In a .NetCore web api we have:

The program.cs , and the startup.cs

In the program.cs we have

1. Host builder

Host builder is an object and it is used to add some features in the application

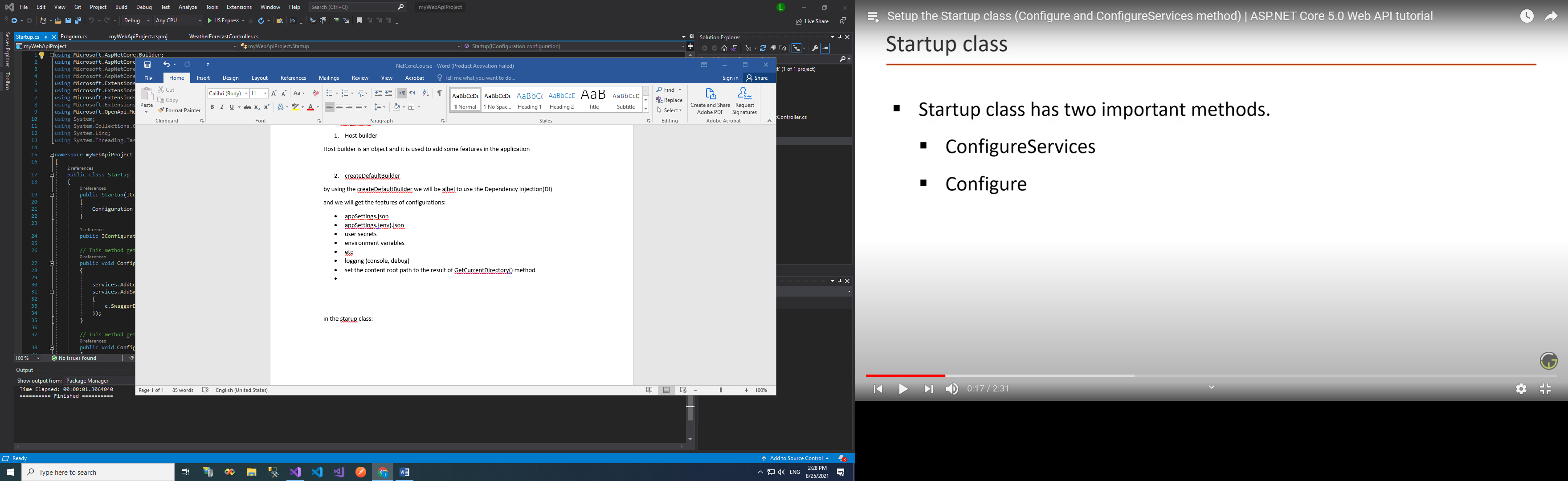
1. createDefaultBuilder

by using the createDefaultBuilder we will be albel to use the Dependency Injection(DI)

and we will get the features of configurations:

* appSettings.json
* appSettings.{env}.json
* user secrets
* environment variables
* etc
* logging (console, debug)
* set the content root path to the result of GetCurrentDirectory() method

in the starup class:



The configureServices is used to configure all the services that we want to use in this particular application, which means where we INJECT the services that we want to use in the app.

And to inject the services we need to call

* the IServiceCollection from the namespace : Microsoft.extension.dependencyInjection
* from the argument of the method ConfigureServices

