**Статические файлы**

**Установка каталога статических файлов. UseStaticFiles**

Ранее уже рассматривалась отправка статических файлов. В частности, для отправки файлов мы могли использовать метод SendFileAsync() объекта HttpResponse:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | var builder = WebApplication.CreateBuilder();  var app = builder.Build();    app.Run(async (context) => await context.Response.SendFileAsync("index.html"));    app.Run(); |

Однако ASP.NET Core также предоставляет специальный встроенный middleware, который подключается с помощью метода **UseStaticFiles()** и который упрощает работу со статическими файлами.

При использовании этого middleware применяются некоторые условности. В частности, по умолчанию для определения пути хранения статических файлов в проекте используются два параметра **ContentRoot** и **WebRoot**, а сами статические файлы должны помещаться в каталог ContentRoot/WebRoot. На стадии разработки параметр "ContentRoot" соответствует каталогу текущего проекта. А параметр "WebRoot" по умолчанию представляет папку **wwwroot** в рамках каталога ContentRoot. То есть, исходя из значений по умолчанию, то статические файлы следует располагать в папке "wwwroot", которая должна находиться в текущем проекте. Однако эти параметры при необходимости можно переопределить.

В разных типах проектов ASP NET Core данная папка может уже быть по умолчанию в проекте, а может отсутствовать. Например, в проекте по типу Empty данная папка отсутствует, поэтому ее надо добавлять вручную.

Итак, возьмем проект по типу Empty и добавим в него новую папку **wwwroot**. Далее добавим в папку wwwroot новый файл **index.html**. То есть у нас получится следующий проект:

Изменим код файла index.html, например, следующим образом:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | <!DOCTYPE html>  <html>  <head>      <meta charset="utf-8" />      <title>METANIT.COM</title>  </head>  <body>      <h1>Index Page</h1>  </body>  </html> |

Для того, чтобы клиенты могли обращаться к этому файлу, подключим соответствующий компонент middleware с помощью метода UseStaticFiles():

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | var builder = WebApplication.CreateBuilder();  var app = builder.Build();    app.UseStaticFiles();   // добавляем поддержку статических файлов    app.Run(async (context) => await context.Response.WriteAsync("Hello World"));    app.Run(); |

Для подключения функциональности работы со статическими файлами применяется метод **UseStaticFiles()**, который реализован как метод расширения для типа IApplicationBuilder.

Теперь, если мы обратимся к добавленному файлу, например, по пути /index.html, то нам отобразится содержимое данной веб-страницы:

По всем остальным запросам браузер выводил бы строку "Hello World".

Если бы index.html находился бы в какой-то вложенной папке, например, в *wwwroot/html/*, то для обращения к нему мы могли бы использовать путь */html/index.html*. То есть middleware для работы со статическими сайтами автоматически сопоставляет запросы с путями к статическим файлам в рамках папки wwwroot.

**Изменение пути к статическим файлам**

Что делать, если нас не устраивает стандартная папка wwwroot. И мы, к примеру, хотим, чтобы все статические файлы в проекте находились в папке **static**. Для этого добавим в проект папку **static** в проект и определим в ней какой-нибудь html-файл. Пусть он будет называться **content.html**:

Допустим, в этом файле будет какое-нибудь простейшее содержимое:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | <!DOCTYPE html>  <html>  <head>      <meta charset="utf-8" />      <title>METANIT.COM</title>  </head>  <body>      <h1>Content from Static folder</h1>  </body>  </html> |

Чтобы приложение восприняло эту папку, изменим код создания хоста в файле Program.cs:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | var builder = WebApplication.CreateBuilder(      new WebApplicationOptions { WebRootPath = "static"});  // изменяем папку для хранения статики    var app = builder.Build();    app.UseStaticFiles();   // добавляем поддержку статических файлов    app.Run(async (context) => await context.Response.WriteAsync("Hello World"));    app.Run(); |

Для добавления пути к файлам используется перегруженная версия метода **CreateBuilder()**, которая в качестве параметра принимает объект **WebApplicationOptions**. Его свойство **WebRootPath** позволяет установить папку для статических файлов.

И после этого мы также сможем обращаться к статическим файлам из папки **static**:

**Работа со статическими файлами**

Рассмотрим некоторые возможности, которые предоставляет нам ASP.NET Core для работы со статическими файлами.

**Файлы по умолчанию**

Допустим, в проекте в папке **wwwroot** располагается файл **index.html**:

С помощью специального метода расширения **UseDefaultFiles()** можно настроить отправку статических веб-страниц по умолчанию без обращения к ним по полному пути:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | var builder = WebApplication.CreateBuilder();  var app = builder.Build();    app.UseDefaultFiles();  // поддержка страниц html по умолчанию  app.UseStaticFiles();    app.Run(async (context) => await context.Response.WriteAsync("Hello World"));    app.Run(); |

В этом случае при отправке запроса к корню веб-приложения типа *http://localhost:xxxx/* приложение будет искать в папке wwwroot следующие файлы:

* default.htm
* default.html
* index.htm
* index.html

Если файл будет найден, то он будет отправлен в ответ клиенту.

