Create Project

Sunday, January 7, 2024 11:50 AM

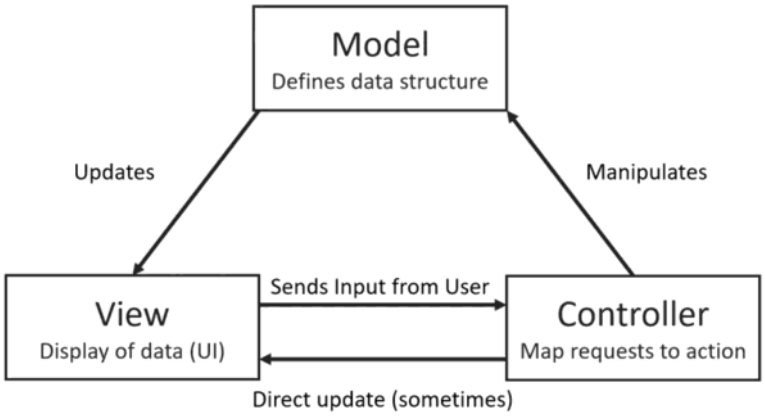
Differences Minimal API & MVC API

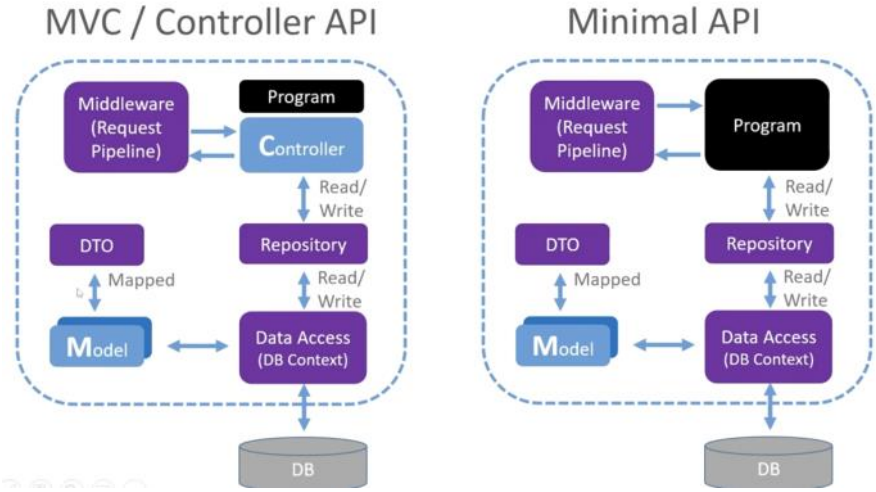
https://learn.microsoft.com/en-us/aspnet/core/fundamentals/apis?view=aspnetcore-8.0

Model – View - Controller

- Software design pattern (created @ Xerox Parc in the late 1970s)
- Splits program logic into three (interconnected) elements
- Widely adopted in web development frameworks:
  - Ruby on Rails
  - Spring
  - Django
  - ASP.NET MVC

Model – View - Controller





Controllers

- An action is a C# method that handles HTTP requests.
- Then the class that contains the action methods is a controller.
- Also, the controller will allow us to define global logic that will be applied to each action.

```
[ApiController]
[Route("[controller]")]
public class WeatherForecastController : ControllerBase
{
    private static readonly string[] Summaries = new[]
    {
    "Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot",
"Sweltering", "Scorching"
};

    private readonly ILogger<WeatherForecastController> _logger;

    public WeatherForecastController(ILogger<WeatherForecastController> logger)
    {
        _logger = logger;
    }

    [HttpGet]
    public IEnumerable<WeatherForecast> Get()
    {
        return Enumerable.Range(1, 5).Select(index => new WeatherForecast
        {
            Date = DateOnly.FromDateTime(DateTime.Now.AddDays(index)),
            TemperatureC = Random.Shared.Next(-20, 55),
            Summary = Summaries[Random.Shared.Next(Summaries.Length)]
        })
        .ToArray();
    }
}
```

Minimal APIs

- Allows us to build faster Web APIs with less code
- We will work directly with endpoints
- Although less powerful than controllers, they are faster and more efficient

```
var summaries = new[]
{
    "Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot",
"Sweltering", "Scorching"
};

app.MapGet("/weatherforecast", () =>
{
    var forecast = Enumerable.Range(1, 5).Select(index =>
        new WeatherForecast
        (
            DateOnly.FromDateTime(DateTime.Now.AddDays(index)),
            Random.Shared.Next(-20, 55),
            summaries[Random.Shared.Next(summaries.Length)]
        ))
        .ToArray();
    return forecast;
});
```

This means*...

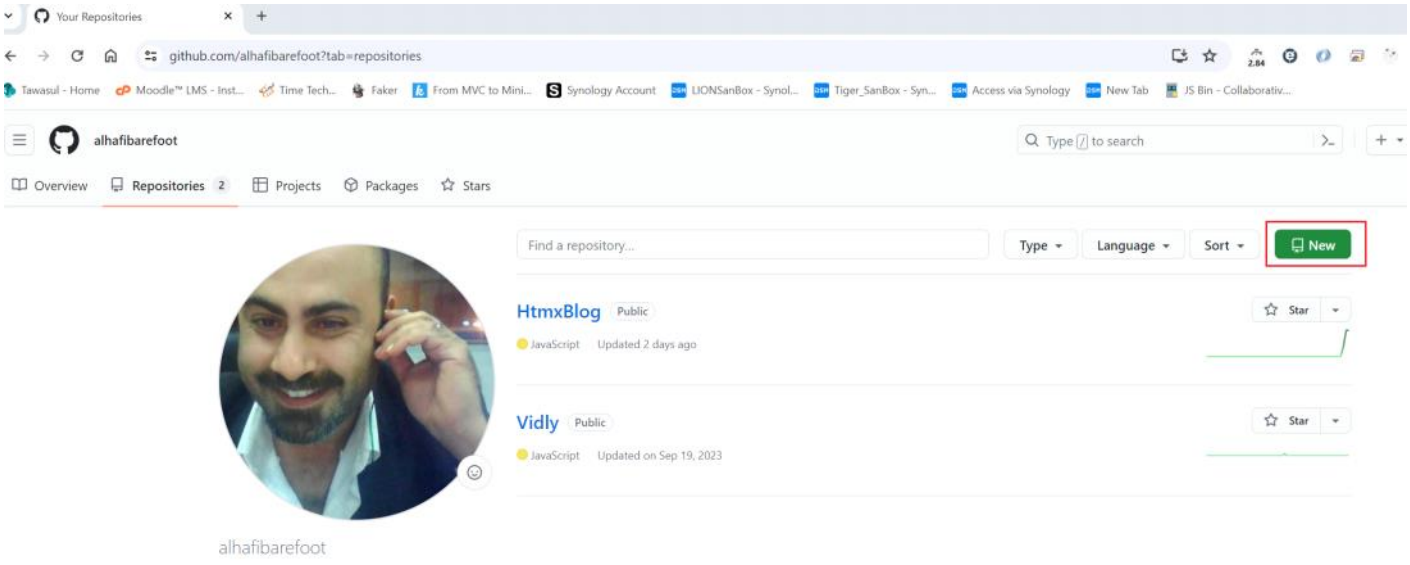
Minimal APIs:

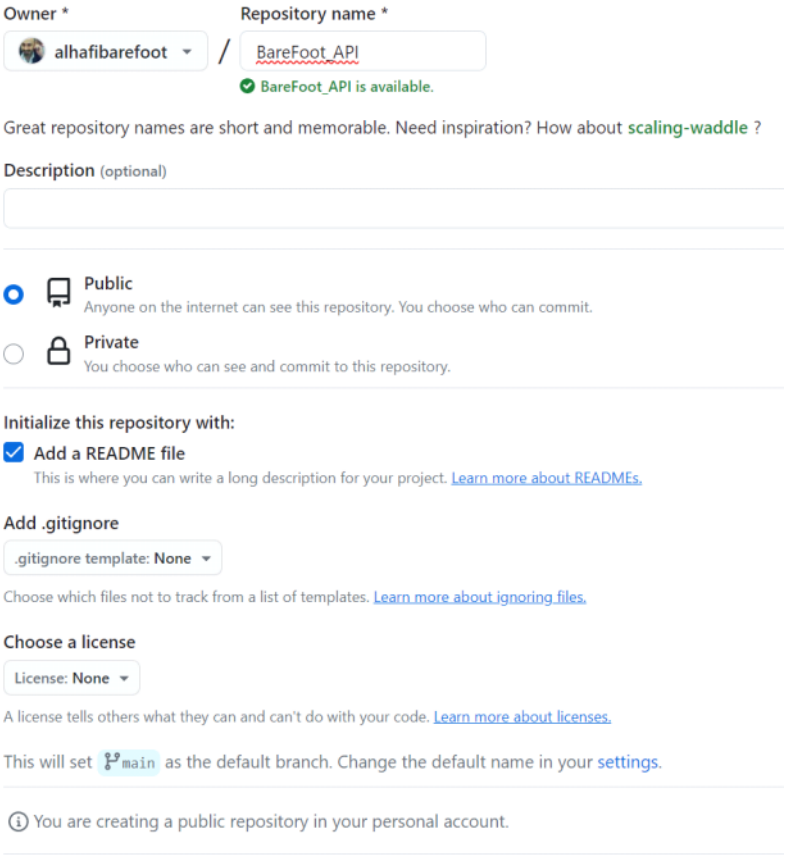
- Don't support model validation
- Don't support for JSONPatch
- Don't support filters
- Don't support *custom* model binding (Support for IModelBinder)

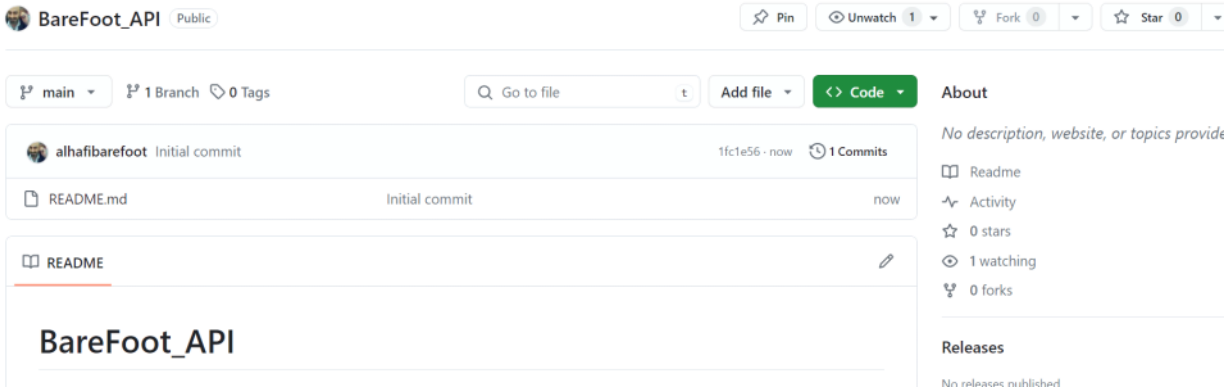
Limitations of Minimal API

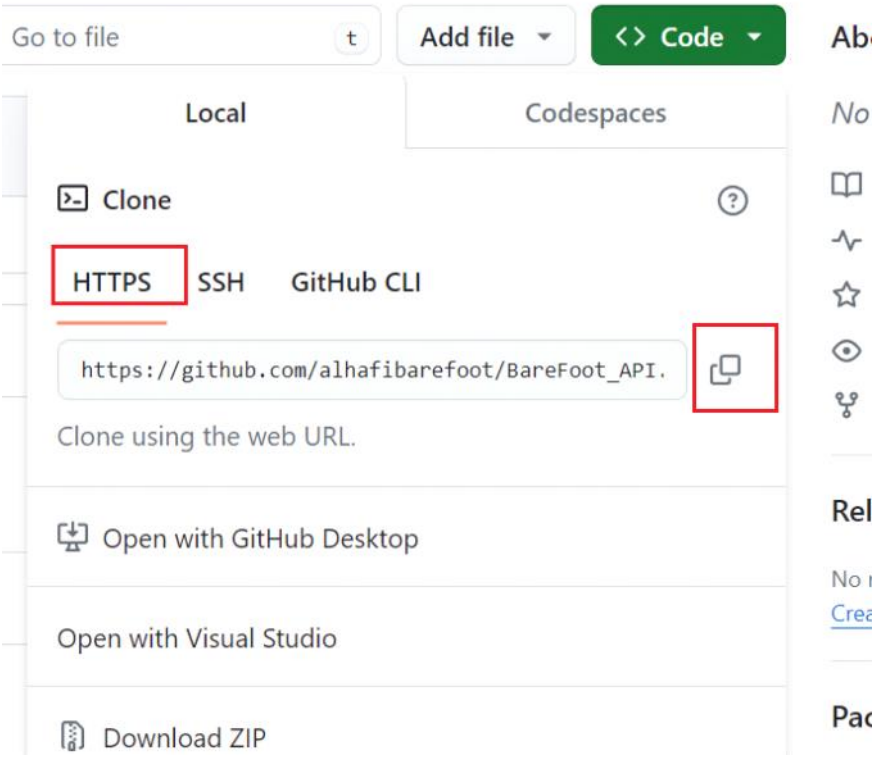
1. No support for filters: For example, no support for IAsyncAuthorizationFilter, IAsyncActionFilter, IAsyncExceptionFilter, IAsyncResultFilter, and IAsyncResourceFilter.
2. No support for model binding, i.e. IModelBinderProvider, IModelBinder. Support can be added with a custom binding shim.
3. No support for binding from forms. This includes binding IFormFile. We plan to add support for IFormFile in the future.
4. No built-in support for validation, i.e. IModelValidator
5. No support for application parts or the application model. There's no way to apply or build your own conventions.
6. No built-in view rendering support. We recommend using Razor Pages for rendering views.
7. No support for JsonPatch
8. No support for OData
9. No support for ApiVersioning. See this issue for more details.

GITHUB


Your Repositories
github.com/alhafibarefoot?tab=repositories
alhafibarefoot
Type / to search
Overview
Repositories 2
Projects
Packages
Stars
Find a repository...
Type
Language
Sort
New
HtmxBlog
Public
JavaScript
Updated 2 days ago
Star
Vidly
Public
JavaScript
Updated on Sep 19, 2023
Star
alhafibarefoot


Owner *
alhafibarefoot
/
Repository name *
BareFoot_API
BareFoot_API is available.
Great repository names are short and memorable. Need inspiration? How about scaling-waddle ?
Description (optional)
Public
Anyone on the internet can see this repository. You choose who can commit.
Private
You choose who can see and commit to this repository.
Initialize this repository with:
Add a README file
This is where you can write a long description for your project. Learn more about READMEs.
Add .gitignore
.gitignore template: None
Choose which files not to track from a list of templates. Learn more about ignoring files.
Choose a license
License: None
A license tells others what they can and can't do with your code. Learn more about licenses.
This will set main as the default branch. Change the default name in your settings.
You are creating a public repository in your personal account.


BareFoot_API
Public
Pin
Unwatch 1
Fork 0
Star 0
main
1 Branch
0 Tags
Go to file
t
Add file
Code
About
alhafibarefoot Initial commit
1fc1e56 · now
1 Commits
README.md
Initial commit
now
README
BareFoot_API
No description, website, or topics provide
Readme
Activity
0 stars
1 watching
0 forks
Releases
No releases published

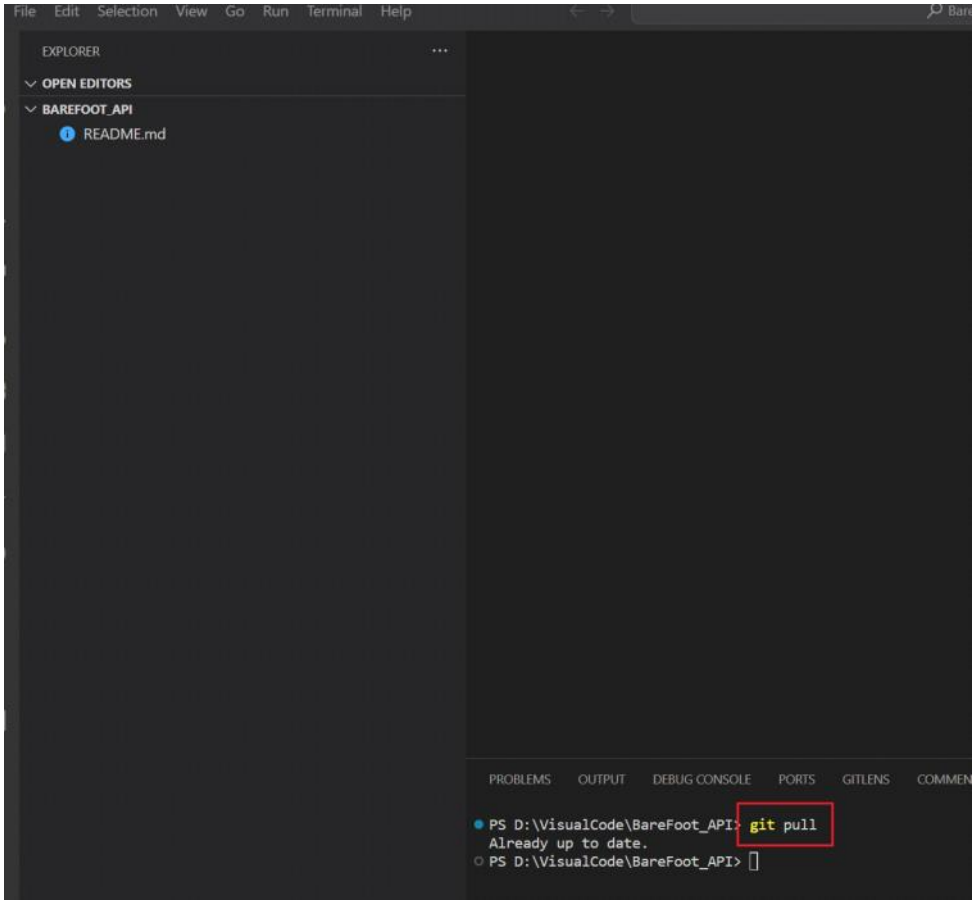


git clone https://github.com/alhafibarefoot/BareFoot_API.git

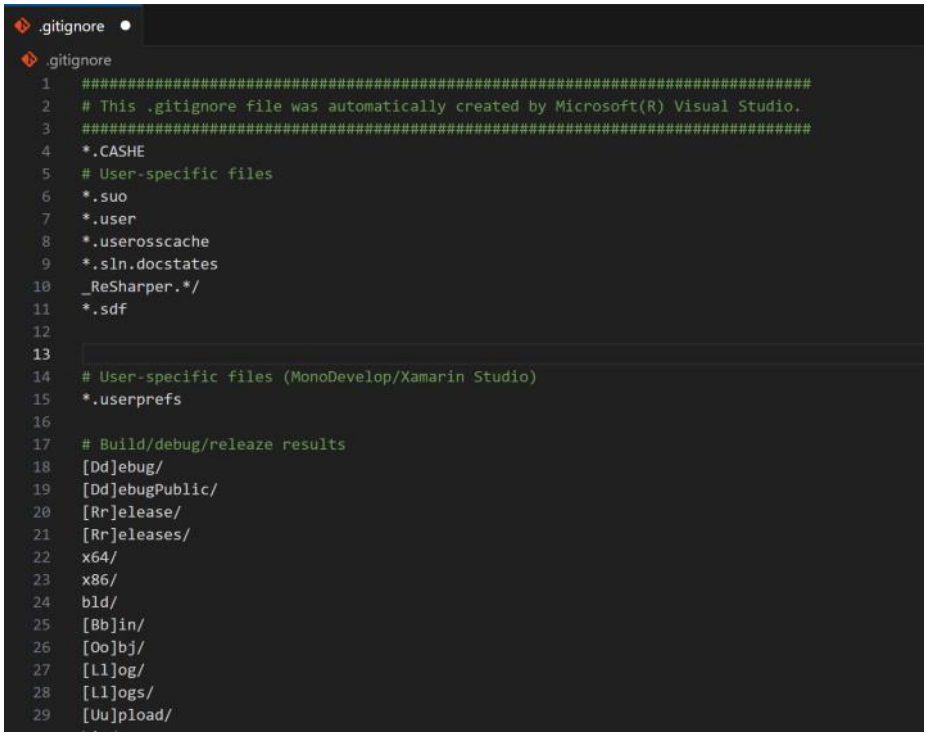
```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.

D:\VisualCode>git clone https://github.com/alhafibarefoot/BareFoot_API.git
```

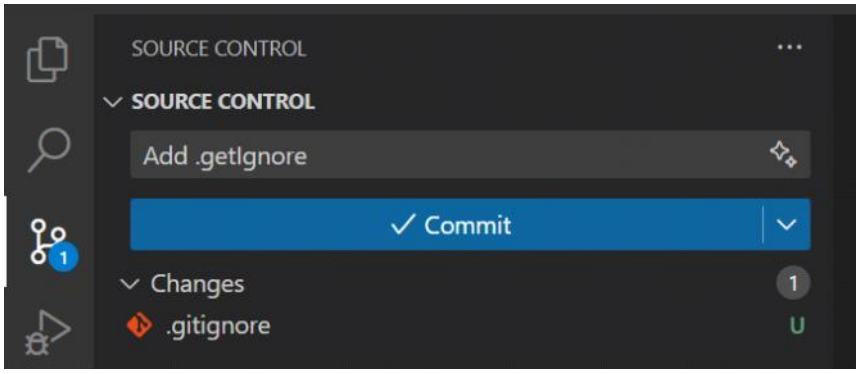
Using Visual Code

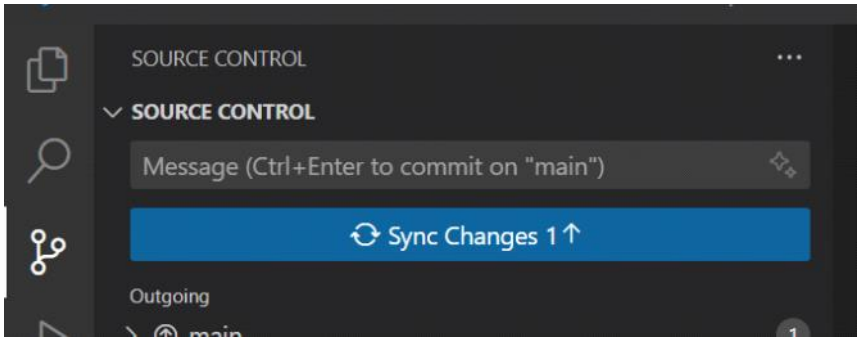
```
D:\VisualCode\BareFoot_API>code .
```



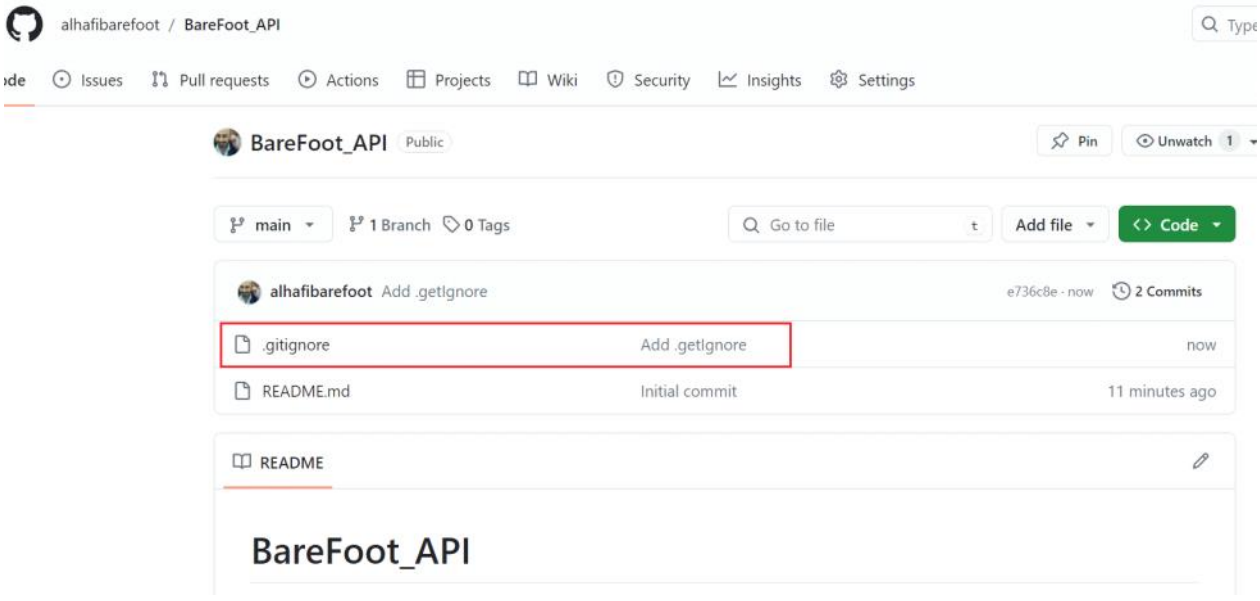
Before Adding Project Let we add ignore

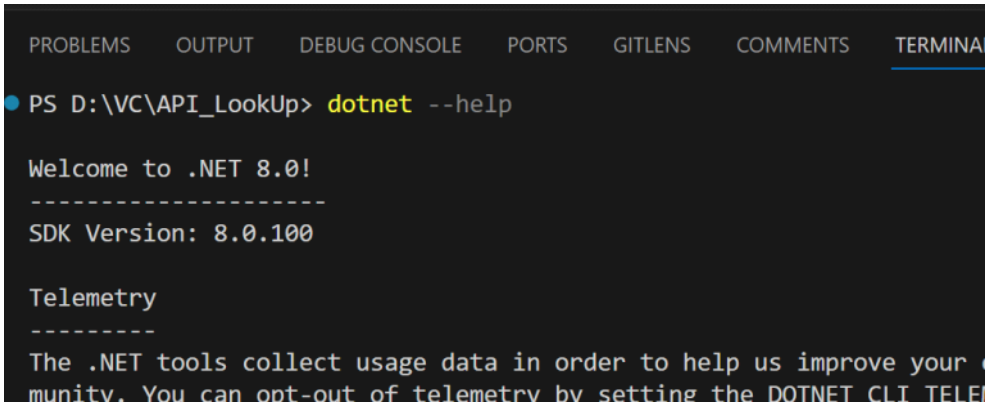






Checking Update git hub



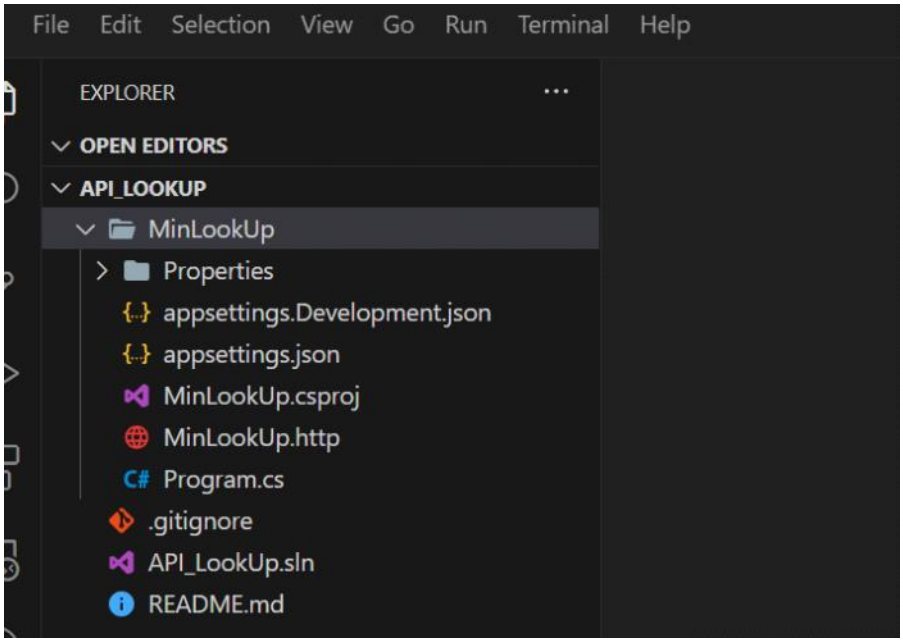


```
PS D:\VC\API_LookUp> dotnet --list-sdks
8.0.100 [C:\Program Files\dotnet\sdk]
PS D:\VC\API_LookUp>
```

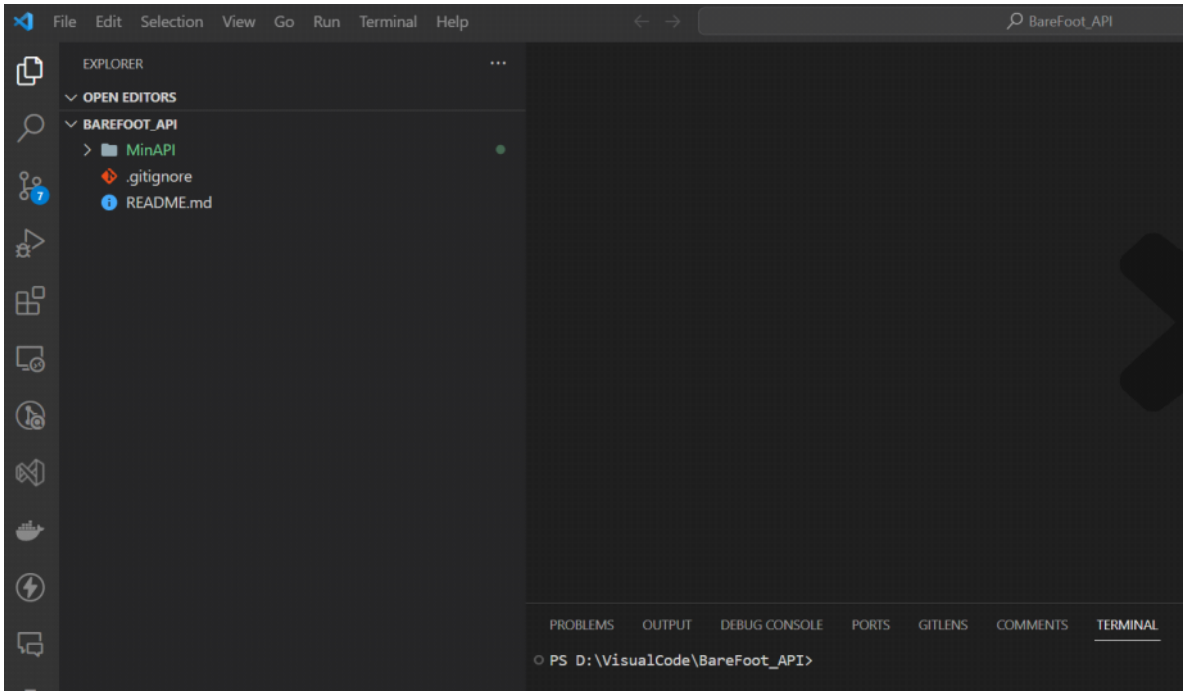
```
PS D:\VC\API_LookUp> dotnet new list
These templates matched your input:

Template Name       Short Name                Language    Tags
------------------  ------------------------  ----------  ----------------------
.NET MAUI App       maui                      [C#]        MAUI/Android/iOS/macOS/M
.NET MAUI Blazo...  maui-blazor               [C#]        MAUI/Android/iOS/macOS/M
.NET MAUI Class...  mauilib                   [C#]        MAUI/Android/iOS/macOS/M
.NET MAUI Conte...  maui-page-csharp          [C#]        MAUI/Android/iOS/macOS/M
.NET MAUI Conte...  maui-page-xaml            [C#]        MAUI/Android/iOS/macOS/M
.NET MAUI Conte...  maui-view-csharp          [C#]        MAUI/Android/iOS/macOS/M
.NET MAUI Conte...  maui-view-xaml            [C#]        MAUI/Android/iOS/macOS/M
.NET MAUI Resou...  maui-dict-xaml            [C#]        MAUI/Android/iOS/macOS/M
Android Activit...  android-activity          [C#]        Android/Mobile
Android Applica...  android                   [C#]        Android/Mobile
Android Class L...  androidlib                [C#]        Android/Mobile
Android Java Li...  android-bindinglib        [C#]        Android/Mobile
Android Layout ...  android-layout            [C#]        Android/Mobile
Android Wear Ap...  androidwear               [C#]        Android/Mobile
API Controller      apicontroller             [C#]        Web/ASP.NET
ASP.NET Core Empty  web                       [C#],F#     Web/Empty
ASP.NET Core gR...  grpc                      [C#]        Web/gRPC/API/Service
ASP.NET Core We...  webapi                    [C#],F#     Web/WebAPI/Web API/API
ASP.NET Core We...  webapiaot                 [C#]        Web/Web API/API/Service
```

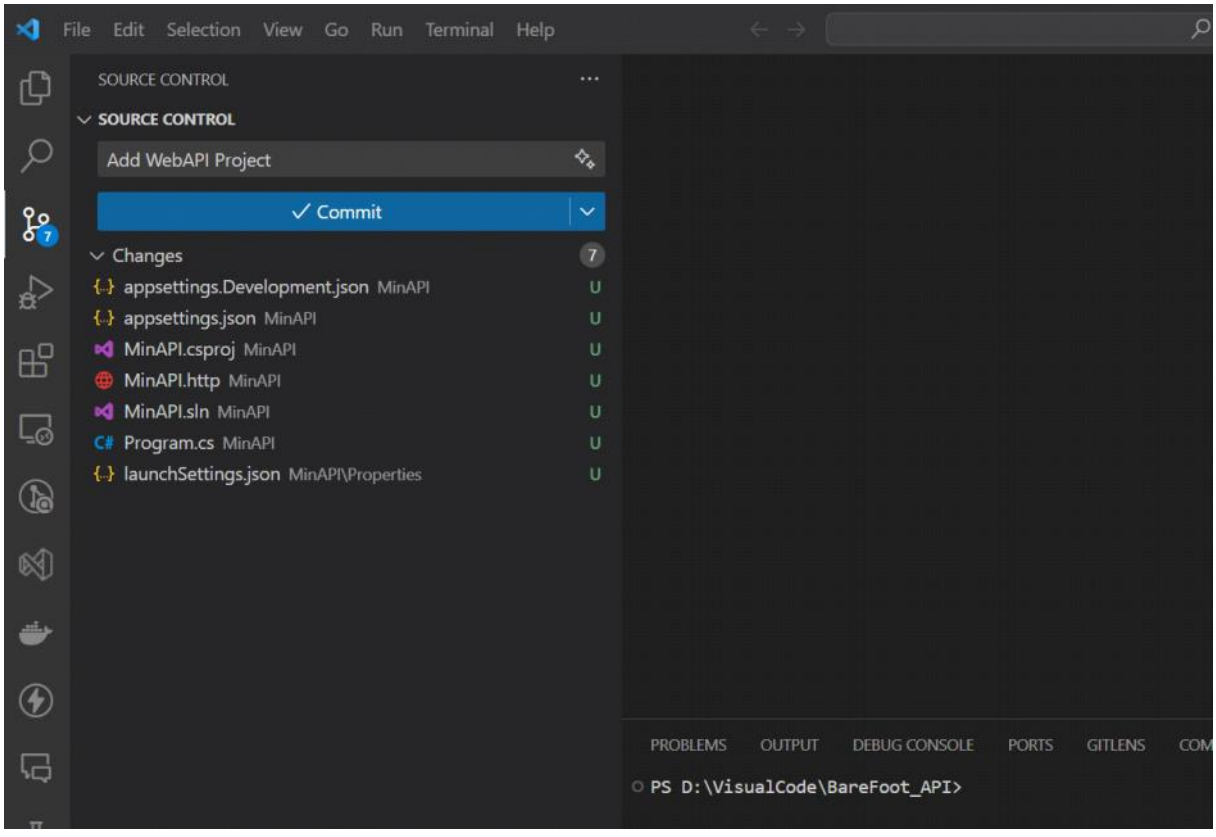
Creating New Project

dotnet new webapi -minimal -n MinAPI



CD to project



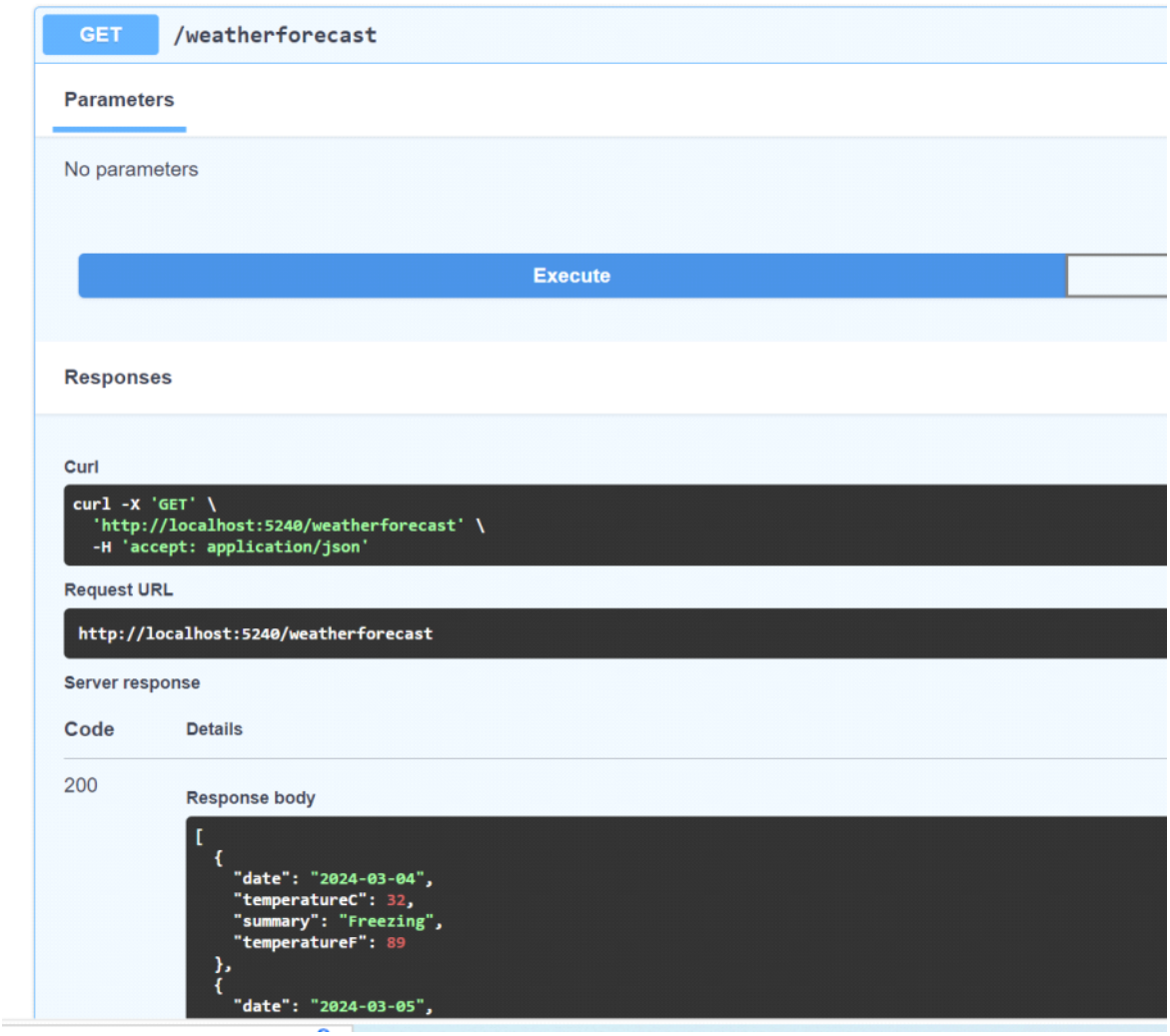


Testing

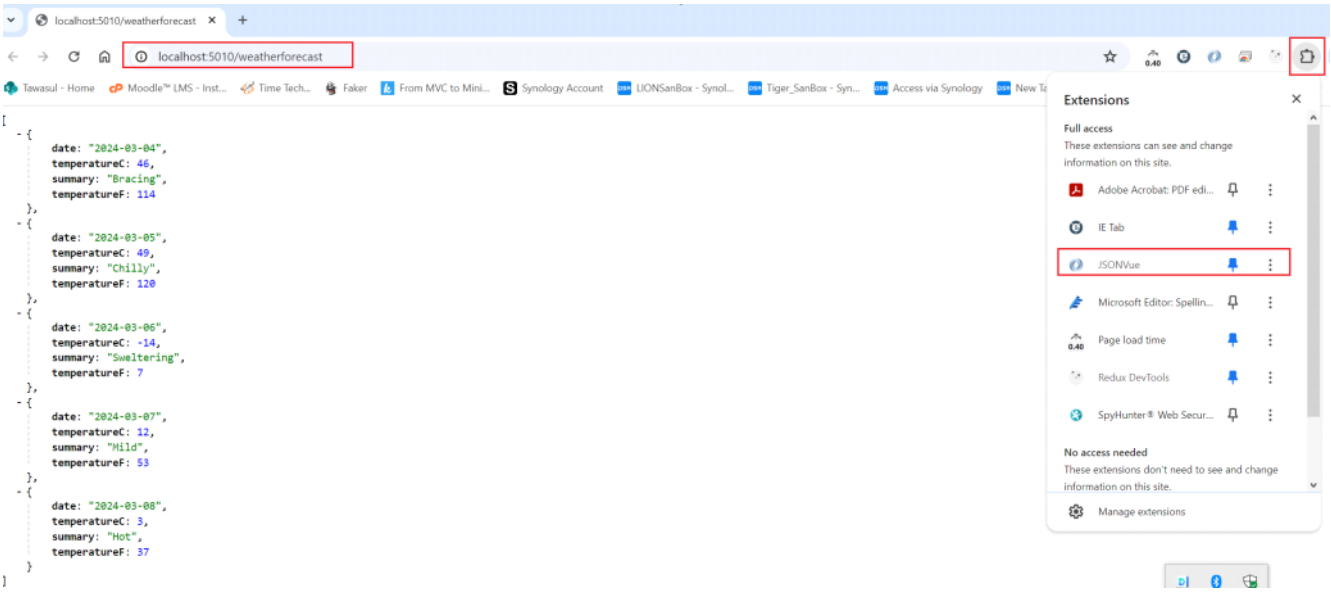
```
PS D:\VisualCode\BareFoot_API> cd minapi
PS D:\VisualCode\BareFoot_API\minapi> dotnet watch run
```

