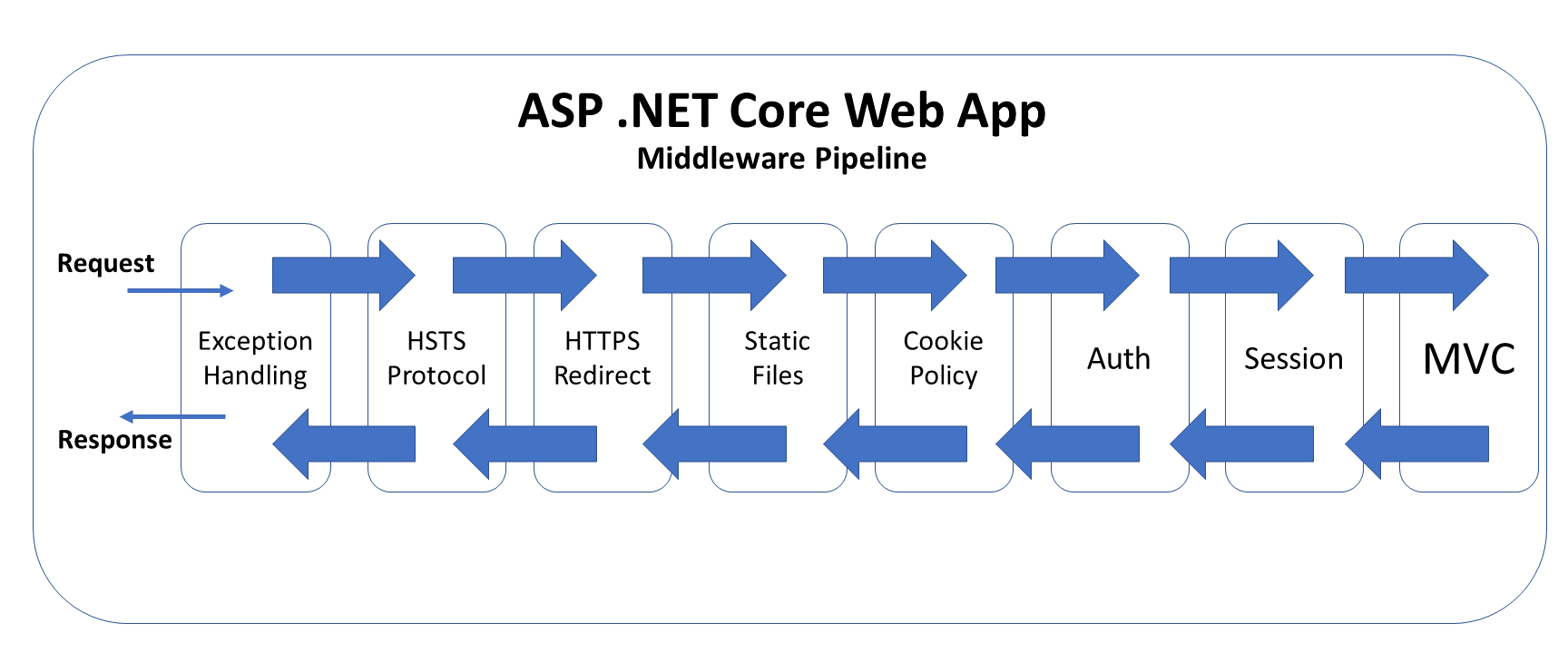
1. Middleware -Handling requests

<https://codingblast.com/asp-net-core-middleware/>

* Is a software that app assembles into the pipeline to handle requests and responses
* Be setup inside of Startup.cs
* Control how the app responds to HTTP requests
* The below diagram illustrates the typical order of middleware layers in an ASP .NET Core web application. The order is very important and it is the order in Configure() method

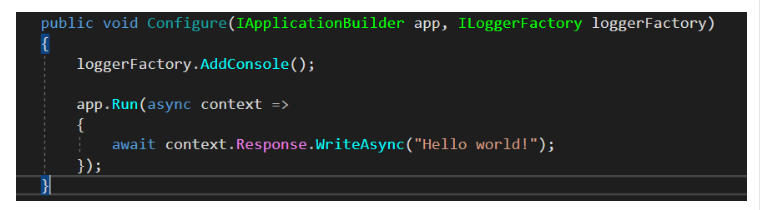


* **Apply cho project ntn????????**

1. **App startup process:**

* Program.cs is
* + where everything gets started then is in Startup.cs
* + Where to define services that app will use them
* Process:

