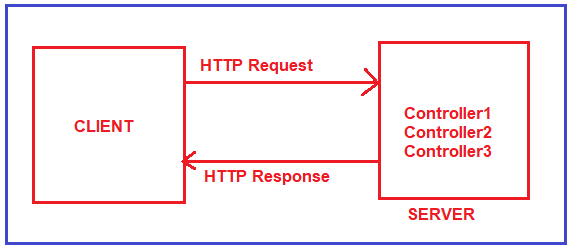
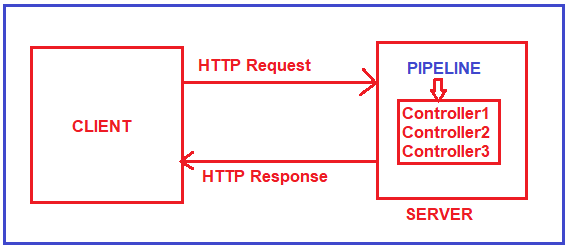
**Middleware in ASP.NET Core Web API**

##### **HTTP Request Pipeline:**

Before understanding ASP.NET Core Middleware components, let us first understand what is HTTP Request Pipeline and how does it work. Please have a look at the following image for a better understanding of the HTTP Request Pipeline. As you can see in the below image, on the left-hand side, we have the client i.e. a browser and on the right-hand side, we have the server where our ASP.NET Core Web API application is hosted. Further, the Web API Application has three controllers. So, when the client sends a request to the server, we generally believe that it is the controller action method that is going to serve the request and then we get the response.



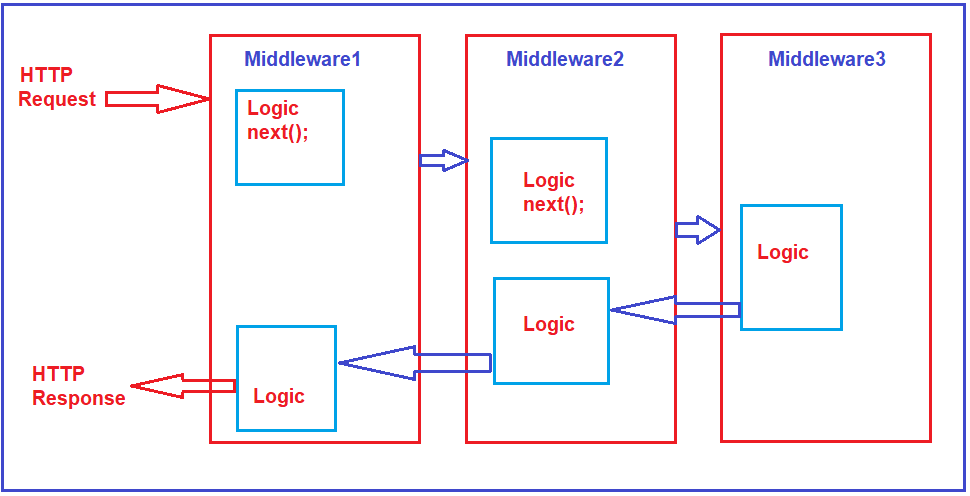
But in reality, before hitting the controller action method, the request has to pass through a pipeline. Once the pipeline is completed, then only it navigates the request to the corresponding controller action method as shown in the below image.



Let us understand what exactly a Pipeline is in detail.

##### **HTTP Request Pipeline in ASP.NET Core Web API Application:**

The Request Pipeline in ASP.NET Core Web API Application can have multiple middlewares as shown in the below image. If you are confusing what is Middleware, let us assume a middleware is a piece of code with some logic. Whenever a request comes from a client to the server, then the request comes to the first middleware which is registered in the request pipeline. In our case it is Middleware1. The code or logic which is there in Middleware1 will be executed and then if it will call the next method, then the request goes to the next middleware which is registered in the request processing pipeline i.e. Middlware2. The code or logic which is there in Middleware2 will be executed and if it calls the next method, then it navigates to the request to the next middleware i.e. Middleware3. Let us assume, in Middleware3, we don’t have the next method. So, the code or logic which is there in Middleware3 will be executed and as there is no next method call, so the request will come back to the previous middleware i.e. Middleware2. And if there is some code after the next method in Middleware2, then those codes will be executed and once the code executed the request again comes back to the previous middleware i.e. Middlwware1. Similarly, if there is some code after the next method in Middleware1, then those codes will be executed and once the code gets executed the final response sends to the client who initially made the request.



