Middleware Components in ASP.NET CORE

What are the ASP.NET Core Middleware Components?

- The **ASP.NET Core Middleware Components** are the software components (C# Classes) that are assembled into the application pipeline to handle the HTTP Requests and Responses. Each middleware component in ASP.NET Core Application performs the following tasks.
  - **Chooses whether to pass the HTTP Request to the next component in the pipeline.**
  - **Can perform work before and after the next component in the pipeline.**
- In ASP.NET Core there are so many built-in Middleware components are already available that you can directly use. If you want then you can also create your own Middleware components.
- The most important point that you need to remember is, in ASP.NET Core a give Middleware component should only have a specific purpose.

Where we use Middleware Components?

- Some of the examples of using Middleware components in the ASP.NET Core application are as follows
  1. We may have a Middleware component for authenticating the user
  2. Another Middleware component may be used to log the request and response
  3. Similarly, we may have a Middleware component that is used to handle the errors
  4. We may have a Middleware component that is used to handle the static files such as images, Javascript or CSS files, etc.
  5. Another Middleware component may be used to Authorize the users while accessing a specific resource
- The Middleware components are the components that we use to set up the request processing pipeline in the ASP.NET Core application. If you have worked with previous versions of ASP.NET then you may know, we use HTTP Handlers and HTTP Modules to set up the request process pipeline. It is this pipeline which will determine how the HTTP request and response is going to be processed.

How to Configure Middleware Components?

- In ASP.NET Core application, you need to configure the Middleware components within the **Configure()** method of the **Startup** class which is present within the **Startup.cs** file.
- This is the class that is going to run when the application starts. When we create an ASP.NET Core application with Empty Template, then by default the **Startup class** is created with the **Configure()** method as shown in the below image.

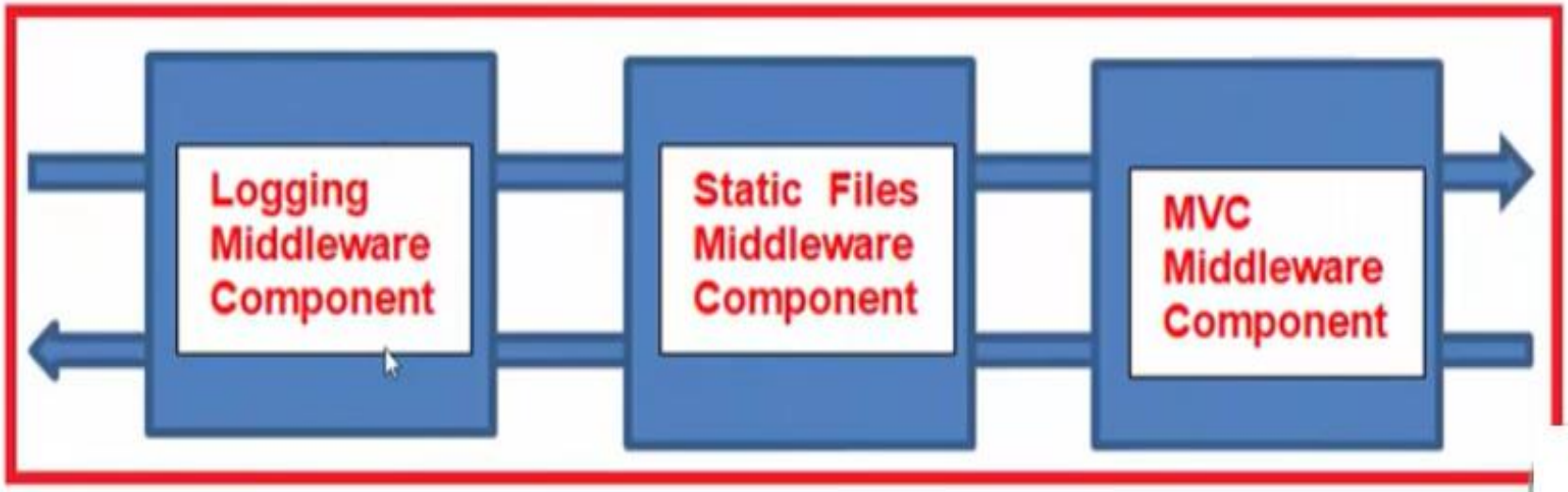
```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.Run(async (context) =>
    {
        await context.Response.WriteAsync("Hello world!");
    });
}
```

How to Configure Middleware Components?

- So, whenever you want to configure any middleware components, then you need to configure it within the **Configure()** method of the **Startup** class by calling the **Use*** methods on the **IApplicationBuilder** object. As you can see from the previous image, the **configuration()** method sets up the request processing pipeline with just two middleware components are as follows
  - **UseDeveloperExceptionPage() Middleware component**
  - **Run() Middleware component**

Understanding the Middleware Components?

- The below diagram explains what the middleware components are and how they used in the request processing pipeline of an ASP.NET Core application.



Understanding the Middleware Components?

- In ASP.NET Core application, the Middleware component can have access to both the incoming **HTTP Request** and outgoing **HTTP Response**.
- So a Middleware component in ASP.NET Core can
  - **Handle the incoming HTTP request by generating an HTTP response.**
  - **Process the incoming HTTP request, modify it, and then pass it to the next middleware component**
  - **Process the outgoing HTTP response, modify it, and then pass it on to either the next middleware component or to the ASP.NET Core web server.**

Example

- As shown in the above image, we have a logging middleware component. This component simply logs the request time and then passes the request to the next middleware component i.e. Static Files Middleware component in the request pipeline for further processing.
- A middleware component in ASP.NET Core may also handle the HTTP Request by generating an HTTP Response. The ASP.NET Core Middleware component may also decide not to call the next middleware component in the request pipeline. This concept is called **short-circuiting the request pipeline**.
- **For example**, we have a static file middleware component. And if the incoming HTTP request comes for some static files such as images, CSS files, etc. then this Static Files Middleware component can handle the request and then short-circuit the request pipeline by not calling to the next component in pipeline i.e. the MVC Middleware component.
- As we already discussed the ASP.NET Core middleware components can have access to both the HTTP request and response in the pipeline. So a middleware component can also process the outgoing response. For example, the logging middleware component in our case may log the time when the response is sent back to the client.

Execution Order?

- It is very important to understand the execution order of Middleware components. The ASP.NET Core middleware components are executed in the same order as they are added to the pipeline.
- You can add more than one component depending on your application needs.
- Example:
  - if you are developing a static web application with some static HTML pages and images, then you may require only **"StaticFiles"** middleware components in the request processing pipeline.
  - But, if you are developing a secure dynamic data-driven web application then you may require several middleware components such as **Logging Middleware, Authentication middleware, Authorization middleware, MVC middleware**, etc.

Thanks