



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

**Факультет Информационных технологий**  
**Кафедра Информатики и информационных технологий**

направление подготовки

**09.03.02 «Информационные системы и технологии»**

**ЛАБОРАТОРНАЯ РАБОТА № 5**

**Дисциплина: «Backend»**

**Тема:** *Добавление в веб-приложение на основе ASP.NET Core статических файлов*

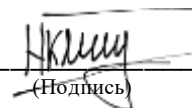
**Выполнил: студент группы: 231-339**

Карапетян Нвер Каренович

(Фамилия И.О.)

**Дата, подпись:** 05.05.25

(Дата)

  
(Подпись)

**Проверил:** \_\_\_\_\_

(Фамилия И.О., степень, звание)

(Оценка)

**Дата, подпись** \_\_\_\_\_

(Дата)

(Подпись)

**Москва**  
**2025**

## Цель:

Ознакомиться с процессом добавления и обслуживания статических файлов в веб-приложениях на платформе ASP.NET Core.

## Задачи:

- Добавить статические файлы (например, изображения, таблицы стилей, скрипты) в проект веб-приложения на ASP.NET Core.
- Настроить маршрутизацию для обслуживания статических файлов из соответствующих каталогов.
- Использовать различные подкаталоги и вложенные структуры для организации статических файлов.
- Протестировать работу приложения, обеспечив корректную доставку статических ресурсов.

## Ход работы

**Статические файлы** — это любые ресурсы веб-приложения, которые не формируются динамически на сервере, а отдаются «как есть» клиенту. К ним относятся: **изображения** (PNG, JPEG, SVG и др.), **таблицы стилей** (CSS), **скрипты** (JavaScript, TypeScript в скомпилированном виде), **шрифты** (WOFF, TTF), **HTML-страницы**, JSON-файлы, файлы конфигурации и пр.

В реальных проектах статические файлы используются для оформления (CSS-фреймворки, собственные стили), клиентской логики (JavaScript/TypeScript-библиотеки, фреймворки), хранения медиа-контента (картинки, видео) и дополнительных данных (например, локальных JSON-справочников).

## Настройка и организация статических файлов

В исходном WebAPI-проекте не было папки для статических файлов, поэтому в корне приложения вручную создана папка **wwwroot**. Внутри нее организована структура:

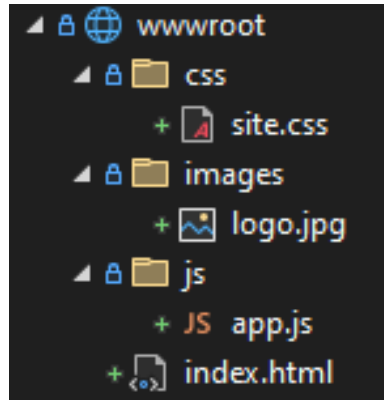


Рисунок 1. Структура папки **wwwroot**.

Каждый файл выполняет следующие роли: **site.css** задает стили страницы (в данном примере фон страницы и цвет заголовка); **app.js** выводит сообщение в консоль и навешивает обработчик на кнопку; **logo.png** — любое изображение для демонстрации; **index.html** подключает все ресурсы и содержит простую разметку с заголовком, картинкой и кнопкой.

## Настройки маршрутизации статических файлов

В файле **Program.cs** прямо после создания **app** добавлены два **middleware**:

Листинг 1. Подключение **middleware** для работы со статическими файлами.

```
app.UseDefaultFiles(); // 1) Раздача файла index.html при обращении к корню '/'
app.UseStaticFiles();  // 2) Раздача всех содержимых в wwwroot
```

Метод **UseDefaultFiles()** позволяет отдавать **index.html** без явного указания имени, а **UseStaticFiles()** обслуживает любые запросы к ресурсам внутри **wwwroot**.

Таким образом запросы к **/css/site.css**, **/js/app.js**, **/images/logo.png** обрабатываются без участия контроллеров.