This is how the Request Pipeline works in ASP.NET Core Web API Application. These Middleware are nothing but a piece of code or we can say these are some functionalities that we want to insert in our ASP.NET Core Web API Application.

##### **Middleware in ASP.NET Core Web API:**

Middleware is a piece of code that is used in the HTTP Request Pipeline. An ASP.NET Core Web API Application can have n numbers of middleware. So, depending upon the requirement, we can configure n numbers of middleware in the application request processing pipeline.

The order of middleware matters a lot in the execution. That means in the order they are configured into the request processing pipeline; in the same order, they are going to be executed when a request comes. Each middleware in the ASP.NET Core Web API Application performs the following tasks.

1. Chooses whether to pass the HTTP Request to the next component in the pipeline. This can be achieved by calling the next() method within the middleware.
2. Can perform work before and after the next component in the pipeline.

ASP.NET Core provides some built-in middleware that is ready to be used, even if you want then you can also create your own custom middleware. The most important point that you need to keep in mind is, in ASP.NET Core a given Middleware component should only have a specific purpose i.e. single responsibility.

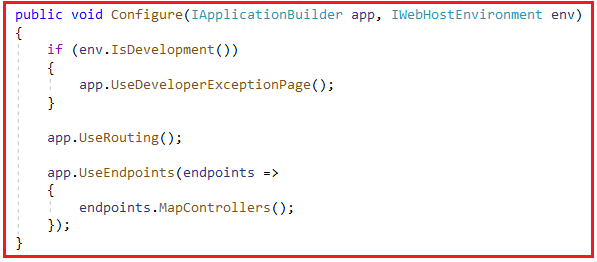
##### **Middleware Examples:**

1. **Routing**: If you want to implement Routing in your application, then you need to use Routing Middleware in the HTTP Request Processing pipeline.
2. **Authentication**: If you want to authenticate the user then you need to use Authentication Middleware.
3. **Authorize**: The Authorize Middleware is used to Authorize the users while accessing a specific resource.
4. **Log**: If you want to log request and response while processing, then you need Middleware.
5. **Exception Middleware:** You can also use Middleware to handle the exception globally.

**Note**: The Middleware in ASP.NET Core Web API Application is used to set up the HTTP Request processing pipeline. If you have prior experience of the previous .NET Framework then you may know, HTTP Handlers and HTTP Modules which are basically used to set up the request processing pipeline. It is this pipeline that will determine how the HTTP request and response are going to be processed.

##### **How to Configure Middleware Components in ASP.NET Core application?**

In the ASP.NET Core Web API application, the Middleware components are configured within the Configure method of the Startup class. The Startup class is the class that is going to run when the application starts. The following is the Configure method of the Startup class that we have created in our previous article. Even though if you created an ASP.NET Core Application with an Empty Project template, then also you will find the following code within the Configure method of the Startup class.



As you can see in the above image, within the Configure method we have configured three Middleware components to the HTTP Request Processing Pipeline. They are as follows.

1. **UseDeveloperExceptionPage() Middleware:**The UseDeveloperExceptionPage() middleware will come into picture only when the hosting environment is set to “development”. The UseDeveloperExceptionPage middleware is going to execute when there is an unhandled exception that occurred in the application and since it is in development mode, it is going to show you the detailed information of the exception.
2. **UseRouting() Middleware:**The UseRouting middleware is used to add Endpoint Routing Middleware to the request processing pipeline i.e. it will map the URL (or incoming HTTP Request) to a particular resource.
3. **UseEndpoints() Middleware:**In this middleware, the routing decisions are going to be taken using the Map extension method.

So, if you want to configure any middleware components in any type of ASP.NET Core applications, then you need to configure it within the Configure method by calling the **Use\*** methods on the IApplicationBuilderobject.

**Note:**The ASP.NET Core Middleware components are executed in the same order as they are added to the pipeline. So, it is our key responsibility to take care when adding middleware components.

##### **Run, Use, Next, and Map Methods in Middleware:**

In order to work with ASP.NET Core Middleware Components, we need to learn about few methods are as follows:

1. **Run() Method:** The [**Run() Extension Method**](https://dotnettutorials.net/lesson/run-method-in-asp-net-core/) is used to complete the Middleware Execution.
2. **Use() Method:** The Use() Extension Method is used to insert a new Middleware component to the Request Processing Pipeline.
3. **Next() Method:** The Next() Extension Method is used to call the next middleware component in the request processing pipeline.
4. **Map() Method:** The Map() Extension Method is used to map the Middleware to a specific URL.