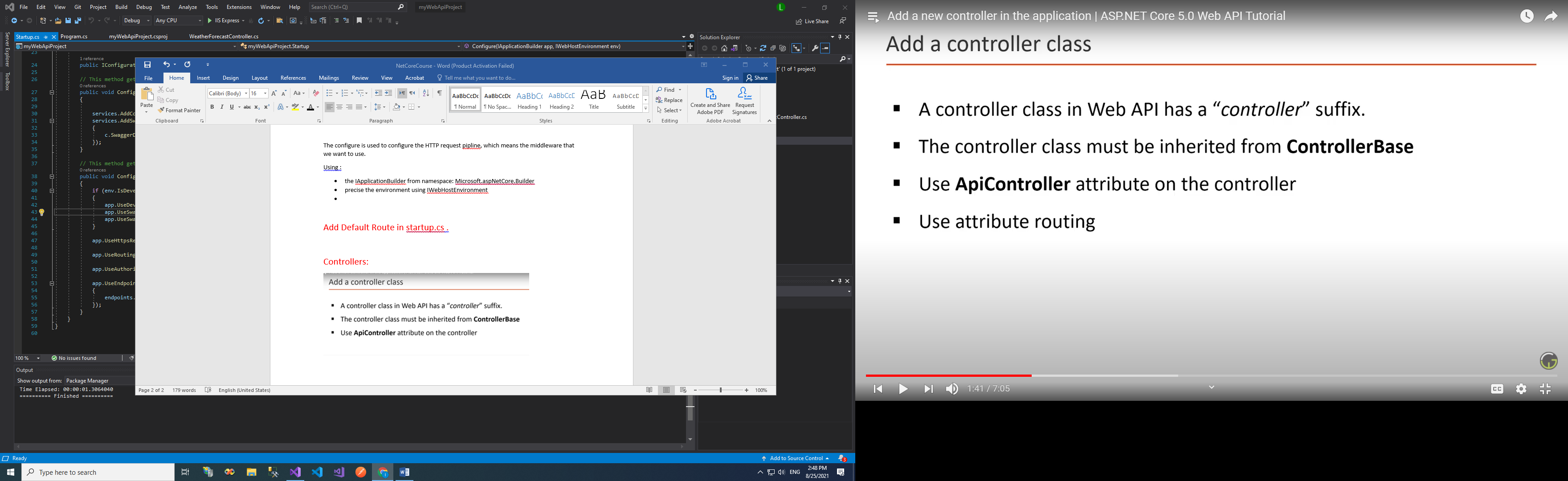
The configure is used to configure the HTTP request pipline, which means the middleware that we want to use.

Using :

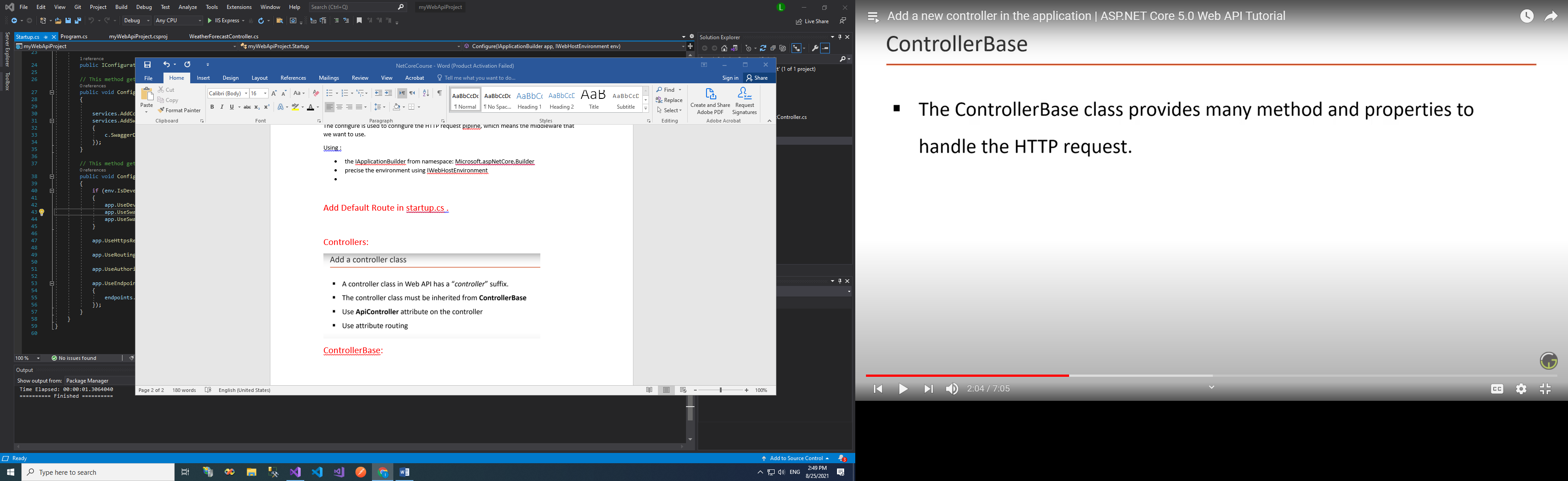
* the IApplicationBuilder from namespace: Microsoft.aspNetCore.Builder
* precise the environment using IWebHostEnvironment

Add Default Route in startup.cs .

