**Static files in asp.net core**

**We will discuss how to make ASP.NET Core application serve static files such as HTML, Images, CSS and JavaScript files**  
  
  
**Static Files**

* By default, an asp.net core application will not serve static files
* The default directory for static files is wwwroot and this directory must be in the root project folder

Copy and paste an image in **wwwroot**folder. let's assume the name of the file is banner.jpg. To be able to access this file from the browser we use  
  
http://{{serverName}}/banner.jpg   
  
In our case we are running on our local machine so the URL would be the following. The port number may be different on your machine.   
  
http://localhost:49119/banner.jpg   
  
On my machine when I navigate to the above URL, I still see the response produced by the middleware I have registered using the Run() method. I do not see the image banner.jpg.   
  
This is because, at the moment our application request processing pipeline does not have the required middleware that can serve static files. The middleware that we need is UseStaticFiles() middleware.    
  
Modify the code in Configure() method to add UseStaticFiles() middleware to our application's request processing pipeline as shown below.

public void Configure(IApplicationBuilder app, IHostingEnvironment env)  
{  
    if (env.IsDevelopment())  
    {  
        app.UseDeveloperExceptionPage();  
    }   
  
    // Add Static Files Middleware  
    app.UseStaticFiles();   
  
    app.Run(async (context) =>  
    {  
        await context.Response.WriteAsync("Hello World!");  
    });  
}

Instead of having all files types like images, css and JavaScript files flat in the wwwroot folder, it is common to have separate folders for css, images and JavaScript under wwwroot as shown below. Consider the following folder hierarchy.   
  
   
  
To be able to access **image1.jpg** from the browser we use   
http://localhost:49119/images/image1.jpg   
  
**Serving static files outside of wwwroot folder**   
  
By default, UseStaticFiles() middleware only serves the static files that are in wwwroot folder. We can also server static files outside of the wwwroot folder if you want to.   
  
**Serving a default document**   
  
Most web applications have a default document and it is the document that is displayed when a user visits the root URL of your application. For example, you have a file with name default.html and you want to serve it when the user visits the root url of your application i.e http://localhost:49119/   
  
At the moment, on my machine when I navigate to the root URL, I see the response produced by the middleware I have registered using the Run() method. I do not see the content of the default document default.html. To be able to serve default page we have to plug in the UseDefaultFiles() middleware in our application's request processing pipeline.   
  
// Add Default Files Middleware  
app.UseDefaultFiles();  
// Add Static Files Middleware  
app.UseStaticFiles();   
  
**Please Note :**UseDefaultFiles must be called before UseStaticFiles to serve the default file. UseDefaultFiles is a URL rewriter that doesn't actually serve the file. It simply rewrites the URL to the default document which will then be served by the Static Files Middleware. The URL displayed in the address bar still reflects the root URL and not the rewritten URL.   
  
The following are the default files which UseDefaultFiles middleware looks for  
index.htm  
index.html  
default.htm  
default.html   
  
If you want to use another document like **foo.html**for example as your default document, you can do so using the following code.   
  
// Specify foo.html as the default document  
DefaultFilesOptions defaultFilesOptions = new DefaultFilesOptions();  
defaultFilesOptions.DefaultFileNames.Clear();  
defaultFilesOptions.DefaultFileNames.Add("foo.html");  
// Add Default Files Middleware  
app.UseDefaultFiles(defaultFilesOptions);  
// Add Static Files Middleware  
app.UseStaticFiles();   
  
**UseFileServer Middleware**  
  
UseFileServer combines the functionality of UseStaticFiles, UseDefaultFiles and UseDirectoryBrowser middleware. DirectoryBrowser middleware, enables directory browsing and allows users to see files within a specified directory. We could replace UseStaticFiles and UseDefaultFiles middlewares with UseFileServer Middleware.   
  
// Use UseFileServer instead of UseDefaultFiles & UseStaticFiles  
FileServerOptions fileServerOptions = new FileServerOptions();  
fileServerOptions.DefaultFilesOptions.DefaultFileNames.Clear();  
fileServerOptions.DefaultFilesOptions.DefaultFileNames.Add("foo.html");  
app.UseFileServer(fileServerOptions);  
  
The important point to note here is the pattern that we use to add middleware to our application's request processing pipeline. In most cases we add middleware using the extension methods that start with the word **USE**. For example,

* UseDeveloperExceptionPage()
* UseDefaultFiles()
* UseStaticFiles()
* UseFileServer()

If you want to customise these middleware components, we use the respective OPTIONS object. For example notice the respective **OPTIONS**objects we use.

|  |  |
| --- | --- |
| **Middleware** | **Options Object** |
| UseDeveloperExceptionPage | DeveloperExceptionPageOptions |
| UseDefaultFiles | DefaultFilesOptions |
| UseStaticFiles | StaticFileOptions |
| UseFileServer | FileServerOptions |