Program.cs/main => Startup.cs/ConfigureServices() => Startup.cs/Configure(), where will use *IApplicationBuilder* to set up middleware



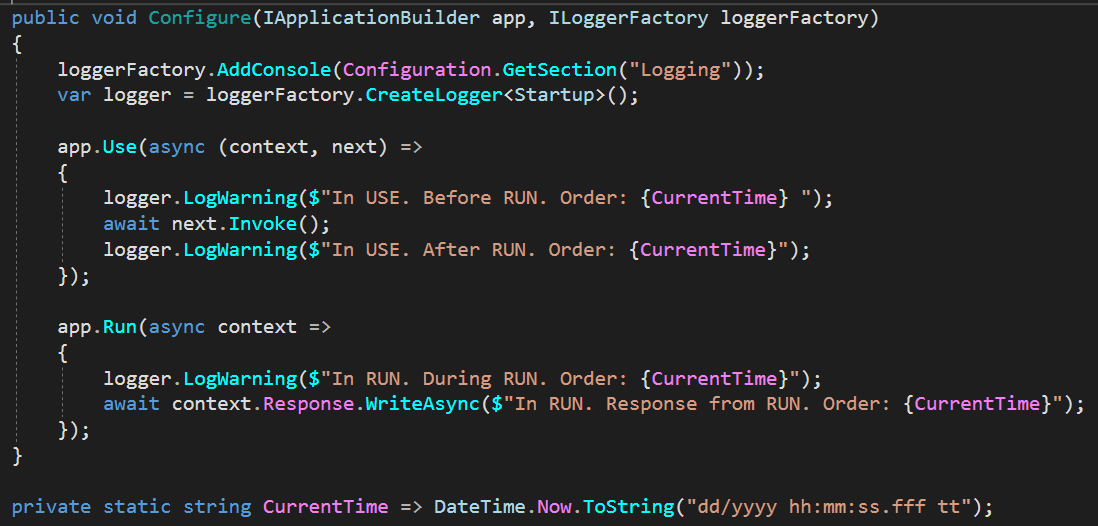
Explain:

* loggerFactory is a middleware in the pipeline.
* It can read all data from the incoming HTTP request(path, query string, headers, cookies, etc…).
* It can log any information about the request.
* Then it passes the request to next piece of middleware

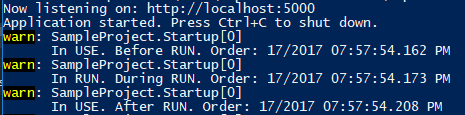
1. **Request delegates CHƯA HIỂU**

* 4 methods available:
* + **Use**: add a middleware/ pass the request to next delegate/ end the request
* **+ Map**: connect a request path with another middleware to use any of the delegated request
* **+ MapWhen**: same as ***Map***except that we can specify the detailed condition by using a HttpContext object
* **+ Run:**