Controllers:

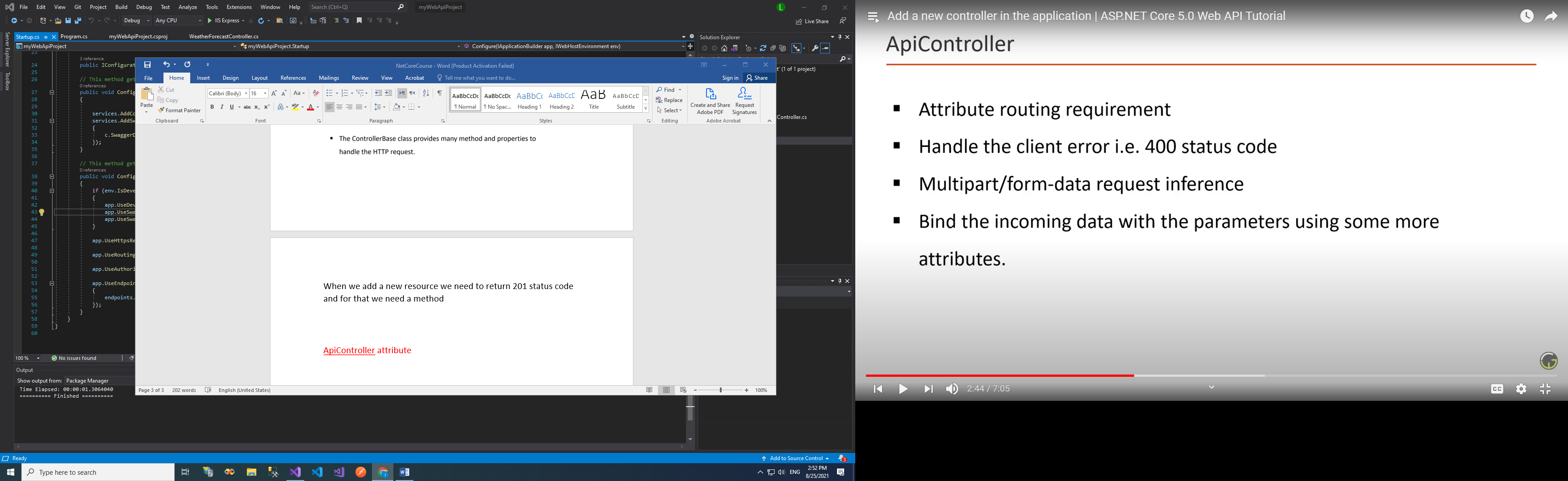


ControllerBase:



When we add a new resource we need to return 201 status code and for that we need a method. This controllerBase class has all these methods.

ApiController attribute



[ApiController]

[Route("test/[action]")]

public class TestController : ControllerBase

{//https://localhost:44381/test

public string Get() =>

"hello world";

public string Get2() =>

"hello world22222";

