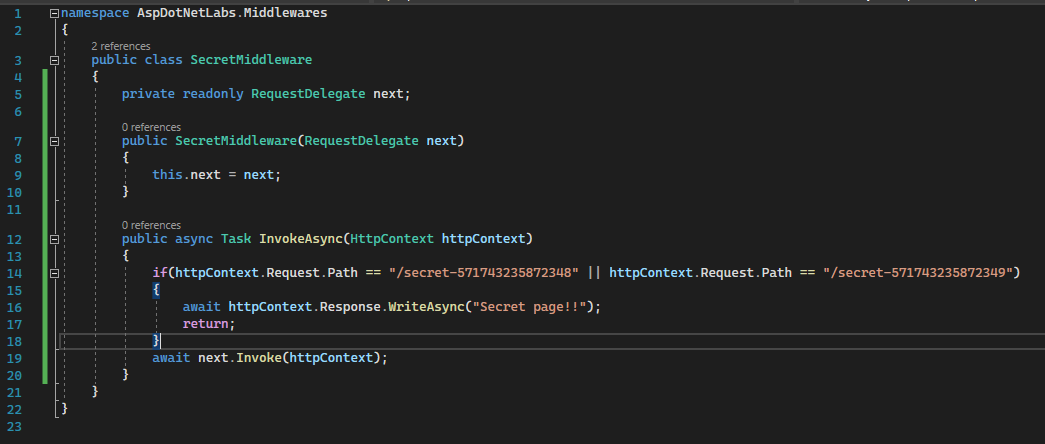


Рис. 13. Код класу SecretMiddleware

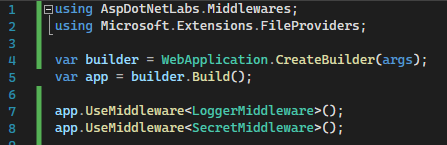


Рис. 14. Зареєстровано SecretMiddleware

**Висновок:** на цій лабораторній роботі ми ознайомилися зі структурою ASP.NET Code додатків, навчилися створювати найпростіші та використовувати наявні компоненти middleware у ASP.NET Core.