Check your swagger

MinAPI



Or You can browser it by direct address
And to enhance Json format add plugin to chrome **JsonVue**



custom out port of http--->5010 and https--->7010

launchSettings.json M

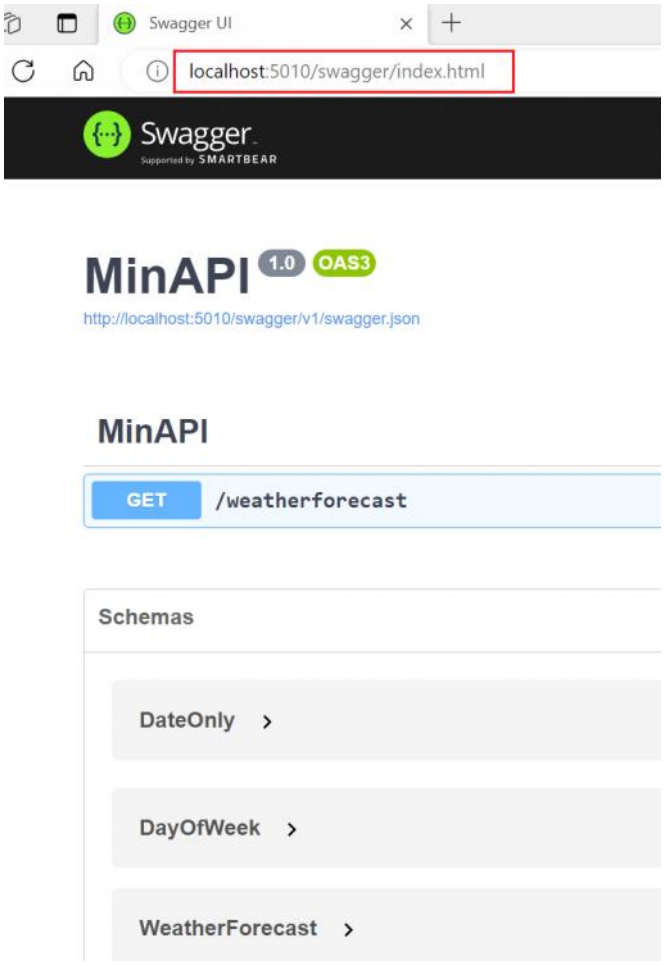
MinAPI > Properties > launchSettings.json > { } profiles > { } https > applicationUrl

```
    "sslPort": 44346
  }
},
"profiles": {
  "http": {
    "commandName": "Project",
    "dotnetRunMessages": true,
    "launchBrowser": true,
    "launchUrl": "swagger",
    "applicationUrl": "http://localhost:5010",
    "environmentVariables": {
      "ASPNETCORE_ENVIRONMENT": "Development"
    }
  },
  "https": {
    "commandName": "Project",
    "dotnetRunMessages": true,
    "launchBrowser": true,
    "launchUrl": "swagger",
    "applicationUrl": "https://localhost:7010;http://localhost:5010",
    "environmentVariables": {
      "ASPNETCORE_ENVIRONMENT": "Development"
```

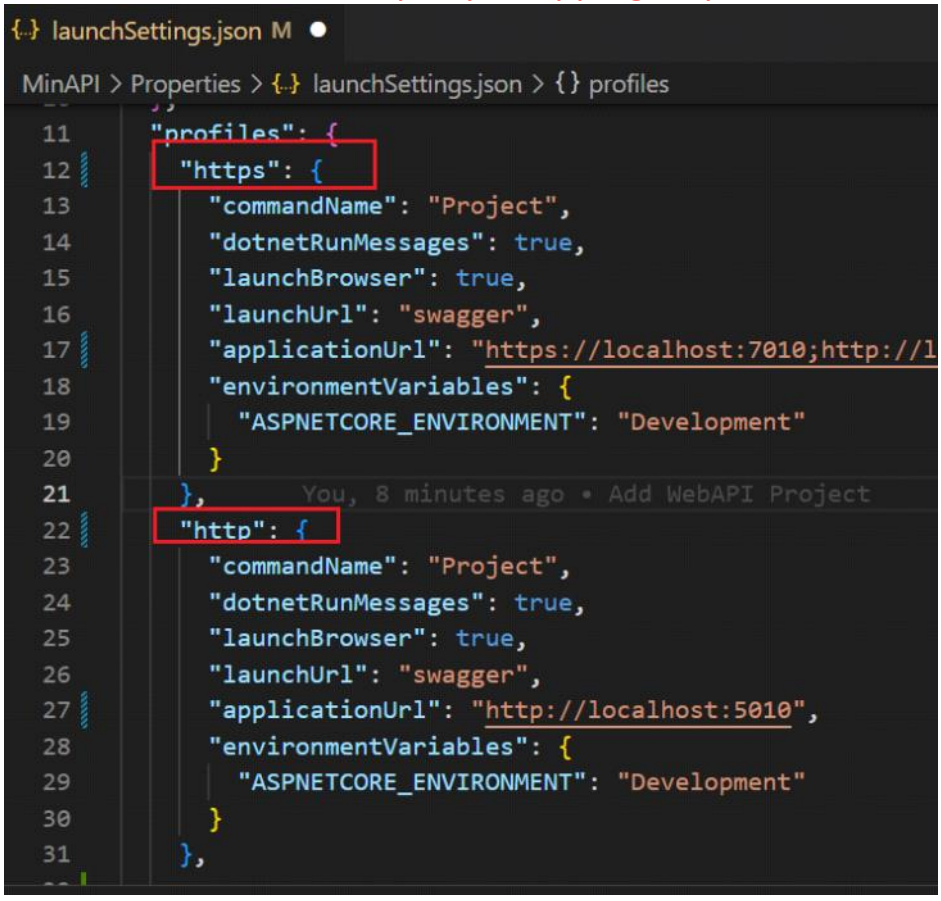
PROBLEMS OUTPUT DEBUG CONSOLE PORTS GITLENS COMMENTS TERMINAL

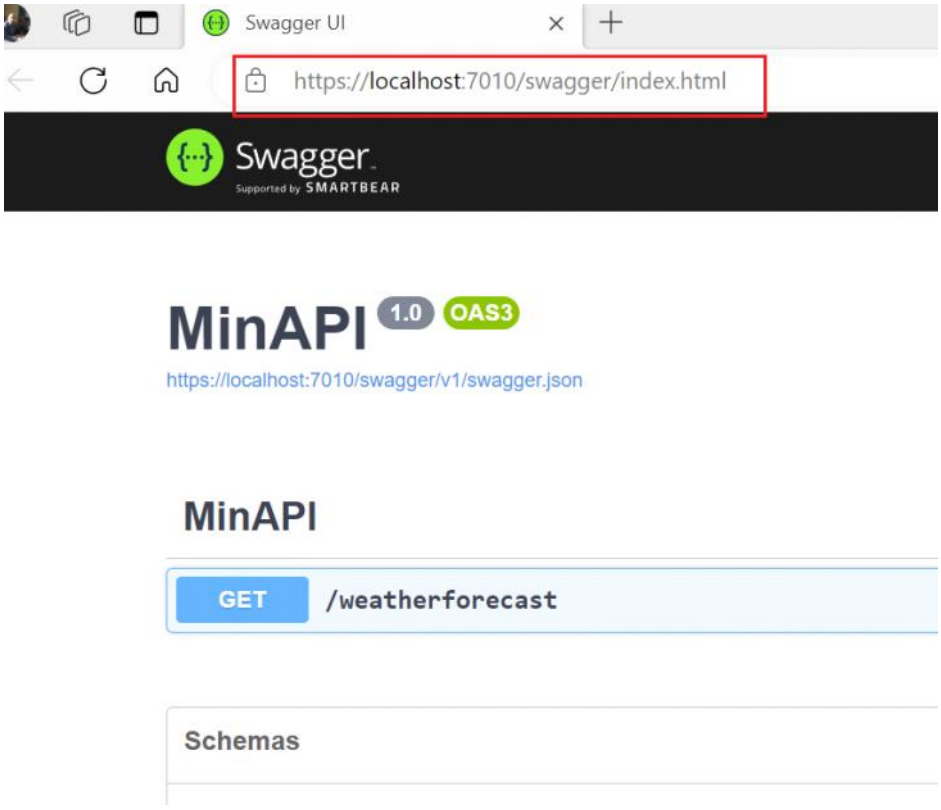
```
PS C:\DevGIT\skitnetCore6\api> dotnet watch run
```

It work as http



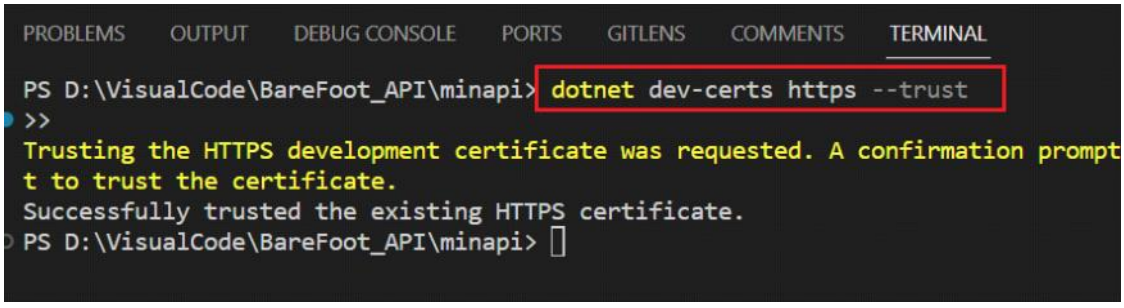
Let we make default https by swapping https first





Adding HTTPs developer trust certification

dotnet dev-certs https --trust

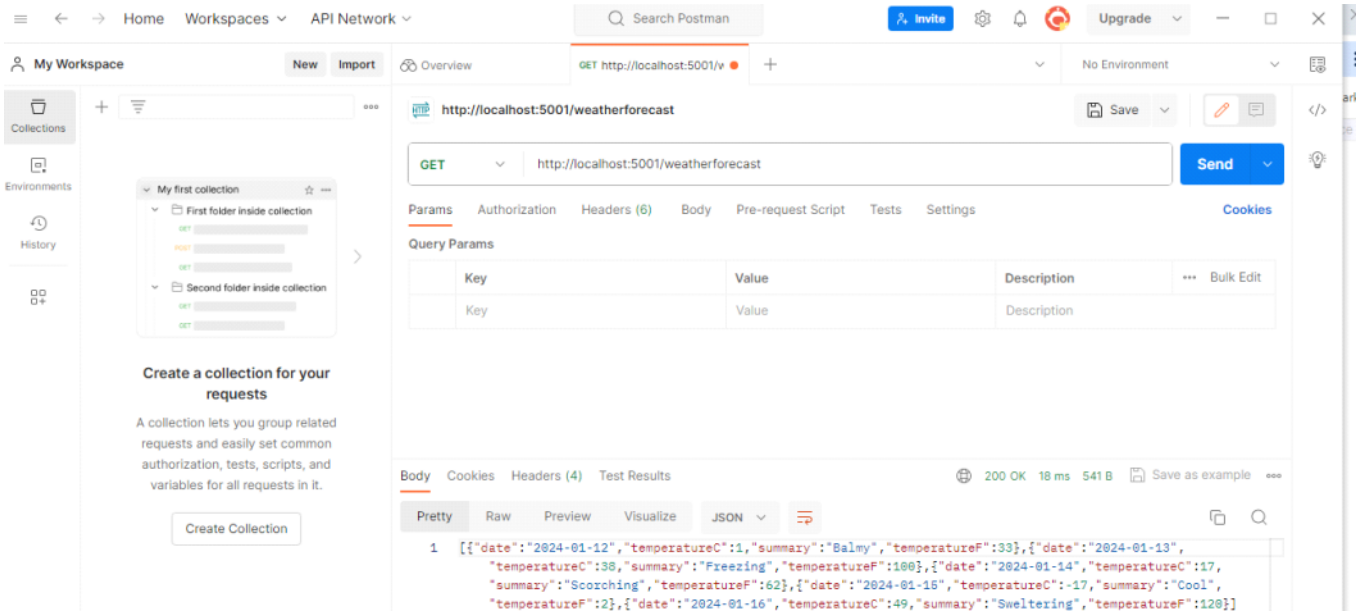


Now our project working as default in https mode , but we can run also in http by

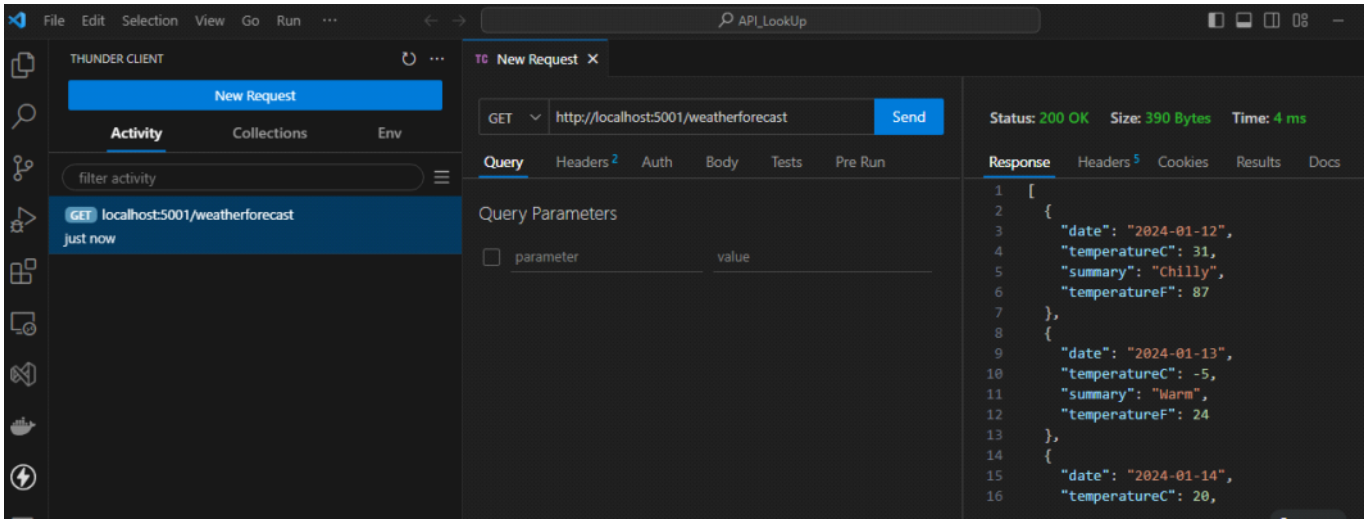
dotnet watch run --launch-profile http

We can use third parties such as **postman**

POSTMAN



Or extension in Visual code **Thunder Client**



Viewing Web API

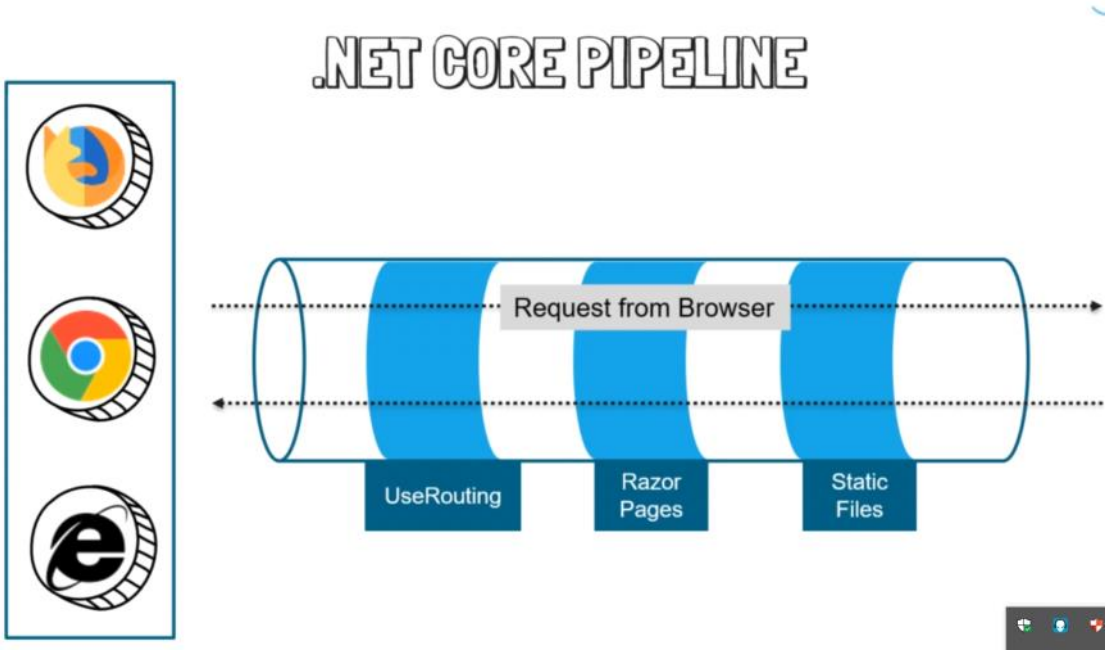
Tuesday, January 9, 2024 12:33 PM

@ programs.cs

Services

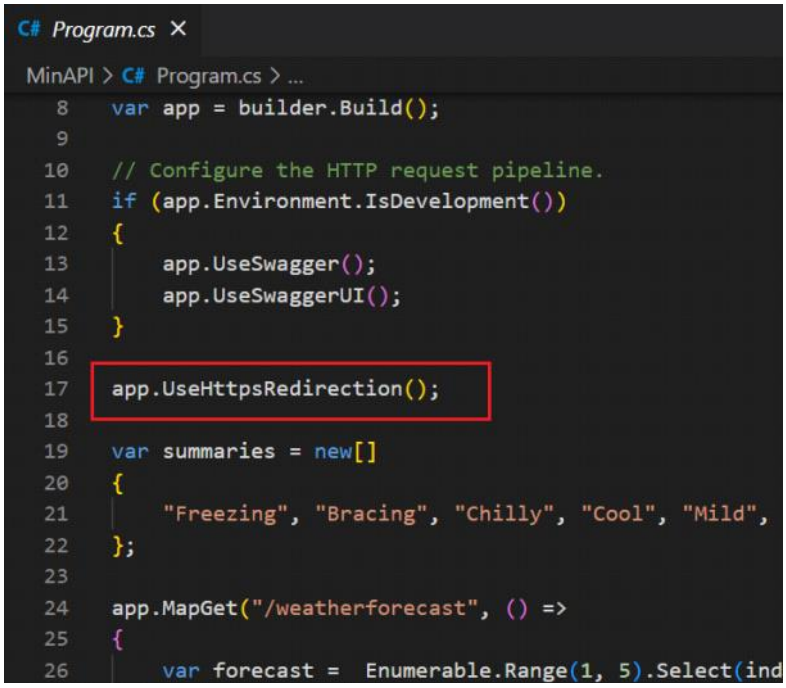
```
inAPI > C# Program.cs > ...
      You, 21 minutes ago | 1 author (You)
1     var builder = WebApplication.CreateBuilder(args);      You, 21 minutes ago • Add WebAPI
2     
3     // Add services to the container.
4     // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckl
5     builder.Services.AddEndpointsApiExplorer();
6     builder.Services.AddSwaggerGen();
7
```

pipeline



Middle Points

```
// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}
```



End Points

```
app.MapGet("/weatherforecast", () =>
{
    var forecast =  Enumerable.Range(1, 5).Select(index =>
        new WeatherForecast
        (
            DateOnly.FromDateTime(DateTime.Now.AddDays(index)),
            Random.Shared.Next(-20, 55),
            summaries[Random.Shared.Next(summaries.Length)]
        ))
        .ToArray();
    return forecast;
})
.WithName("GetWeatherForecast")
.WithOpenApi();
```

==========

Understanding Middle Point **redirect**

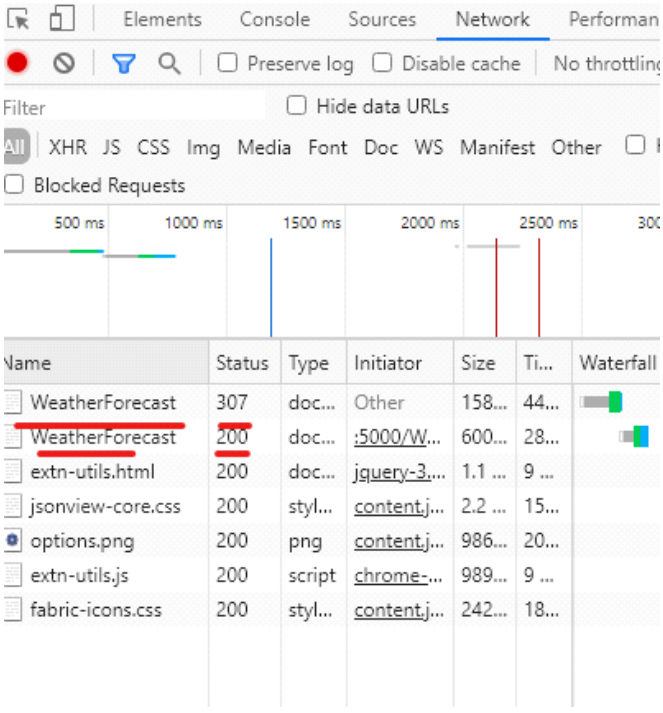
```
app.UseHttpsRedirection();
```

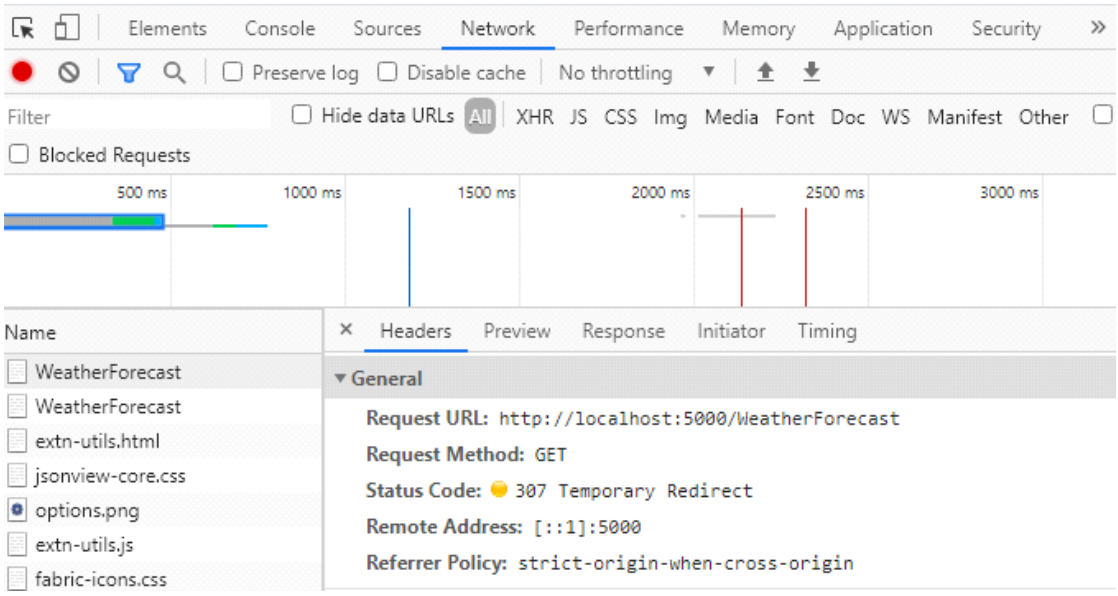
Assume we call HTTP instead of https

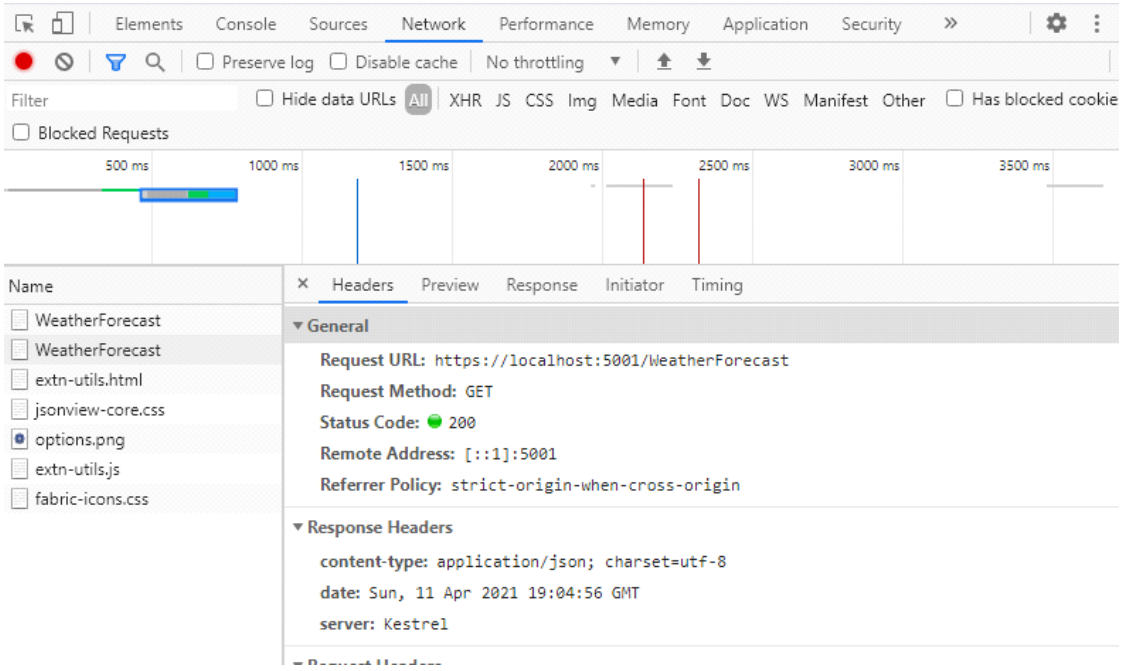
http://localhost:5010/WeatherForecast

```
[
  - {
      date: "2022-08-16T13:55:12.5035678+03:00",
      temperatureC: 31,
      temperatureF: 87,
      summary: "Bracing"
    },
  - {
      date: "2022-08-17T13:55:12.5035755+03:00",
      temperatureC: 54,
      temperatureF: 129,
      summary: "Warm"
    },
  - {
      date: "2022-08-18T13:55:12.5035759+03:00",
      temperatureC: 6,
      temperatureF: 42,
      summary: "Bracing"
    },
  - {
      date: "2022-08-19T13:55:12.5035761+03:00",
      temperatureC: 22,
      temperatureF: 71,
      summary: "Sweltering"
    },
  - {
      date: "2022-08-20T13:55:12.5035772+03:00",
      temperatureC: -1,
      temperatureF: 31,
      summary: "Scorching"
    }
]
```


307 =redirect
200 response ok to https







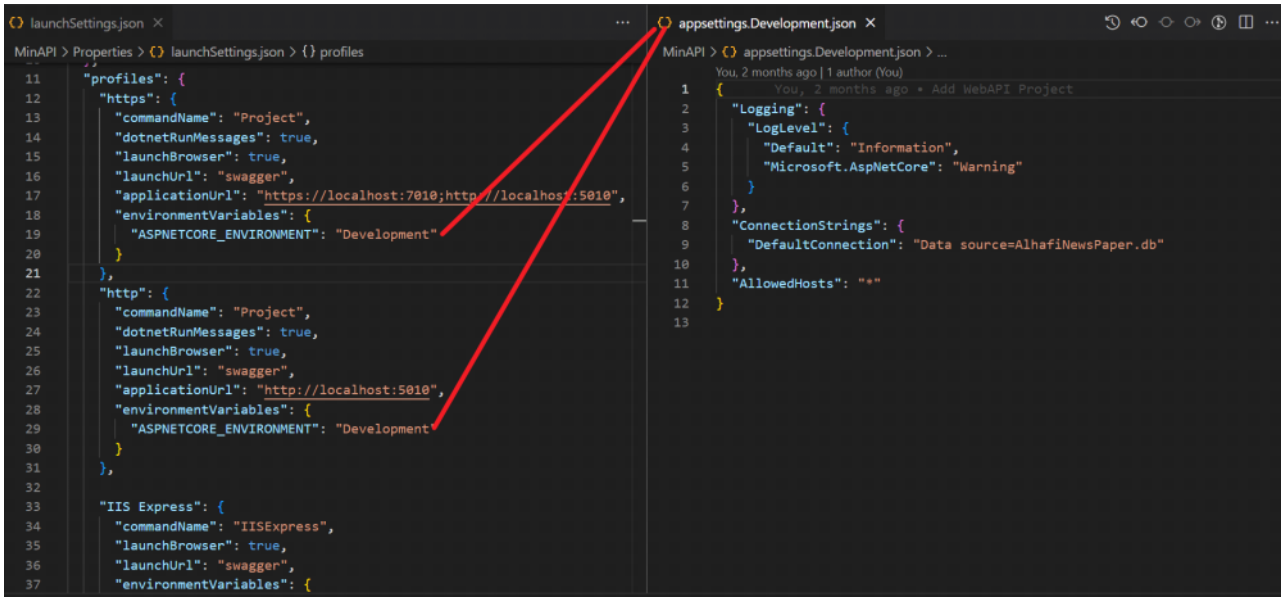
==========
==========

Let we update Environment

```
// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}
```

```
// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}
if (app.Environment.IsStaging())
{
    // your code here
}
if (app.Environment.IsProduction())
{
    // your code here
}
```

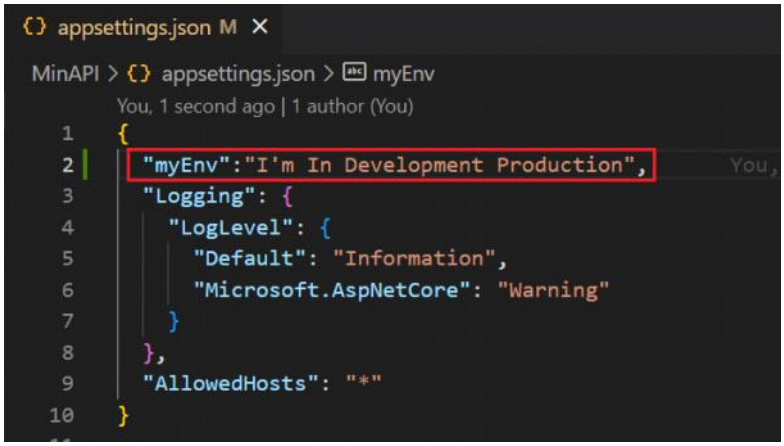
```
// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}
if (app.Environment.IsStaging())
{
    // your code here
}
if (app.Environment.IsProduction())
{
     // your code here
}
```



Understanding environments

How to check in which environment you are stand , So Let we add string to environment


```
"myEnv":"I'm In Development Environment",
```




```
"myEnv":"I'm In Production Environment",
```

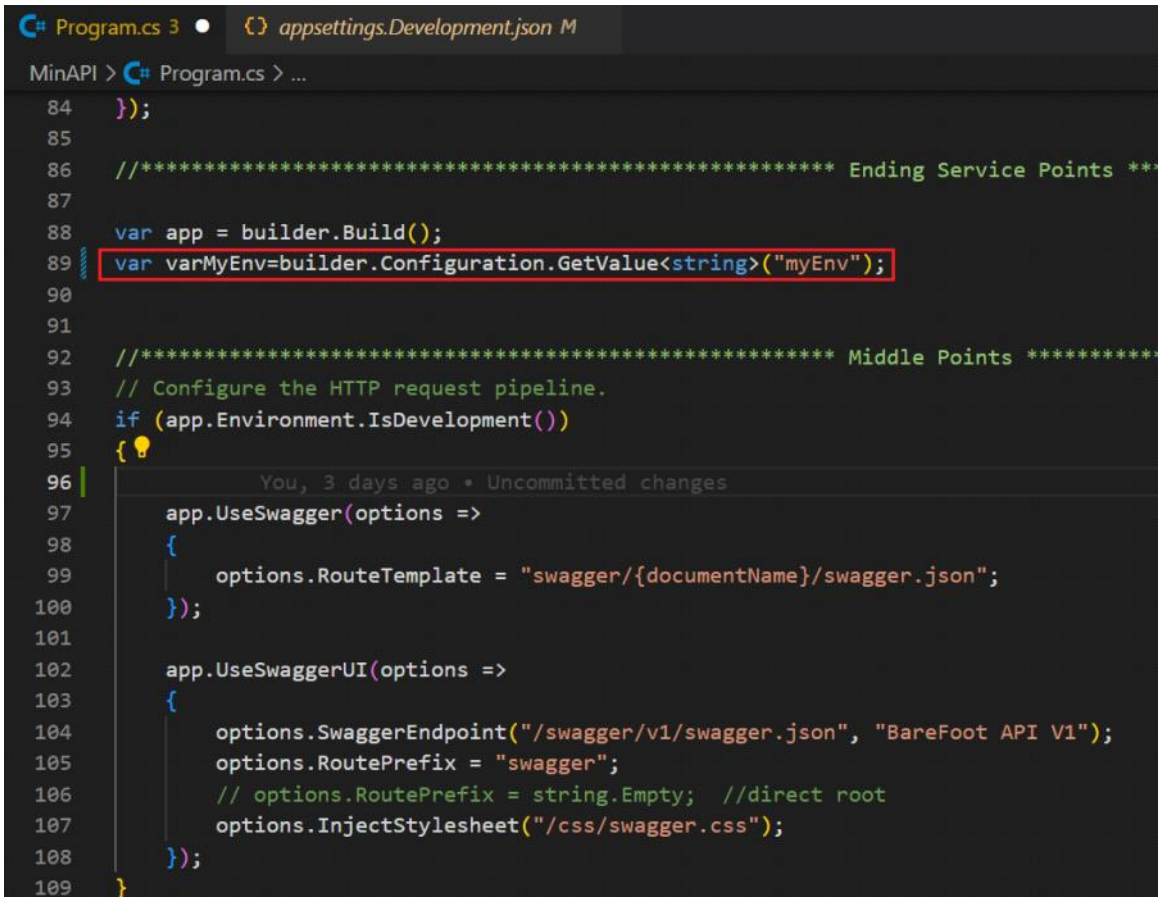
If we want to run in specific environment we can change this Production

But Preferred Method While developing

dotnet watch run --environment Production

dotnet watch run --environment Development

Let we try this

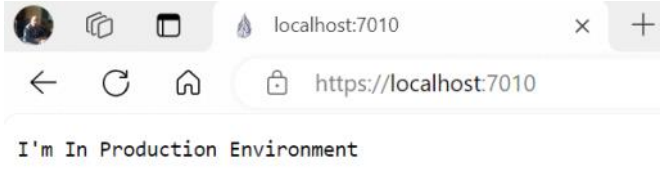



```
var varMyEnv=builder.Configuration.GetValue<string>("myEnv");
```

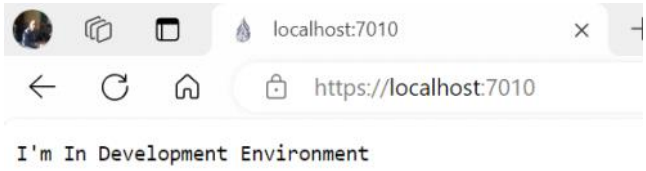
```
Program.cs 3 ●   {} appsettings.Development.json M
MinAPI > Program.cs > ...
113         // your code here
114     }
115     if (app.Environment.IsProduction())
116     {
117
118         // your code here
119     }
120
121     app.UseHttpsRedirection();
122
123     app.UseStaticFiles();
124
125     //************************************************** Ending Middle Points ***
126
127
128     //************************************************** End Points ************
129
130     //************************Static Sample Hello**********************************
131     💡
132     app.MapGet("/",()=>varMyEnv);      You, 3 days ago • Uncommitted changes
133
134     app.MapGet("/hello", () => "[GET] Hello World!").WithTags("Hello");
135     app.MapPost("/hello", () => "[POST] Hello World!").WithTags("Hello");
136     app.MapPut("/hello", () => "[PUT] Hello World!").WithTags("Hello");
137     app.MapDelete("/hello", () => "[DELETE] Hello World!").WithTags("Hello");
138
139     app.MapGet(
```

```
app.MapGet("/",()=>varMyEnv);
```

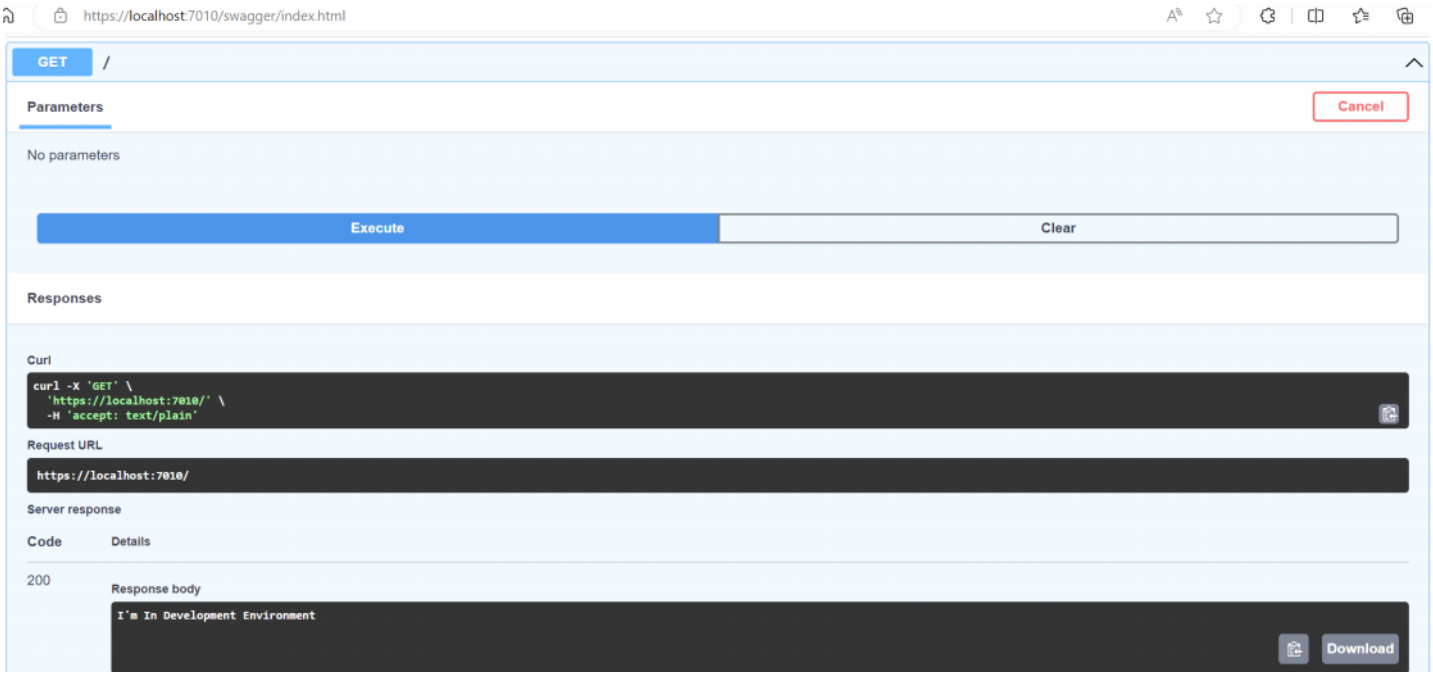
Let we run this
dotnet watch run --environment Production



dotnet watch run --environment Development



Or by swagger

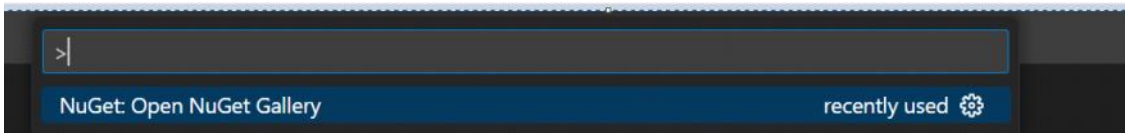


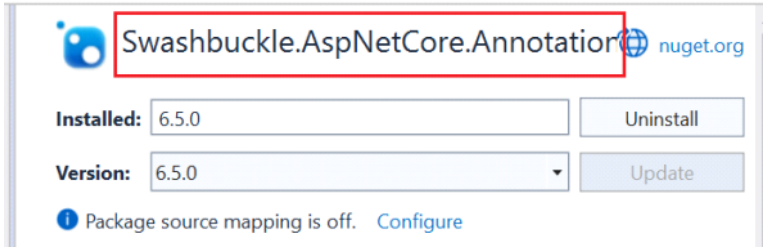
==========

=====

Customize Swagger UI in ASP.NET Core

CTRL+Shift+P in visual code





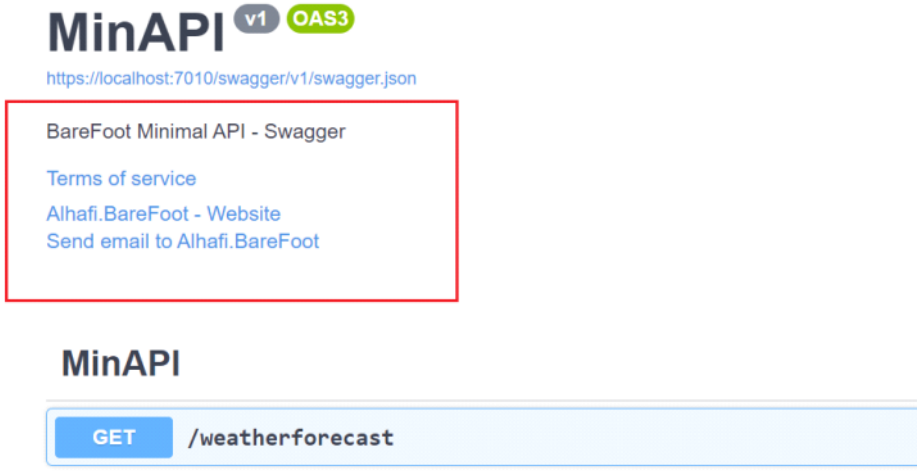
And now let we add Documentation

```
builder.Services.AddSwaggerGen(c =>
{
    c.EnableAnnotations();
    c.SwaggerDoc(
        "v1",
        new OpenApiInfo
        {
            Title = "Barefoot API",
            Version = "v1",
            Contact = new()
            {
                Name = "Alhafi.BareFoot",
                Email = "alhafi@hotmail.com",
                Url = new Uri("https://www.alhafi.org/")
            },
            Description =
                " BareFoot Minimal API Build in <b>dotnet new webapi -minimal</b>  Hosted at github  <a href='htt
            License = new Microsoft.OpenApi.Models.OpenApiLicense(),
            TermsOfService = new("https://www.alhafi.org/")
        }
    );
});
```