}

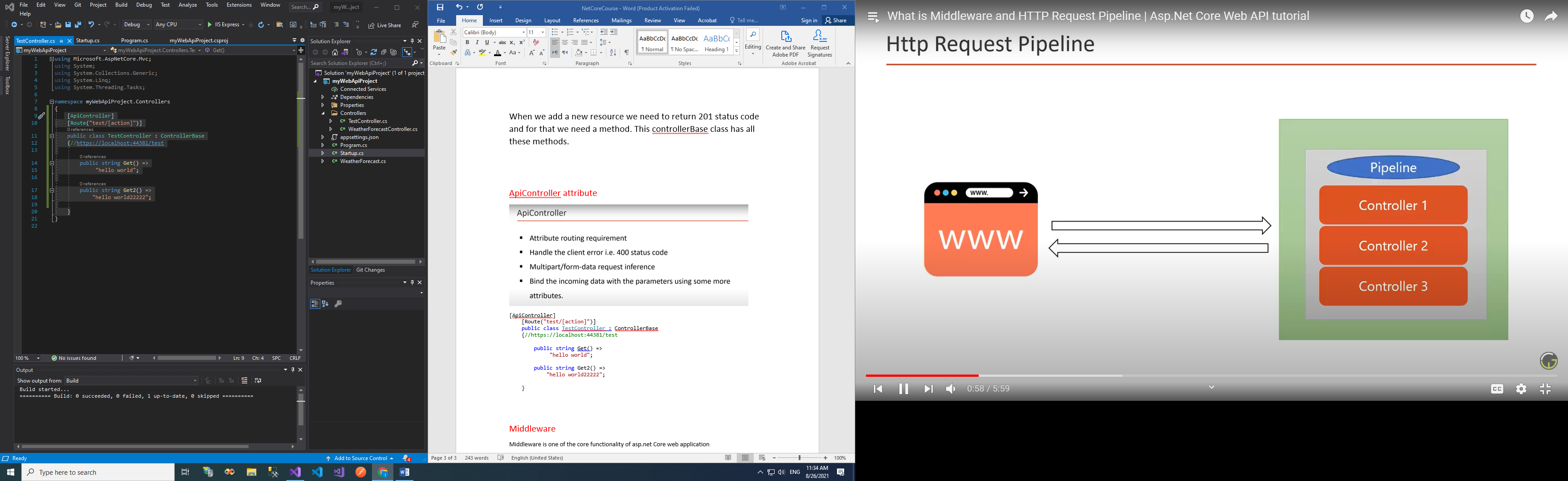
Middleware and Request pipeline:

Middleware is one of the core functionality of asp.net Core web application

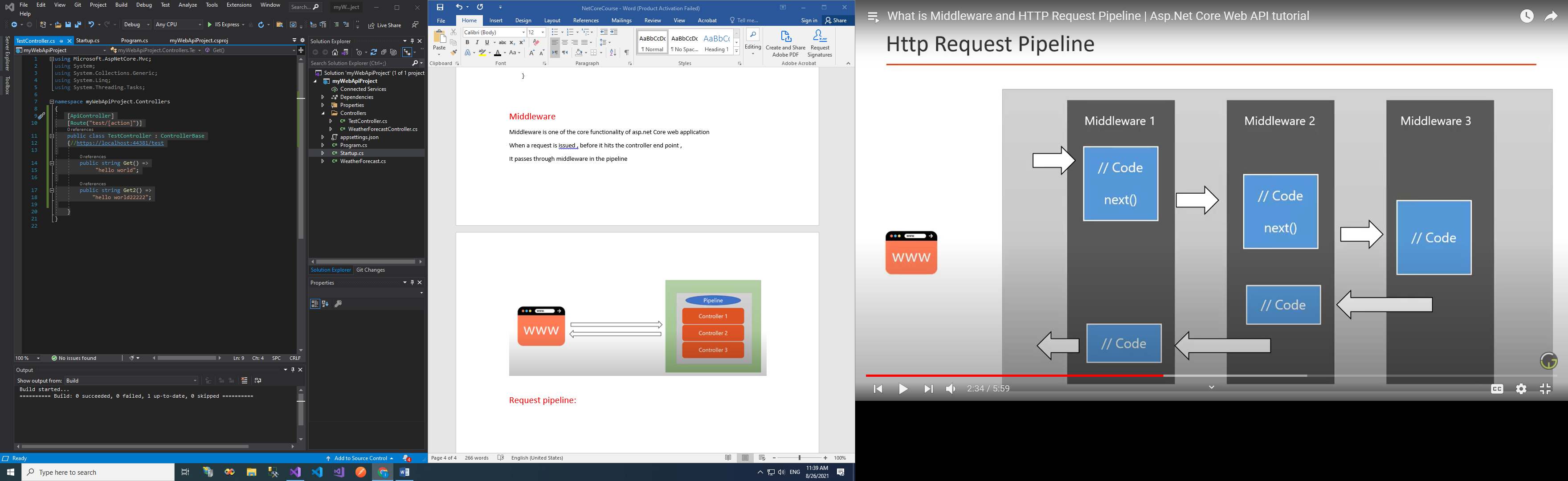
When a request is issued , before it hits the controller end point ,

It passes through middleware in the pipeline.

