ASP.NET Core - Serving Static Files

Here, we will learn how to serve static files such as html, JavaScript, CSS, or image files on HTTP request without any server-side processing.

ASP.NET Core application cannot serve static files by default. We must include Microsoft.AspNetCore.StaticFiles middleware in the request pipeline.

Install StaticFiles Middleware

The **Microsoft.AspNetCore.StaticFiles** middleware package is already included in the meta package Microsoft.AspNetCore.All, so we don't need to install it separately in ASP.NET Core 2.x application.

To install StaticFiles middleware in ASP.NET Core 1.x application, open NuGet package manager by right clicking on project in the solution explorer and select **Manage NuGet Packages..**. Search for staticfiles in the search box in the browse tab. This will display **Microsoft.AspNetCore.StaticFiles** middleware as shown below.

[A screenshot of a computer

Description automatically generated](https://www.tutorialsteacher.com/Content/images/core/install-staticfiles-middleware.png)Install StaticFiles Middleware

Click on the **Install** button on the right pane to install it. Once installed, the Microsoft.AspNetCore.StaticFiles is automatically included in the dependencies section of the project.json.

[A screenshot of a computer program

Description automatically generated](https://www.tutorialsteacher.com/Content/images/core/staticfiles.png)StaticFiles Dependency in project.json

2.5M

JavaScript Quiz

Using StaticFiles Middleware

By default, all the static files of a web application should be located in the web root folder **wwwroot**. To understand this, let's create a simple default.html in the wwwroot folder with the following content.

[A screenshot of a computer

Description automatically generated](https://www.tutorialsteacher.com/Content/images/core/staticfile-html2.png)Default.html

Now, to serve the above Default.html static file, we must add StaticFiles middleware in the Configure() method of Startup file as shown below.

public class Startup

{

public Startup()

{

}

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

app.UseStaticFiles();

app.Run(async (context) =>

{

await context.Response.WriteAsync("Hello World");

});

}

}

As you can see above, the app.UseStaticFiles() method adds StaticFiles middleware into the request pipeline. The UseStaticFiles is an extension method included in the StaticFiles middleware so that we can easily configure it.

Now, open the browser and send http request *http://localhost:<port>/default.html* which will display default.html as a response as shown below.

[A screenshot of a computer

Description automatically generated](https://www.tutorialsteacher.com/Content/images/core/staticfiles2.png)Serving HTML File

This way we can serve any other file stored in wwwroot folder or sub-folder. For example, consider the following test.js file in the wwwroot folder.

[A screenshot of a computer

Description automatically generated](https://www.tutorialsteacher.com/Content/images/core/staticfile-jscript.png)test.js

Now, we can access this file by sending http://localhost:<port>/test.js request.

[A screenshot of a computer

Description automatically generated](https://www.tutorialsteacher.com/Content/images/core/staticfiles3.png)Serving JS File

Suppose, you want to serve files from the outside of web root folder (wwwroot). For example, you include images in the following Images folder as shown below.

[A screenshot of a computer

Description automatically generated](https://www.tutorialsteacher.com/Content/images/core/staticfiles7.png)Serving Static Files

Now, specify StaticFileOptions parameter in the UseStaticFiles method to serve images from the Images folder as shown below.

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

app.UseStaticFiles(); // For the wwwroot folder

app.UseStaticFiles(new StaticFileOptions()

{

FileProvider = new PhysicalFileProvider(

Path.Combine(Directory.GetCurrentDirectory(), @"Images")),

RequestPath = new PathString("/app-images")

});

}

As you can see, we used FileProvider option to specify Images folder from which static files will be served. The RequestPath option specifies the relative path in the URL which maps to the static folder.

Now, a request to *http://localhost/app-images/MyImage.png* will serve the MyImage.png file.

Set Default File

As we have seen above, default.html or test.js was served on the specific request for it. However, what if we want to serve default html file on the root request?

Currently, when you send *http://localhost:<port>* request, it will be handled by run method and display the following result.

[A screenshot of a computer

Description automatically generated](https://www.tutorialsteacher.com/Content/images/core/staticfiles5.png)

To serve default.html on the root request *http://localhost:<port>*, call UseDefaultFiles() method before UseStaticFiles() in the Configure method as shown below.

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

app.UseDefaultFiles();

app.UseStaticFiles();

app.Run(async (context) =>

{

await context.Response.WriteAsync("Hello World");

});

}

The UseDefaultFiles configures the DefaultFiles middleware which is a part of StaticFiles middleware. This will automatically serve html file named default.html, default.htm, index.html or index.htm on the http request *http://localhost:<port>*. The above example will display default.html file on *http://localhost:<port>* as shown below.

[A screenshot of a computer

Description automatically generated](https://www.tutorialsteacher.com/Content/images/core/staticfiles6.png)Serving Static Files

 Note:

Order of middleware is very important. app.UseDefaultFiles() should be added before app.UseStaticFiles() in the request pipeline.

FileServer

The FileServer middleware combines the functionalities of UseDefaultFiles and UseStaticFiles middlware. So, instead of using both the middlware, just use UseFileServer in the Configure method.

UseFileServer = UseDefaultFiles + UseStaticFiles

Example: UseFileServer

 Copy

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

app.UseFileServer();

app.Run(async (context) =>

{

await context.Response.WriteAsync("Hello World");

});

}

Thus, we can serve static files on http requests.