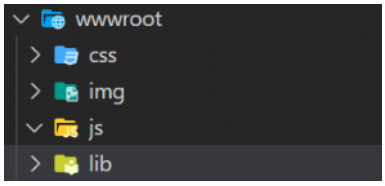
```
builder.Services.AddSwaggerGen(c =>
{
    c.EnableAnnotations();
    c.SwaggerDoc(
        "v1",
        new OpenApiInfo
        {
            Title = "Barefoot API",
            Version = "v1",
            Contact = new()
            {
                Name = "Alhafi.BareFoot",
                Email = "alhafi@hotmail.com",
                Url = new Uri("https://www.alhafi.org/")
            },
            Description =
                " BareFoot Minimal API Build in <b>dotnet new webapi -
minimal</b>  Hosted at github  <a
href='https://github.com/alhafibarefoot/BareFoot_API'>here</a>",
            License = new Microsoft.OpenApi.Models.OpenApiLicense(),
            TermsOfService = new("https://www.alhafi.org/")
        }
    );
});
```

Testing

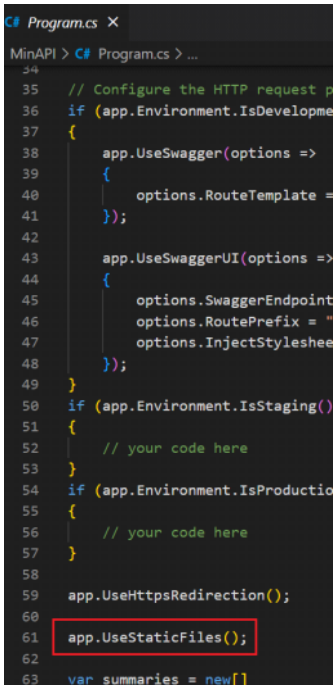
```
PS D:\VisualCode\BareFoot_API\minapi> dotnet watch run
```



add a folders t wwwroot with sub folders (ccs/img/js/lib)

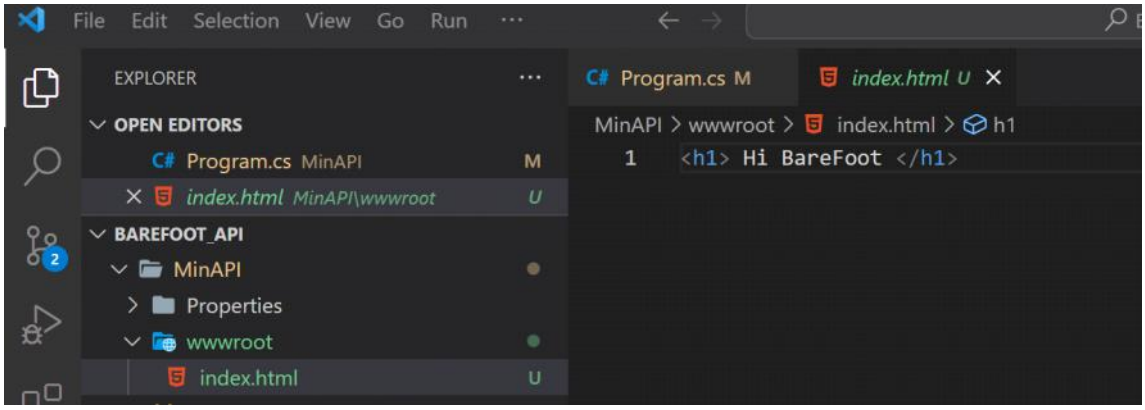


Add middle point

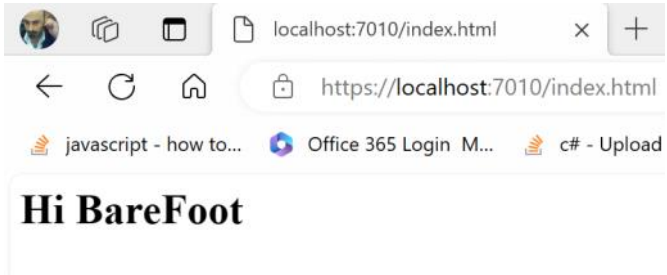



```
app.UseStaticFiles();
```

Test the static folder wwwroot by adding static html



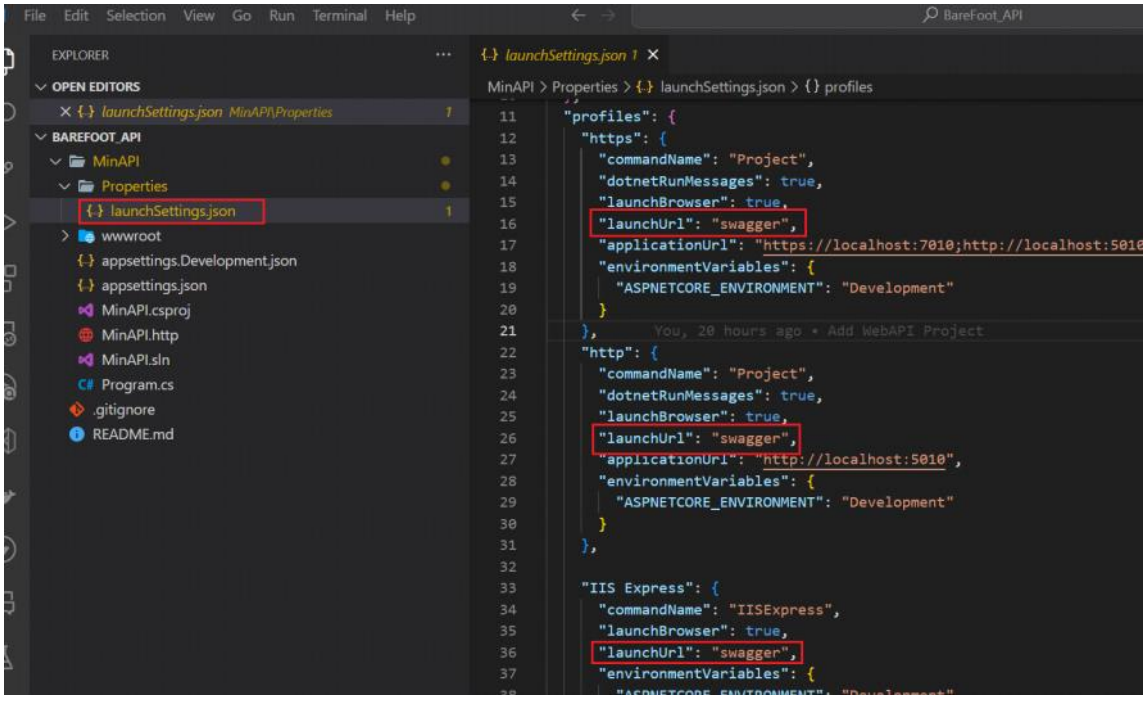
Testing



Route Update

```
Program.cs
MinAPI > Program.cs > ...
        }
    );
});

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger(options =>
    {
        options.RouteTemplate = "swagger/{documentName}/swagger.json";
    });

    app.UseSwaggerUI(options =>
    {
        options.SwaggerEndpoint("/swagger/v1/swagger.json", "BareFoot API V1");
        options.RoutePrefix = "swagger";

    });
}
if (app.Environment.IsStaging())
{
    // your code here
}
if (app.Environment.IsProduction())
{
    // your code here
}
```

```
 app.UseSwagger(options =>
    {
        options.RouteTemplate = "swagger/{documentName}/swagger.json";
    });
    app.UseSwaggerUI(options =>
    {
        options.SwaggerEndpoint("/swagger/v1/swagger.json", "BareFoot API
V1");
        options.RoutePrefix = "swagger";
           });
```



Testing

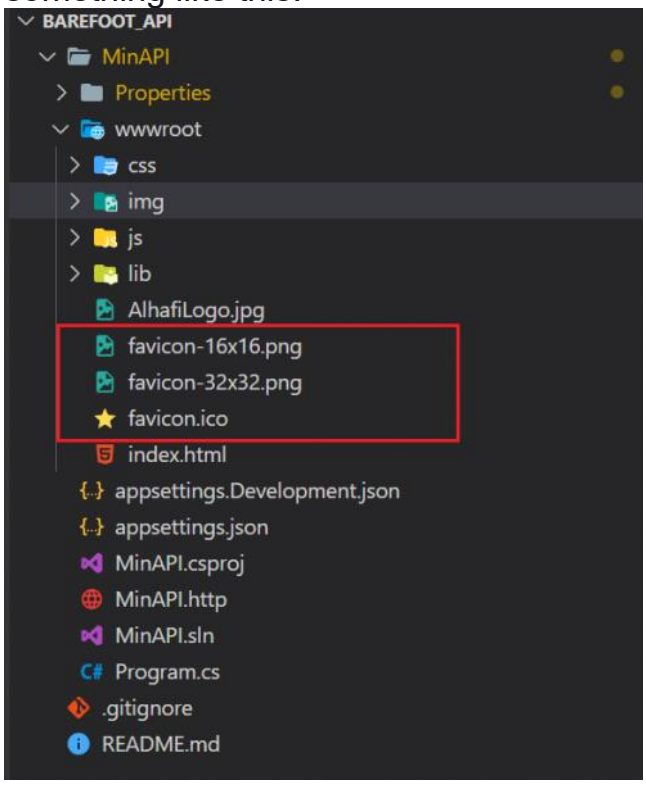
```
PS D:\VisualCode\BareFoot_API\minapi> dotnet watch run
```

Replace Swagger Icon/Logo

The Swagger logo can be seen both in the favicon and in the top bar, on the top left corner of the site. **Favicon**

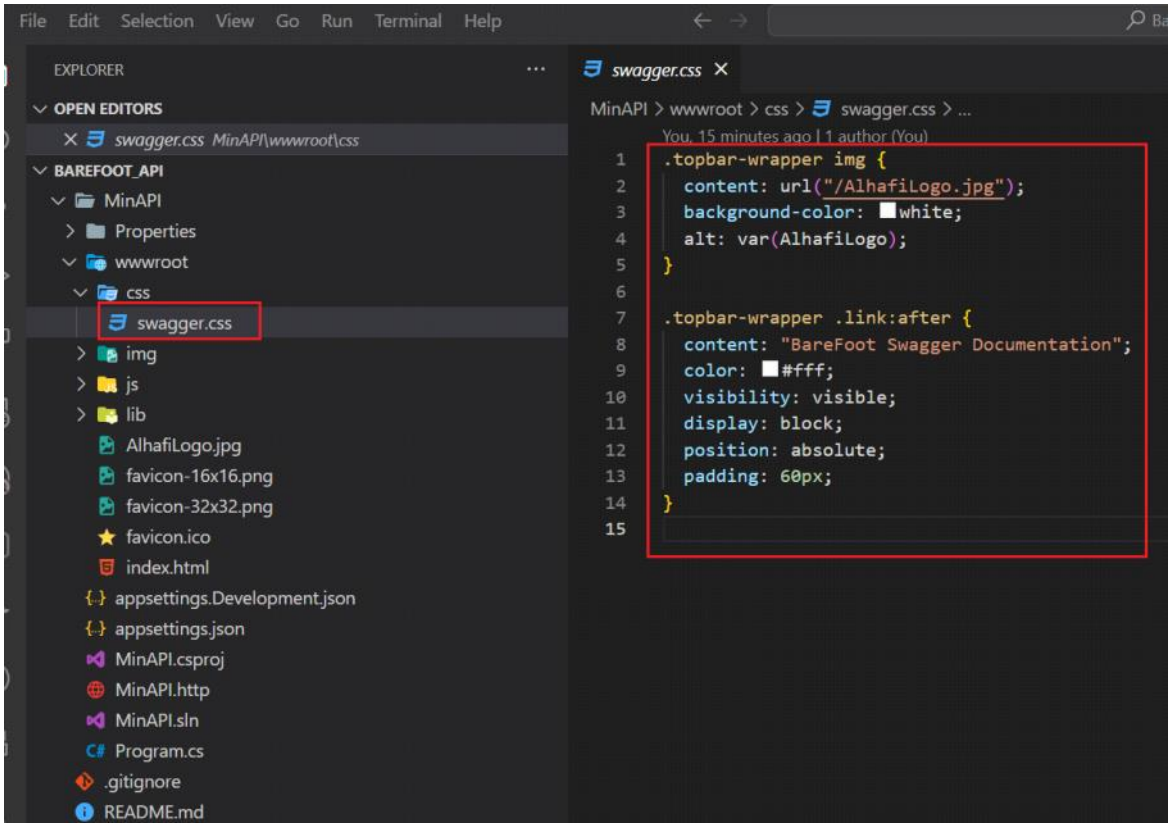
To change the favicon,
https://realfavicongenerator.net/favicon/aspnet_core?ref=halldorstefans.com

You then copy-paste your favicon into that folder. Make sure the files favicon.ico, favicon-16x16.png,
and favicon-32x32.png are included. So, you should see something like this:



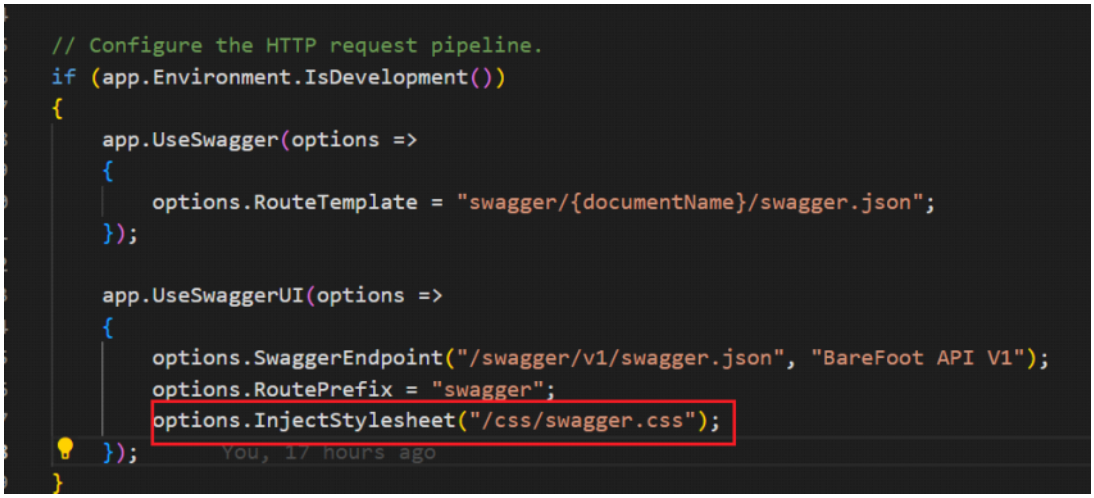
Top bar Logo

Replacing the logo at the top bar means we need to do a little bit of CSS




```
.topbar-wrapper img {
  content: url("/AlhafiLogo.jpg");
  background-color: white;
  alt: var(AlhafiLogo);
}
.topbar-wrapper .link:after {
  content: "BareFoot Swagger Documentation";
  color: #fff;
  visibility: visible;
  display: block;
  position: absolute;
  padding: 60px;
}
```

Then Let we inject CSS




```
app.UseSwagger(options =>
    {
        options.RouteTemplate = "swagger/{documentName}/swagger.json";
    });
    app.UseSwaggerUI(options =>
```

```
    {
        options.SwaggerEndpoint("/swagger/v1/swagger.json", "BareFoot API
V1");
        options.RoutePrefix = "swagger";
        options.InjectStylesheet("/css/swagger.css");
    });
```

Testing

```
PS D:\VisualCode\BareFoot_API\minapi> dotnet watch run
```

==========

Swagger Themes

From location below download your flavour theme

https://ostranme.github.io/swagger-ui-themes/

And update CSS/swager.css

==========

Let we Clean program.cs from weather API

```
app.MapGet(
        "/weatherforecast",
        () =>
        {
            var forecast = Enumerable
                .Range(1, 5)
                .Select(index => new WeatherForecast(
                    DateOnly.FromDateTime(DateTime.Now.AddDays(index)),
                    Random.Shared.Next(-20, 55),
                    summaries[Random.Shared.Next(summaries.Length)]
                ))
                .ToArray();
            return forecast;
        }
    )
    .WithName("GetWeatherForecast")
    .WithOpenApi();       You, 20 hours ago
```

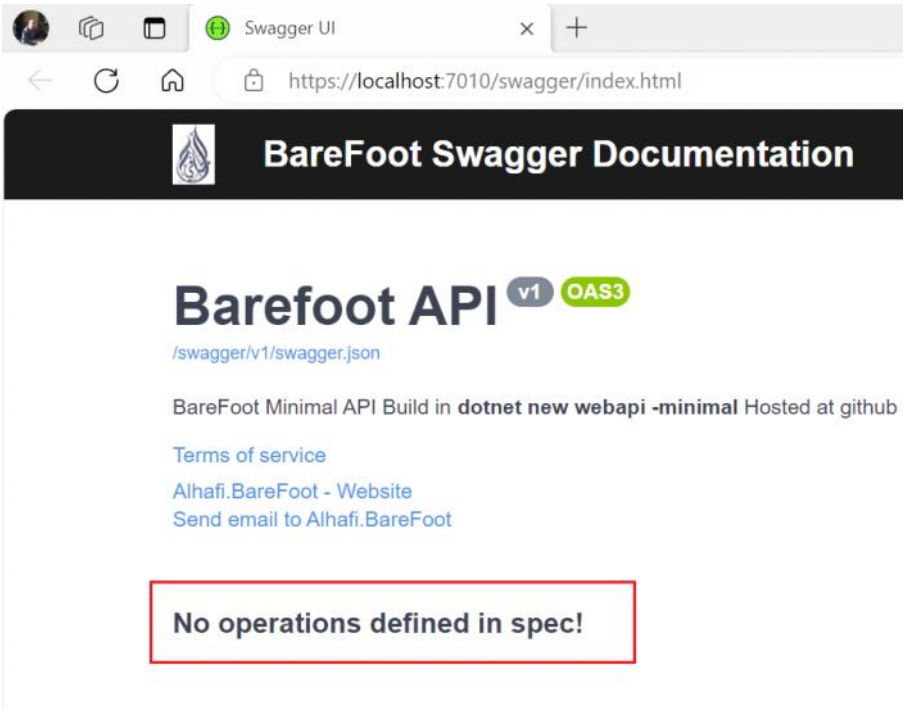
```
record WeatherForecast(DateOnly Date, int TemperatureC, string? Summary)
{
    0 references
    public int TemperatureF => 32 + (int)(TemperatureC / 0.5556);
}
```

And format you program

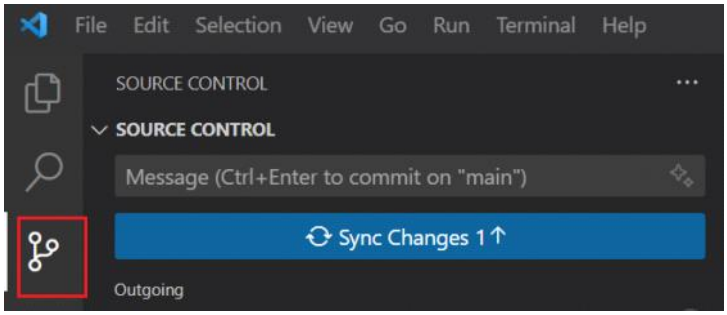
ALT+Shift+F then save

Testing

```
PS D:\VisualCode\BareFoot_API\minapi> dotnet watch run
```



Commit all changes and sync



Static Data

Sunday, August 27, 2023 12:25 PM

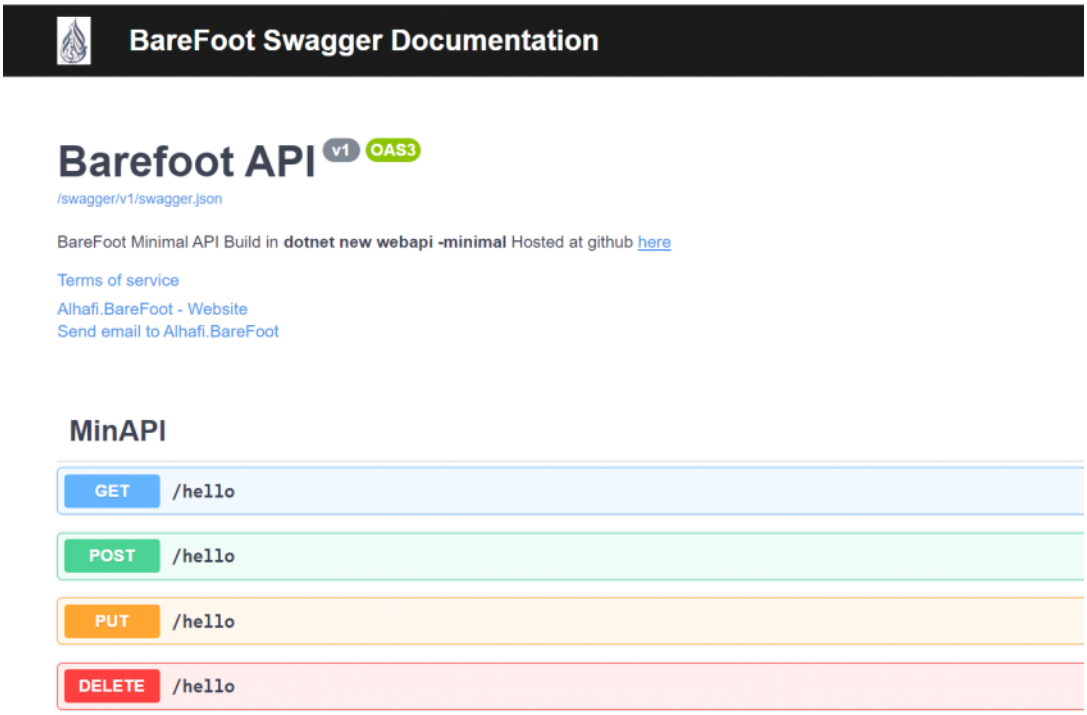
Start Up by simple static Hello endpoints API @ Program.cs

```
app.MapGet("/hello", () => "[GET] Hello World!");
app.MapPost("/hello", () => "[POST] Hello World!");
app.MapPut("/hello", () => "[PUT] Hello World!");
app.MapDelete("/hello", () => "[DELETE] Hello World!");


app.Run();
```

```
app.MapGet("/hello", () => "[GET] Hello World!");
app.MapPost("/hello", () => "[POST] Hello World!");
app.MapPut("/hello", () => "[PUT] Hello World!");
app.MapDelete("/hello", () => "[DELETE] Hello World!");
```

Testing

```
PS D:\VisualCode\BareFoot_API\minapi> dotnet watch run
```



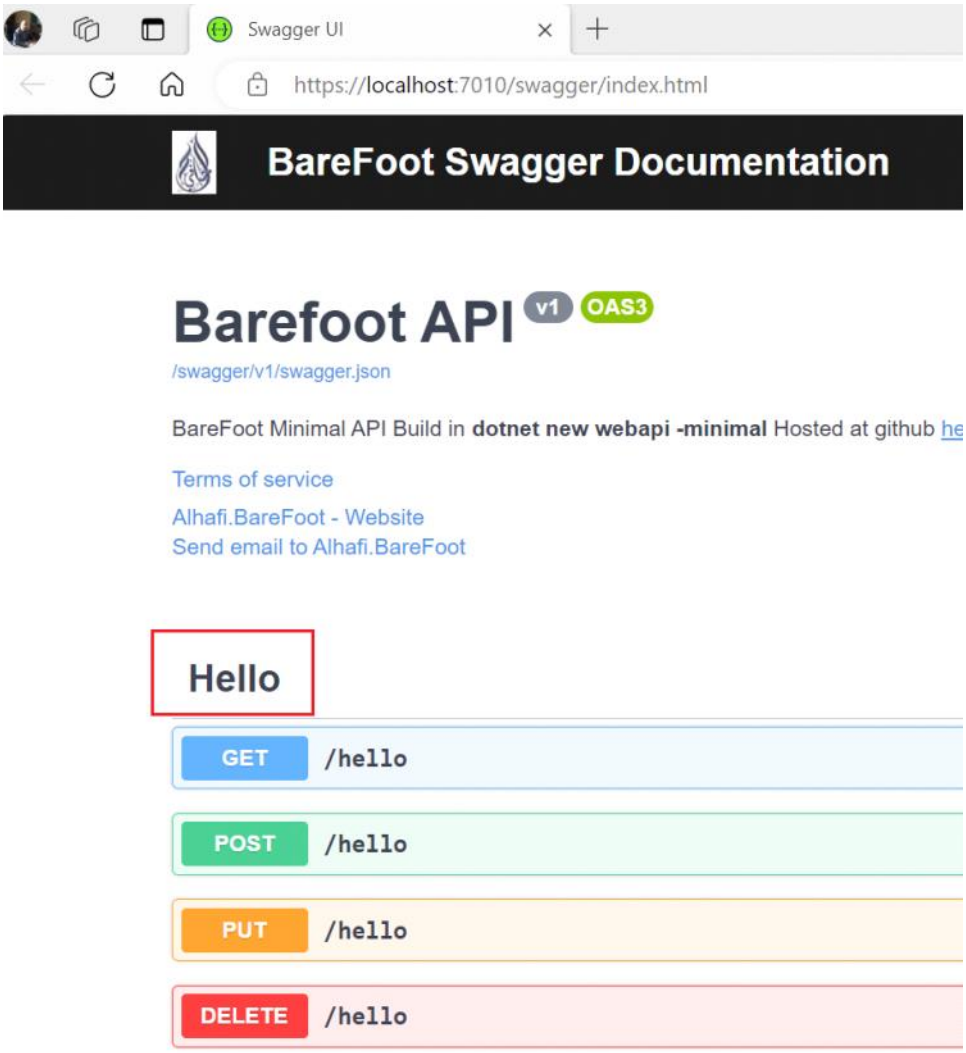
Let we tag name

```
app.MapGet("/hello", () => "[GET] Hello World!").WithTags("Hello");
app.MapPost("/hello", () => "[POST] Hello World!").WithTags("Hello");
app.MapPut("/hello", () => "[PUT] Hello World!").WithTags("Hello");
app.MapDelete("/hello", () => "[DELETE] Hello World!").WithTags("Hello");
```

```
app.MapGet("/hello", () => "[GET] Hello World!").WithTags("Hello");
app.MapPost("/hello", () => "[POST] Hello World!").WithTags("Hello");
app.MapPut("/hello", () => "[PUT] Hello World!").WithTags("Hello");
app.MapDelete("/hello", () => "[DELETE] Hello World!").WithTags("Hello");
```

Testing

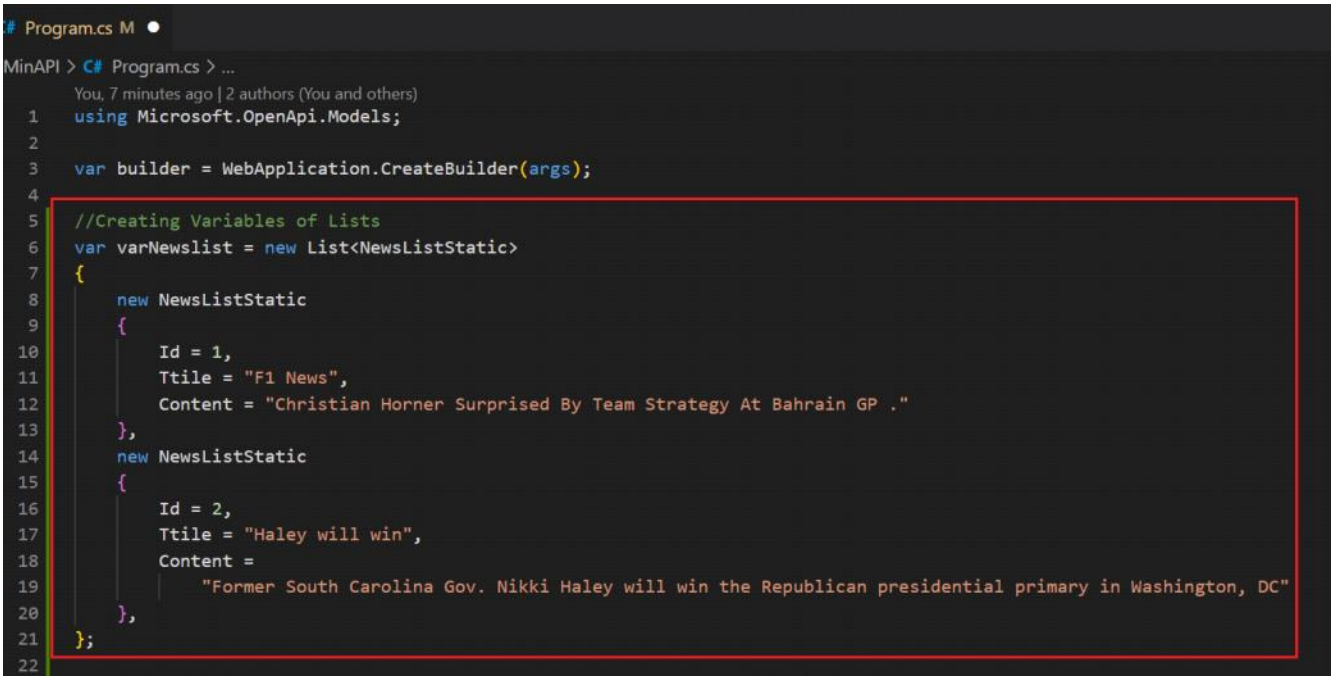
```
PS D:\VisualCode\BareFoot_API\minapi> dotnet watch run
```



Creating Static Record


```
record NewsListStatic
    {
        public int Id { get; set; }
        public string? Title { get; set; }
        public string? Content { get; set; }
    }
```

Define Variable



```
//Creating Variables of Lists
var varNewslist = new List<NewsListStatic>
{
    new NewsListStatic
    {
        Id = 1,
        Title= "F1 News",
        Content = "Christian Horner Surprised By Team Strategy At Bahrain GP ."
    },
    new NewsListStatic
    {
        Id = 2,
        Title= "Haley will win",
        Content =
            "Former South Carolina Gov. Nikki Haley will win the Republican presidential primary in
Washington, DC"
    },
};
```

Creating Endpoints

```
Program.cs 2
MinAPI > C# Program.cs > ...
83   app.MapGet("/hello", () => "[GET] Hello World!").WithTags("Hello");
84   app.MapPost("/hello", () => "[POST] Hello World!").WithTags("Hello");
85   app.MapPut("/hello", () => "[PUT] Hello World!").WithTags("Hello");
86   app.MapDelete("/hello", () => "[DELETE] Hello World!").WithTags("Hello");
87
88   //****************Static Record End Points(Data Will not save after close )****************
89   //Get All record
90   app.MapGet(
91           "/staticNews",
92           () =>       alhafi, 2 hours ago
93           {
94               return Results.Ok(varNewslist);
95           }
96       )
97       .WithTags("Static News");
98
```

```
//Get All record
app.MapGet(
        "/staticNews",
        () =>
        {
            return Results.Ok(varNewslist);
        }
    )
    .WithTags("Static News");
```

```
//Get specif Record
app.MapGet(
        "/staticNews/{id}",
        (int id) =>
        {
            var varNews = varNewslist.Find(c => c.Id == id);
            if (varNews == null)
                return Results.NotFound("Sorry this News doesn't exsists");
            return Results.Ok(varNews);
        }
    )
    .WithTags("Static News");
```

```
//Get specif Record
app.MapGet(
        "/staticNews/{id}",
```

```
        (int id) =>
        {
            var varNews = varNewslist.Find(c => c.Id == id);
            if (varNews == null)
                return Results.NotFound("Sorry this News doesn't exsists");
            return Results.Ok(varNews);
        }
    )
    .WithTags("Static News");
```

```
//Update specif Record
app.MapPut(
        "/staticNews/{id}",
        (NewsListStatic UpdateNewsListStatic, int id) =>
        {
            var varNews = varNewslist.Find(c => c.Id == id);
            if (varNews == null)
                return Results.NotFound("Sorry this command doesn't exsists");

            varNews.Title = UpdateNewsListStatic.Title;
            varNews.Content = UpdateNewsListStatic.Content;

            return Results.Ok(varNews);
        }       alhafi, 3 hours ago
    )
    .WithTags("Static News");
```

```
//Update specif Record
app.MapPut(
        "/staticNews/{id}",
        (NewsListStatic UpdateNewsListStatic, int id) =>
        {
            var varNews = varNewslist.Find(c => c.Id == id);
            if (varNews == null)
                return Results.NotFound("Sorry this command doesn't exsists");
            varNews.Title = UpdateNewsListStatic.Title;
            varNews.Content = UpdateNewsListStatic.Content;
            return Results.Ok(varNews);
        }
    )
    .WithTags("Static News");
```

```
//Add New News

app.MapPost(
        "/staticNews",
        (NewsListStatic NewNewsListStatic) =>
        {
            if (NewNewsListStatic.Id != 0 || string.IsNullOrEmpty(NewNewsListStatic.Title))
            {
                return Results.BadRequest("Invalid Id or HowTo filling");
            }
            if (
                varNewslist.FirstOrDefault(
                    c => c.Title.ToLower() == NewNewsListStatic.Title.ToLower()
                ) != null
            )
            {
                return Results.BadRequest("HowTo Exsists");
            }

            NewNewsListStatic.Id = varNewslist.OrderByDescending(c => c.Id).FirstOrDefault().Id + 1;
            varNewslist.Add(NewNewsListStatic);
            return Results.Ok(varNewslist);
        }
    )
    .WithTags("Static News");
```

```
//Add New News
```

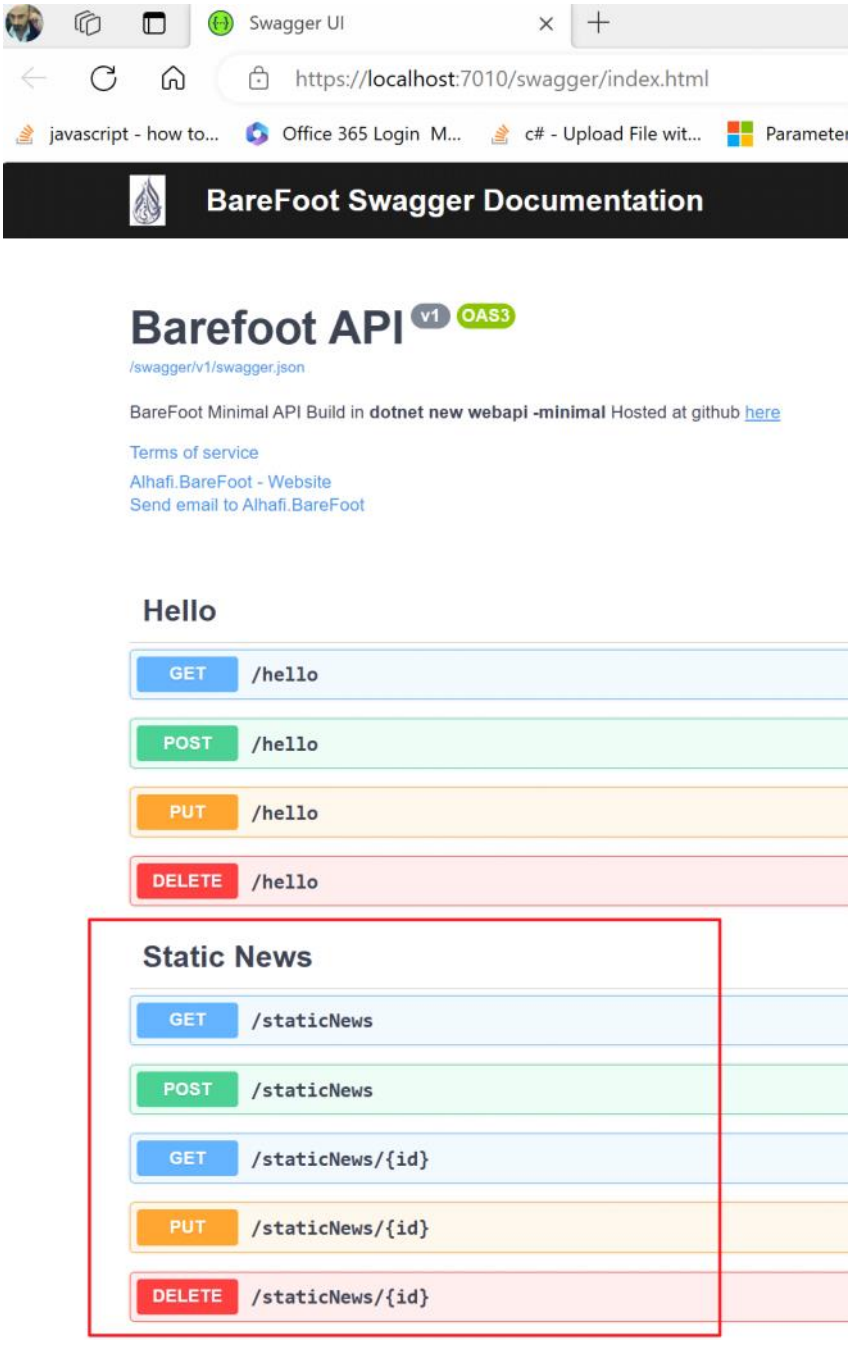
```
app.MapPost(
        "/staticNews",
        (NewsListStatic NewNewsListStatic) =>
        {
            if (NewNewsListStatic.Id != 0 || string.IsNullOrEmpty(NewNewsListStatic.Title))
            {
                return Results.BadRequest("Invalid Id or HowTo filling");
            }
            if (
                varNewslist.FirstOrDefault(
                    c => c.Title.ToLower() == NewNewsListStatic.Title.ToLower()
                ) != null
            )
            {
                return Results.BadRequest("HowTo Exsists");
            }
            NewNewsListStatic.Id = varNewslist.OrderByDescending(c => c.Id).FirstOrDefault().Id + 1;
            varNewslist.Add(NewNewsListStatic);
            return Results.Ok(varNewslist);
        }
    )
    .WithTags("Static News");
```

```
//Delete Specific News
app.MapDelete(
        "/staticNews/{id}",
        (int id) =>
        {
            var varNews = varNewslist.Find(c => c.Id == id);
            if (varNews == null)
                return Results.NotFound("Sorry this News doesn't exsists");
            varNewslist.Remove(varNews);
            return Results.Ok(varNews);
        }
    )
    .WithTags("Static News");
```

```
//Delete Specific News
app.MapDelete(
        "/staticNews/{id}",
        (int id) =>
        {
            var varNews = varNewslist.Find(c => c.Id == id);
            if (varNews == null)
                return Results.NotFound("Sorry this News doesn't exsists");
            varNewslist.Remove(varNews);
            return Results.Ok(varNews);
        }
    )
    .WithTags("Static News");
```

Testing

```
PS D:\VisualCode\BareFoot_API\minapi> dotnet watch run
```



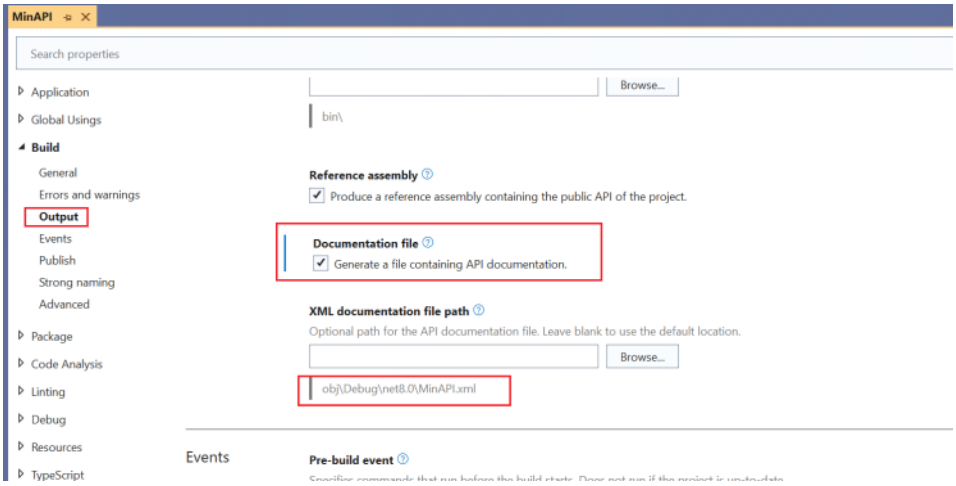
==========

Enhance Swagger Documentation

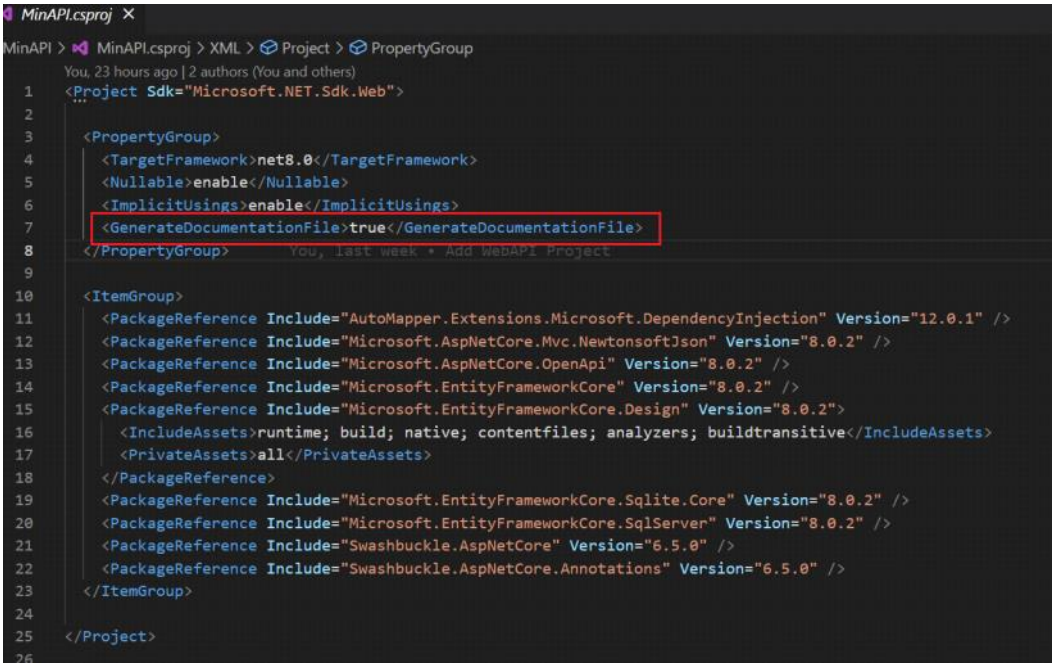
Enhance Swagger Documentation with Annotations in ASP.NET core | by Nitesh Singhal | Medium

By Generating XML Documentation Files

Either by Project setting

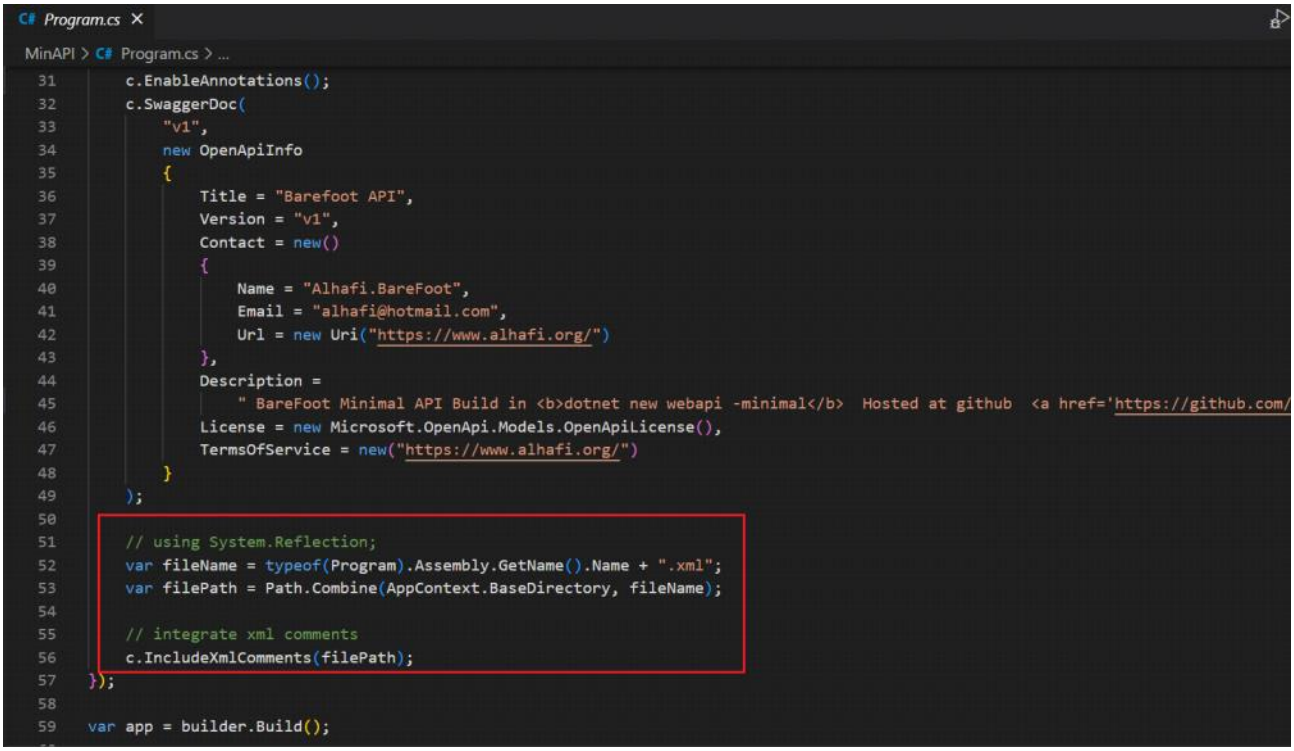


Or By editing project




```
<GenerateDocumentationFile>true</GenerateDocumentationFile>
```

Reflection will be generate in xml




```
// using System.Reflection;
    var fileName = typeof(Program).Assembly.GetName().Name + ".xml";
    var filePath = Path.Combine(AppContext.BaseDirectory, fileName);
    // integrate xml comments
    c.IncludeXmlComments(filePath);
```

is a feature of .NET that allows you to document your code inline with detailed information and later pull that information out into reusable XML files. For example, if you've ever used a /// in your codebase, then you've used this feature. For this case, you'll be using XML documentation to document our OpenAPI schema and detail information about our endpoints.

