namespace WebApplication1

{

public class Program

{

public static void Main(string[] args)

{

var builder = WebApplication.CreateBuilder(args);

var app = builder.Build();

app.Run(c => c.Response.WriteAsync("Hello1"));

app.Run(c => c.Response.WriteAsync("Hello2"));

app.MapGet("/", () => "Hello World!");

app.Run();

}

}

}

---------------------------------------------------------------------------------

app.Run(c => c.Response.WriteAsync("Hello1"));

app.Run(c => c.Response.WriteAsync("Hello2"));

---------------------------------------------------------------------------------

app.Use(async(context, next) => {

await context.Response.WriteAsync("Use1");

next(); }

);

---------------------------------------------------------------------------------

app.Map("/user", a => a.Run(c => c.Response.WriteAsync("user1")));

---------------------------------------------------------------------------------

app.Map("/user1",

a =>

{

a.Map("/test1", c => c.Use(async (context, next) =>

{

await context.Response.WriteAsync("test1");

next();

}));

});

---------------------------------------------------------------------------------

app.Map("/newbranch", a => {

a.Map("/branch1", brancha => brancha

.Run(c => c.Response.WriteAsync("Running from the newbranch/branch1 branch!")));

a.Map("/branch2", brancha => brancha

.Run(c => c.Response.WriteAsync("Running from the newbranch/branch2 branch!")));

a.Run(c => c.Response.WriteAsync("Running from the newbranch branch!"));

});

---------------------------------------------------------------------------------

app.Use(async (context, next) =>

{

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

await context.Response.WriteAsync("\nThis is after another Hello, World!");

await context.Response.WriteAsync(builder.Configuration["Message"]);

next();

});

---------------------------------------------------------------------------------

app.UseStaticFiles();

app.UseDefaultFiles();

app.UseStaticFiles();

OR

app.UseFileServer();

OR

DefaultFilesOptions options = new DefaultFilesOptions();

options.DefaultFileNames.Clear();

options.DefaultFileNames.Add("home.html");

app.UseDefaultFiles(options);

app.UseStaticFiles();

app.UseDeveloperExceptionPage();

app.Run((c) => throw new Exception("ERROR"));

Custom Middleware

public class MyMiddleware

{

private readonly RequestDelegate \_next;

private readonly ILogger \_logger;

public MyMiddleware(RequestDelegate next, ILoggerFactory logFactory)

{

\_next = next;

\_logger = logFactory.CreateLogger("MyMiddleware");

}

public async Task Invoke(HttpContext httpContext)

{

\_logger.LogInformation("MyMiddleware executing..");

await \_next(httpContext); // calling next middleware

}

}

// Extension method used to add the middleware to the HTTP request pipeline.

public static class MyMiddlewareExtensions

{

public static IApplicationBuilder UseMyMiddleware(this IApplicationBuilder builder)

{

return builder.UseMiddleware<MyMiddleware>();

}

}

Now, we need to add our custom middleware in the request pipeline by using Use extension method as shown below.

app.UseMyMiddleware();