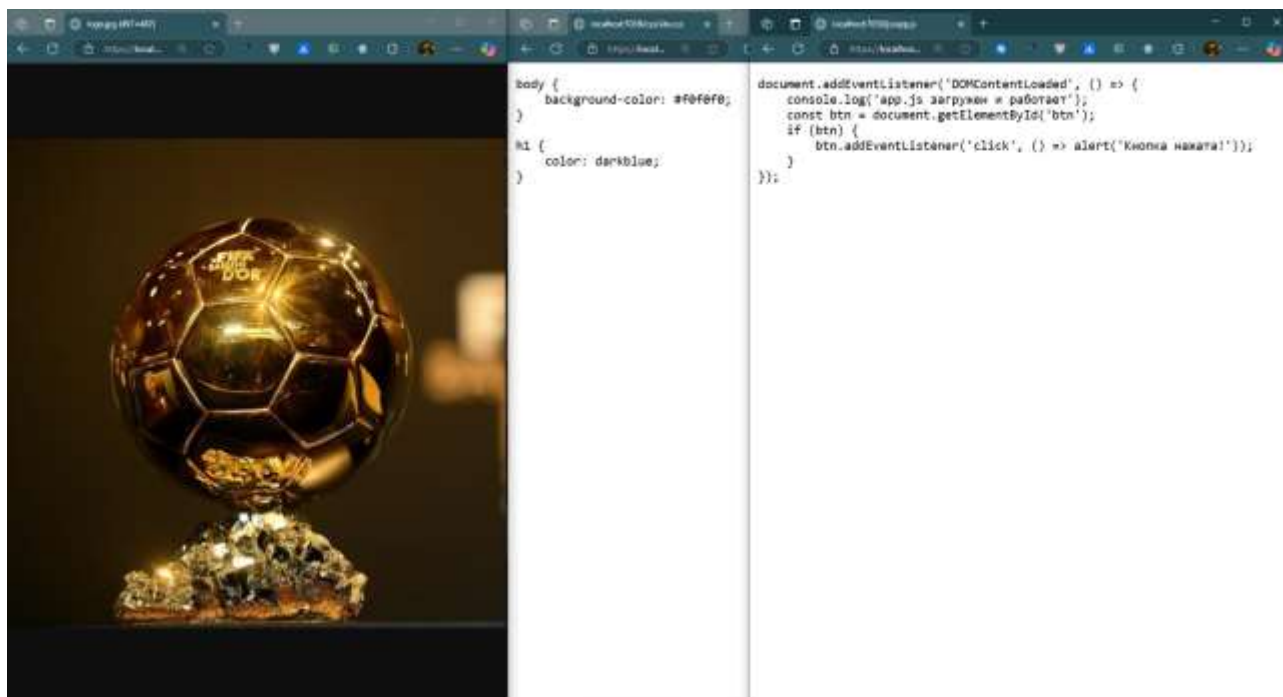


Рисунок 2. В браузере по соответствующим адресам можно открыть статические файлы.

А при переходе на <https://localhost:7039/> загружается **index.html**, применяются стили из `/css/site.css`, скрипт `/js/app.js` выводит сообщение в консоль при нажатии на кнопку, а картинка из `/images/logo.png` отображается:

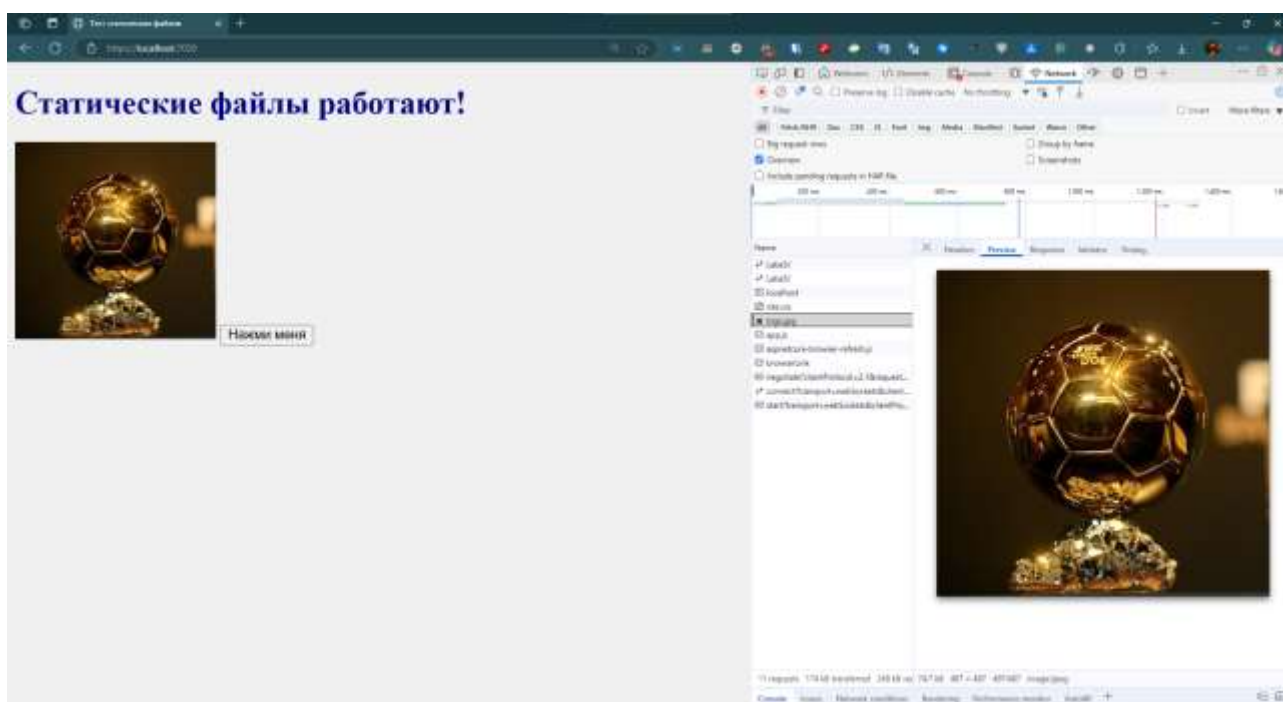


Рисунок 3. Содержимое `index.html` отображается корректно.