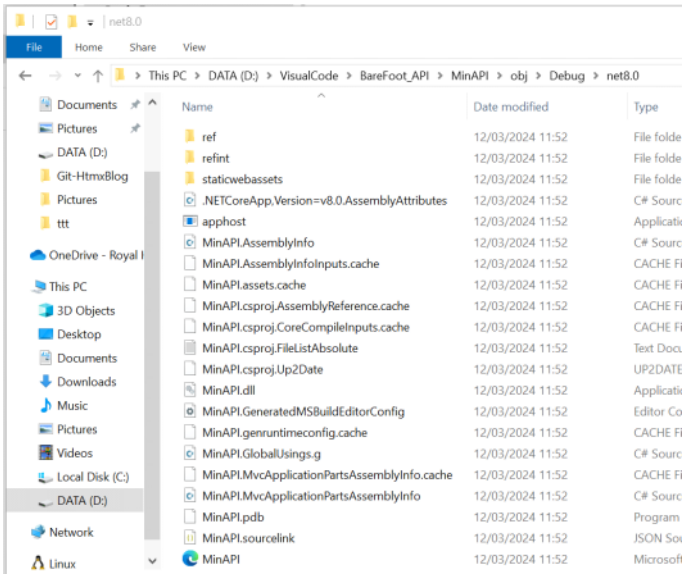
Let We Document Fields in Models

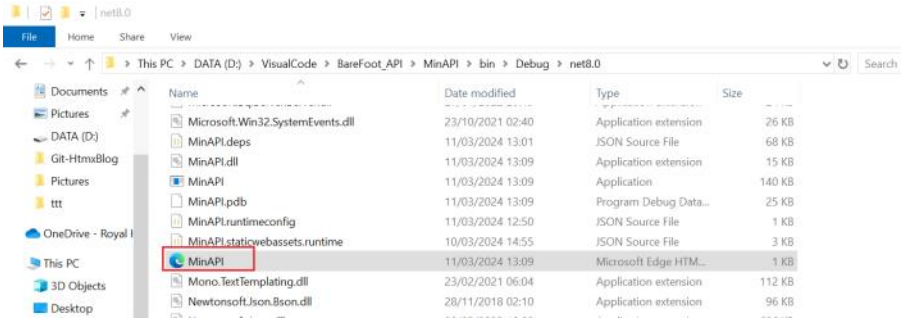
```
C# Program.cs ×
MinAPI > C# Program.cs > ...
205  //*************************************************************************************************
206
207  app.Run();
208
     You, 6 minutes ago | 1 author (You) | 5 references
209  record NewsListStatic
210  {
         4 references
211      public int Id { get; set; }
212
213      /// <summary>
214      /// Title is Any headline news
215      /// </summary>
216      /// <example>Formula One World Championship</example>
         5 references
217      public string? Title { get; set; }
218
219      /// <summary>
220      /// Contenty contain details of the news
221      /// </summary>
222      /// <example>
223      ///  2024 FIA Formula One World Championship is a motor racing championship
224      ///  for Formula One cars and is the 75th running of the Formula One World Championship.
225      /// </example>
         3 references
226      public string? Content { get; set; }
227  }
```

Testing

```
PS D:\VisualCode\BareFoot_API\minapi> dotnet watch run
```

Now Searching xml in obj or bin folder





Open this XML , So Our generation work well

MinAPI.xml

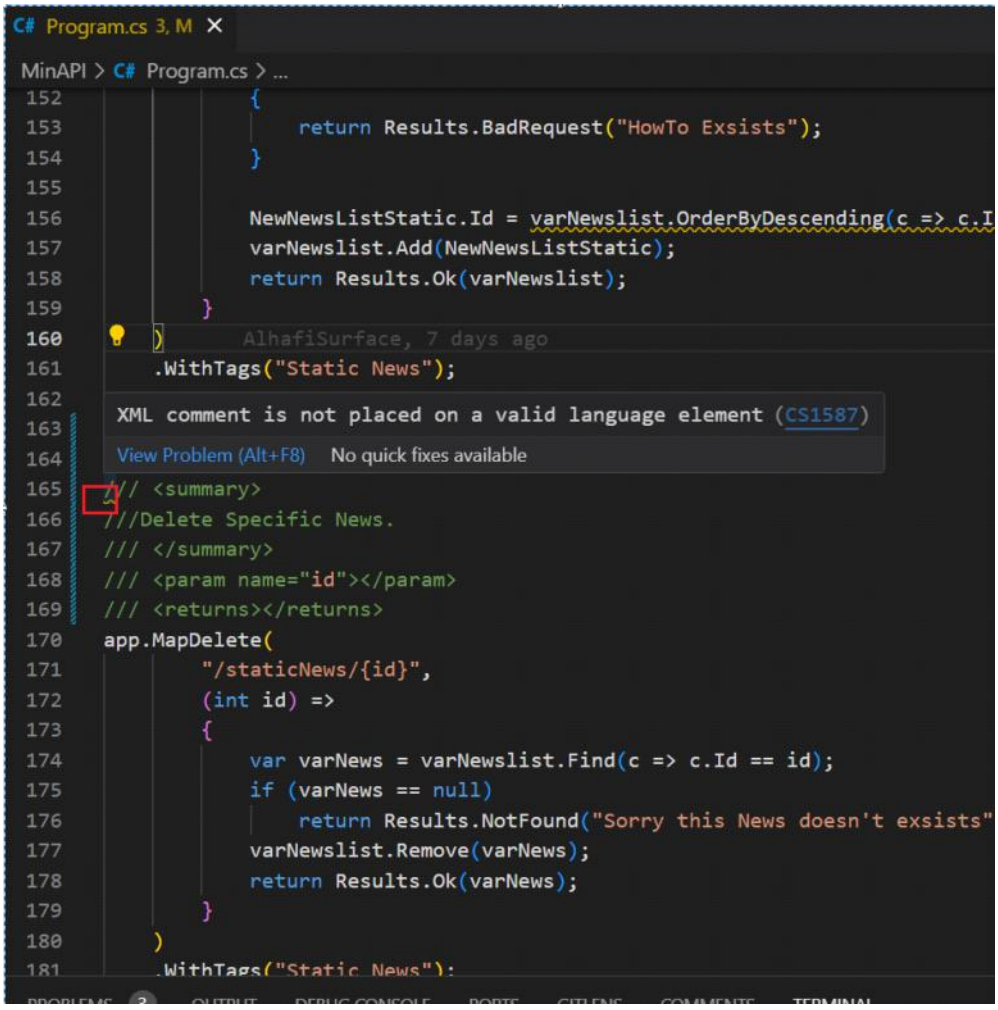
```xml
<?xml version="1.0"?>
<doc>
    <assembly>
        <name>MinAPI</name>
    </assembly>
    <members>
        <member name="P:NewsListStatic.Title">
            <summary>
            Title is Any headline news
            </summary>
            <example>Formula One World Championship</example>
        </member>
        <member name="P:NewsListStatic.Content">
            <summary>
            Contenty contain details of the news
            </summary>
            <example>
             2024 FIA Formula One World Championship is a motor racing championship
             for Formula One cars and is the 75th running of the Formula One World Championship
            </example>
        </member>
    </members>
</doc>
```

Let We Document End Point by adding ///

```
/// <summary>
///Delete Specific News.
/// </summary>
/// <param name="id"></param>
/// <returns></returns>
app.MapDelete(
        "/staticNews/{id}",
        (int id) =>
        {
            var varNews = varNewslist.Find(c => c.Id == id);
            if (varNews == null)
                return Results.NotFound("Sorry this News doesn't exsists");
            varNewslist.Remove(varNews);
            return Results.Ok(varNews);
        }
    )
    .WithTags("Static News");
```

BLEMS 3 OUTPUT DEBUG CONSOLE PORTS GITLENS COMMENTS TERMINAL

Some error shows



Let we disable errors 1587

MinAPI.csproj

MinAPI > MinAPI.csproj > XML > Project > PropertyGroup > NoWarn

You, 1 minute ago | 2 authors (You and others)

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net8.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
    <GenerateDocumentationFile>true</GenerateDocumentationFile>
    <NoWarn>$(NoWarn);1591;1587</NoWarn>        You, 1 minute ago
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="AutoMapper.Extensions.Microsoft.DependencyInjection" Version="12.0.1" />
    <PackageReference Include="Microsoft.AspNetCore.Mvc.NewtonsoftJson" Version="8.0.2" />
    <PackageReference Include="Microsoft.AspNetCore.OpenApi" Version="8.0.2" />
    <PackageReference Include="Microsoft.EntityFrameworkCore" Version="8.0.2" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.Design" Version="8.0.2">
      <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
      <PrivateAssets>all</PrivateAssets>
    </PackageReference>
    <PackageReference Include="Microsoft.EntityFrameworkCore.Sqlite.Core" Version="8.0.2" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="8.0.2" />
    <PackageReference Include="Swashbuckle.AspNetCore" Version="6.5.0" />
    <PackageReference Include="Swashbuckle.AspNetCore.Annotations" Version="6.5.0" />
  </ItemGroup>

</Project>

```

```
<NoWarn>$(NoWarn);1591;1587</NoWarn>
```

Error Notes gone

```
/// <summary>
///Delete Specific N
/// </summary>
/// <param name="id"
```

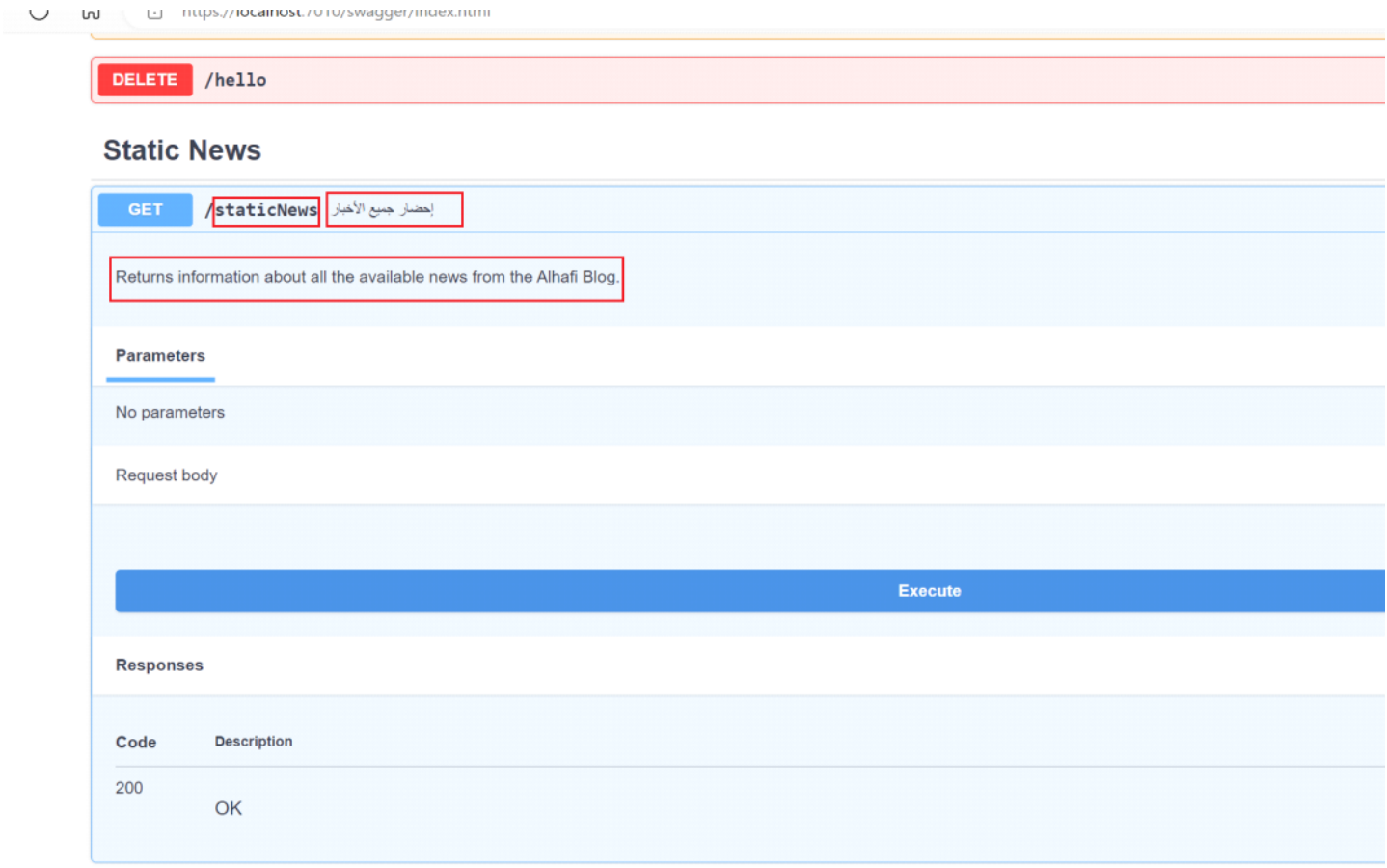
```
/// <summary>
///Delete Specific News.
/// </summary>
/// <param name="id"></param>
/// <returns></returns>
```

Unfortunately XML generation not working perfect Minimal API , So let we use Open Api swagger documentation

```
app.MapGet(
        "/staticNews",
        () =>
        {
            return Results.Ok(varNewslist);
        }
    )
    .WithOpenApi(x => new OpenApiOperation(x)
    {
        Summary = "إحضار جميع الأخبار!",
        Description = "Returns information about all the available news from the Alhafi Blog.",
        Tags = new List<OpenApiTag> { new() { Name = "Static News" } }
    });
```

```
app.MapGet(
        "/staticNews",
        () =>
        {
            return Results.Ok(varNewslist);
        }
    )
```

```
    .WithOpenApi(x => new OpenApiOperation(x)
    {
        Summary = "إحضار جميع الأخبار",
        Description = "Returns information about all the available news from the Alhafi Blog.",
        Tags = new List<OpenApiTag> { new() { Name = "Static News" } }
    });
```



Or Another Shape

```
app.MapGet(
        "/staticNews/{id}",
        (int id) =>
        {
            var varNews = varNewslist.Find(c => c.Id == id);
            if (varNews == null)
                return Results.NotFound("Sorry this News doesn't exsists");
            return Results.Ok(varNews);
        }
    )
    .WithDescription("return one news ")
    .WithSummary("إحضار خبر واحد بناء على قيمة رقم السجل")
    .WithName("GetStaticNewsbyID")
    .WithTags("Static News")
    .WithOpenApi();
```

```
app.MapGet(
        "/staticNews/{id}",
        (int id) =>
        {
            var varNews = varNewslist.Find(c => c.Id == id);
            if (varNews == null)
                return Results.NotFound("Sorry this News doesn't exsists");
            return Results.Ok(varNews);
        }
    )
    .WithDescription("return one news ")
    .WithSummary("إحضار خبر واحد بناء على قيمة رقم السجل")
```

```
.WithName("GetStaticNewsbyID")
.WithTags("Static News")
.WithOpenApi();
```

Dynamic Data -ContextDB

11 February 2024 10:52

Understanding Minimal API by using Class Model (Direct Data Access with DB context)

Adding Packaging

Databases

Microsoft.EntityFrameworkCore by Microsoft

Microsoft.EntityFrameworkCore. Design by Microsoft

SQLite

Microsoft.EntityFrameworkCore.Sqlite.Core by Microsoft

MSSQL

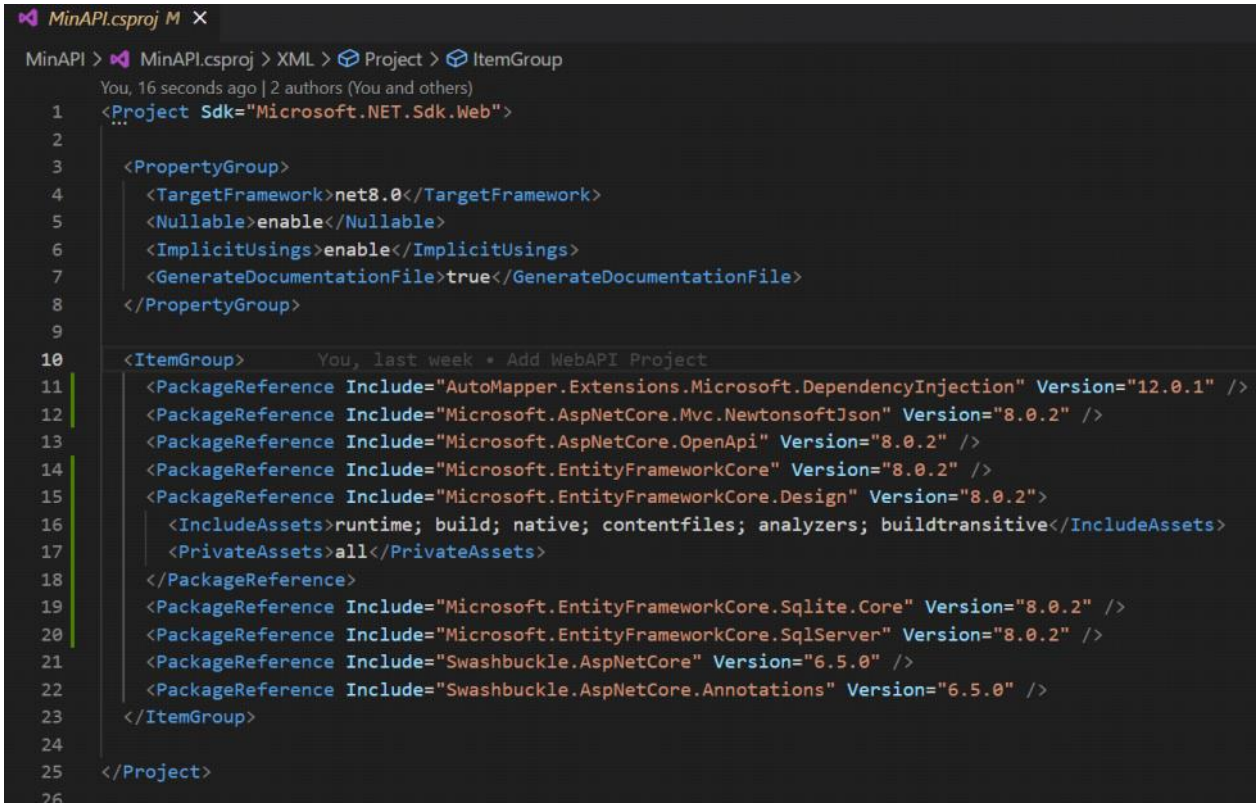
Microsoft.EntityFrameworkCore.SqlServer by Microsoft

Utilities

Microsoft.AspNetCore.Mvc.NewtonsoftJson by Microsoft

AutoMapper.Extensions.Microsoft.DependencyInjection by Microsoft

Our project installed packages



Approaches to communicate to Databases

1) Indirect Access to database Using API

A) Code First

The Code First approach is a more traditional approach to building APIs, with the development of code happening after the business requirements are laid out, eventually generating the documentation from the code. The Design First approach advocates for designing the API's contract first before writing any code.

B) Database First

Development using Entity Framework. Database First allows you to reverse engineer a model from an existing database. The model is stored in an EDMX file (.edmx extension) and can be viewed and edited in the Entity Framework Designer. The classes that you interact with in your application are automatically generated from the EDMX file.

https://learn.microsoft.com/en-us/ef/ef6/modeling/designer/workflows/database-first

1. **Abstraction**: APIs provide an abstraction layer that decouples your application from the underlying database implementation. This can make it easier to switch databases or scale your system in the future.
2. **Security**: APIs can enforce access control and provide a secure way to interact with the database, reducing the risk of exposing sensitive data or opening up security vulnerabilities.
3. **Scalability**: APIs can help improve the scalability of your system by allowing you to introduce caching, load balancing, and other optimizations at the API layer.
4. **Versioning and Documentation**: APIs can provide clear interfaces with versioning and documentation, making it easier for developers to understand how to interact with the database.

Direct Access to database

Direct database access lets client applications connect directly with databases and request information. For example, people using mobile apps designed to provide driving directions might ask for instructions to reach a specific destination. The app would send the query to a database, which would return real-time driving directions to the user.
Most of the work takes place in the backend, so users never see the underlying processes that deliver information to their apps. It's a relatively straightforward approach to client-server connections that can produce results quickly.

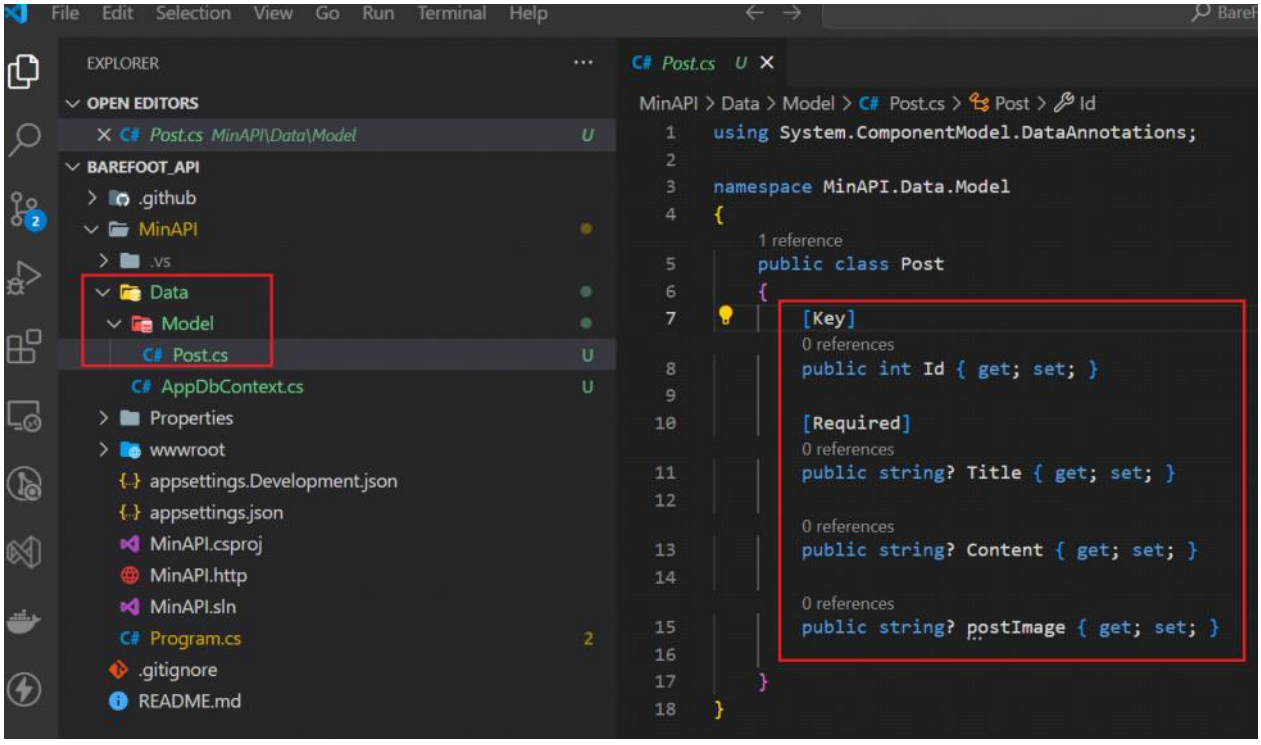
5. **Performance**: Direct database access can be faster and more efficient than going through an API layer, as it eliminates the overhead of serialization, deserialization, and network communication involved in API calls.
6. **Complexity**: Direct database access can simplify the architecture of your application by removing an additional layer (the API) between your application and the database.
7. **Control**: Direct access gives you more control over the queries and transactions executed on the database. This can be beneficial if you need to optimize specific queries or leverage advanced database features.
8. **Security**: Direct database access requires careful handling of security concerns such as SQL injection attacks. You need to ensure that your application is secure and that sensitive data is properly protected.

API Access:

In general, if you are building a simple application with straightforward database interactions and performance is a critical factor, direct database access may be a suitable choice. However, if you anticipate the need for scalability, flexibility, security, or abstraction from the underlying database, using an API can offer several advantages.

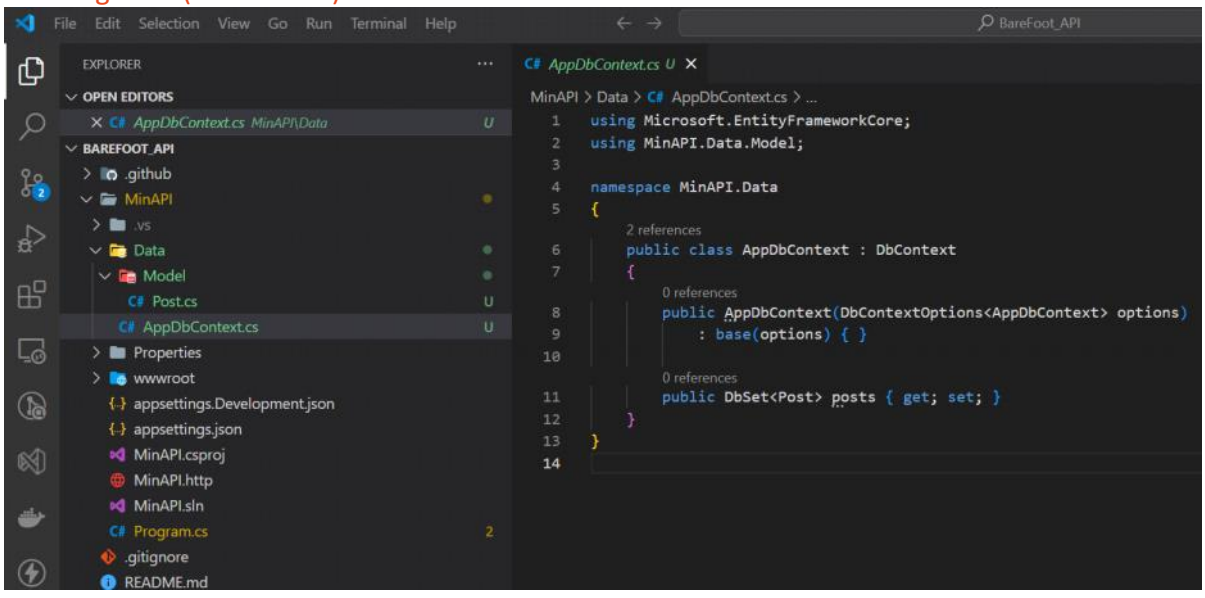
Our Target Using Indirect Access to database API & First Code

Creating Model




```
using System.ComponentModel.DataAnnotations;
namespace MinAPI.Data.Model
{
    public class Post
    {
        [Key]
        public int Id { get; set; }
        [Required]
        public string? Title { get; set; }
        public string? Content { get; set; }
        public string? postImage { get; set; }
    }
}
```

Creating Data (DB Context)



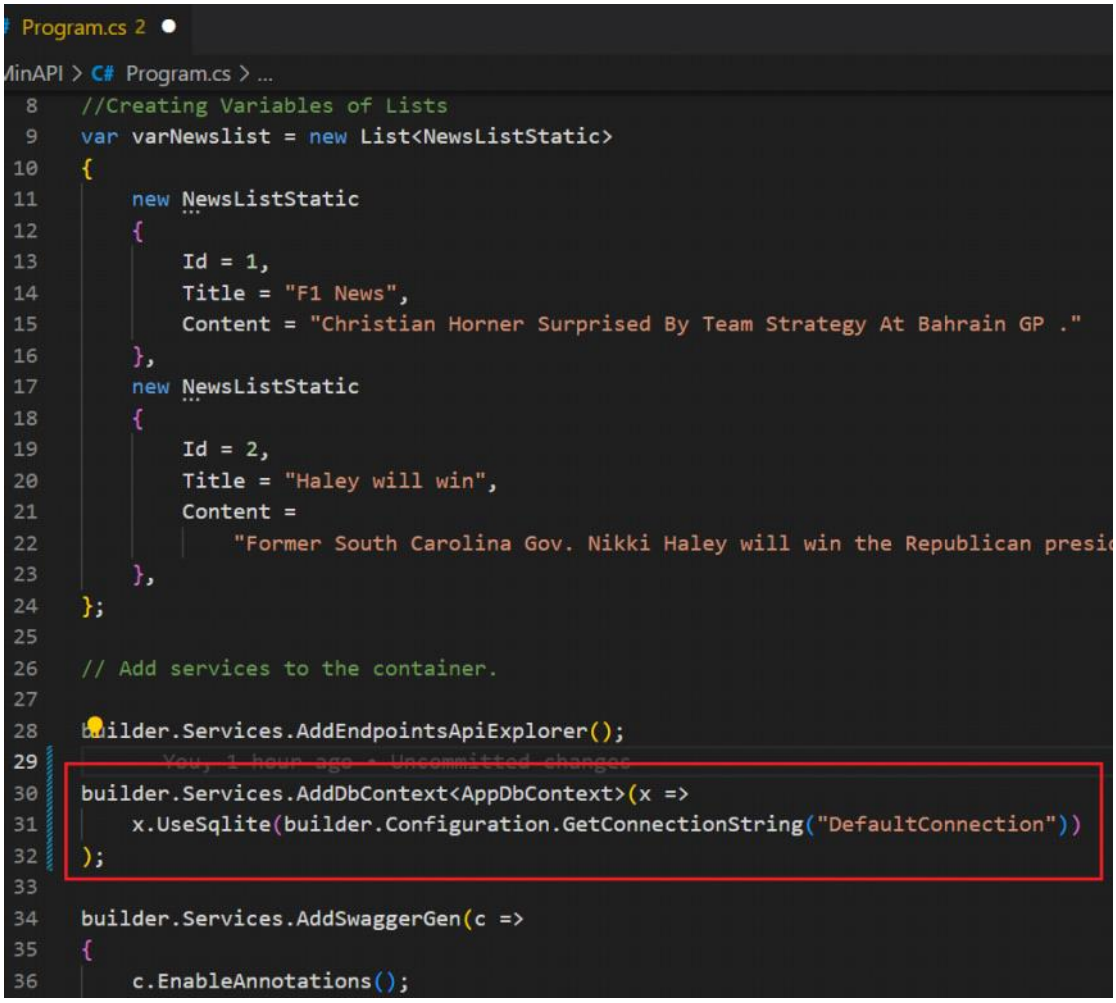

```
using Microsoft.EntityFrameworkCore;
using MinAPI.Data.Model;
namespace MinAPI.Data
{
    public class AppDbContext : DbContext
```

```
    {
        public AppDbContext(DbContextOptions<AppDbContext> options)
            : base(options) { }
        public DbSet<Post> Posts { get; set; }
    }
}
```

Adding Connection String to the Database

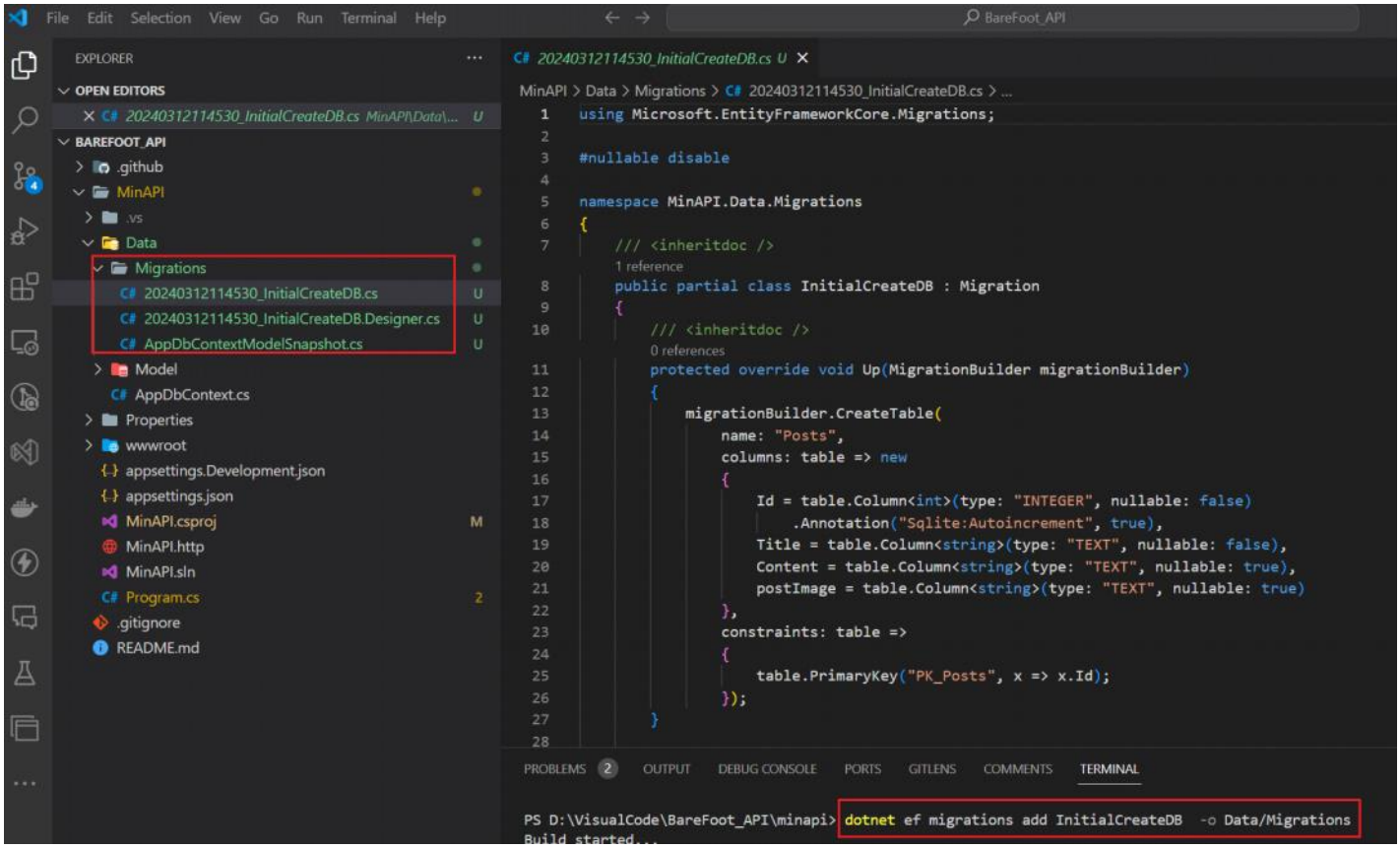

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "ConnectionStrings": {
    "DefaultConnection": "Data source=AlhafiNewsPaper.db"
  },
  "AllowedHosts": "*"
}
```

Define Connection & Register DBConext as Service




```
builder.Services.AddDbContext<AppDbContext>(x =>
    x.UseSqlite(builder.Configuration.GetConnectionString("DefaultConnection"))
);
```

Migration Database



[EF Core tools reference (.NET CLI) - EF Core | Microsoft Learn]

Update utilities

dotnet tool update --global dotnet-ef

Add Migration

dotnet ef migrations add InitialCreateDB -o Data/Migrations

EF-Migration Tips

- dotnet ef migrations add
- dotnet ef migrations remove
- dotnet ef migrations remove -force
- dotnet ef migrations List
- dotnet ef database update
- dotnet ef database update MyFirstMigration

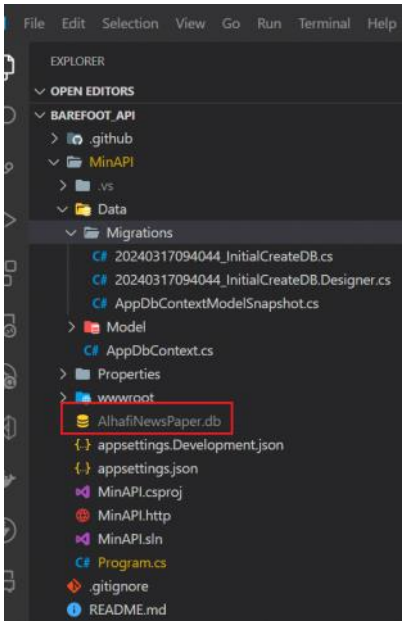
List Migration

```
PS D:\VisualCode\BareFoot_API\minapi> dotnet ef migrations list
Build started...
Build succeeded.
20240317094044_InitialCreateDB (Pending)
```

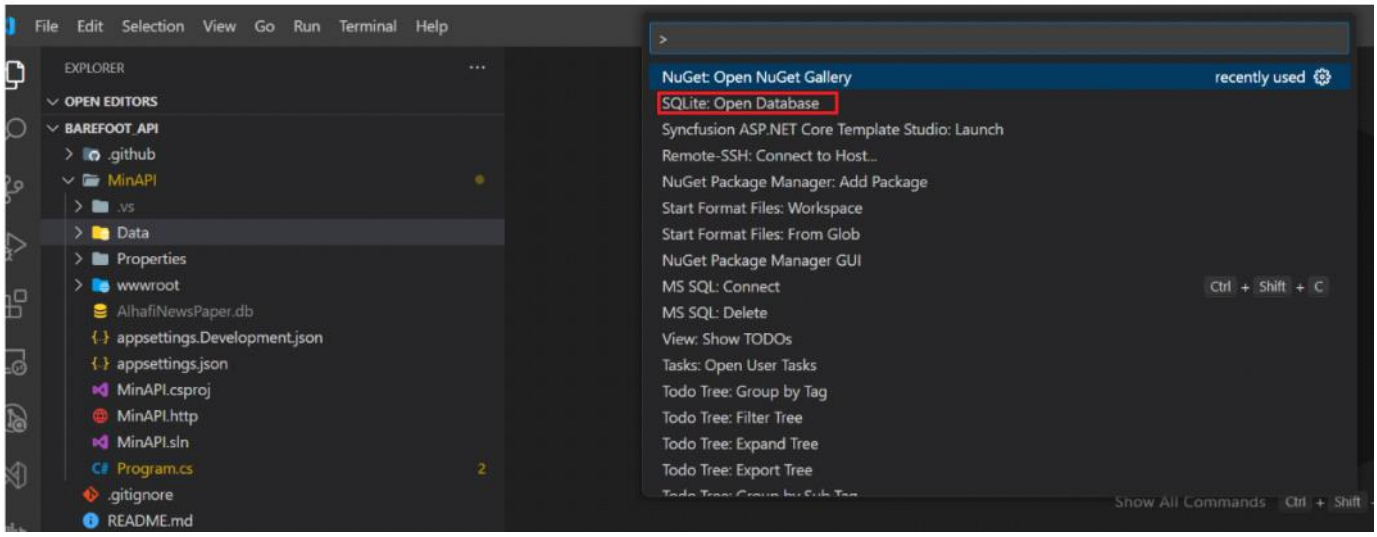
Creating Database

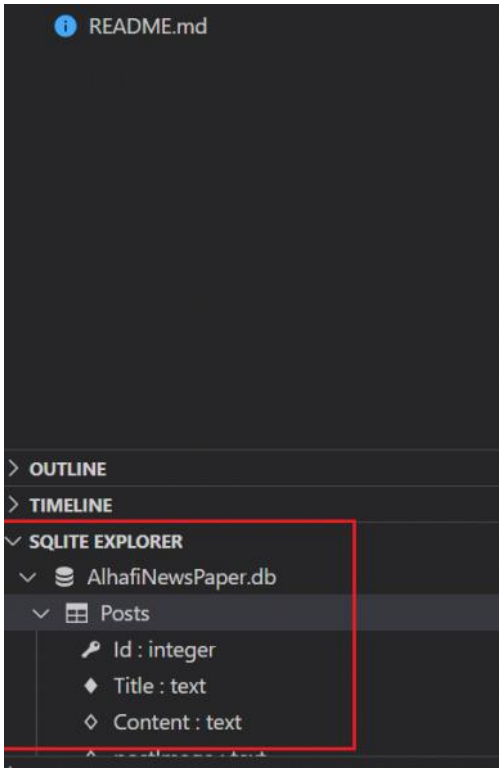
dotnet ef database update

Check your DB


File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
BAREFOOT_API
.github
MinAPI
.vs
Data
Migrations
20240317094044_InitialCreateDB.cs
20240317094044_InitialCreateDB.Designer.cs
AppDbContextModelSnapshot.cs
Model
AppDbContext.cs
Properties
wwwroot
AlhafiNewsPaper.db
appsettings.Development.json
appsettings.json
MinAPI.csproj
MinAPI.http
MinAPI.sln
Program.cs
.gitignore
README.md