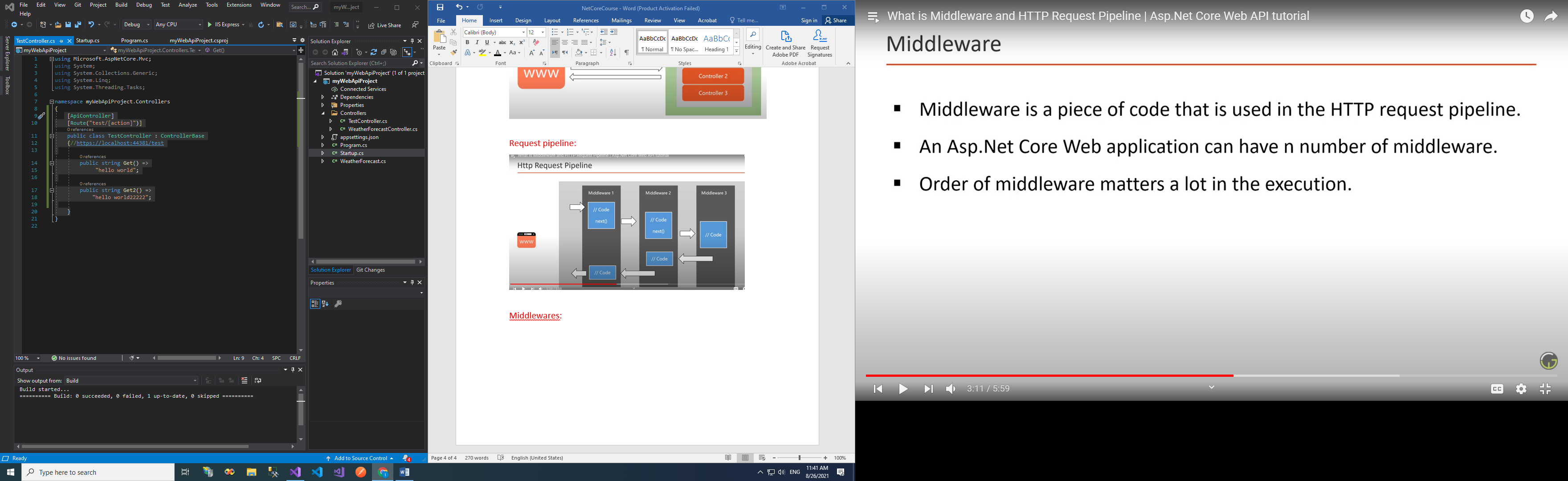
Middlewares are used in the HTTP request pipeline.

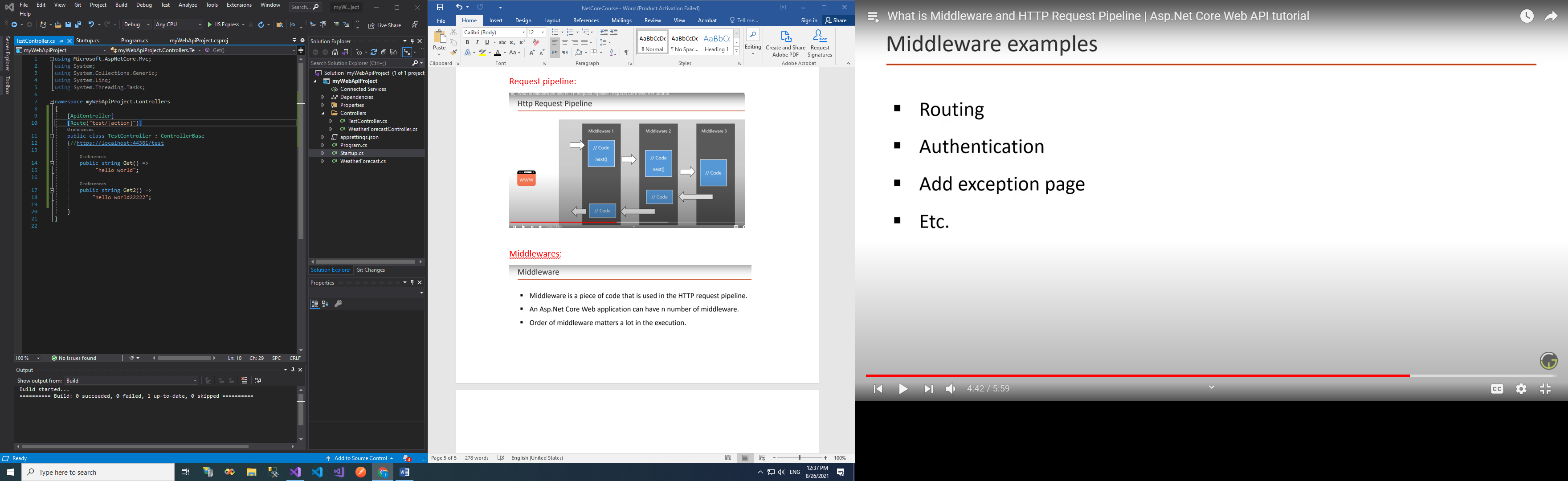


Request pipeline:



Middlewares:





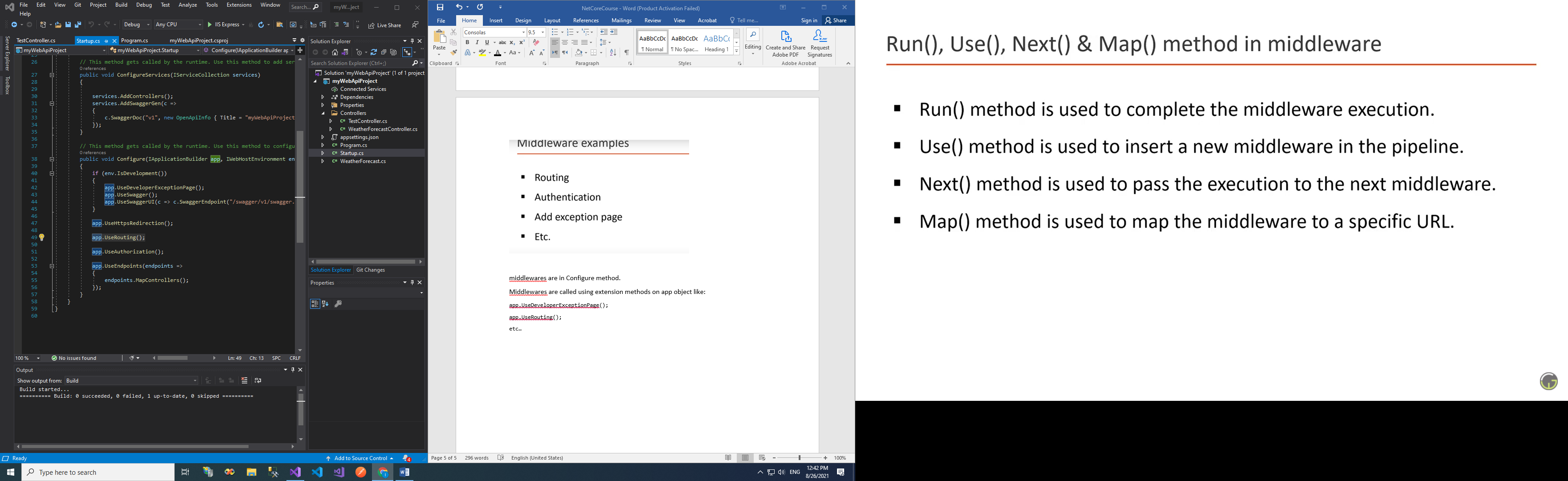
middlewares are in Configure method.

Middlewares are called using extension methods on app object like:

app.UseDeveloperExceptionPage();

app.UseRouting();

etc…



Run()

Adds a terminal middleware delegate to the application's request pipeline.

It complete and terminate the middleware execution, el request men 3anda bterja3 rjou3 3ala el middleware anterieur (li abla), so the request travels back to the previous middleware

Use()

Adds a middleware delegate to the application's request pipeline.

We can use the Use() method without the next() instead of Run() method.