Если же файл не будет найден, то продолжается обычная обработка запроса с помощью следующих компонентов middleware. То есть фактически это будет аналогично, как будто мы обращаемся к файлу: *http://localhost/index.html*

Если же мы хотим использовать файл, название которого отличается от вышеперечисленных, то нам надо в этом случае применить объект **DefaultFilesOptions**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | var builder = WebApplication.CreateBuilder();  var app = builder.Build();    DefaultFilesOptions options = new DefaultFilesOptions();  options.DefaultFileNames.Clear(); // удаляем имена файлов по умолчанию  options.DefaultFileNames.Add("hello.html"); // добавляем новое имя файла  app.UseDefaultFiles(options); // установка параметров    app.UseStaticFiles();    app.Run(async (context) => await context.Response.WriteAsync("Hello World"));    app.Run(); |

В этом случае в качестве страницы по умолчанию будет использоваться файл **hello.html**, который должен располагаться в папке wwwroot.

**Метод UseDirectoryBrowser**

Метод **UseDirectoryBrowser** позволяет пользователям просматривать содержимое каталогов на сайте:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var builder = WebApplication.CreateBuilder();  var app = builder.Build();    app.UseDirectoryBrowser();  app.UseStaticFiles();    app.Run(); |

Данный метод имеет перегрузку, которая позволяет сопоставить определенный каталог на жестком диске или в проекте с некоторой строкой запроса и тем самым потом отобразить содержимое этого каталога:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | using Microsoft.Extensions.FileProviders;  var builder = WebApplication.CreateBuilder();  var app = builder.Build();    app.UseDirectoryBrowser(new DirectoryBrowserOptions()  {      FileProvider = new PhysicalFileProvider(Path.Combine(Directory.GetCurrentDirectory(), @"wwwroot\html")),        RequestPath = new PathString("/pages")  });  app.UseStaticFiles();    app.Run(); |

Чтобы задействовать новый функционал, надо подключить пространство имен using Microsoft.Extensions.FileProviders.

В качестве параметра метод UseDirectoryBrowser() принимает объект **DirectoryBrowserOptions**, который позволяет настроить сопоставление путей к файлам с каталогами. Так, в данном случае путь типа http://localhost:xxxx/pages/ будет сопоставляться с каталогом "wwwroot\html".

**Сопоставление каталогов с путями**

Перегрузка метода UseStaticFiles() позволяет сопоставить пути с определенными каталогами:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | using Microsoft.Extensions.FileProviders;    var builder = WebApplication.CreateBuilder();  var app = builder.Build();      app.UseStaticFiles();  app.UseStaticFiles(new StaticFileOptions() // обрабатывает запросы к каталогу wwwroot/html  {      FileProvider = new PhysicalFileProvider(              Path.Combine(Directory.GetCurrentDirectory(), @"wwwroot\html")),      RequestPath = new PathString("/pages")  });    app.Run(); |

Первый вызов app.UseStaticFiles() обрабатывает запросы к файлам в папке wwwroot. Второй вызов принимает те же параметры, что и метод app.UseDirectoryBrowser() в предыдущем примере. И в отличие от первого вызова он обрабатывает запросы по пути http://localhost:xxxx/pages, сопоставляя данные запросы с папкой **wwwroot/html**. К примеру, по запросу http://localhost:xxxx/pages/index.html мы можем обратиться к файлу **wwwroot/html/index.html**.

**Метод UseFileServer**

Метод **UseFileServer()** объединяет функциональность сразу всех трех вышеописанных методов UseStaticFiles, UseDefaultFiles и UseDirectoryBrowser:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | var builder = WebApplication.CreateBuilder();  var app = builder.Build();    app.UseFileServer();    app.Run(); |

По умолчанию этот метод позволяет обрабатывать статические файлы и отправлять файлы по умолчанию типа index.html. Если нам надо еще включить просмотр каталогов, то мы можем использовать перегрузку данного метода:

|  |  |
| --- | --- |
| 1 | app.UseFileServer(enableDirectoryBrowsing: true); |

Еще одна перегрузка метода позволяет более точно задать параметры:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | var builder = WebApplication.CreateBuilder();  var app = builder.Build();    app.UseFileServer(new FileServerOptions  {      EnableDirectoryBrowsing = true,      EnableDefaultFiles = false  });    app.Run(); |

Также можно настроить сопоставление путей запроса с каталогами:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | using Microsoft.Extensions.FileProviders;    var builder = WebApplication.CreateBuilder();  var app = builder.Build();    app.UseFileServer(new FileServerOptions  {      EnableDirectoryBrowsing = true,      FileProvider = new PhysicalFileProvider(Path.Combine(Directory.GetCurrentDirectory(), @"wwwroot\html")),      RequestPath = new PathString("/pages"),      EnableDefaultFiles = false  });    app.Run(); |

В этом случае будет разрешен обзор каталога по пути http://localhost:xxxx/pages/, но при этом путь http://localhost:xxxx/html/ работать не будет.