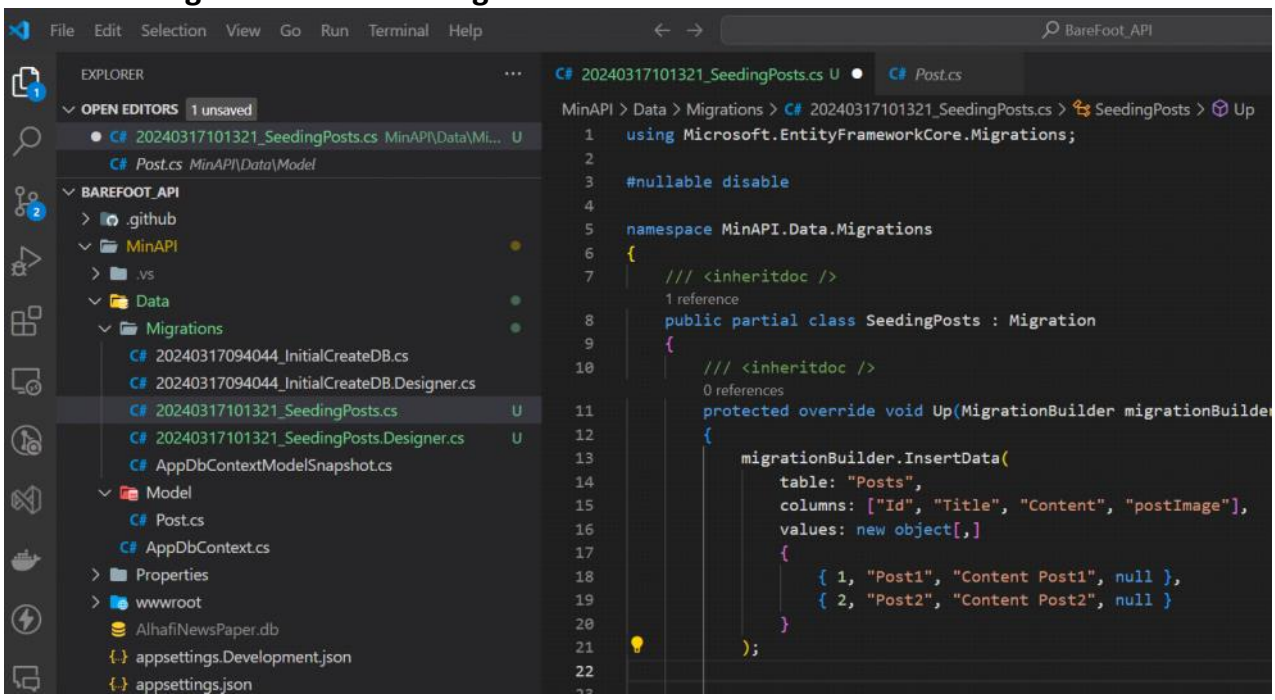
CTRL+Shift+P


File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
BAREFOOT_API
.github
MinAPI
.vs
Data
Properties
wwwroot
AlhafiNewsPaper.db
appsettings.Development.json
appsettings.json
MinAPI.csproj
MinAPI.http
MinAPI.sln
Program.cs
.gitignore
README.md
>
NuGet: Open NuGet Gallery
recently used
SQLite: Open Database
Syncfusion ASP.NET Core Template Studio: Launch
Remote-SSH: Connect to Host...
NuGet Package Manager: Add Package
Start Format Files: Workspace
Start Format Files: From Glob
NuGet Package Manager GUI
MS SQL: Connect
Ctrl + Shift + C
MS SQL: Delete
View: Show TODOs
Tasks: Open User Tasks
Todo Tree: Group by Tag
Todo Tree: Filter Tree
Todo Tree: Expand Tree
Todo Tree: Export Tree
Show All Commands
Ctrl + Shift


README.md
OUTLINE
TIMELINE
SQLITE EXPLORER
AlhafiNewsPaper.db
Posts
Id : integer
Title : text
Content : text

Seeding stage migration

dotnet ef migrations add SeedingPosts



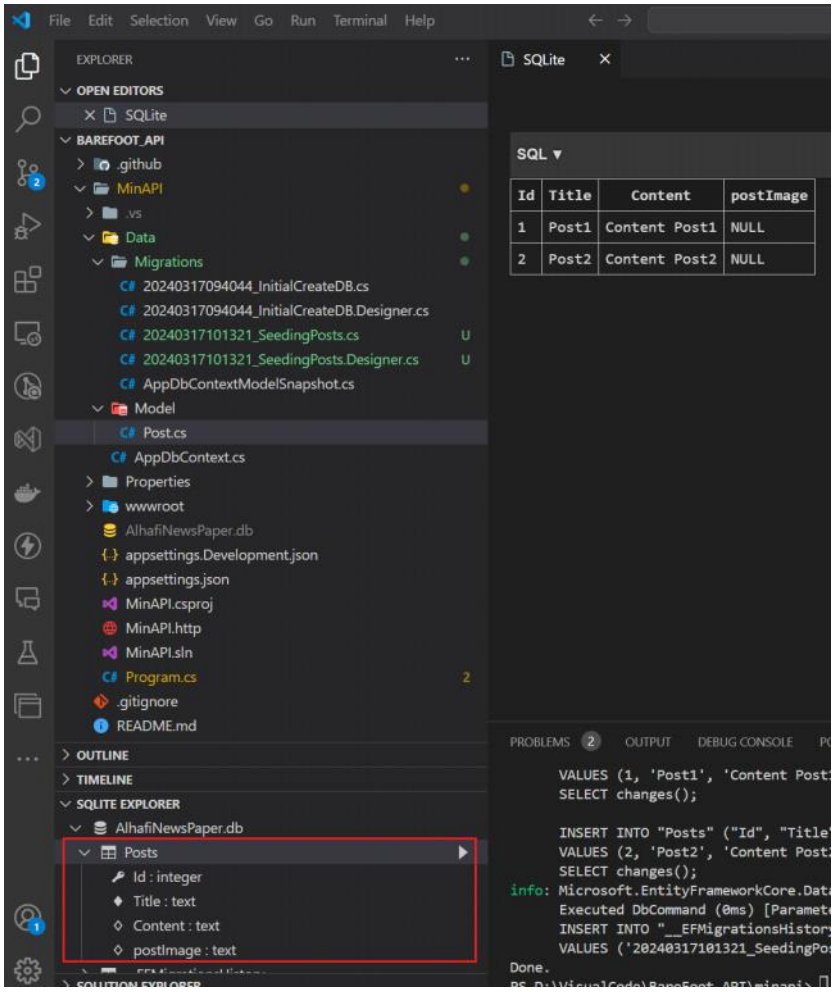
Seeding Data

```
migrationBuilder.InsertData(
                table: "Posts",
                columns: ["Id", "Title", "Content", "postImage"],
                values: new object[,]
                {
                    { 1, "Post1", "Content Post1", null },
                    { 2, "Post2", "Content Post2", null }
                }
            );
```

Update database

dotnet ef database update

Check database

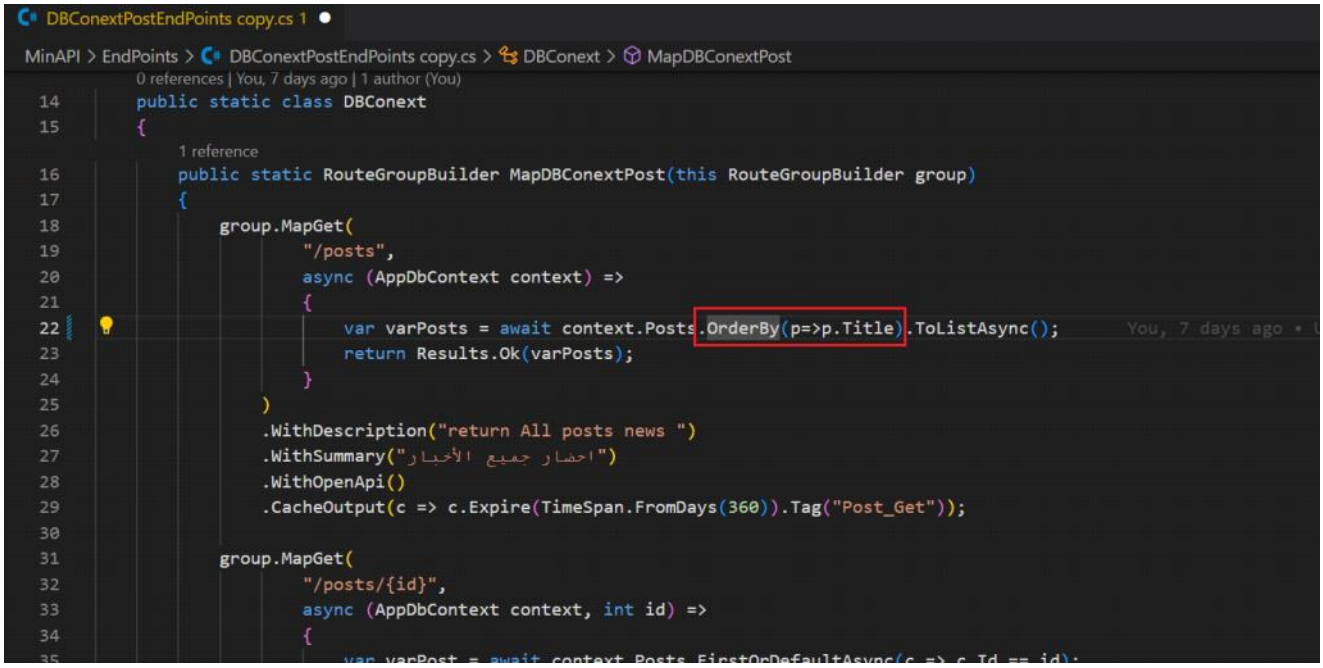


Now our End point API

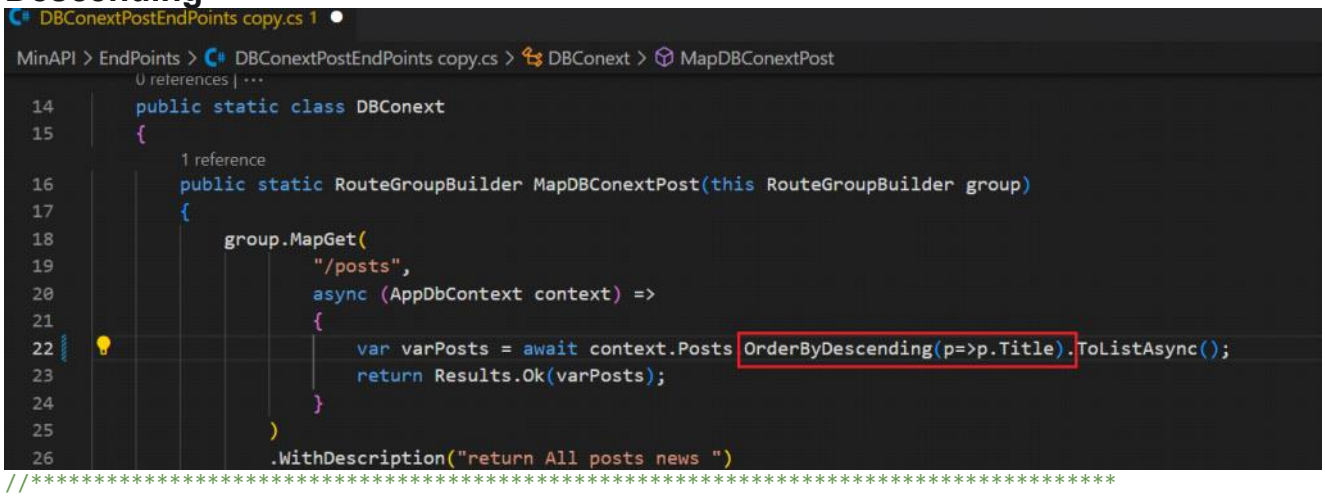
```
//************************Dynamic Data -Repository- Data Access(DB Context)
app.MapGet(
        "/dbcontext/posts",
        async (AppDbContext context) =>
        {
            var varPosts= await context.Posts.ToListAsync();
            return Results.Ok(varPosts);
        }
    )
    .WithDescription("return All posts news ")
    .WithSummary("احضار جميع الأخبار")
    .WithTags("DBContext")
    .WithOpenApi();


app.MapGet(
        "/dbcontext/posts/{id}",
        async (AppDbContext context, int id) =>
        {
            var varPost = await context.Posts.FirstOrDefaultAsync(c => c.Id == id);
            if (varPost != null)
            {
                return Results.Ok(varPost);
            }
            return Results.NotFound();
        }
    )
    .WithDescription("return Only One Post News")
    .WithSummary("احضار خبر واحد ")
    .WithTags("DBContext")
    .WithOpenApi();


app.MapPost(
```

```
        "/dbcontext/posts/{id}",
        async (AppDbContext context, Post poss) =>
        {
            await context.Posts.AddAsync(poss);
            await context.SaveChangesAsync();
            return Results.Created($"/posts/{poss.Id}", poss);
        }
    )
    .WithDescription("Insert New Post News")
    .WithSummary("ادخال خبر جديد ")
    .WithTags("DBContext")
    .WithOpenApi();


app.MapPut(
        "/dbcontext/posts/{id}",
        async (AppDbContext context, int id, Post poss) =>
        {
            var varPost = await context.Posts.FirstOrDefaultAsync(c => c.Id == id);
            if (varPost != null)
            {
                varPost.Title = poss.Title;
                varPost.Content = poss.Content;
                varPost.postImage = poss.postImage;
                await context.SaveChangesAsync();
                return Results.NoContent();
            }
            return Results.NotFound();
        }
    )
    .WithDescription("Update Post News")
    .WithSummary("تعديل خبر  ")
    .WithTags("DBContext")
    .WithOpenApi();

app.MapDelete(
        "/dbcontext/posts/{id}",
        async (AppDbContext context, int id) =>
        {
            var varPost = await context.Posts.FirstOrDefaultAsync(c => c.Id == id);
            if (varPost != null)
            {
                context.Posts.Remove(varPost);
                await context.SaveChangesAsync();
                return Results.NoContent();
            }
            return Results.NotFound();
        }
    )
    .WithDescription("Delete  Post ")
    .WithSummary("حذف خبر  ")
    .WithTags("DBContext")
    .WithOpenApi();

//*****************************************************************************************
```

We can add Ordering by
Ascending



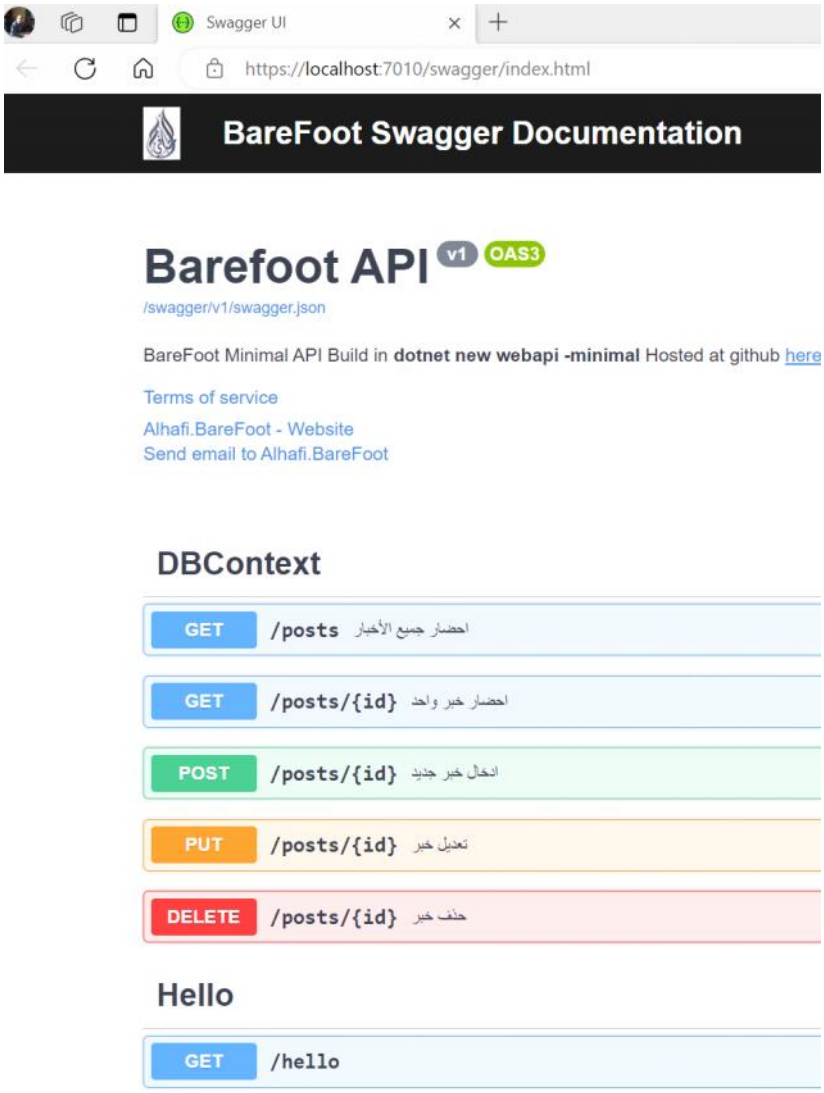
Descending




```
//*************************************************************************************
```

Testing

```
PS D:\VisualCode\BareFoot_API\minapi> dotnet watch run
```

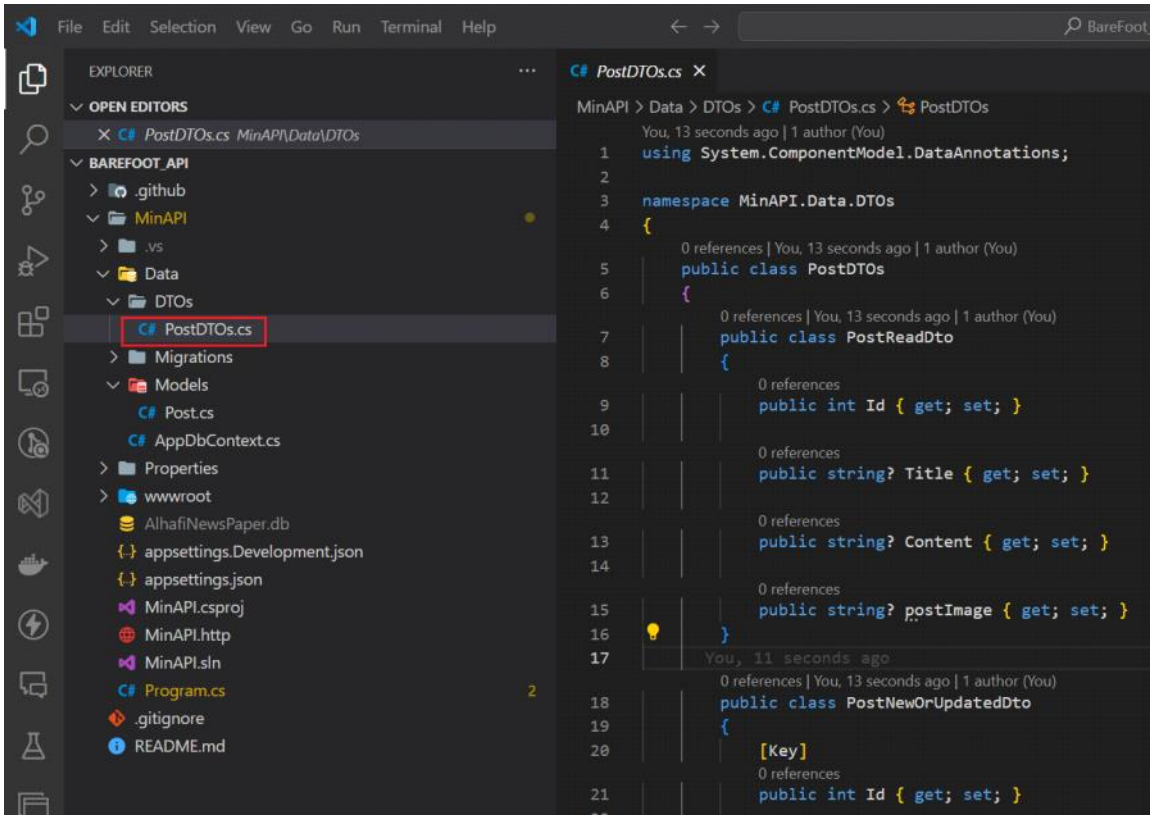

Swagger UI
https://localhost:7010/swagger/index.html
BareFoot Swagger Documentation
Barefoot API
v1
OAS3
/swagger/v1/swagger.json
BareFoot Minimal API Build in dotnet new webapi -minimal Hosted at github here
Terms of service
Alhafi.BareFoot - Website
Send email to Alhafi.BareFoot
DBContext
GET /posts احضار جميع الأخبار
GET /posts/{id} احضار خبر واحد
POST /posts/{id} ادخال خبر جديد
PUT /posts/{id} تعديل خبر
DELETE /posts/{id} حذف خبر
Hello
GET /hello

Dynamic Data -AutoMapper

11 February 2024 12:53

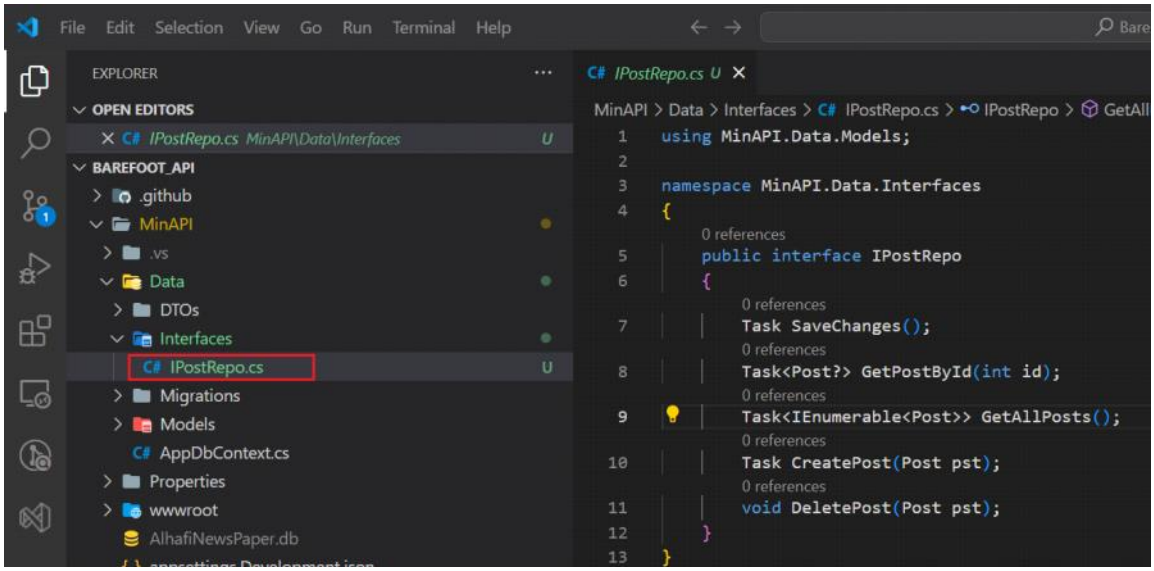
Understanding Minimal API by using Class Model (indirect Data Access with DTO's and AutoMapper)

Creating DTOS



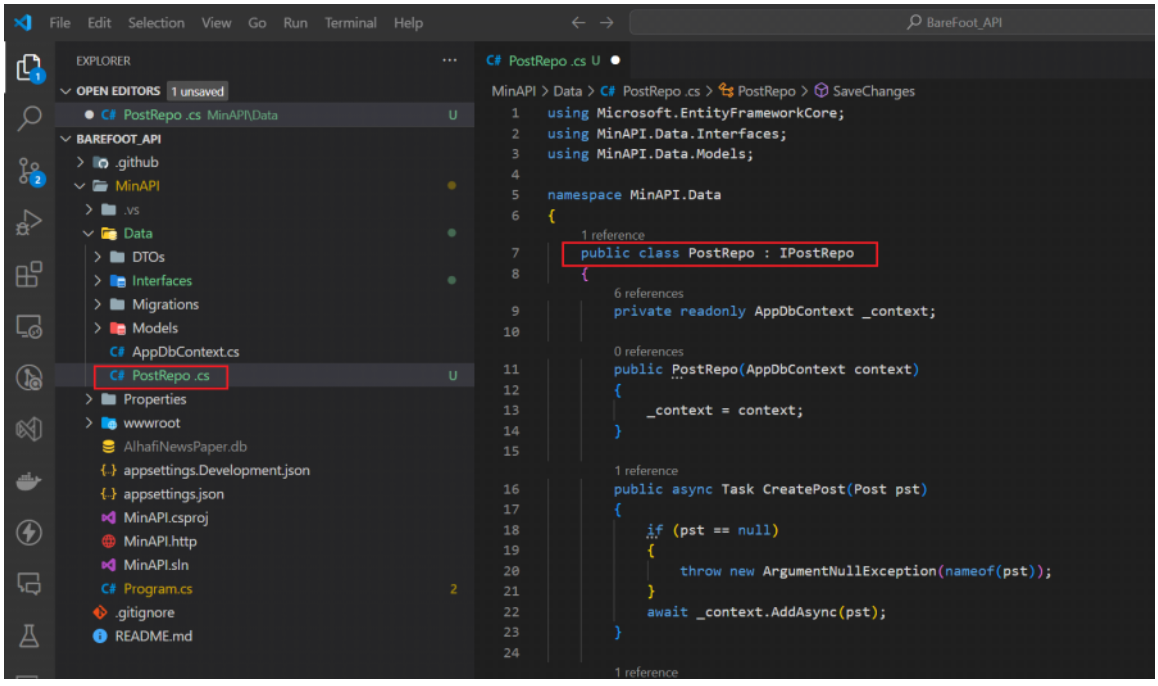

```
using System.ComponentModel.DataAnnotations;
namespace MinAPI.Data.DTOs
{
    public class PostDTOs
    {
        public class PostReadDto
        {
            public int Id { get; set; }
            public string? Title { get; set; }
            public string? Content { get; set; }
            public string? postImage { get; set; }
        }
        public class PostNewOrUpdatedDto
        {

             // [Key]
            // public int Id { get; set; }
            [Required]
            [MinLength(3)]
            public string? Title { get; set; }
            public string? Content { get; set; }
            public string? postImage { get; set; }
        }
    }
}
```

Creating Repository Layers Interface




```
using MinAPI.Data.Models;
namespace MinAPI.Data.Interfaces
{
    public interface IPostRepo
    {
        Task SaveChanges();
        Task<Post?> GetPostById(int id);
        Task<IEnumerable<Post>> GetAllPosts();
        Task CreatePost(Post pst);
        void DeletePost(Post pst);
    }
}
```

Implementation of Interface by class repository

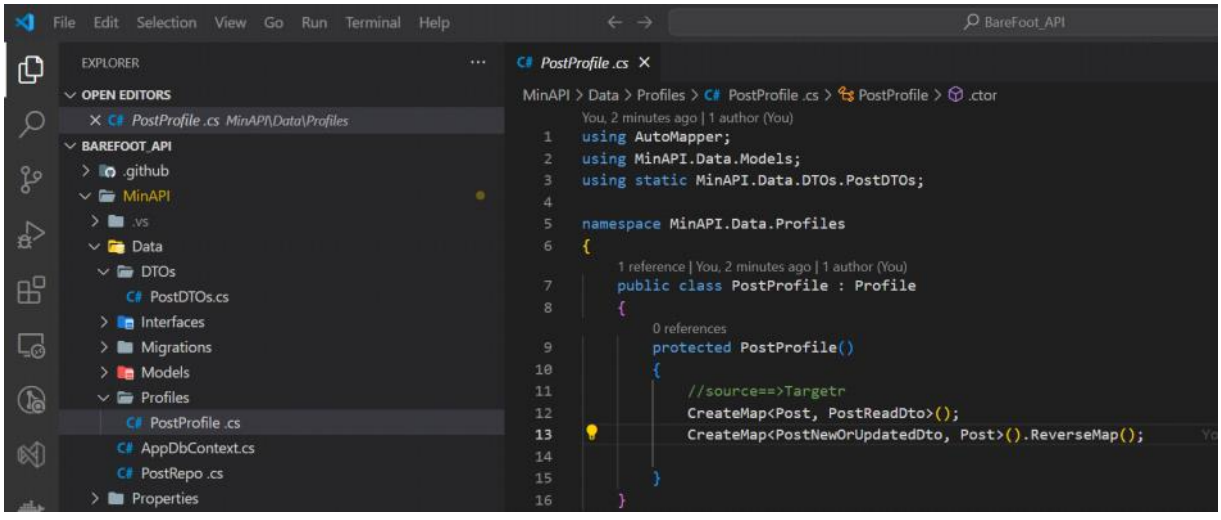



```
using Microsoft.EntityFrameworkCore;
using MinAPI.Data.Interfaces;
using MinAPI.Data.Models;
namespace MinAPI.Data
{
    public class PostRepo : IPostRepo
    {
        private readonly AppDbContext _context;
```

```
        public PostRepo(AppDbContext context)
        {
            _context = context;
        }
        public async Task CreatePost(Post pst)
        {
            if (pst == null)
            {
                throw new ArgumentNullException(nameof(pst));
            }
            await _context.AddAsync(pst);
        }
        public void DeletePost(Post pst)
        {
            if (pst == null)
            {
                throw new ArgumentNullException(nameof(pst));
            }
            _context.Posts.Remove(pst);
        }
        public async Task<IEnumerable<Post>> GetAllPosts()
        {
            return await _context.Posts.ToListAsync();
        }
        public async Task<Post?> GetPostById(int id)
        {
            return await _context.Posts.FirstOrDefaultAsync(c => c.Id == id);
        }
        public async Task SaveChanges()
        {
            await _context.SaveChangesAsync();
        }
    }
}
```

AutoMapper

Mapping source code which is Model to Target which is Dto




```
using AutoMapper;
using MinAPI.Data.Models;
using static MinAPI.Data.DTOs.PostDTOs;
namespace MinAPI.Data.Profiles
{
    public class PostProfile : Profile
    {
        public PostProfile()
        {
            //source==>Targetr
            CreateMap<Post, PostReadDto>();
            CreateMap<PostNewOrUpdatedDto, Post>().ReverseMap();
        }
```

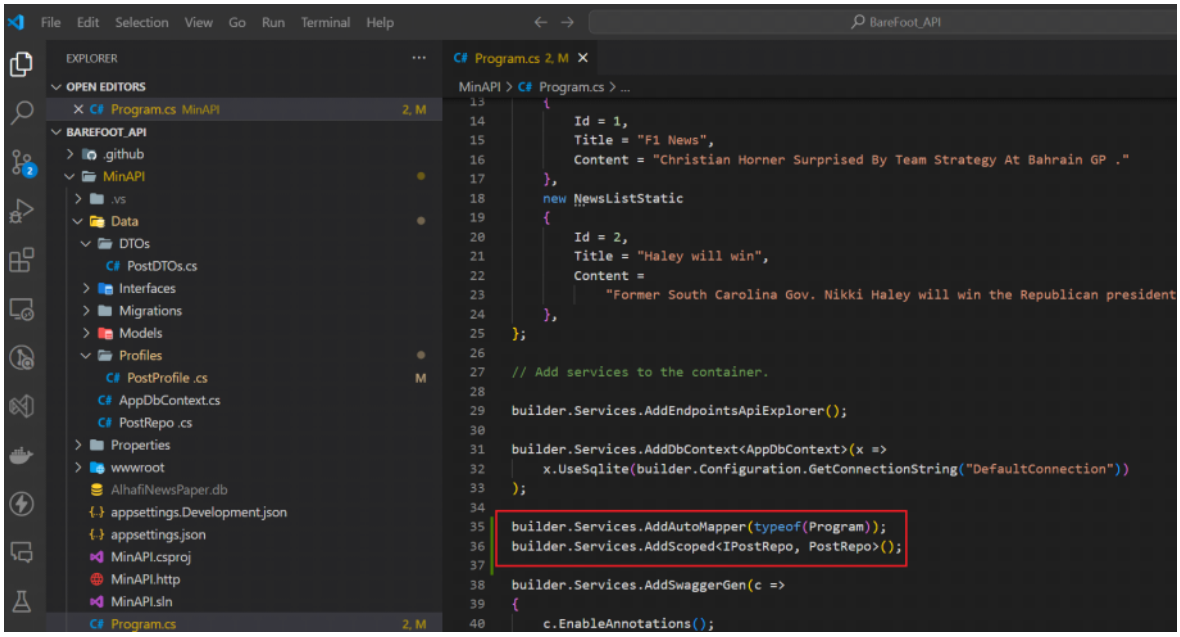
```
    }
}
```

Register Automapper

builder.Services.AddAutoMapper(typeof(Program));

Register Repository by Concrete (Class and Interface)

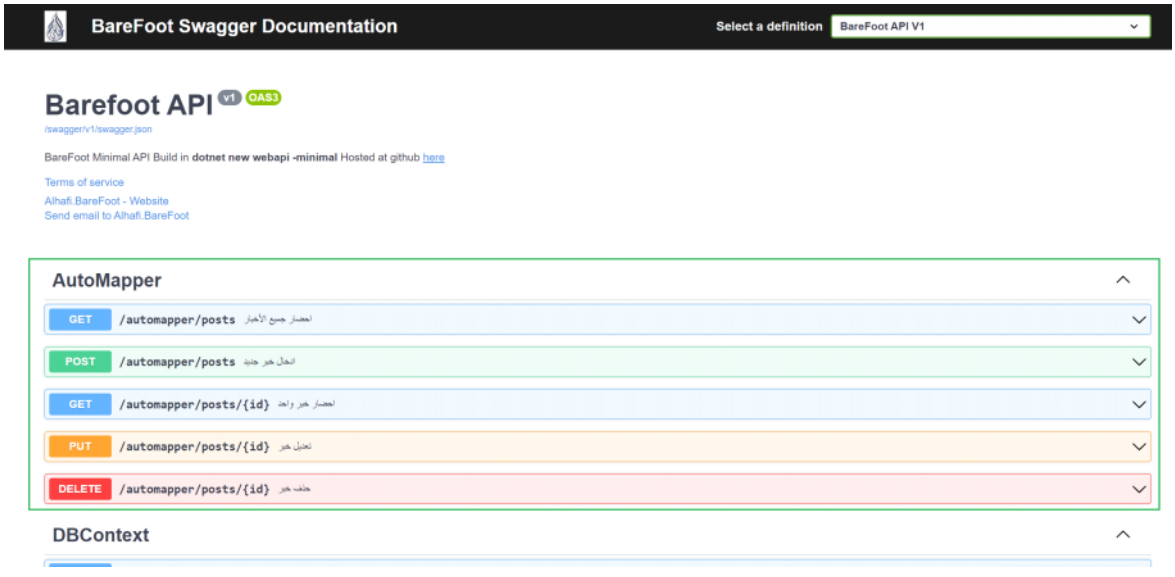
builder.Services.AddScoped<IPostRepo, PostRepo>();



Building Endpoints API

```
//************************Daynamic Data Access Automapper End Point*****************************************
app.MapGet(
        "/automapper/posts",
        async (IPostRepo repo, IMapper mapper) =>
        {
            var varPosts = await repo.GetAllPosts();
            return Results.Ok(mapper.Map<IEnumerable<PostReadDto>>(varPosts));
        }
    )
    .WithDescription("return All posts news ")
    .WithSummary("احضار جميع الأخبار")
    .WithTags("AutoMapper")
    .WithOpenApi();

app.MapGet(
        "/automapper/posts/{id}",
        async (IPostRepo repo, IMapper mapper, int id) =>
        {
            var varPost = await repo.GetPostById(id);
            if (varPost != null)
            {
                return Results.Ok(mapper.Map<PostReadDto>(varPost));
            }
            return Results.NotFound();
        }
    )
    .WithDescription("return Only One Post News")
    .WithSummary("احضار خبر واحد ")
    .WithTags("AutoMapper")
    .WithOpenApi();

app.MapPost(
        "/automapper/posts",
        async (IPostRepo repo, IMapper mapper, PostNewOrUpdatedDto postCreateDto) =>
```

```
        {
            var postModel = mapper.Map<Post>(postCreateDto);
            await repo.CreatePost(postModel);
            await repo.SaveChanges();
            var postReadDto = mapper.Map<PostReadDto>(postModel);
            return Results.Created($"/automapper/posts/{postReadDto.Id}", postReadDto);
        }
    )
    .WithDescription("Insert New Post News")
    .WithSummary("ادخال خبر جديد ")
    .WithTags("AutoMapper")
    .WithOpenApi();

app.MapPut(
        "/automapper/posts/{id}",
        async (IPostRepo repo, IMapper mapper, int id, PostNewOrUpdatedDto postUpdateDto) =>
        {
            var varPost = await repo.GetPostById(id);
            if (varPost == null)
            {
                return Results.NotFound();
            }
            mapper.Map(postUpdateDto, varPost);
            await repo.SaveChanges();
            return Results.NoContent();
        }
    )
    .WithDescription("Update Post News")
    .WithSummary("تعديل خبر  ")
    .WithTags("AutoMapper")
    .WithOpenApi();

app.MapDelete(
        "/automapper/posts/{id}",
        async (IPostRepo repo, IMapper mapper, int id) =>
        {
            var varPost = await repo.GetPostById(id);
            if (varPost == null)
            {
                return Results.NotFound();
            }
            repo.DeletePost(varPost);
            await repo.SaveChanges();
            return Results.NoContent();
        }
    )
    .WithDescription("Delete  Post ")
    .WithSummary("حذف خبر  ")
    .WithTags("AutoMapper")
    .WithOpenApi();

//*************************************************************************************************
```



Model Validation

21 April 2024 11:37

[Coding Shorts: Minimal API Endpoint Filters for Model Validation]

https://benfoster.io/blog/minimal-api-validation-endpoint-filters/

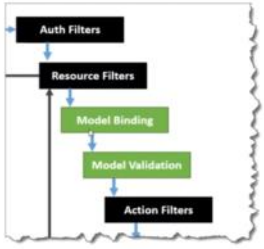
This means*...