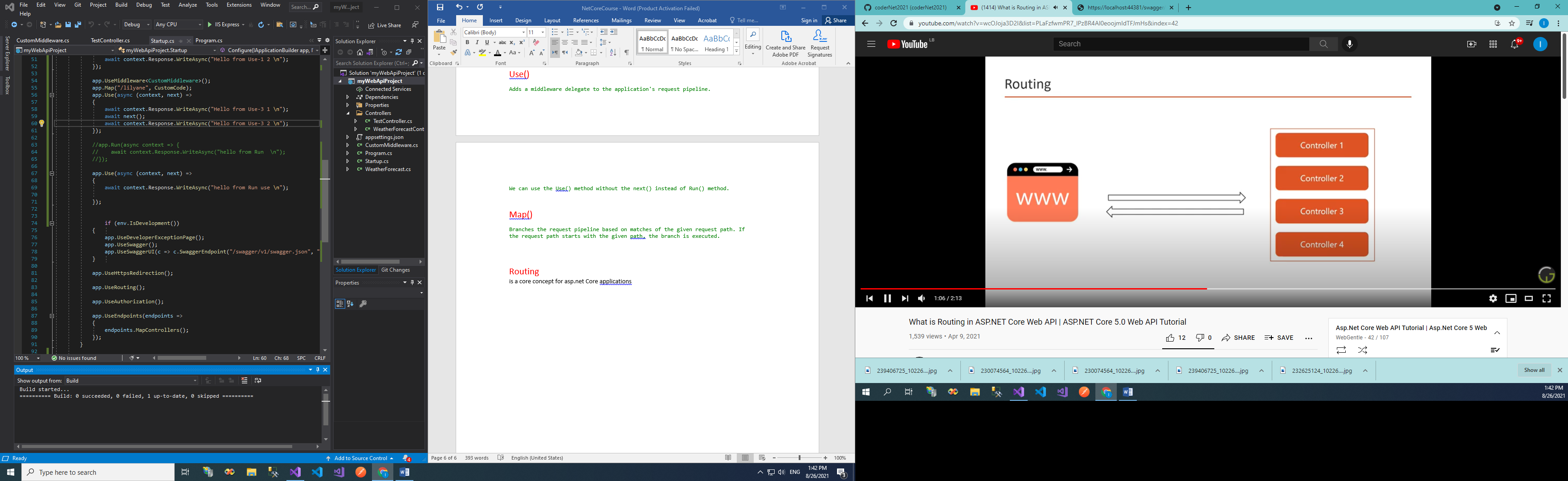
Map()

Branches the request pipeline based on matches of the given request path. If

the request path starts with the given path, the branch is executed.