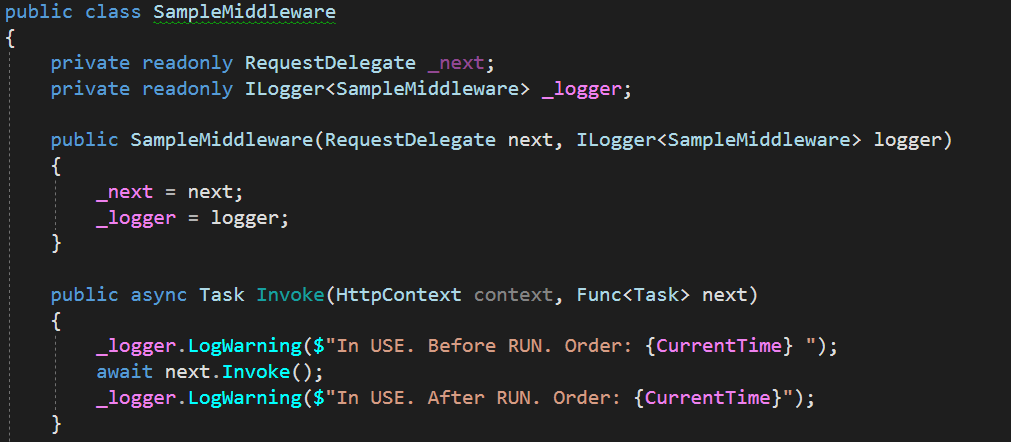
1. **Execution order:**

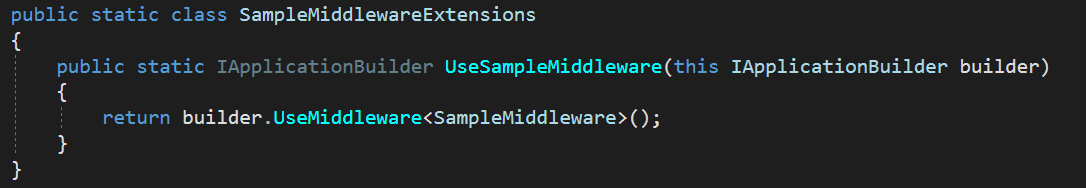


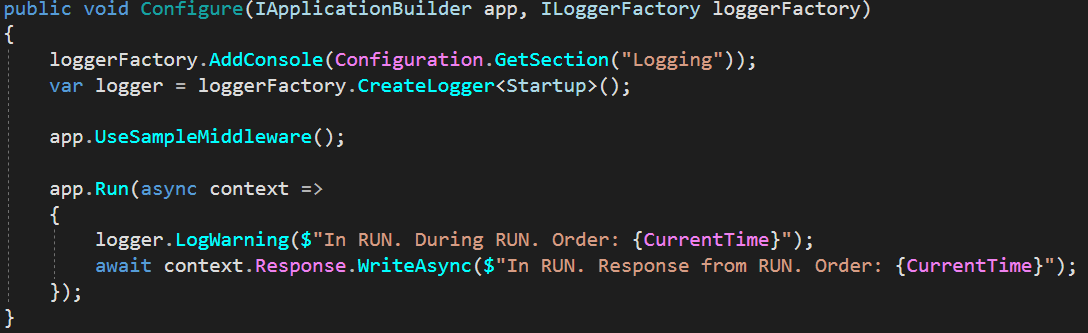
We can see this clearly in the console and in the browser:



1. **Create a simple middleware class**





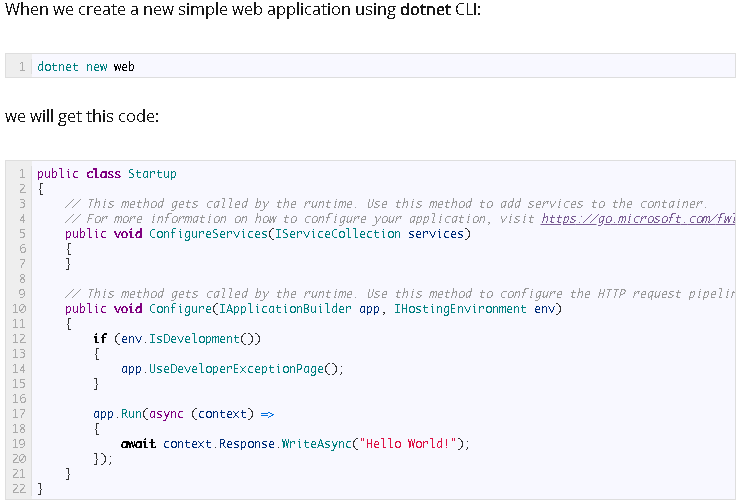


1. Startup.cs, ConfigureServices() vs Configure()

<https://codingblast.com/asp-net-core-configureservices-vs-configure/>

1. **Startup.cs**

* All configuration is done
* Processed once when app starting



1. **ConfigureServices()**

* Option method
* Called by Web host
* adding services to app
* One parameter is *IServiceCollection (container with added services)*
* => those services are available for **dependency injection**
* => can inject those services anywhere in our application



1. **Configure()**

* Set up middleware
* Manage HTTP request pipeline
* Write code that will process every request and make a response



1. dhgfhg