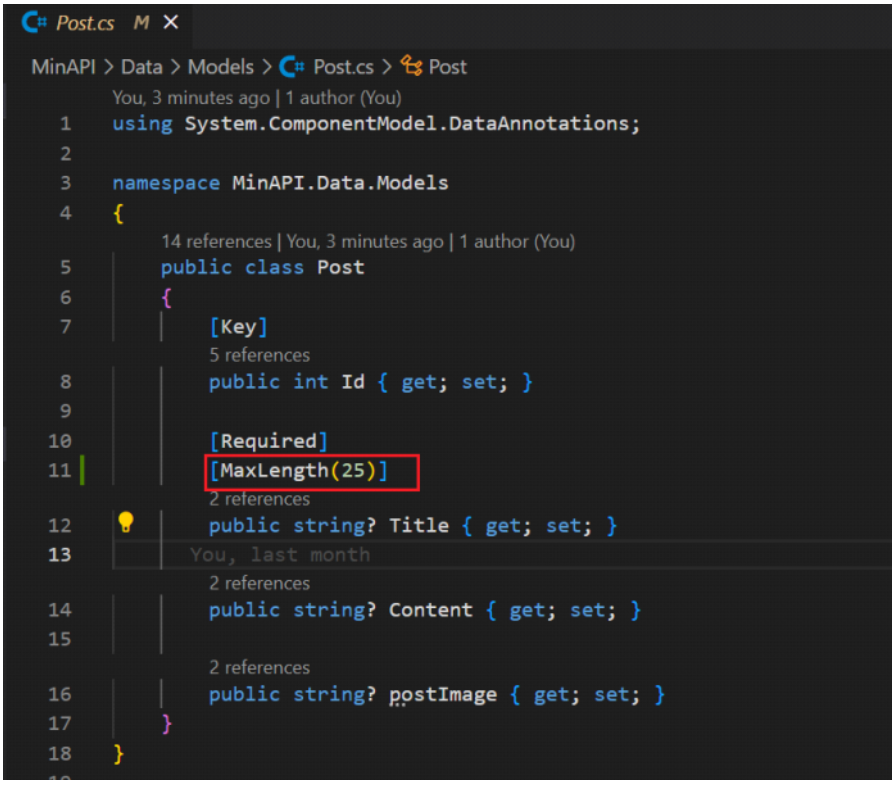
Minimal APIs:

- Don't support model validation
- Don't support for JSONPatch
- Don't support filters
- Don't support *custom* model binding (Support for IModelBinder)

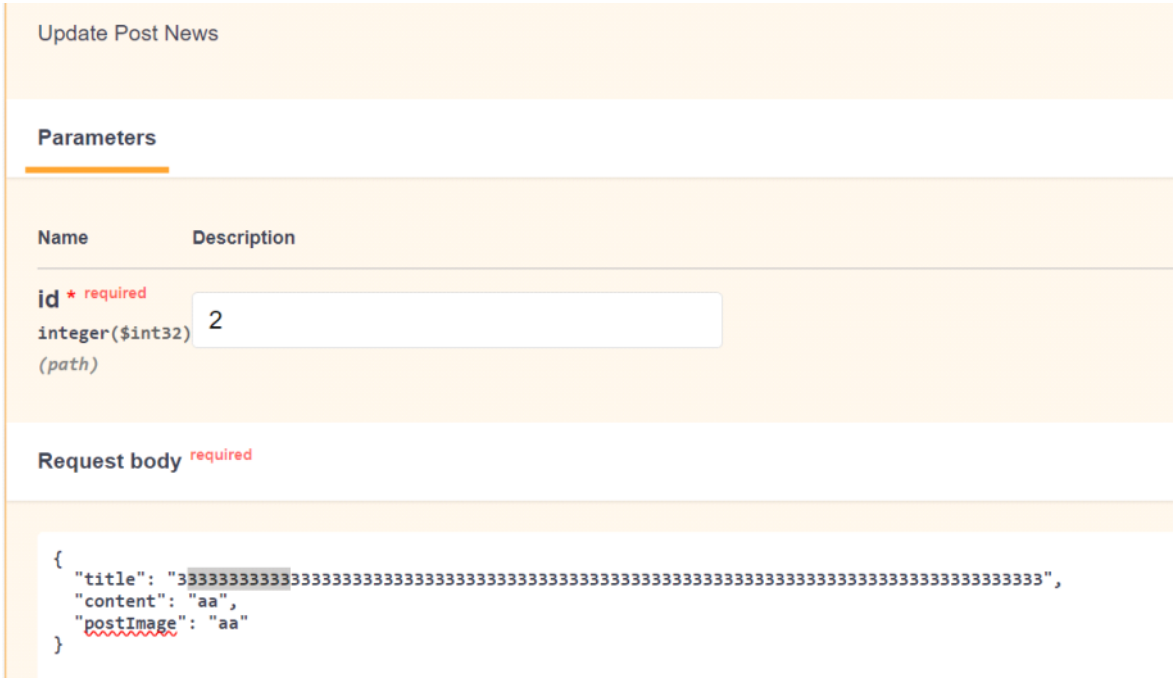
Model Validation

- Model Validation Occurs after Model Binding
- Reports business rule type errors, e.g.
  - Input string value length > max allowed by the model
- Out the box with the [ApiController] attribute
- Validation can be added to .NET 6 Minimal APIs, e.g.:
  - FluentValidation, MinimalValidation

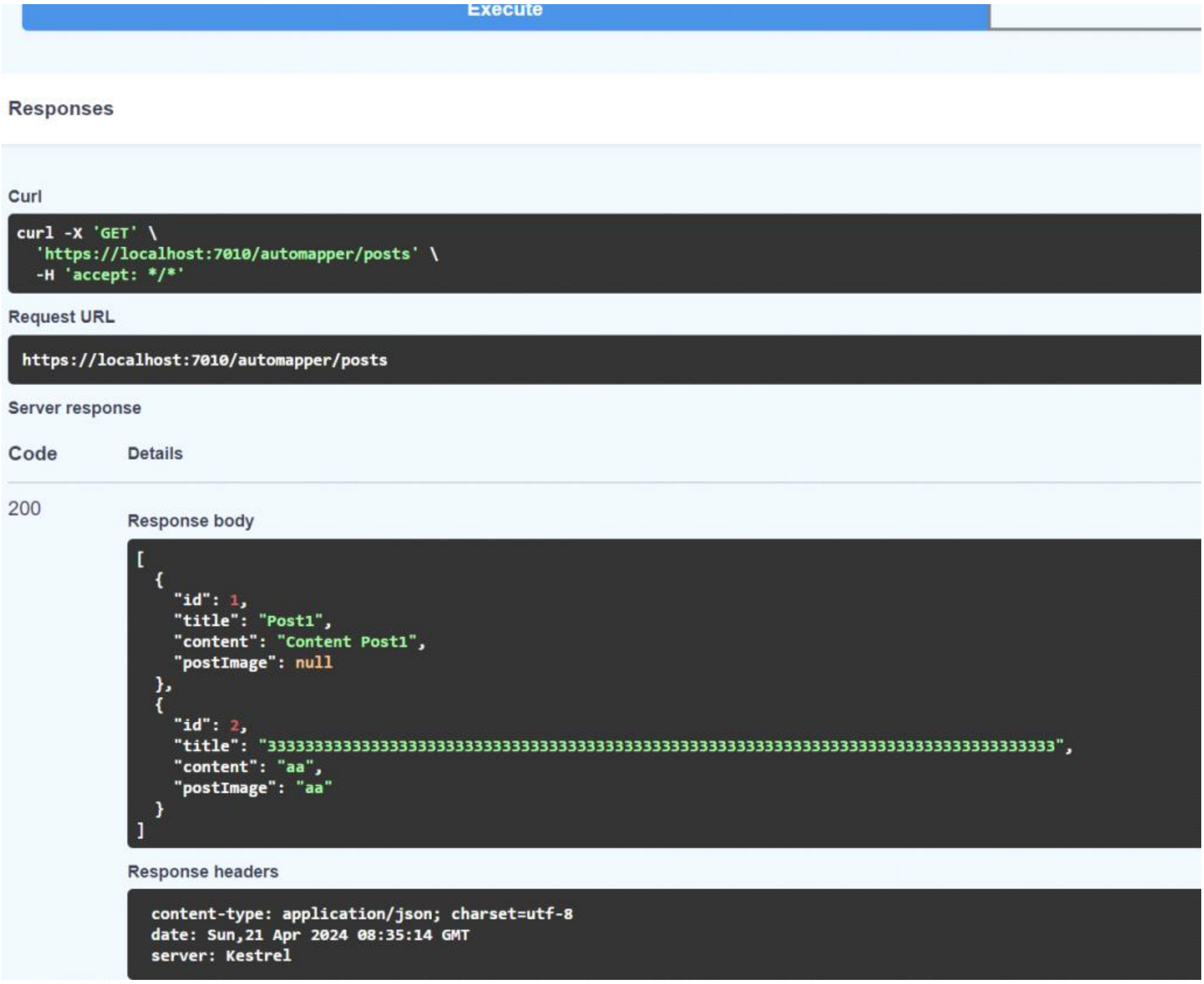




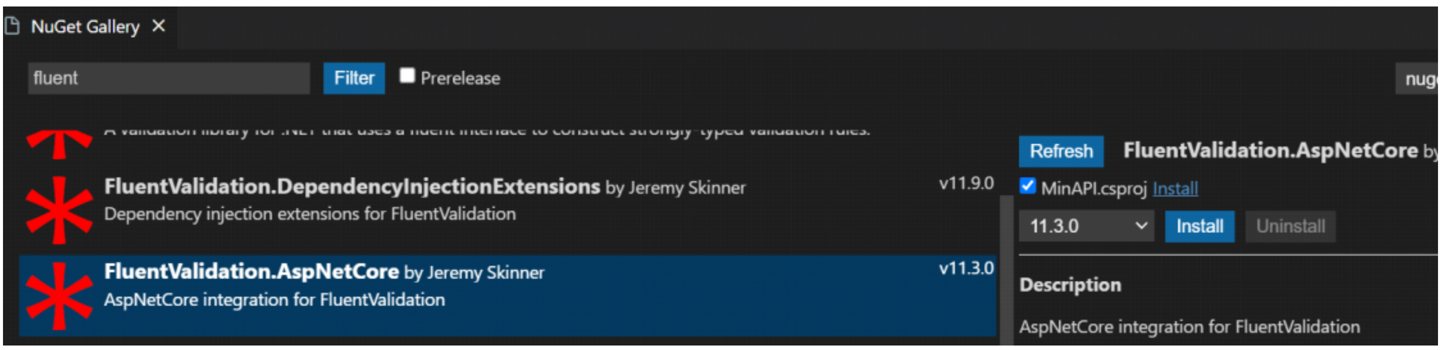
Let We test



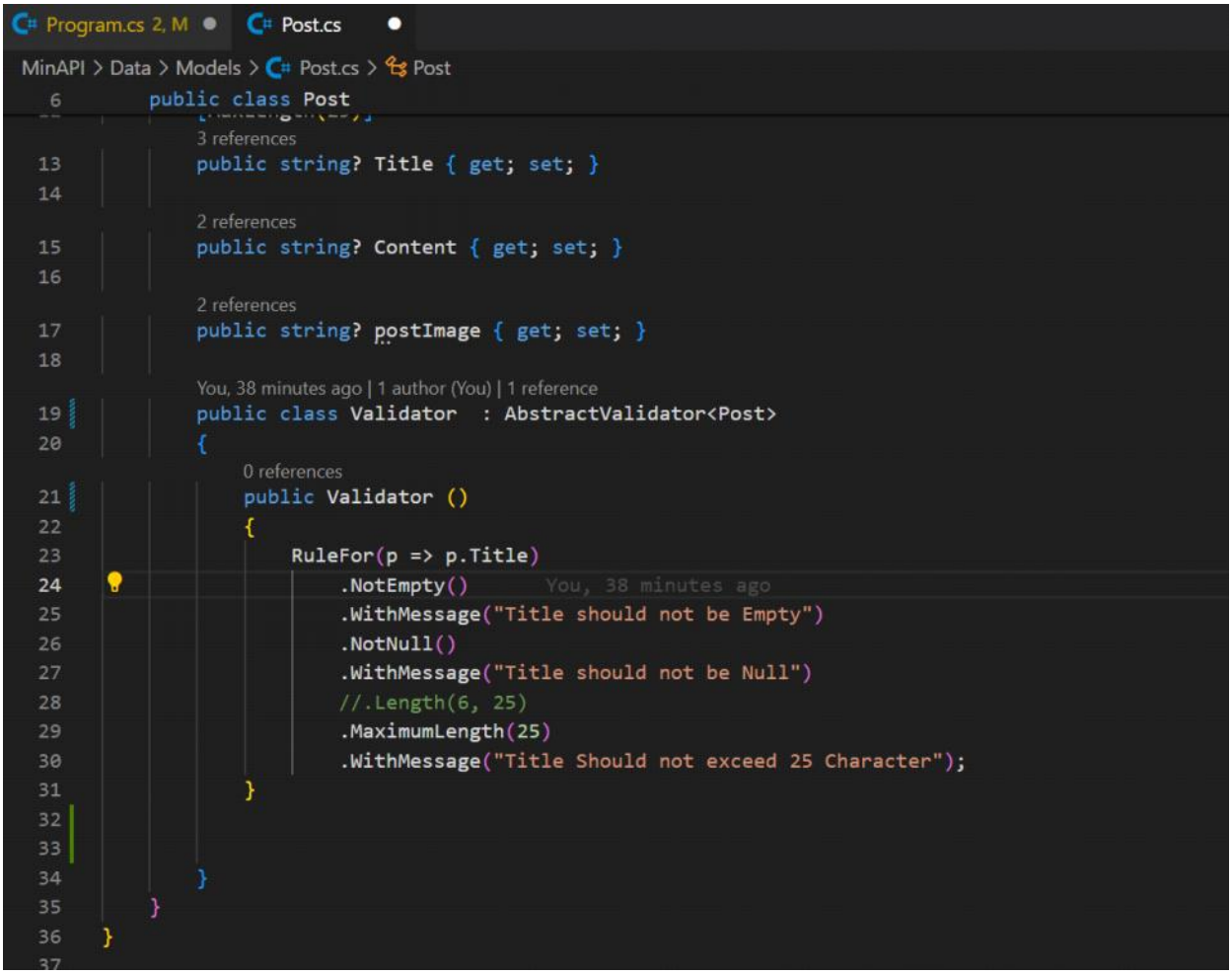
Accepted Not Validate



Let we add Fluent Validation Package

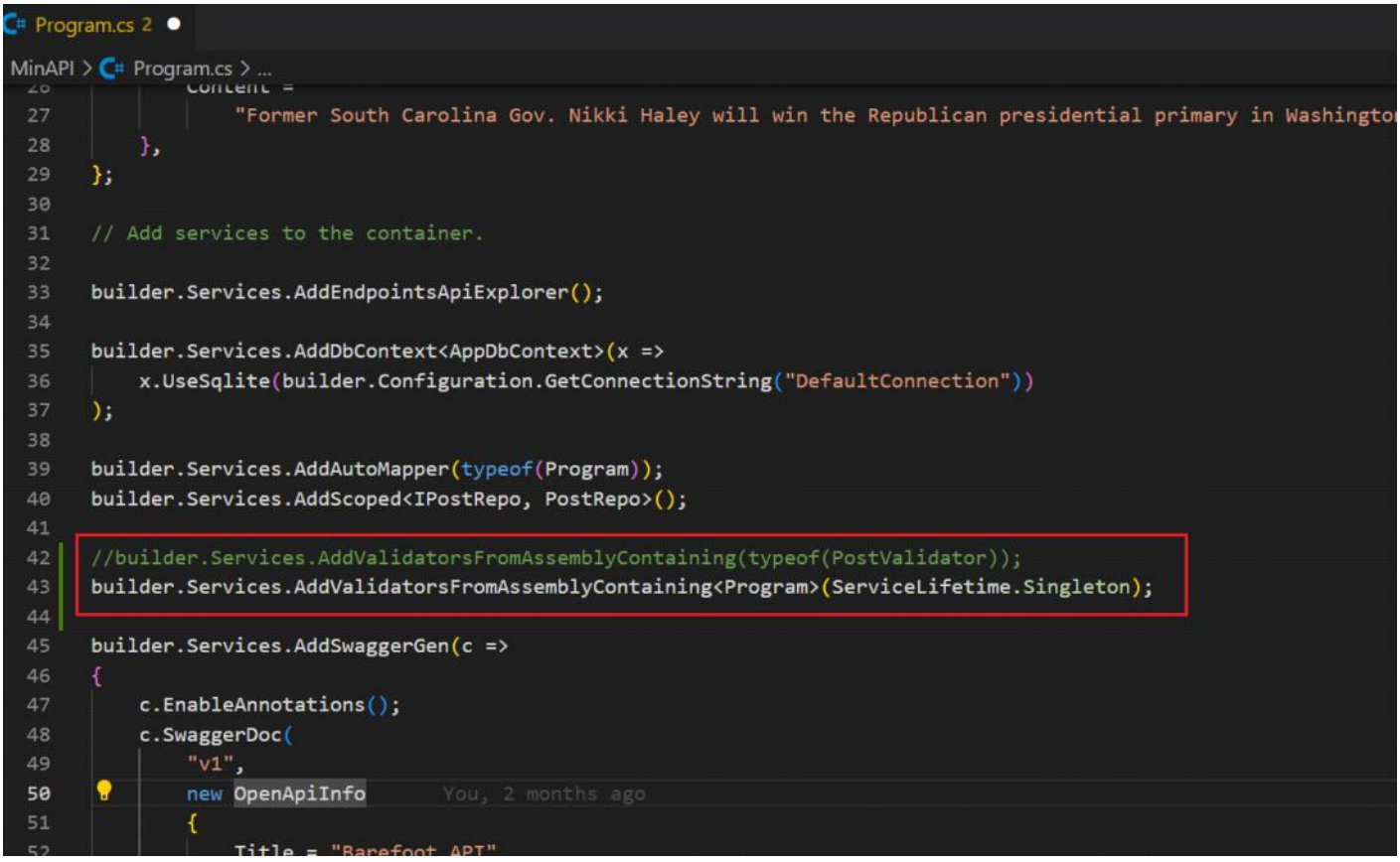


Then Add validation



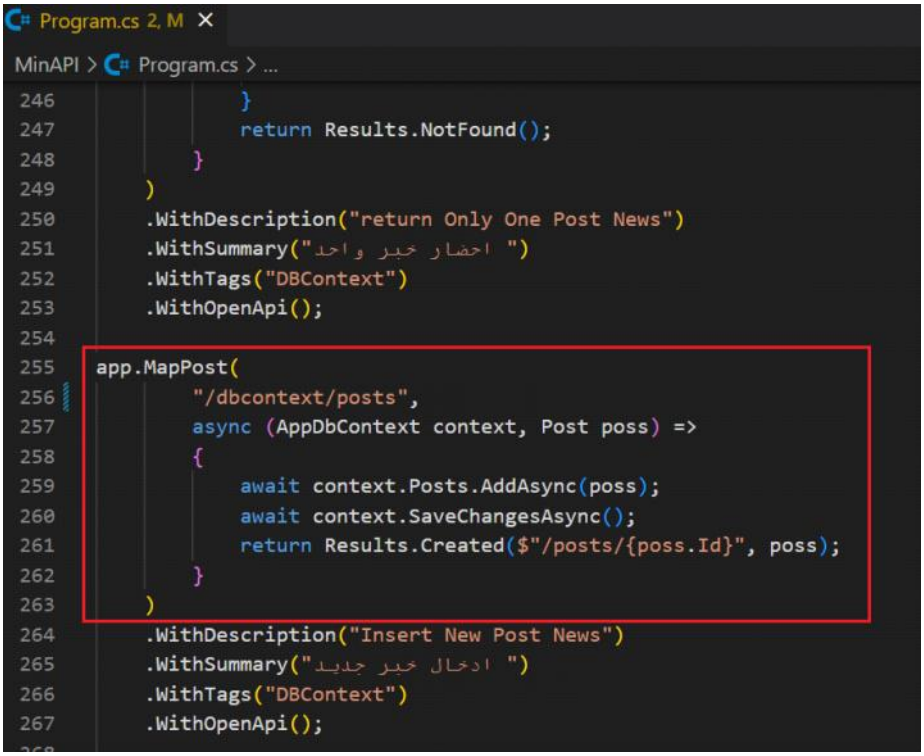

```
public class Validator  : AbstractValidator<Post>
        {
            public Validator ()
            {
                RuleFor(p => p.Title)
                    .NotEmpty()
                    .WithMessage("Title should not be Empty")
                    .NotNull()
                    .WithMessage("Title should not be Null")
                    //.Length(6, 25)
                    .MaximumLength(25)
                    .WithMessage("Title Should not exceed 25 Character");
            }

        }
```

Register validation




```
builder.Services.AddValidatorsFromAssemblyContaining<Program>(ServiceLifetime.Singleton);
```

Let we validate Post



```
app.MapPost(
        "/dbcontext/posts",
        async (AppDbContext context, Post poss, IValidator<Post> validator) =>
        {
            var validationResult = await validator.ValidateAsync(poss);

            if (validationResult.IsValid)
            {
                await context.Posts.AddAsync(poss);
                await context.SaveChangesAsync();
                return Results.Created($"/posts/{poss.Id}", poss);
            }
            return Results.ValidationProblem(
                validationResult.ToDictionary(),
                statusCode: (int)HttpStatusCode.UnprocessableEntity
            );
        }
    )
    .WithDescription("Insert New Post News")
    .WithSummary("ادخال خبر جديد ")
    .WithTags("DBContext")
    .WithOpenApi();
```

```
app.MapPost(
        "/dbcontext/posts",
        async (AppDbContext context, Post poss, IValidator<Post> validator) =>
        {
            var validationResult = await validator.ValidateAsync(poss);
            if (validationResult.IsValid)
            {
                await context.Posts.AddAsync(poss);
                await context.SaveChangesAsync();
                return Results.Created($"/posts/{poss.Id}", poss);
            }
            return Results.ValidationProblem(
                validationResult.ToDictionary(),
                statusCode: (int)HttpStatusCode.UnprocessableEntity
            );
        }
    )
    .WithDescription("Insert New Post News")
    .WithSummary("ادخال خبر جديد ")
    .WithTags("DBContext")
    .WithOpenApi();
```

Let we Test

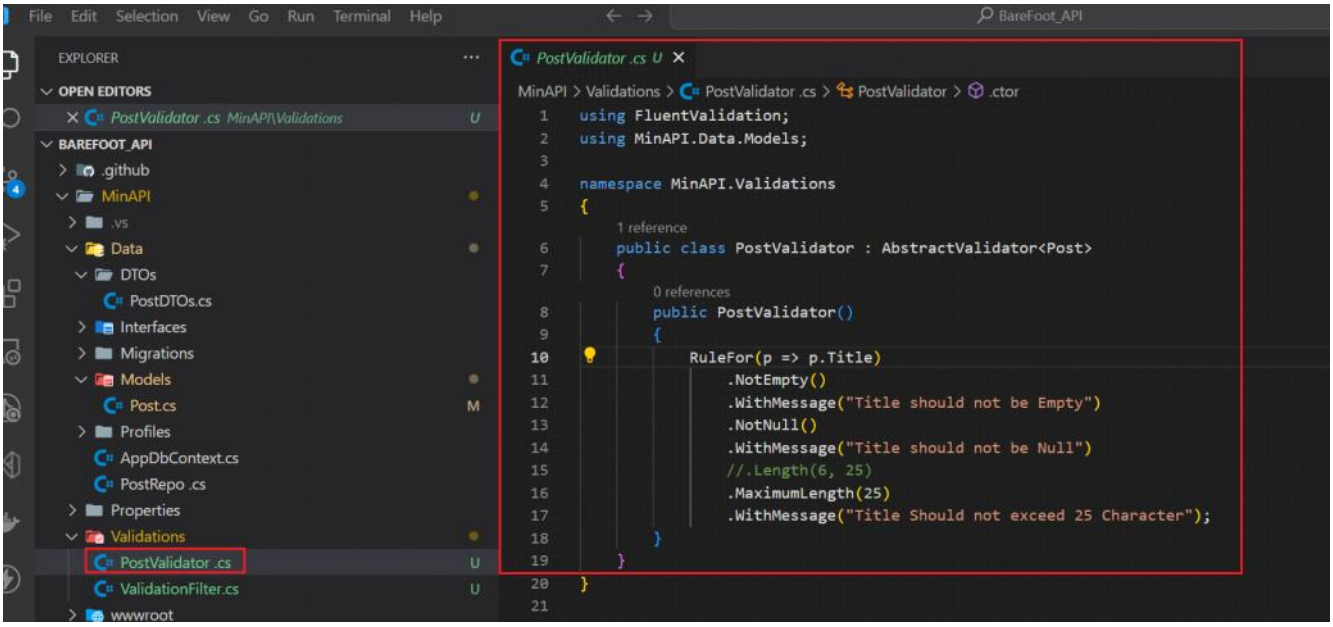
```
{
  "id": 0,
  "title": "5555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555",
  "content": "string",
  "postImage": "string"
}
```

```
{
  "type": "https://tools.ietf.org/html/rfc4918#section-11.2",
  "title": "One or more validation errors occurred.",
  "status": 422,
  "errors": {
    "Title": [
      "Title Should not exceed 25 Character"
    ]
  }
}
```

```
{
  "id": 0,
  "title": "",
  "content": "string",
  "postImage": "string"
}
```

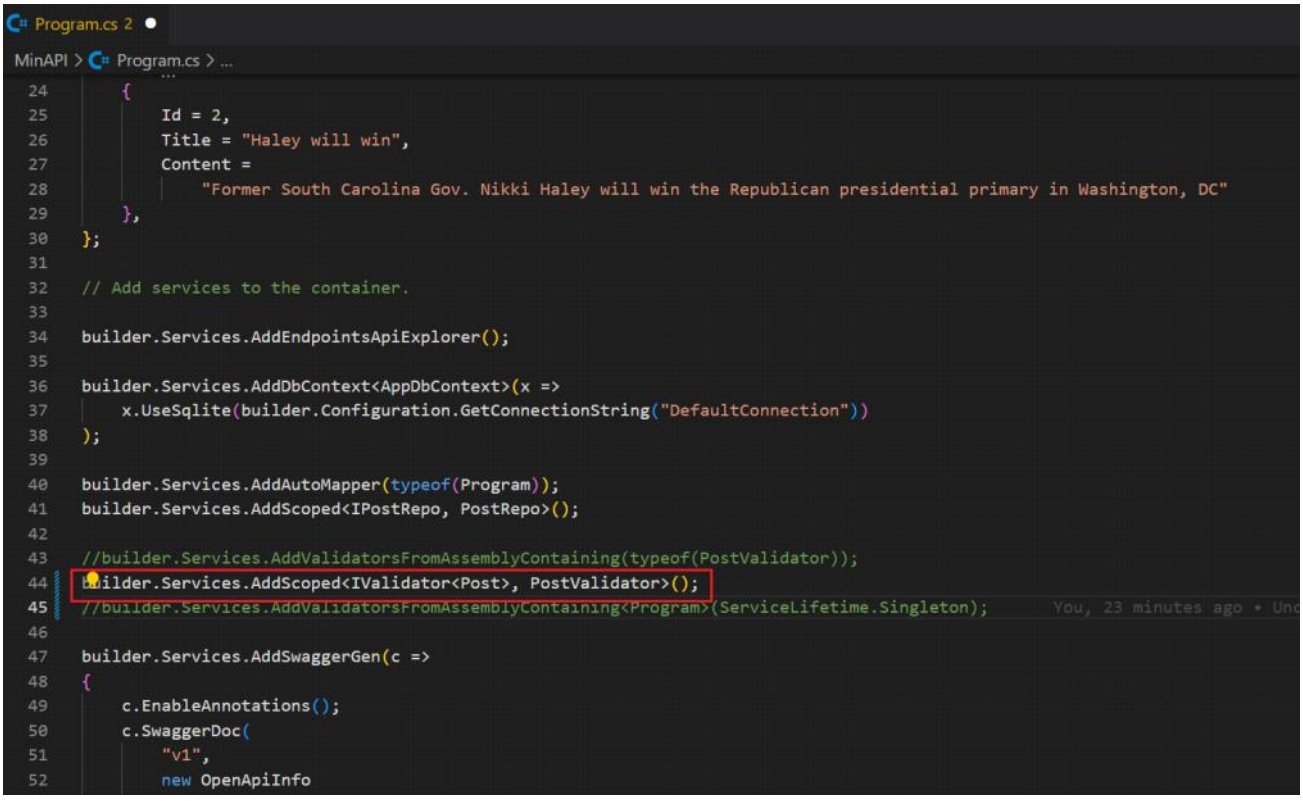
```
{
  "type": "https://tools.ietf.org/html/rfc4918#section-11.2",
  "title": "One or more validation errors occurred.",
  "status": 422,
  "errors": {
    "Title": [
      "Title should not be Empty"
    ]
  }
}
```

We can Separate Validation in different class

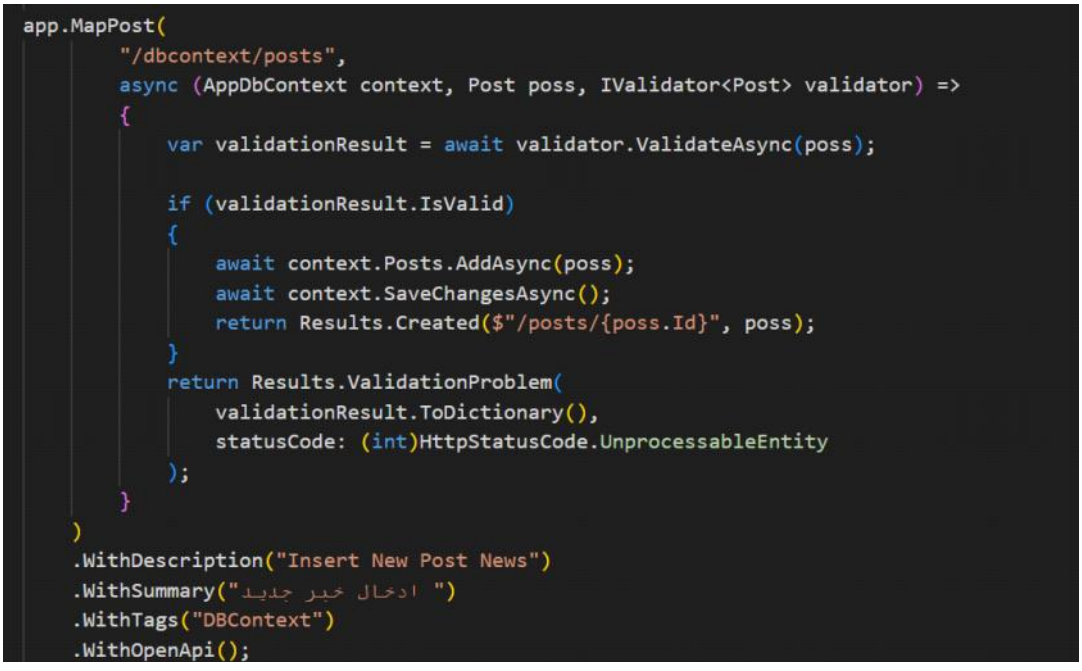



```
using FluentValidation;
using MinAPI.Data.Models;
namespace MinAPI.Validations
{
    public class PostValidator : AbstractValidator<Post>
    {
        public PostValidator()
        {
            RuleFor(p => p.Title)
                .NotEmpty()
                .WithMessage("Title should not be Empty")
                .NotNull()
                .WithMessage("Title should not be Null")
                //.Length(6, 25)
                .MaximumLength(25)
                .WithMessage("Title Should not exceed 25 Character");
        }
    }
}
```

Register the validator



And keep as below

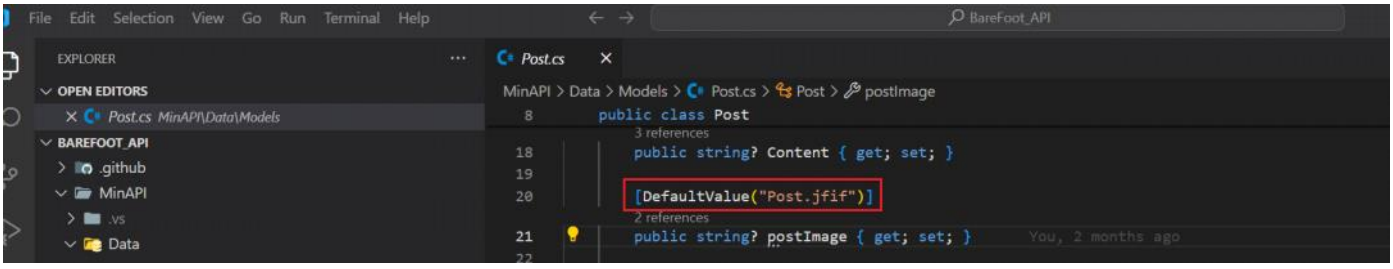



```
app.MapPost(
        "/dbcontext/posts",
        async (AppDbContext context, Post poss, IValidator<Post> validator) =>
        {
            var validationResult = await validator.ValidateAsync(poss);
            if (validationResult.IsValid)
            {
                await context.Posts.AddAsync(poss);
                await context.SaveChangesAsync();
                return Results.Created($"/posts/{poss.Id}", poss);
            }
            return Results.ValidationProblem(
                validationResult.ToDictionary(),
                statusCode: (int)HttpStatusCode.UnprocessableEntity
            );
        }
    )
```

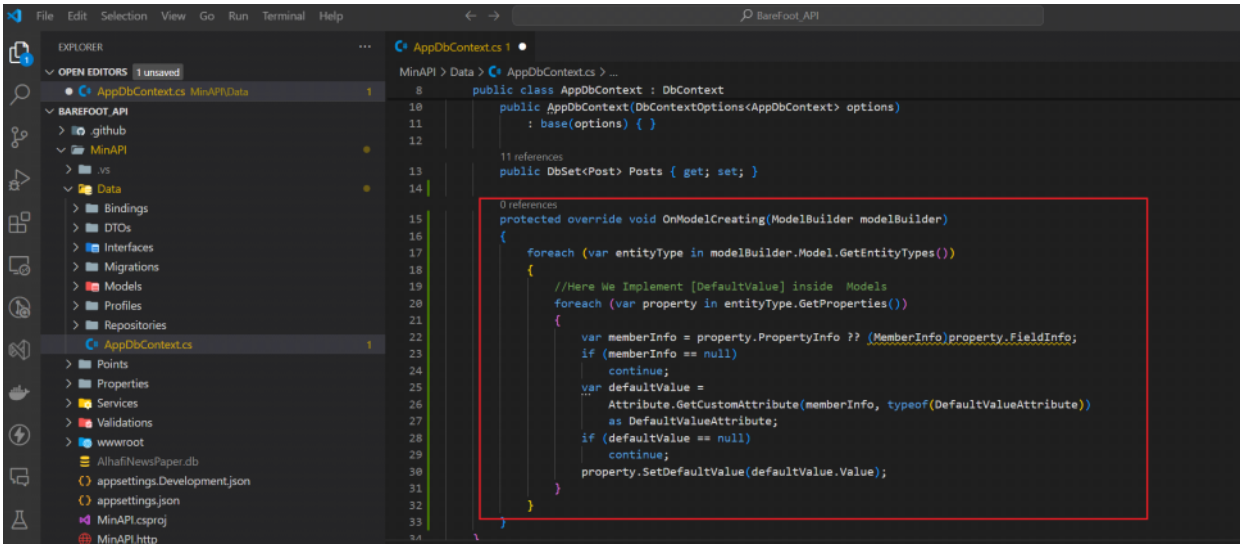
```
    .WithDescription("Insert New Post News")
    .WithSummary("ادخال خبر جديد ")
    .WithTags("DBContext")
    .WithOpenApi();
```

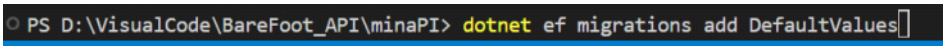
Default Values

Fluent Support validation as above example , but we need default annotation such as below

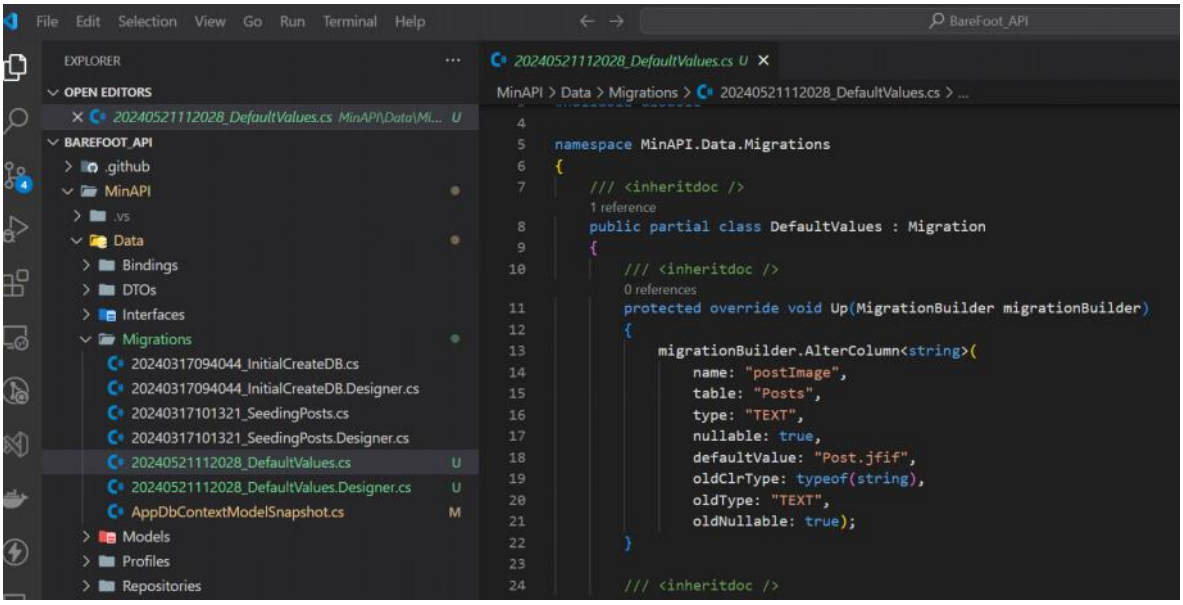


Let we add on Model creation



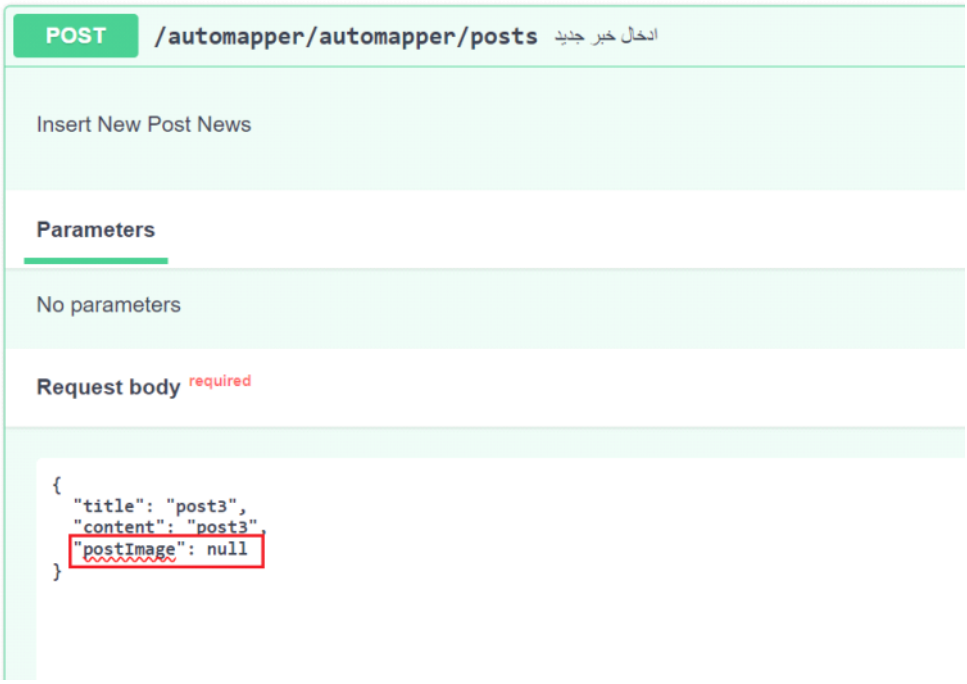


dotnet ef migrations add DefaultValues



```
Done.
PS D:\VisualCode\BareFoot_API\minaPI> dotnet ef database update
```

dotnet ef database update



Default Values for Date & Time

Post.cs

MinAPI > Data > Models > Post.cs > ...

```
9       public class Post
16          public string? Title { get; set; }
17
18          [SwaggerParameter(Description = "Property description", Required = false)]
            3 references
19          public string? Content { get; set; }
20
21          [DefaultValue("Post.jfif")]
            2 references
22          public string? postImage { get; set; }
23
24          [Column("CreatedOn", TypeName = "SmallDateTime")]
25          [DisplayFormat(DataFormatString = "{0:dd-MM-yyyy}", ApplyFormatInEditMode = true)]
            0 references
26          public DateTime? CreatedOn { get; set; }
27
```

AppDbContext.cs 1, M

MinAPI > Data > AppDbContext.cs > AppDbContext > OnModelCreating

```
8       public class AppDbContext : DbContext
            0 references
15          protected override void OnModelCreating(ModelBuilder modelBuilder)
16          {
17              foreach (var entityType in modelBuilder.Model.GetEntityTypes())
18              {
19                  //Here We Implement [DefaultValue] inside  Models
20                  foreach (var property in entityType.GetProperties())
21                  {
22                      var memberInfo = property.PropertyInfo ?? (MemberInfo)property.FieldInfo;
23                      if (memberInfo == null)
24                          continue;
25                      var defaultValue =
26                          Attribute.GetCustomAttribute(memberInfo, typeof(DefaultValueAttribute))
27                          as DefaultValueAttribute;
28                      if (defaultValue == null)
29                          continue;
30                      property.SetDefaultValue(defaultValue.Value);
31                  }
32
33                  //Fixed defaultValue for datetime in SQL for All Entioties as one
34                  entityType.FindProperty("CreatedOn")?.SetDefaultValueSql("GETDATE()");
35              }
36          }       You, 12 minutes ago • Uncommitted changes
37      }
38  }
39
```

dotnet ef migrations add DefaultValuesDateTime

EXPLORER

OPEN EDITORS

20240521113144_DefaultValuesDateTime.cs MinA... U

BAREFOOT_API

- .github
- MinAPI
  - .vs
  - Data
    - Bindings
    - DTOs
    - Interfaces
    - Migrations
      - 20240317094044_InitialCreateDB.cs
      - 20240317094044_InitialCreateDB.Designer.cs
      - 20240317101321_SeedingPosts.cs
      - 20240317101321_SeedingPosts.Designer.cs
      - 20240521112028_DefaultValues.cs U
      - 20240521112028_DefaultValues.Designer.cs U
      - 20240521113144_DefaultValuesDateTime.cs U
      - 20240521113144_DefaultValuesDateTime.Desi... U
      - AppDbContextModelSnapshot.cs M

20240521113144_DefaultValuesDateTime.cs U

MinAPI > Data > Migrations > 20240521113144_DefaultValuesDateTime.cs > ...

```
1   using System;
2   using Microsoft.EntityFrameworkCore.Migrations;
3
4   #nullable disable
5
6   namespace MinAPI.Data.Migrations
7   {
8       /// <inheritdoc />
        1 reference
9       public partial class DefaultValuesDateTime : Migration
10      {
11          /// <inheritdoc />
            0 references
12          protected override void Up(MigrationBuilder migrationBuilder)
13          {
14              migrationBuilder.AddColumn<DateTime>(
15                  name: "CreatedOn",
16                  table: "Posts",
17                  type: "SmallDateTime",
18                  nullable: true,
19                  defaultValueSql: "GETDATE()");
20          }
```

dotnet ef database update

Parameters Binding

21 April 2024 13:55

https://www.infoworld.com/article/3688908/how-to-use-parameter-binding-in-minimal-apis-in-asp-net-core.html

What is parameter binding?

Parameter binding involves mapping incoming HTTP request data to action method parameters, allowing developers to process requests and respond in a structured and efficient manner.
Parameter binding simplifies the process of handling HTTP requests and allows developers to focus on building the logic of their API endpoints. Minimal APIs in ASP.NET Core 7 offer several types of parameter binding including From Query, From Route, From Header, and From Body.