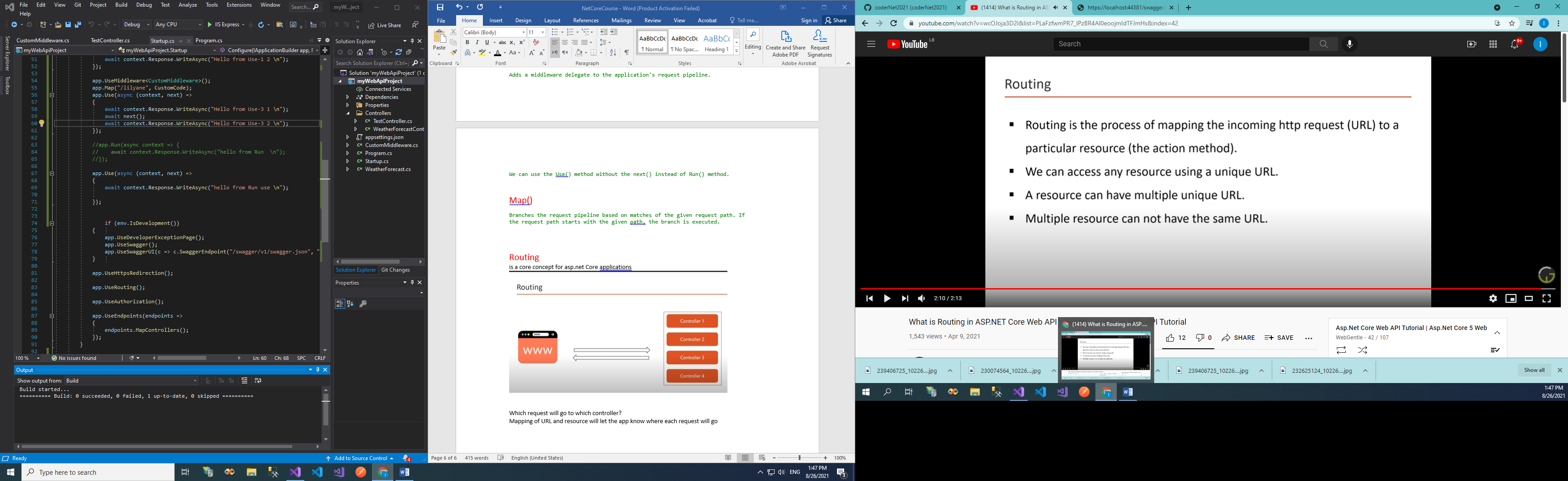
Routing

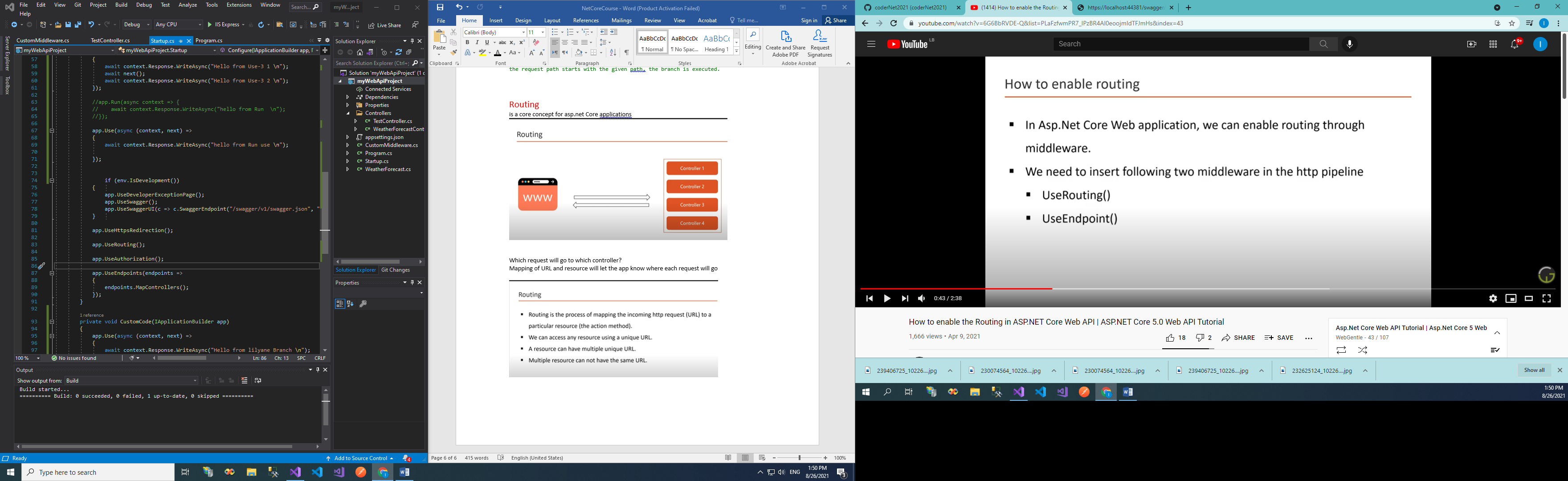
is a core concept for asp.net Core applications



Which request will go to which controller?

Mapping of URL and resource will let the app know where each request will go





2 ways of route mapping:

Conventional mapping

And attribute routing

// [Route("api/[controller]")]

[ApiController]

public class ValuesController : ControllerBase

{

[Route("api/get-all")]

public string GetAll() {

return "Hello from get All";

}

[Route("api/get-all-authors")]

public string GetAllAuthors()

{

return "Hello from get All authors";

}

[Route("books/{id}/author/{authorId}")]

public string GetById(int id ,int authorId)

{

return "Hello : " + id + " author Id = " + authorId;

}

}

Working with Query String in Routing

[Route("search")]

public string SearchBooks(int id, int authorId, string name, int rating, int price)

{

return "Hello : " + id + " author Id = " + authorId+ " name = "+name;

}

Setup multiple URLs for single Resource (action Method)

[Route("api/get-all")]

[Route("getall")]

[Route("get-all")]

public string GetAll() {

return "Hello from get All";

}

Same URL for Multiple Resource

Not possible : ambiguity error

Token replacement in Routing

[Route("api/get-all")]

[Route("getall")]

[Route("get-all")]

[Route("[controller]/[action]")]

public string GetAll() {

return "Hello from get All";

}

Calling it : <https://localhost:44381/values/getall>

Ovverite routes:

//[Route("~/get-all")]

Video 51