Во вкладке Network DevTools при перезагрузке страницы видно запросы к `/css/site.css`, `/js/app.js` и `/images/logo.png` со статусом 200 и правильными MIME-типами (`text/css`, `application/javascript`, `image/png`).

Листинг 2. Содержимое `index.html`.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Тест статических файлов</title>
  <link rel="stylesheet" href="/css/site.css" />
</head>
<body>
  <h1>Статические файлы работают!</h1>
  
  <button id="btn">Нажми меня</button>
  <script src="/js/app.js"></script>
</body>
</html>
```

Если требуется раздавать файлы вне **wwwroot**, можно подключить еще один каталог. В `Program.cs` после `UseStaticFiles()` добавить:

Листинг 3. Подключение middleware для работы со статическими файлами из папки **Content**.

```
app.UseStaticFiles(new StaticFileOptions
{
    FileProvider = new PhysicalFileProvider(Path.Combine(builder.Environment.ContentRootPath, "Content")),
    RequestPath = "/content"
});
```

После этого любой файл, помещенный в папку **Content**, станет доступен по пути `/content/{имя_файла}`, например `/content/book.pdf`:

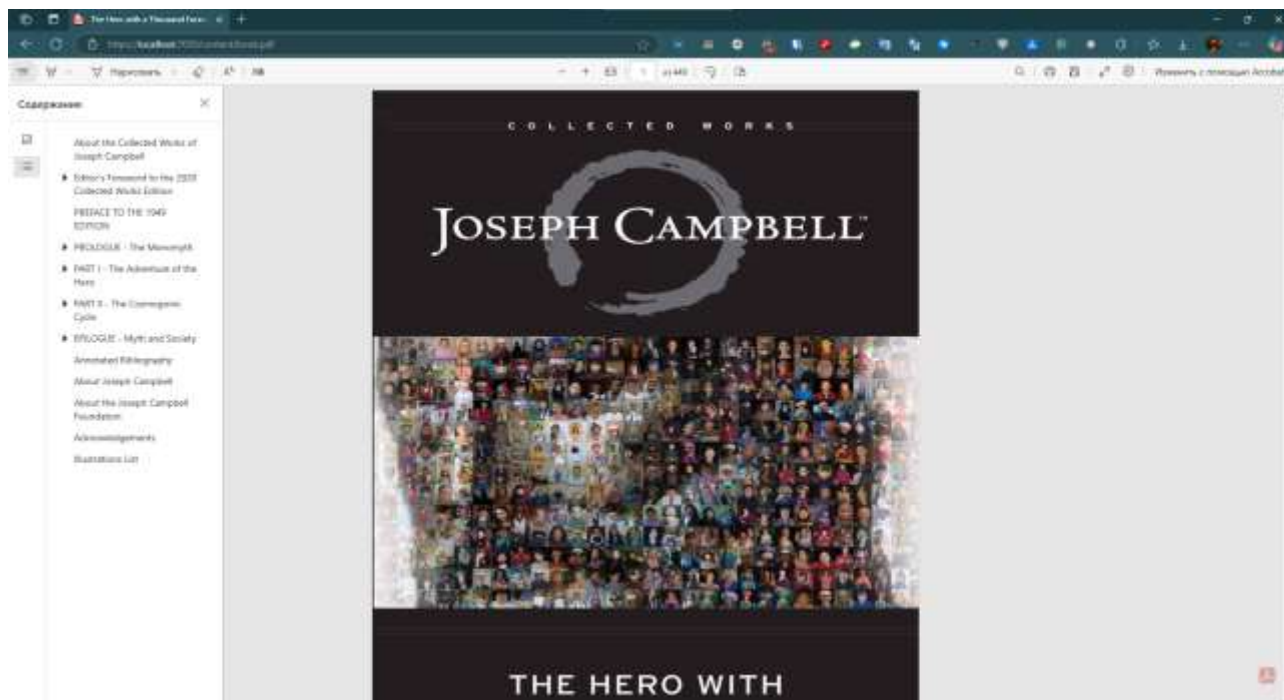


Рисунок 4. Успешно открытый статический PDF-файл из папки Content.