Why use parameter binding?

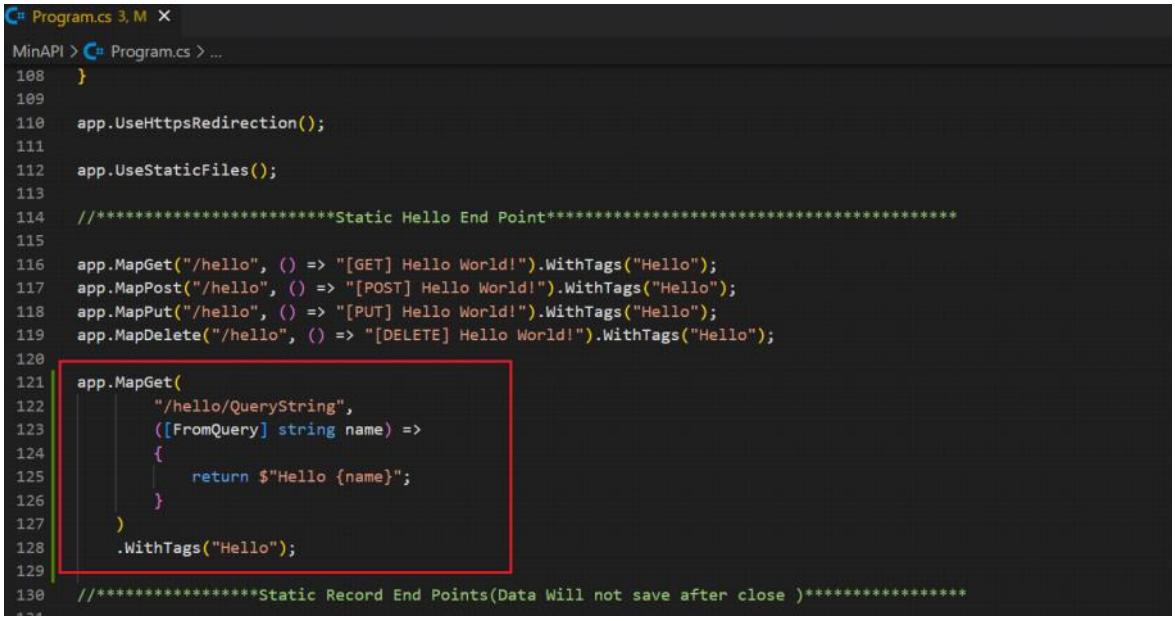
Here are a few reasons why you should use parameter binding in minimal APIs.
To simplify code: Using parameter binding, developers can reduce the boilerplate code required to handle incoming HTTP requests. Instead of manually parsing query string parameters, route data, and request bodies, parameter binding allows developers to define action method parameters and have the framework handle the binding process automatically.
To improve code maintainability: By leveraging parameter binding in minimal APIs, developers can create more maintainable code that is easier to understand and modify over time. The binding process is standardized and predictable, making it easier for developers to reason how data is transmitted between the client and the server.
To enhance application performance: Parameter binding can also help improve performance by reducing unnecessary data processing in the application. For example, by binding a request body to a specific parameter type, the framework can avoid the overhead of parsing and deserializing the entire request body, instead focusing only on the relevant data needed by the application.
To handle complex data types: Parameter binding can be used to handle complex data types such as nested objects, arrays, and collections. By leveraging the built-in mechanisms for binding complex data types, developers can create APIs that address a wide range of data formats without having to write additional code.

How does parameter binding work?

Parameter binding in minimal APIs in ASP.NET Core 7 works similarly to that of traditional ASP.NET Core applications. When a client makes an HTTP request to a minimal API, the request data is automatically mapped to action method parameters based on the parameter names and types. By default, the framework uses a convention-based approach to automatically map request data to action method parameters, but developers can also use explicit parameter binding to gain more control

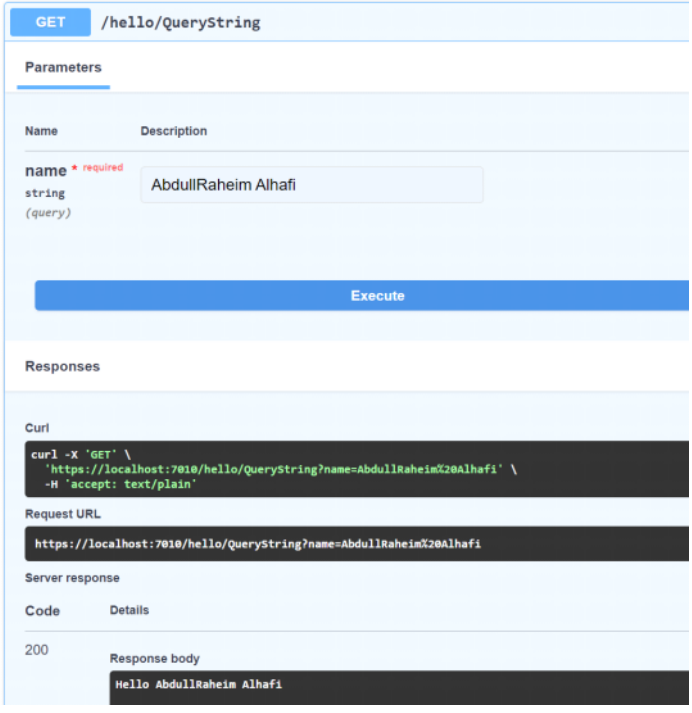
Parameter binding with query strings

To use parameter binding in minimal APIs in ASP.NET Core 7, developers need to define action methods that accept parameters. For example, the following code snippet defines a minimal API endpoint that accepts a parameter from the query string.

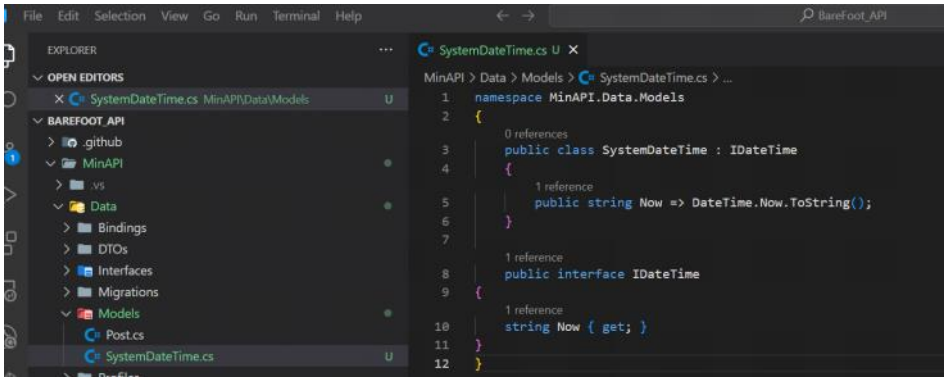



```
app.MapGet(
        "/hello/QueryString",
        ([FromQuery] string name) =>
        {
            return $"Hello {name}";
        }
    )
    .WithTags("Hello");
```

In this example, the [FromQuery] attribute tells the framework to bind the name parameter to the value of the name query string parameter in the HTTP request.



Parameter binding with dependency injection

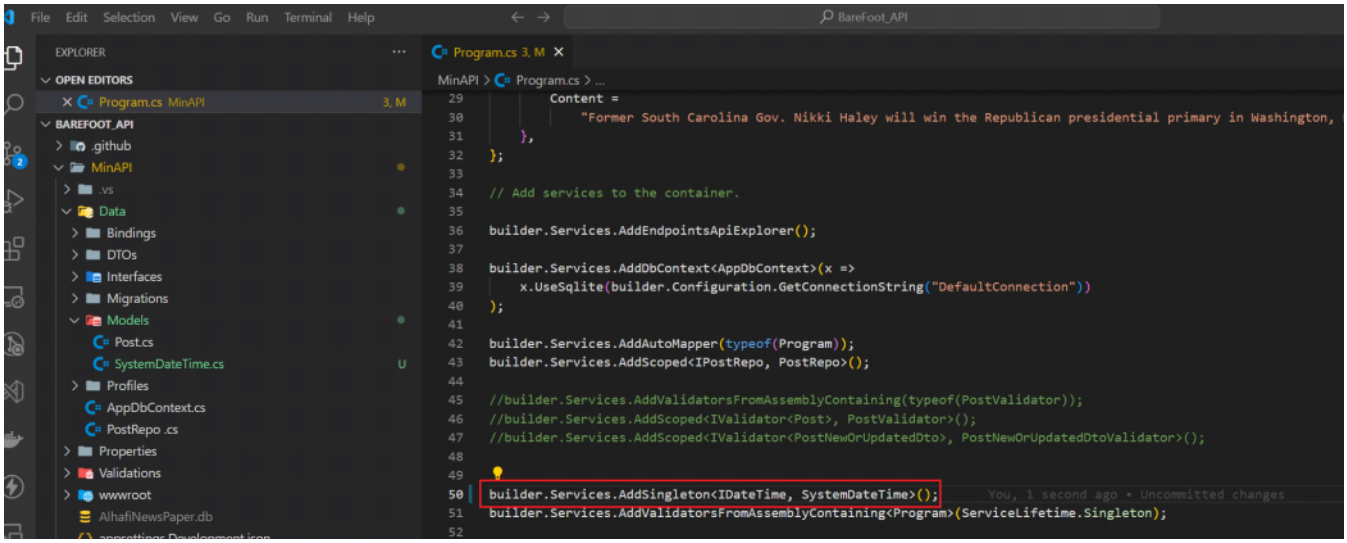



```
namespace MinAPI.Data.Models
{
    public class SystemDateTime : IDateTime
    {
        public string Now => DateTime.Now.ToString();
    }
    public interface IDateTime
{
    string Now { get; }
}
}
```

With ASP.NET Core 7, you can take advantage of dependency injection to bind parameters in the action methods of your API controllers. If the type is configured as a service, you no longer need to add the [FromServices] attribute to your method parameters. Consider the following code snippet.

```
[Route("[controller]")]
[ApiController]
public class MyDemoController : ControllerBase
{
    public ActionResult Get(IDateTime dateTime) => Ok(dateTime.Now);
}
```

If the type is configured as a service, you don't need to use the [FromServices] attribute to bind parameters. Instead, you can use the following piece of code to bind parameters using dependency injection.

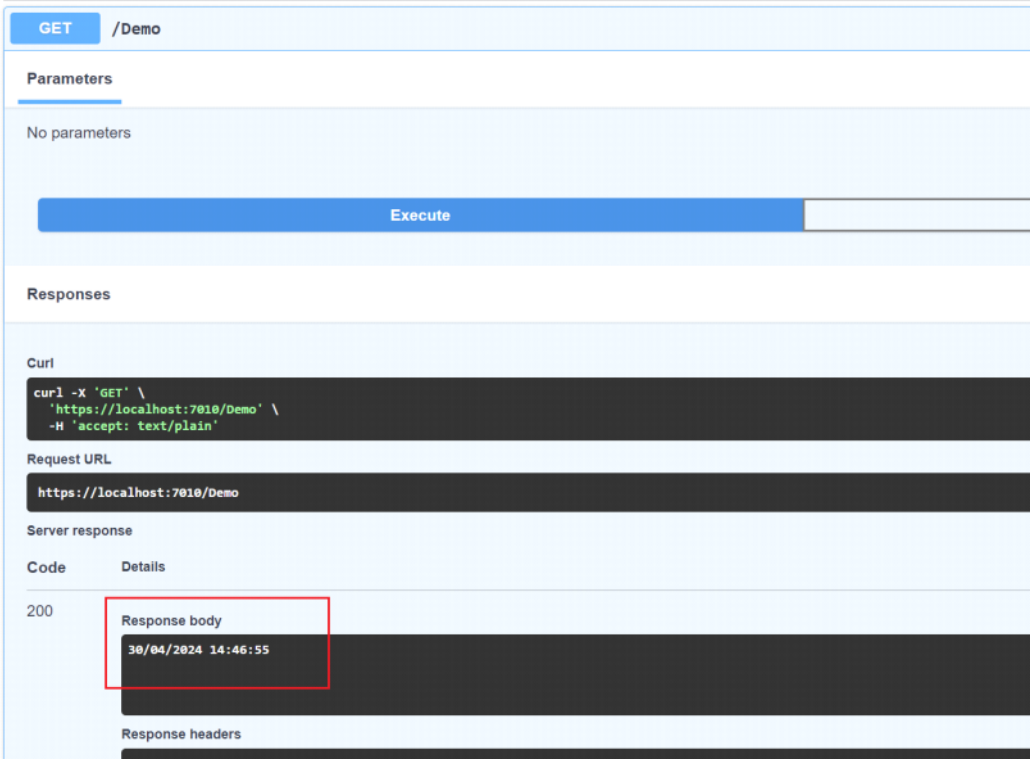



```
builder.Services.AddSingleton<IDateTime, SystemDateTime>();
```

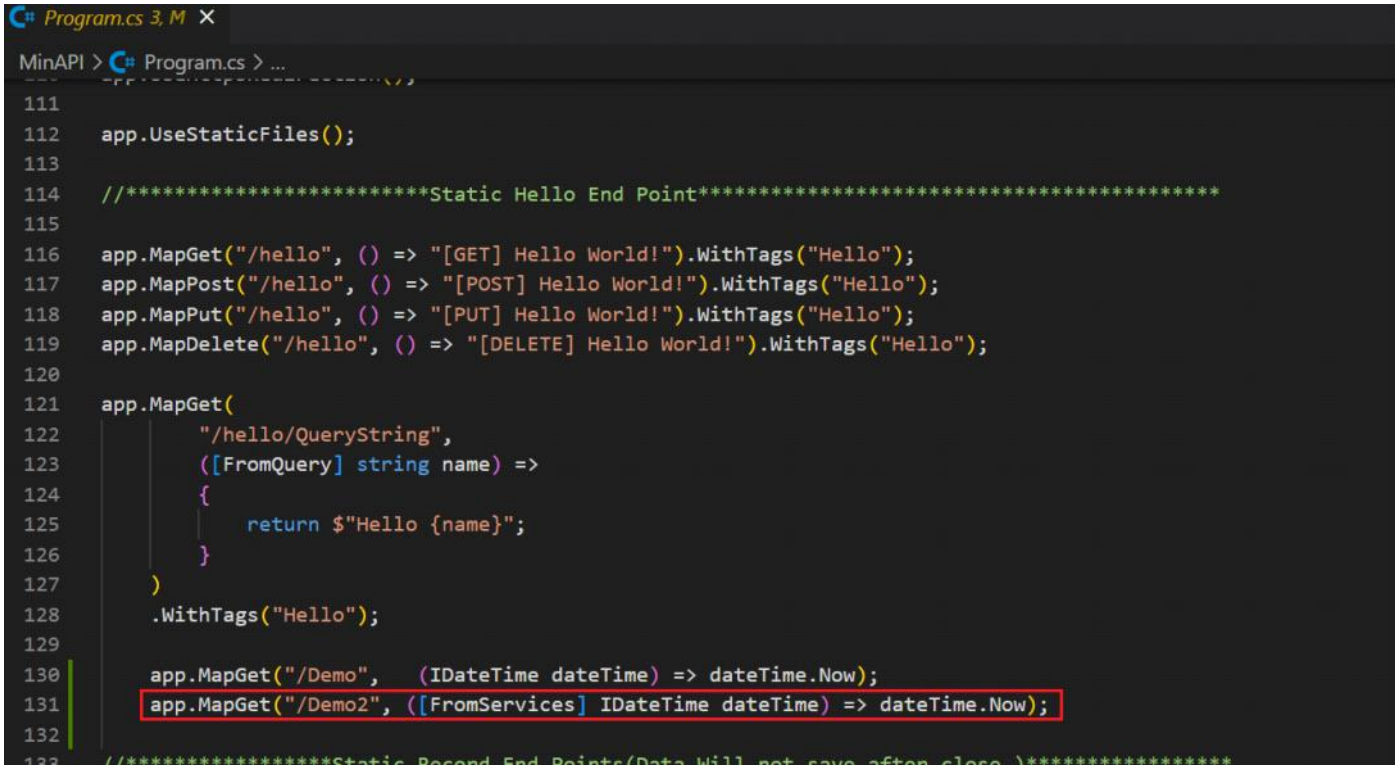
```
Program.cs 3, M
MinAPI > Program.cs > ...
111
112   app.UseStaticFiles();
113
114   //************************Static Hello End Point*****************************************
115
116   app.MapGet("/hello", () => "[GET] Hello World!").WithTags("Hello");
117   app.MapPost("/hello", () => "[POST] Hello World!").WithTags("Hello");
118   app.MapPut("/hello", () => "[PUT] Hello World!").WithTags("Hello");
119   app.MapDelete("/hello", () => "[DELETE] Hello World!").WithTags("Hello");
120
121   app.MapGet(
122           "/hello/QueryString",
123           ([FromQuery] string name) =>
124           {
125               return $"Hello {name}";
126           }
127       )
128       .WithTags("Hello");
129
130       app.MapGet("/Demo",    (IDateTime dateTime) => dateTime.Now);
131
```

```
app.MapGet("/Demo",    (IDateTime dateTime) => dateTime.Now);
```

MinAPI



It's not necessary to explicitly apply it as below



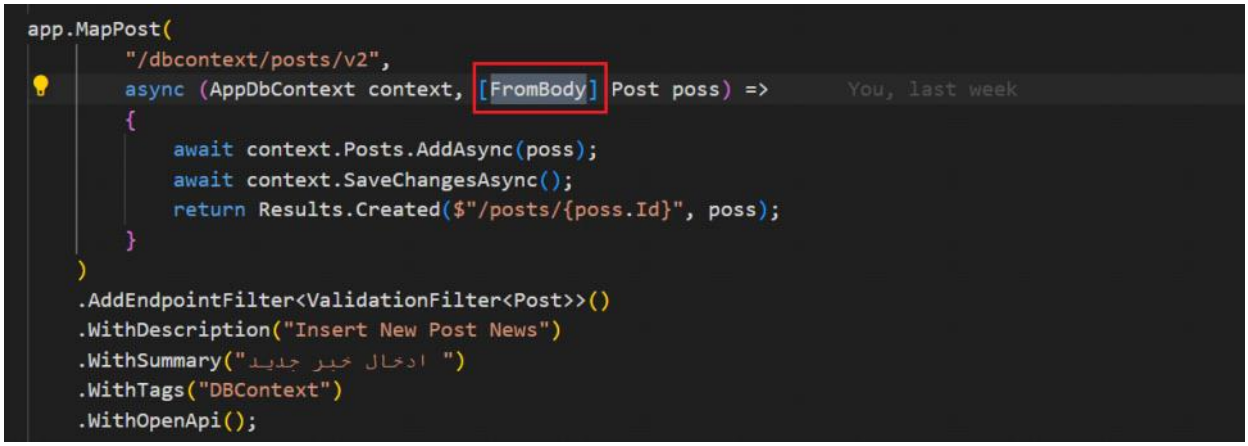

```
app.MapGet("/Demo2", ([FromServices] IDateTime dateTime) => dateTime.Now);
```

Explicit parameter binding in minimal APIs

Explicit parameter binding in minimal APIs in ASP.NET Core 7 is a technique that allows developers to have more control over the binding process by explicitly specifying the source of the data for a given parameter. This is useful in situations where the binding behavior cannot be inferred from the parameter's name or type alone. In ASP.NET Core 7 minimal APIs, developers can use the following attributes to explicitly specify the source of data for a parameter:

- [FromQuery] specifies that the parameter value should be derived from the HTTP query string.
- [FromRoute] specifies that the parameter value should be derived from the HTTP request's route data.
- [FromHeader] specifies that the parameter value should be taken from the HTTP request header.
- [FromBody] specifies that the parameter value should come from the HTTP request body.

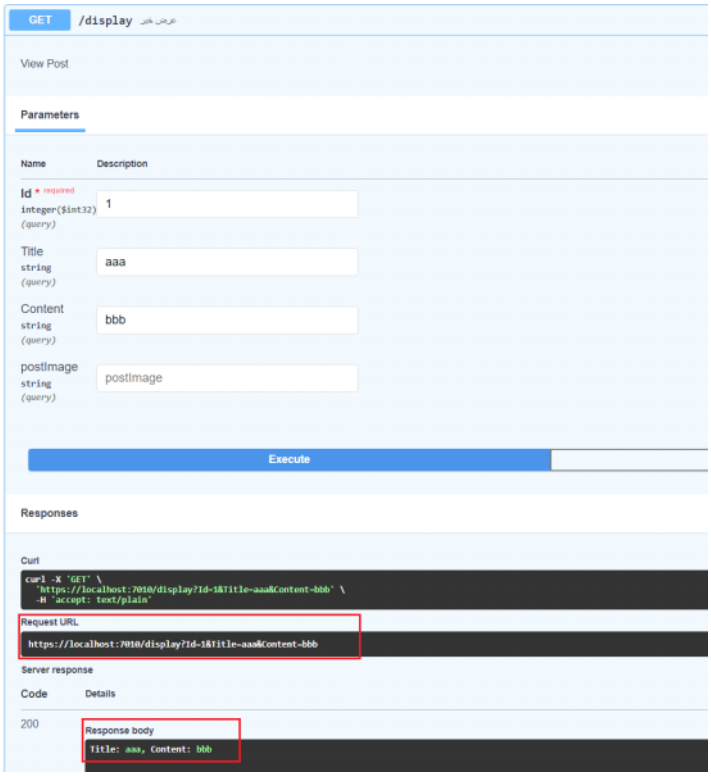
For example, consider the following minimal API endpoint that accepts an instance of type Post in the request body.



In this case, the [FromBody] attribute tells the framework to bind the parameter to the data in the request body. If this attribute is not specified, the framework will try to bind the parameter using other available sources, such as query string or route data, which is likely not what we want in this scenario.

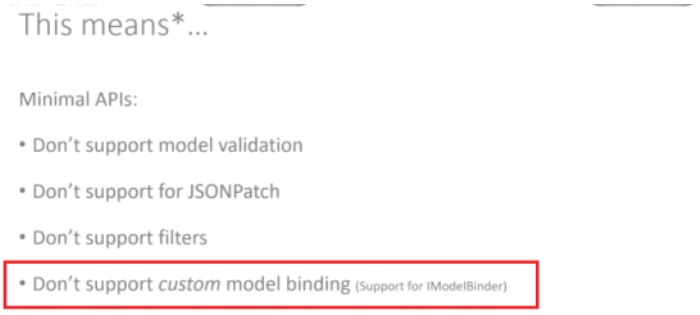
Note that you can also use the [AsParameters] attribute to map query parameters directly to an object without having to use the BindAsync or TryParse methods.


```
app.MapGet(
        "/display",
        ([AsParameters] Post post) =>
        {
            return $"Title: {post.Title}, Content: {post.Content}";
        }
    )
    .WithDescription("View  Post ")
    .WithSummary("عرض خبر  ")
    .WithTags("DBContext")
    .WithOpenApi();
```



https://code-maze.com/aspnetcore-query-string-parameters-minimal-apis/

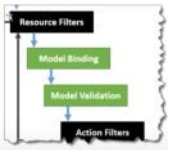
Custom model binding in minimal APIs



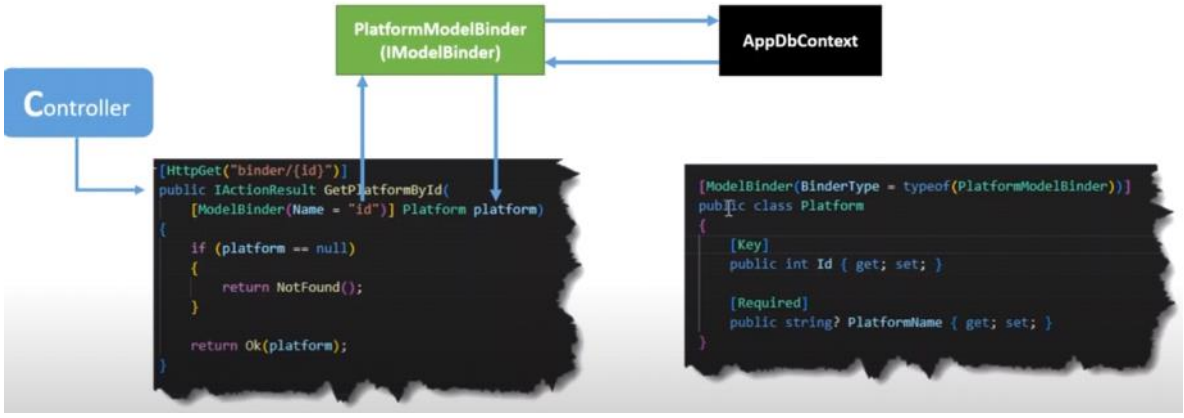
Custom model binding allows developers to define their own binding logic for complex data types or scenarios that cannot be handled by the default binding mechanisms. Custom binding is particularly useful when working with APIs that require data transformation or normalization before the data can be used by the application.
In ASP.NET Core 7 minimal APIs, custom model binding is achieved by implementing the IModelBinder interface or by using the IModelBinderProvider interface to provide a custom implementation of the IModelBinder interface for a specific data type.
To create a custom model binder, you need to implement the IModelBinder interface and override the BindModelAsync method. This method takes a BindingContext parameter, which contains information about the request and the model being bound.
In the BindModelAsync method, you can perform any necessary data transformation or validation before returning the bound model.

Custom Model Binding (IModelBinder)

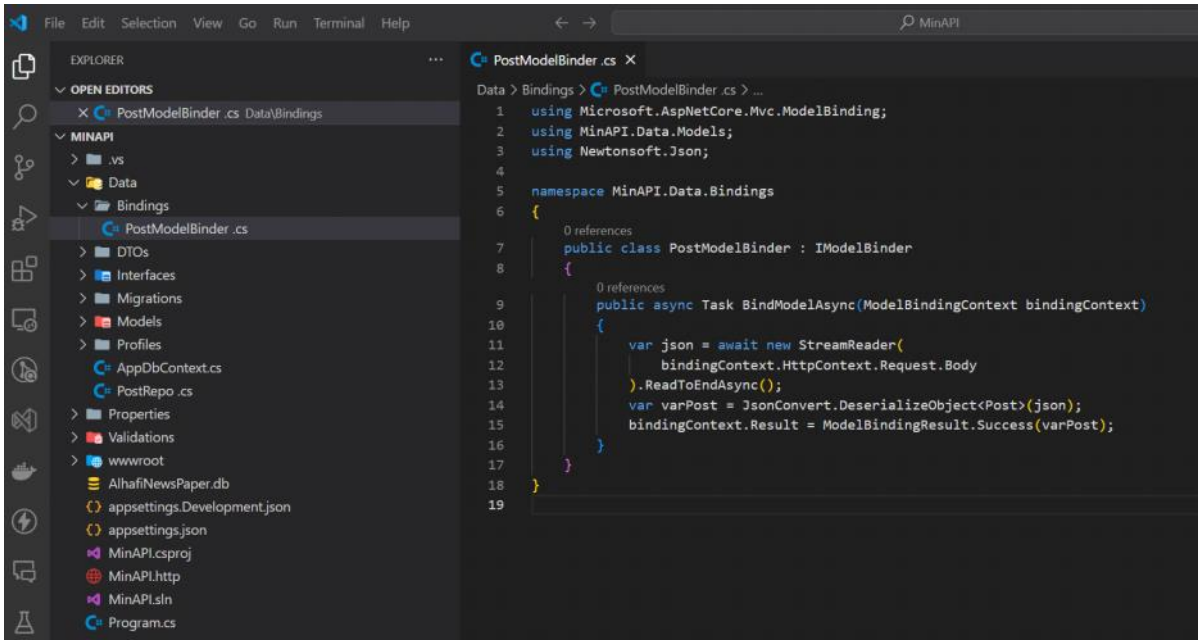
- Allow Controller Actions to work directly with Model Types
  - (Rather than HTTP requests)
- Can be used in more complex “binding scenarios”
  - E.g. if you need to transform data
- Default model binders support most common .NET types
  - They meet most developers needs!
- Model Binding Occurs before Model Validation



Example*



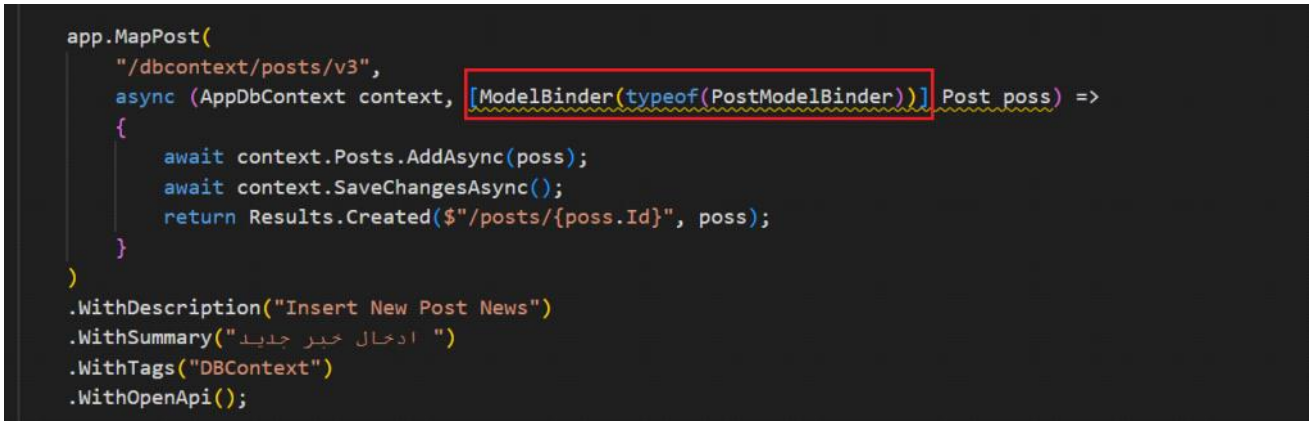
Custom Post Model Binding




```
using Microsoft.AspNetCore.Mvc.ModelBinding;
using MinAPI.Data.Models;
using Newtonsoft.Json;
namespace MinAPI.Data.Bindings
{
    public class PostModelBinder : IModelBinder
    {
        public async Task BindModelAsync(ModelBindingContext bindingContext)
        {
```

```
            var json = await new StreamReader(
                bindingContext.HttpContext.Request.Body
            ).ReadToEndAsync();
            var varPost = JsonConvert.DeserializeObject<Post>(json);
            bindingContext.Result = ModelBindingResult.Success(varPost);
        }
    }
}
```

In this example, the PostModelBinder class implements the IModelBinder interface and provides a custom implementation of the BindModelAsync method. The method reads the JSON payload from the HTTP request body and deserializes it into a Post object using the Newtonsoft.Json library. The resulting Post object is then returned as a successful ModelBindingResult.
To use this custom model binder in a minimal API endpoint, you can use the [ModelBinder] attribute on the parameter.




```
app.MapPost(
        "/dbcontext/posts/v3",
        async (AppDbContext context, [ModelBinder(typeof(PostModelBinder))] Post poss) =>
        {
            await context.Posts.AddAsync(poss);
            await context.SaveChangesAsync();
            return Results.Created($"/posts/{poss.Id}", poss);
        }
    )
    .WithDescription("Insert New Post News")
    .WithSummary("ادخال خبر جديد ")
    .WithTags("DBContext")
    .WithOpenApi();
```

In the preceding code example, the [ModelBinder] attribute specifies that the Post parameter should be bound using the PostModelBinder class.
Parameter binding simplifies the writing of code for handling HTTP requests. It allows developers to more easily handle complex data types, while simplifying code and improving code maintainability, while building the logic of their API endpoints. By taking advantage of parameter binding in minimal APIs, developers can create efficient, maintainable, and scalable APIs that meet the needs of their applications and users.

Custom Model Binding In ASP.NET Core MVC (c-sharpcorner.com)

Endpoint Filters

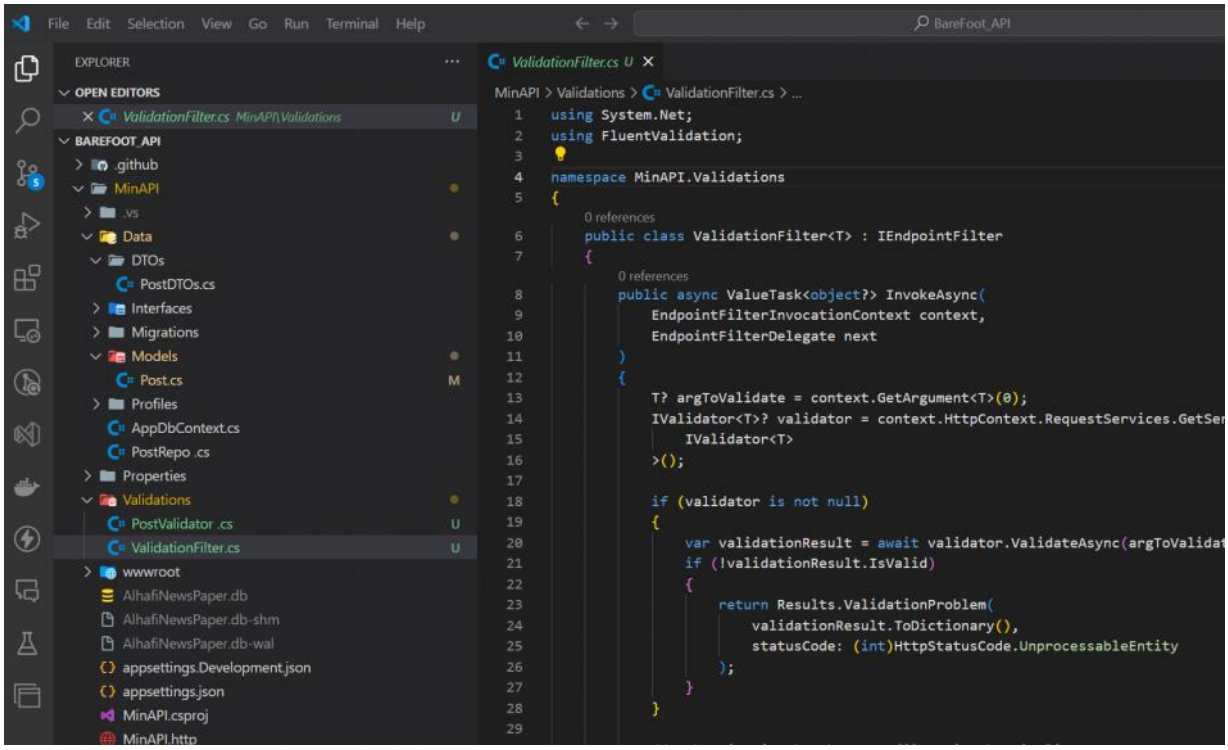
21 April 2024 13:55

This means*...

Minimal APIs:

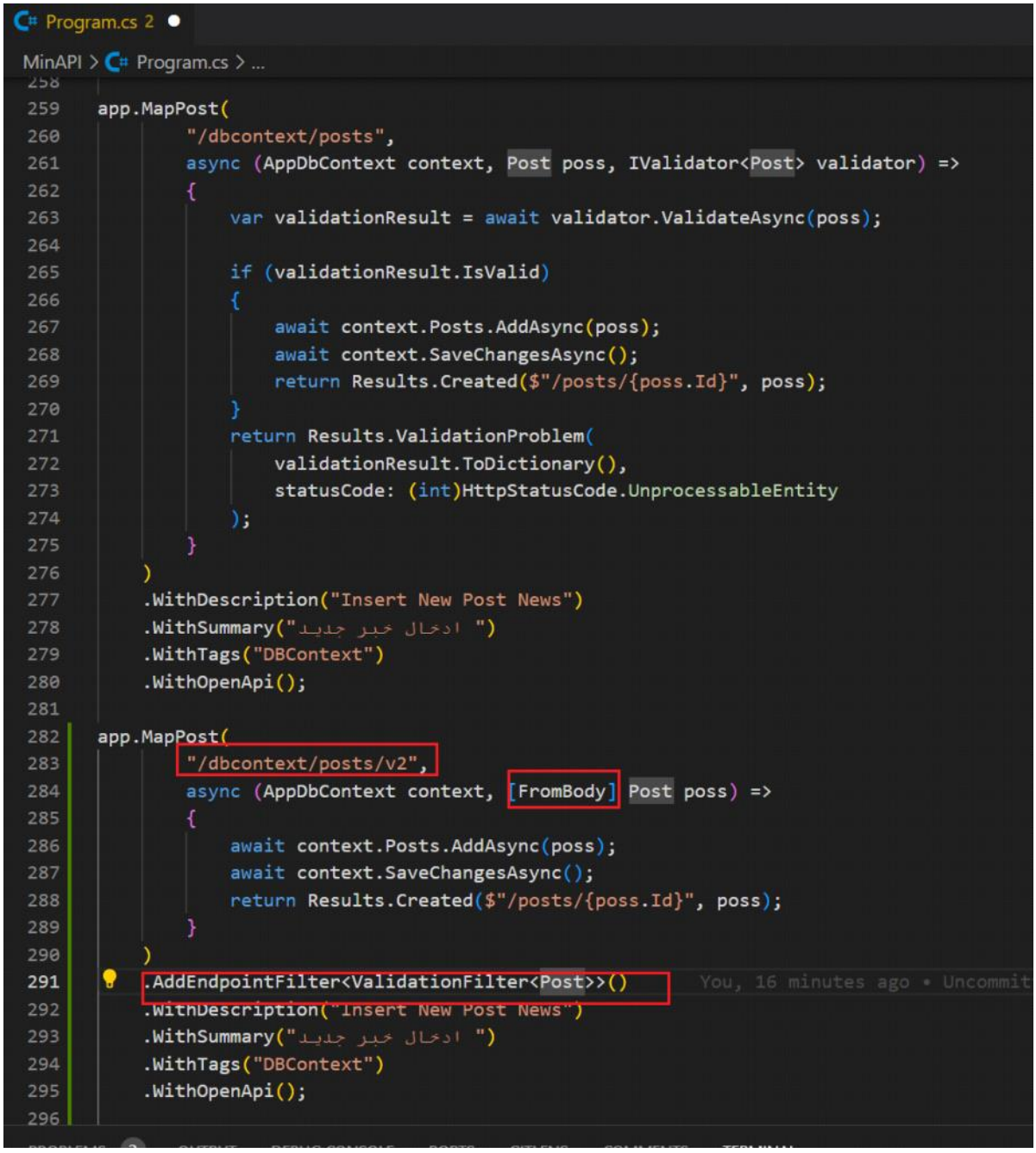
- Don't support model validation
- Don't support for JSONPatch
- Don't support filters
- Don't support *custom* model binding (Support for IModelBinder)

The great news is that as of ASP.NET 7.0 we have the ability to "opt-in" to a filter pipeline and create custom endpoint filters that work in a similar way to filters in MVC.
We can now improve on the above example by moving our validation logic into a filter:




```
using System.Net;
using FluentValidation;
namespace MinAPI.Validations
{
    public class ValidationFilter<T> : IEndpointFilter
    {
        public async ValueTask<object?> InvokeAsync(
            EndpointFilterInvocationContext ctx,
            EndpointFilterDelegate next
        )
        {
            var validator = ctx.HttpContext.RequestServices.GetService<IValidator<T>>();
            if (validator is not null)
            {
                var entity = ctx.Arguments.OfType<T>()
                    .FirstOrDefault(a => a?.GetType() == typeof(T));
                if (entity is not null)
                {
                    var validation = await validator.ValidateAsync(entity);
```

```
                if (validation.IsValid)
                {
                    return await next(ctx);
                }
                return Results.ValidationProblem(
                    validation.ToDictionary(),
                    statusCode: (int)HttpStatusCode.UnprocessableEntity
                );
            }
            else
            {
                return Results.Problem("Could not find type to validate");
            }
        }
        return await next(ctx);
    }
  }
}
```

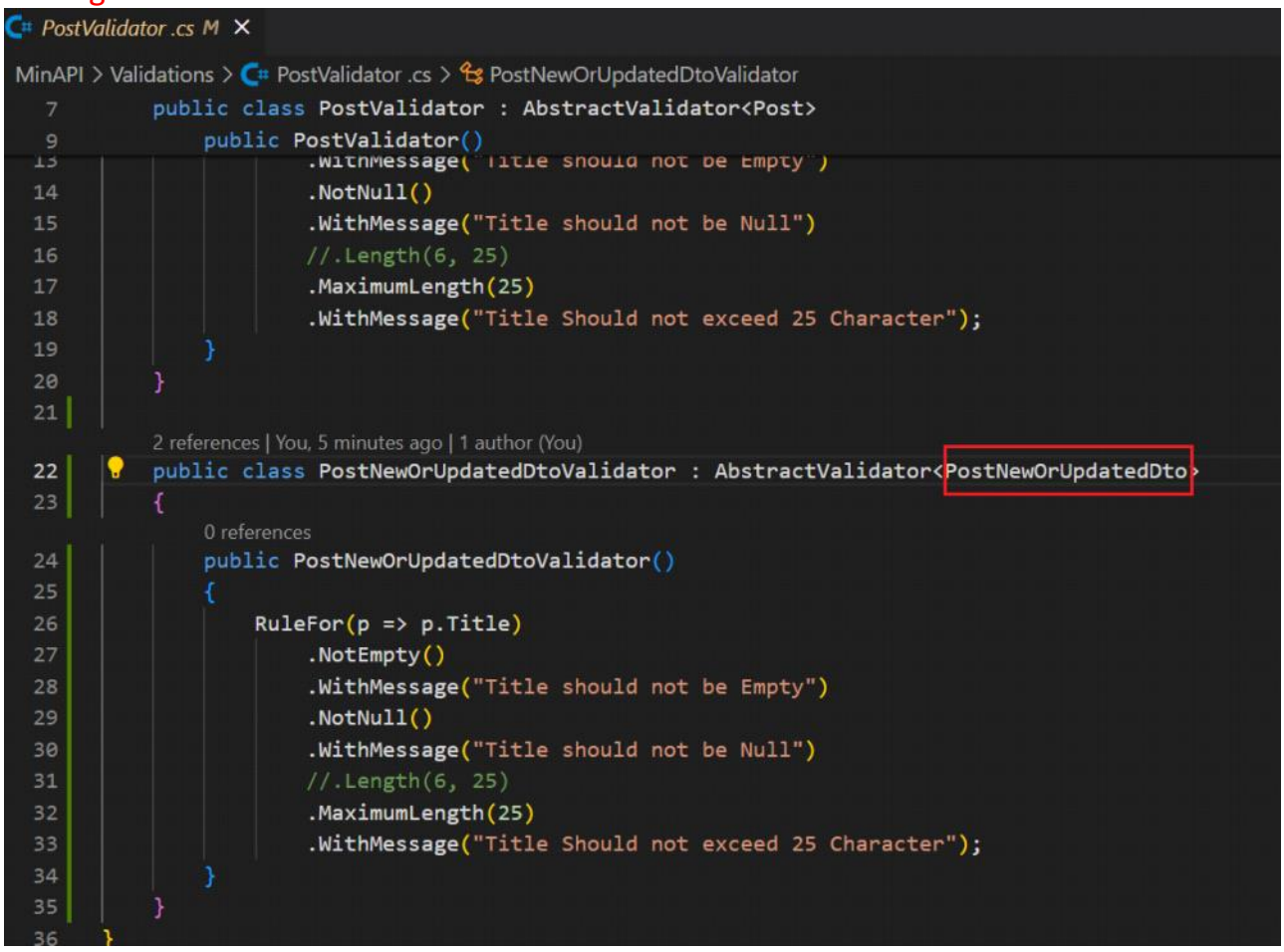



```
app.MapPost(
        "/dbcontext/posts/v2",
        async (AppDbContext context, [FromBody] Post poss) =>
        {
            await context.Posts.AddAsync(poss);
```

```
            await context.SaveChangesAsync();
            return Results.Created($"/posts/{poss.Id}", poss);
        }
    )
    .AddEndpointFilter<ValidationFilter<Post>>()
    .WithDescription("Insert New Post News")
    .WithSummary("ادخال خبر جديد ")
    .WithTags("DBContext")
    .WithOpenApi();
```

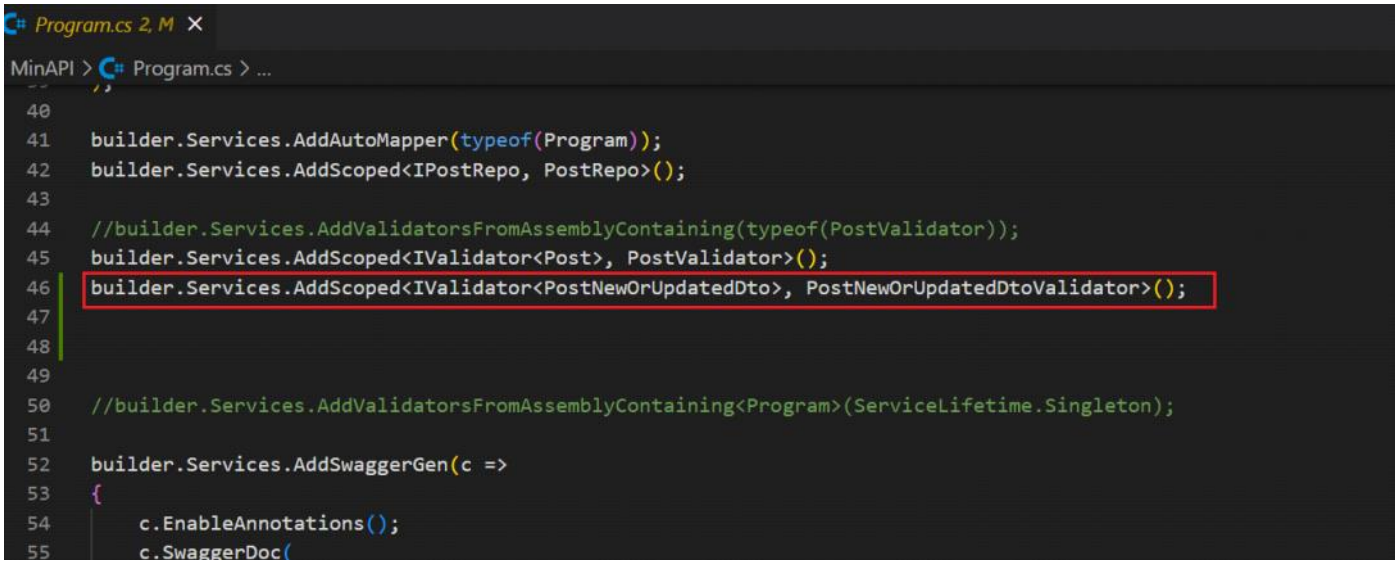
Filtration in Automapping

Adding rule to DTO




```
public class PostNewOrUpdatedDtoValidator : AbstractValidator<PostNewOrUpdatedDto>
    {
        public PostNewOrUpdatedDtoValidator()
        {
            RuleFor(p => p.Title)
                .NotEmpty()
                .WithMessage("Title should not be Empty")
                .NotNull()
                .WithMessage("Title should not be Null")
                //.Length(6, 25)
                .MaximumLength(25)
                .WithMessage("Title Should not exceed 25 Character");
        }
    }
```

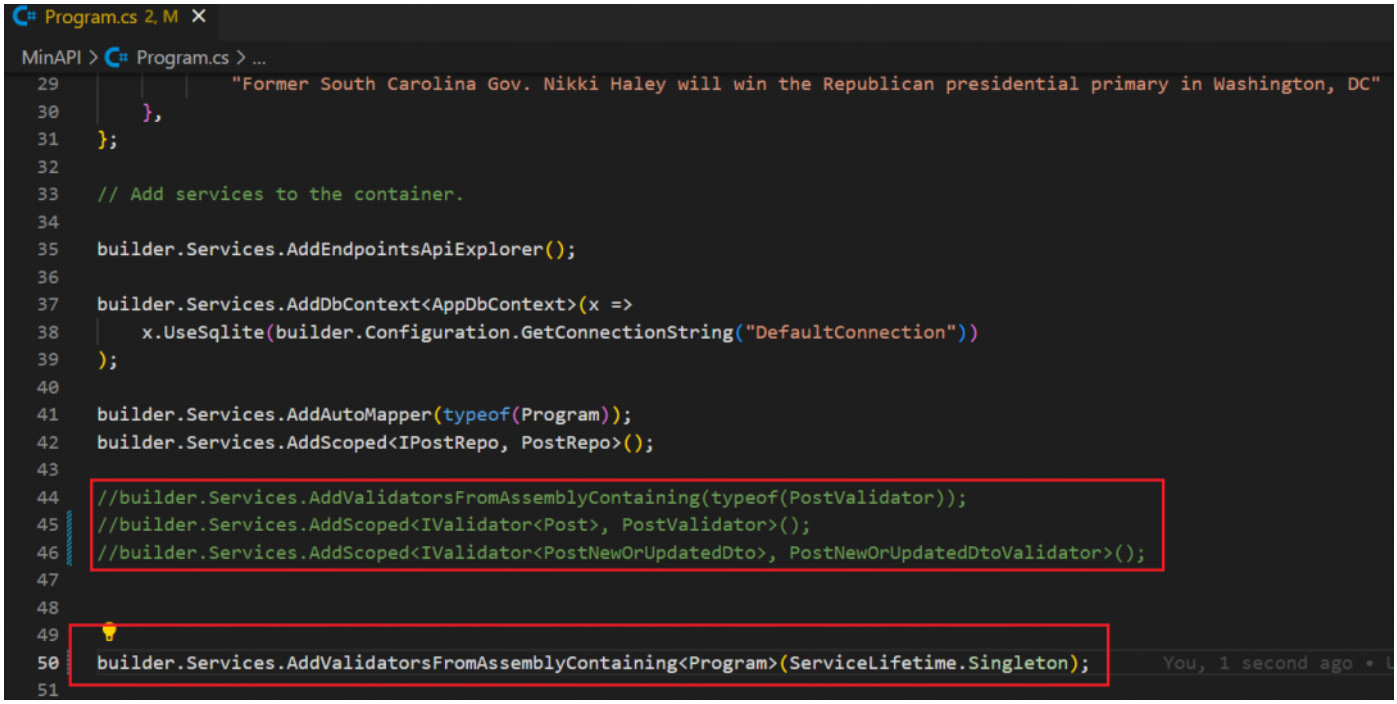
Register the validator




```
builder.Services.AddScoped<IValidator<PostNewOrUpdatedDto>, PostNewOrUpdatedDtoValidator>();
```

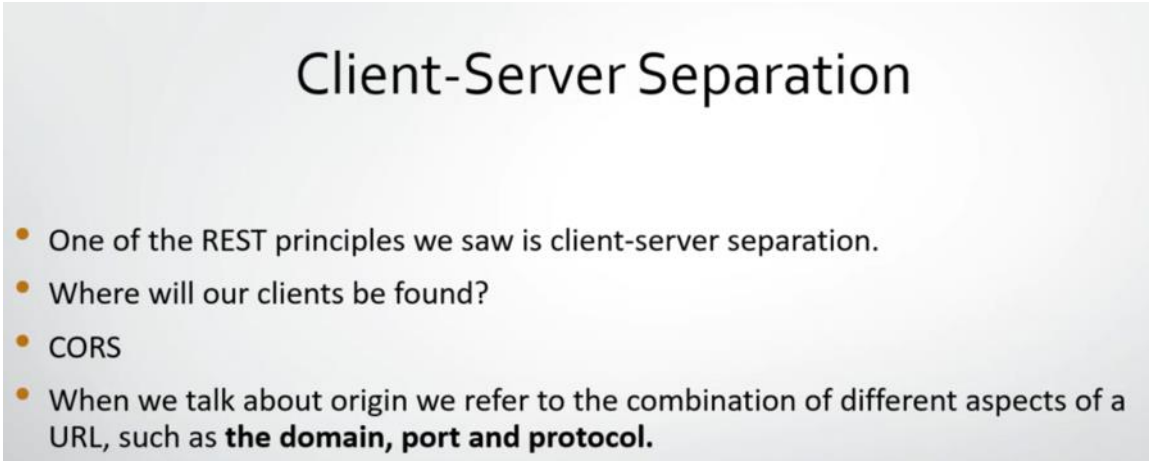

```
app.MapPost(
        "/automapper/posts",
        async (IPostRepo repo, IMapper mapper, [FromBody] PostNewOrUpdatedDto postCreateDto) =>
        {
            var postModel = mapper.Map<Post>(postCreateDto);
            await repo.CreatePost(postModel);
            await repo.SaveChanges();
            var postReadDto = mapper.Map<PostReadDto>(postModel);
            return Results.Created($"/automapper/posts/{postReadDto.Id}", postReadDto);
        }
    )
    .AddEndpointFilter<ValidationFilter<PostNewOrUpdatedDto>>()
    .WithDescription("Insert New Post News")
    .WithSummary("ادخال خبر جديد ")
    .WithTags("AutoMapper")
    .WithOpenApi();
```

Note the following call has been updated to register the validators as singleton:


Program.cs 2, M
MinAPI > Program.cs > ...
"Former South Carolina Gov. Nikki Haley will win the Republican presidential primary in Washington, DC"
},
};
// Add services to the container.
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddDbContext<AppDbContext>(x =>
x.UseSqlite(builder.Configuration.GetConnectionString("DefaultConnection"))
);
builder.Services.AddAutoMapper(typeof(Program));
builder.Services.AddScoped<IPostRepo, PostRepo>();
//builder.Services.AddValidatorsFromAssemblyContaining(typeof(PostValidator));
//builder.Services.AddScoped<IValidator<Post>, PostValidator>();
//builder.Services.AddScoped<IValidator<PostNewOrUpdatedDto>, PostNewOrUpdatedDtoValidator>();
builder.Services.AddValidatorsFromAssemblyContaining<Program>(ServiceLifetime.Singleton);
You, 1 second ago

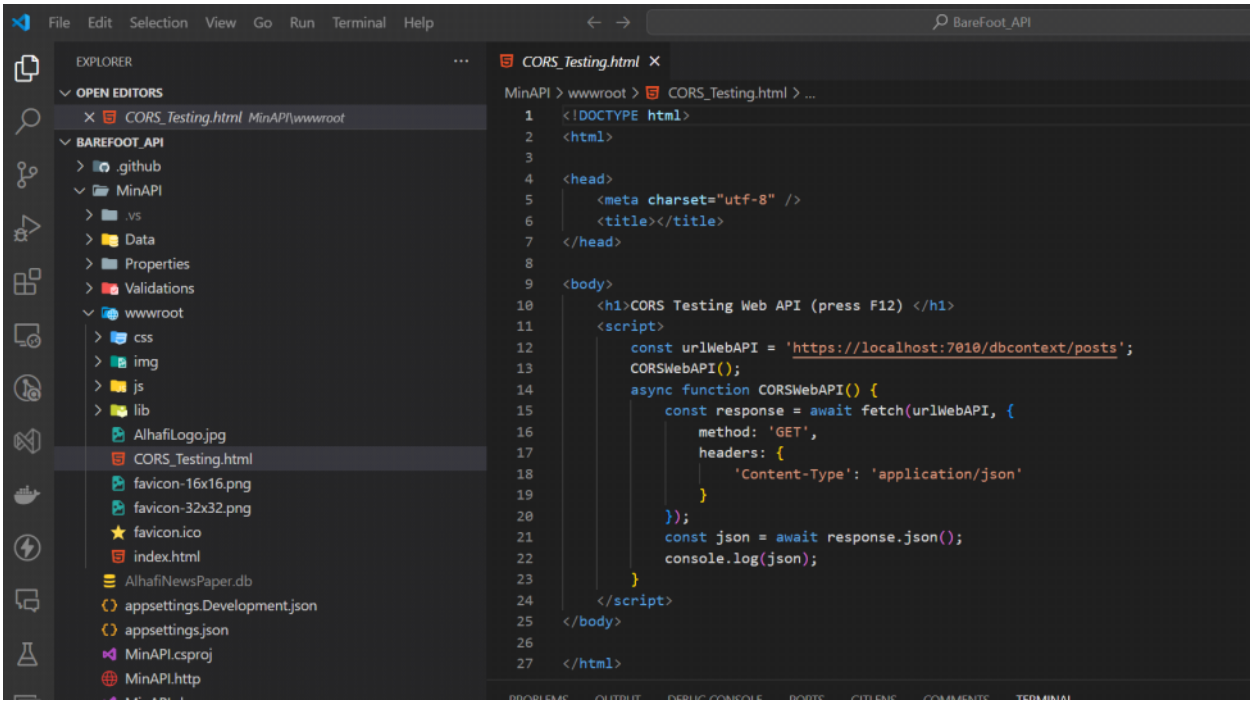
Client-Server Separation - CORS

12 May 2024 10:29



CROS checking

Let We Create HTML




```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title></title>
</head>
<body>
    <h1>CORS Testing Web API (press F12) </h1>
    <script>
        const urlWebAPI = 'https://localhost:7010/dbcontext/posts';
        CORSWebAPI();
        async function CORSWebAPI() {
            const response = await fetch(urlWebAPI, {
                method: 'GET',
                headers: {
                    'Content-Type': 'application/json'
                }
            });
```

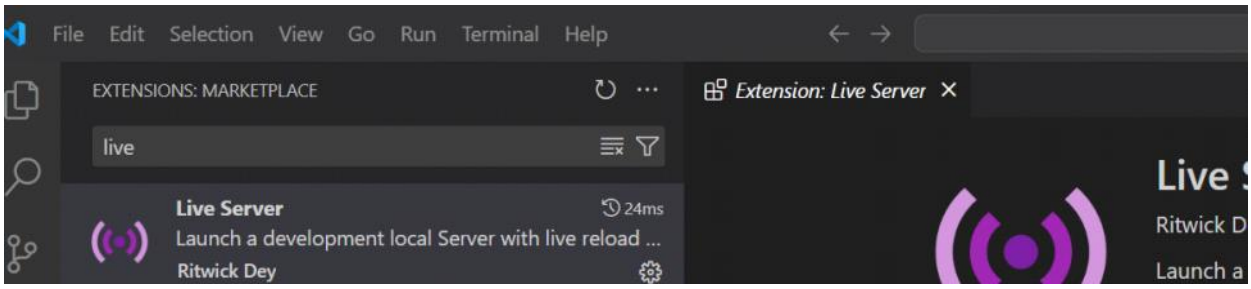
```
            const json = await response.json();
            console.log(json);
        }
    </script>
</body>
</html>
```

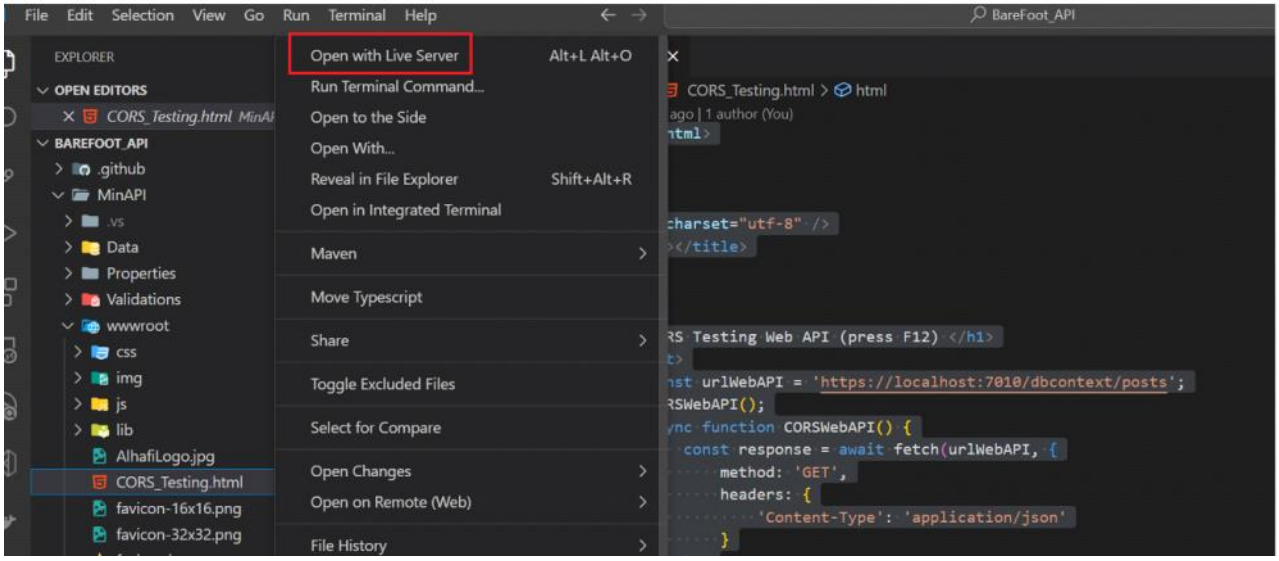
First let we run API

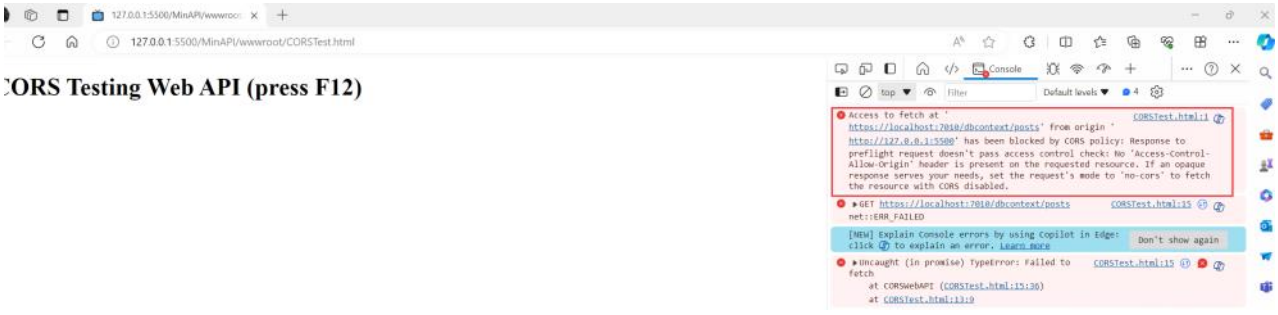
```
PS D:\VisualCode\BareFoot_API\minapi> dotnet watch run
```

Then let we run file HTML

Download extension







Let we authorize origin localhost

```
builder.Services.AddCors(options =>
        {
            options.AddPolicy(
                "MyAllowedOrigins",
                policy =>
                {
                    policy
                        .WithOrigins(
                            "https://localhost/*",
                            "http://localhost/*",
                            "http://127.0.0.1/*",
                            "https://www.alhafi.org"
                        )
                        .AllowAnyHeader()
                        .AllowAnyOrigin()
                        .AllowAnyMethod()
                        .AllowAnyHeader();
                }
            );
        });
```

```
builder.Services.AddCors(options =>
        {
            options.AddPolicy(
                "MyAllowedOrigins",
                policy =>
                {
                    policy
                        .WithOrigins(
                            "https://localhost/*",
                            "http://localhost/*",
                            "http://127.0.0.1/*",
                            "https://www.alhafi.org"
                        )
                        .AllowAnyHeader()
                        .AllowAnyOrigin()
                        .AllowAnyMethod()
                        .AllowAnyHeader();
                }
            );
        });
```

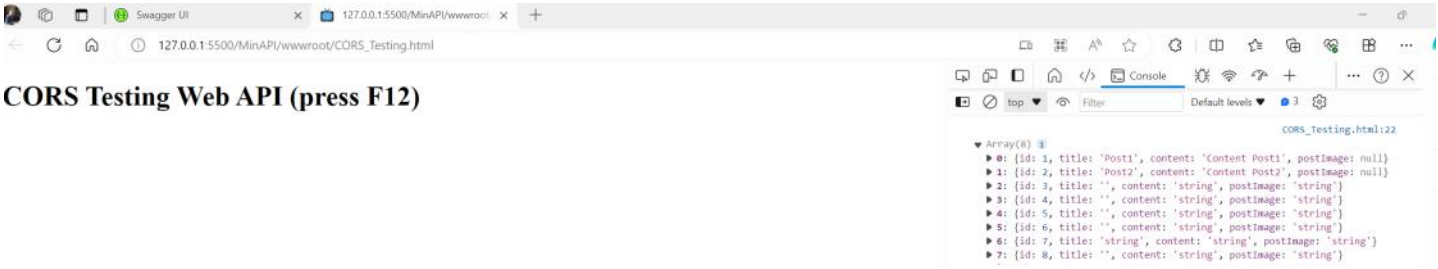
Program.cs 3, M

MinAPI > Program.cs > ...

```
134     });
135 }
136 if (app.Environment.IsStaging())
137 {
138
139     // your code here
140 }
141 if (app.Environment.IsProduction())
142 {
143
144     // your code here
145 }
146
147 app.UseHttpsRedirection();
148
149 app.UseStaticFiles();
150
151 app.UseCors("MyAllowedOrigins");
152
153 //************************************************** Ending Middle Points  *********
154
```

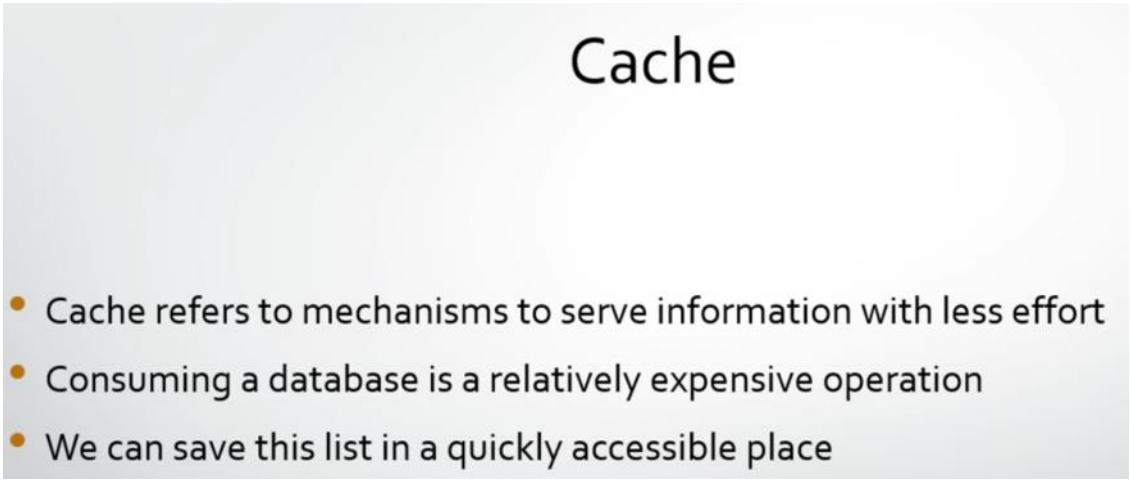
```
app.UseCors("MyAllowedOrigins");
```

Now testing again------ by allowing http requests to be made to our endpoints even they come from other sources



Cache

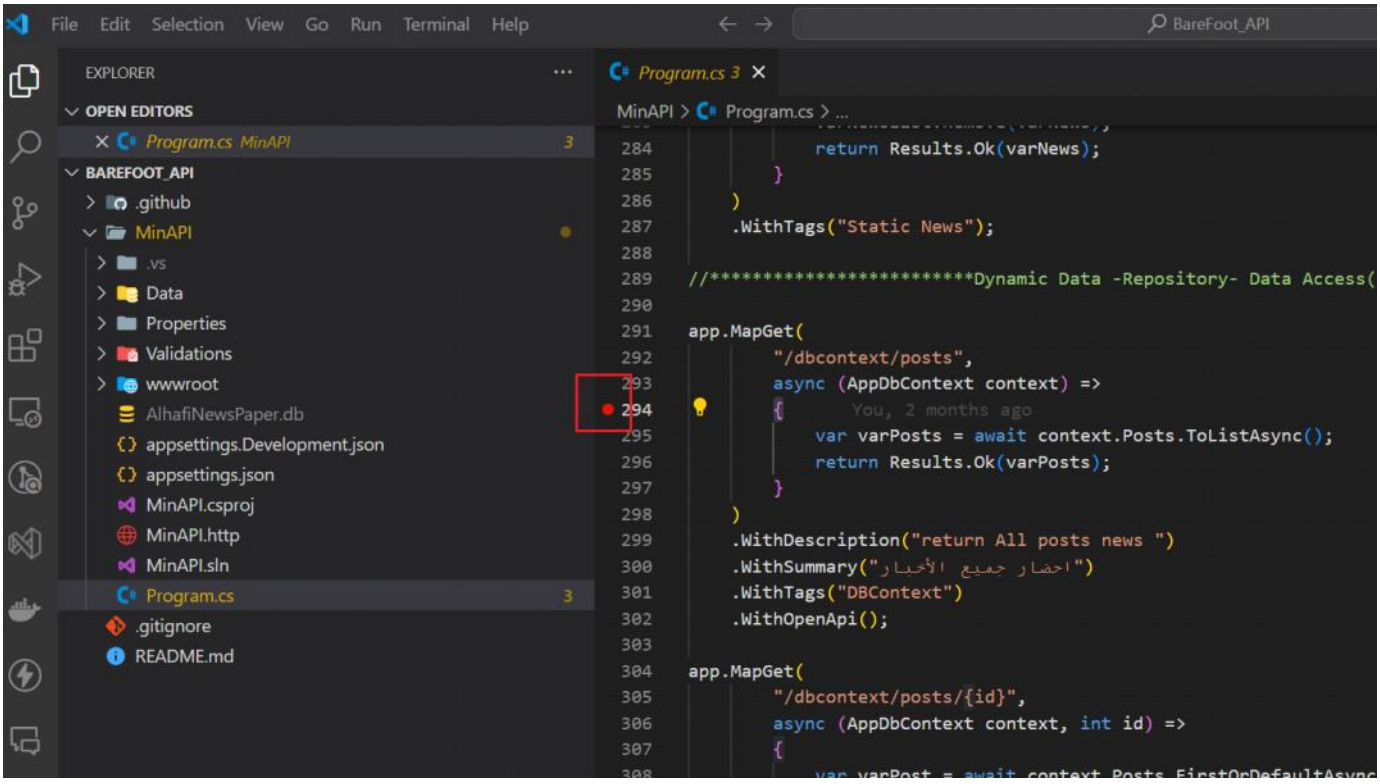
13 May 2024 07:23



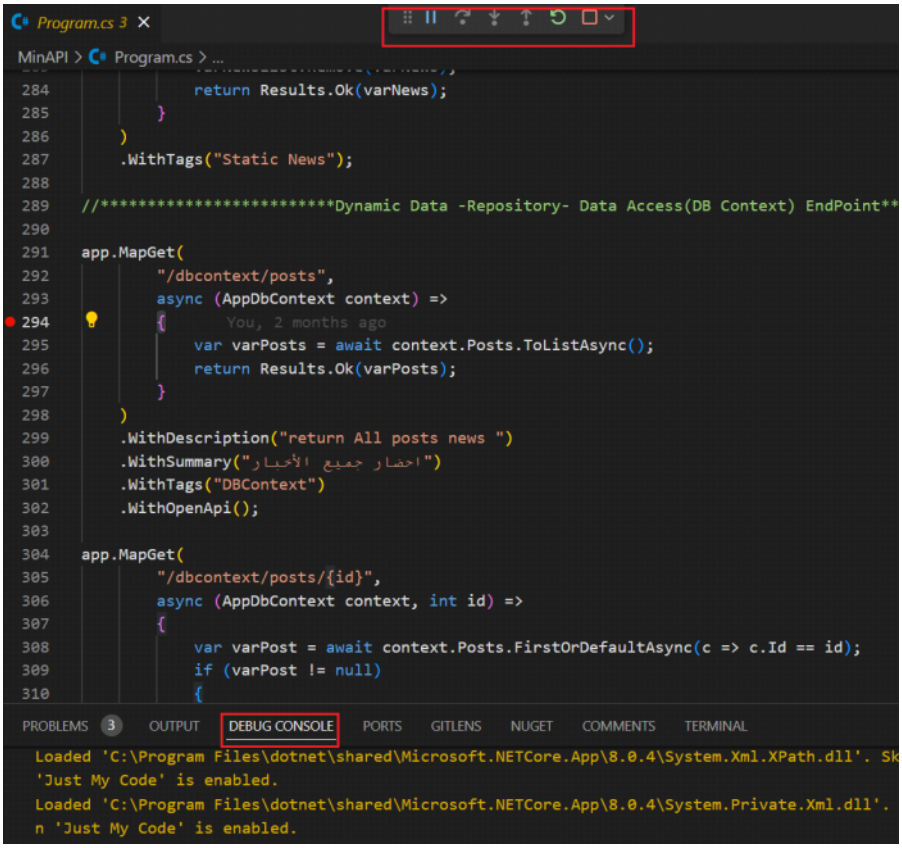
Cache it could be in server side or client side , it better to use in database specially rare to changes such as lookup table sample (countries, Nationalities ,Gender, …etc)

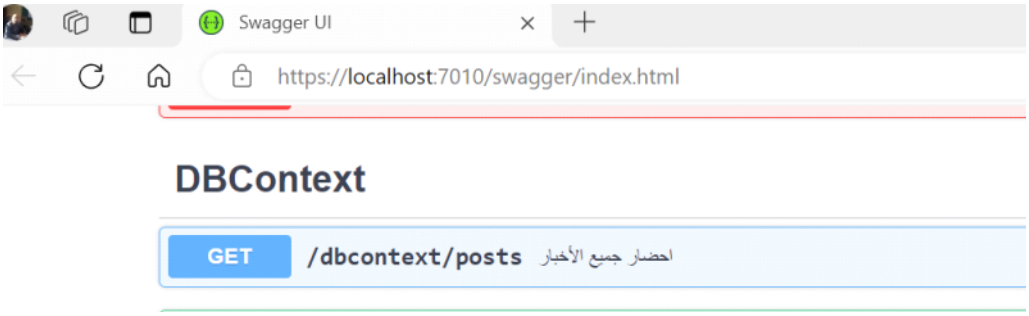
First let we see how program work without Implementing **output cache**

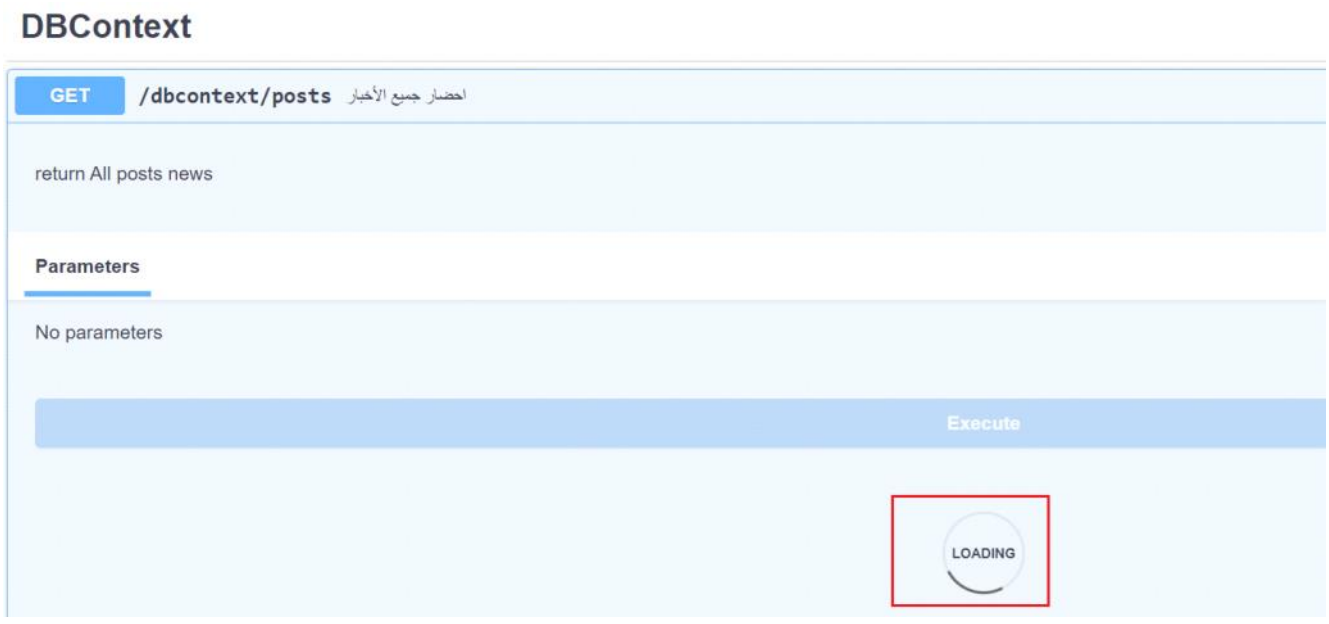
Break Point



Press F5







It will stop @ break point till we press F5

```
Program.cs 3 ×
MinAPI > Program.cs > ...
284             return Results.Ok(varNews);
285         }
286     )
287     .WithTags("Static News");
288
289 //************************Dynamic Data -Repository- Data Access(DB Context) En
290
291 app.MapGet(
292         "/dbcontext/posts",
293         async (AppDbContext context) =>
294         {       You, 2 months ago
295             var varPosts = await context.Posts.ToListAsync();
296             return Results.Ok(varPosts);
```

Then get results

```
curl -X 'GET' \
  'https://localhost:7010/dbcontext/posts' \
  -H 'accept: */*'
```

Request URL

```
https://localhost:7010/dbcontext/posts
```

Server response

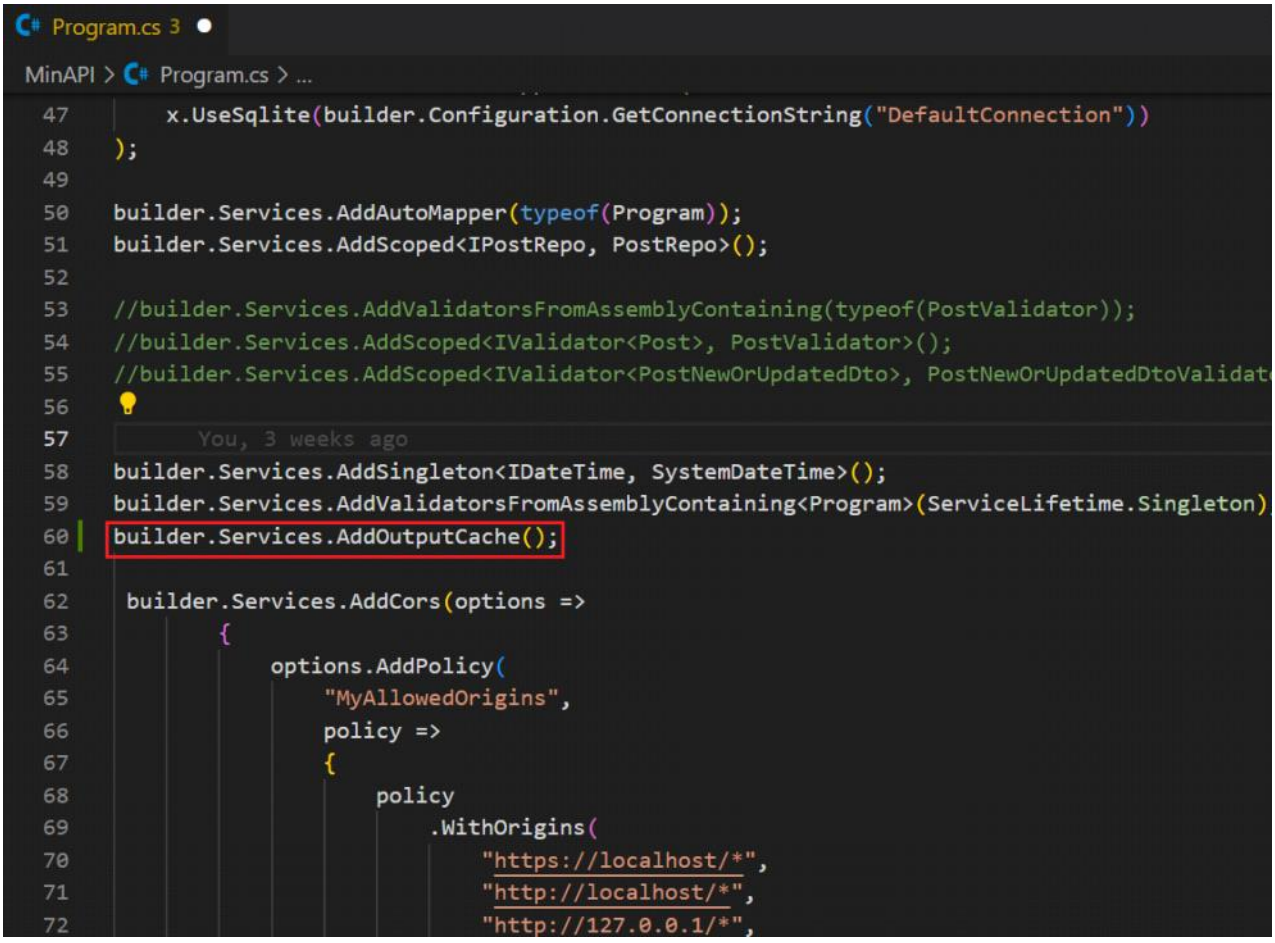
Code	Details
200	Response body

```
[
  {
    "id": 1,
    "title": "Post1",
    "content": "Content Post1",
    "postImage": null
  },
  {
    "id": 2,
    "title": "Post2",
    "content": "Content Post2",
    "postImage": null
  },
```

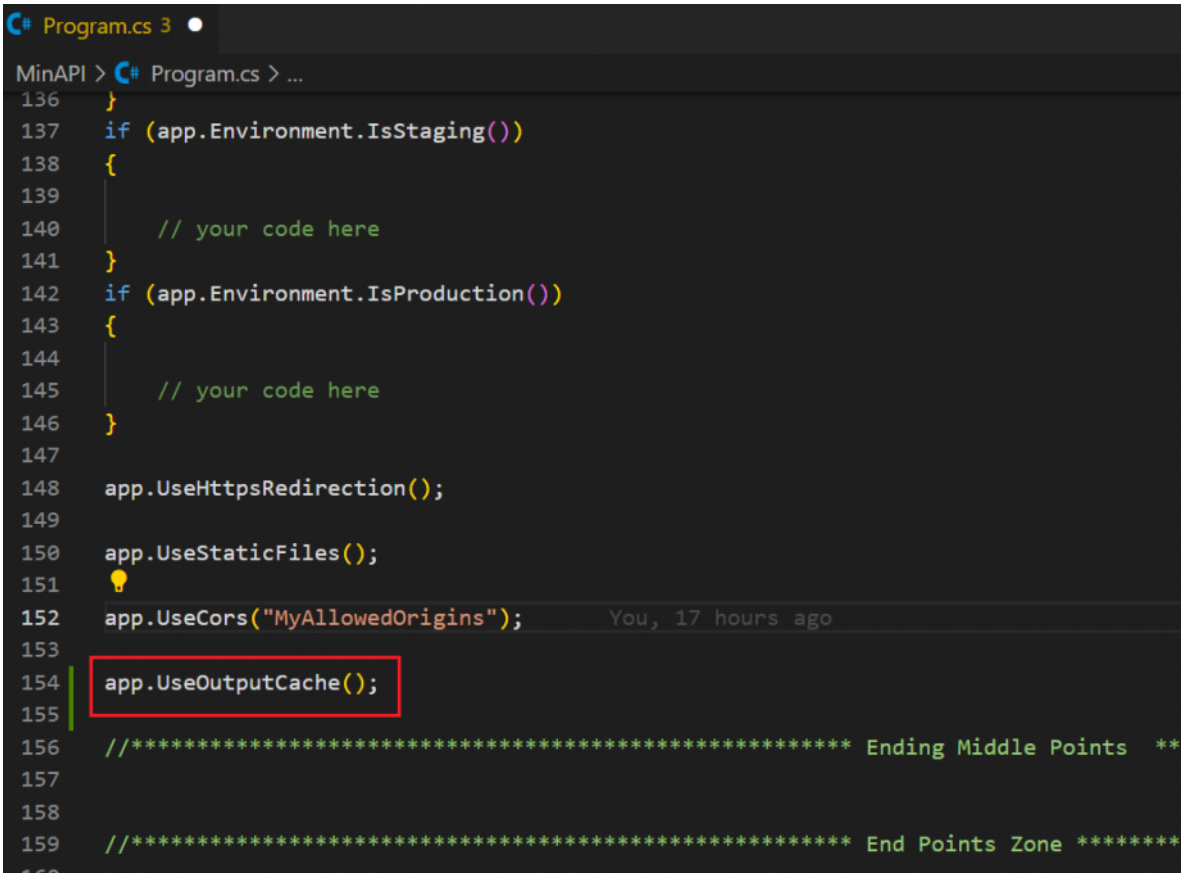
Every time we press execute it will stop @ break point

This process is expensive every time it goes to fetch data from the database and then displays it

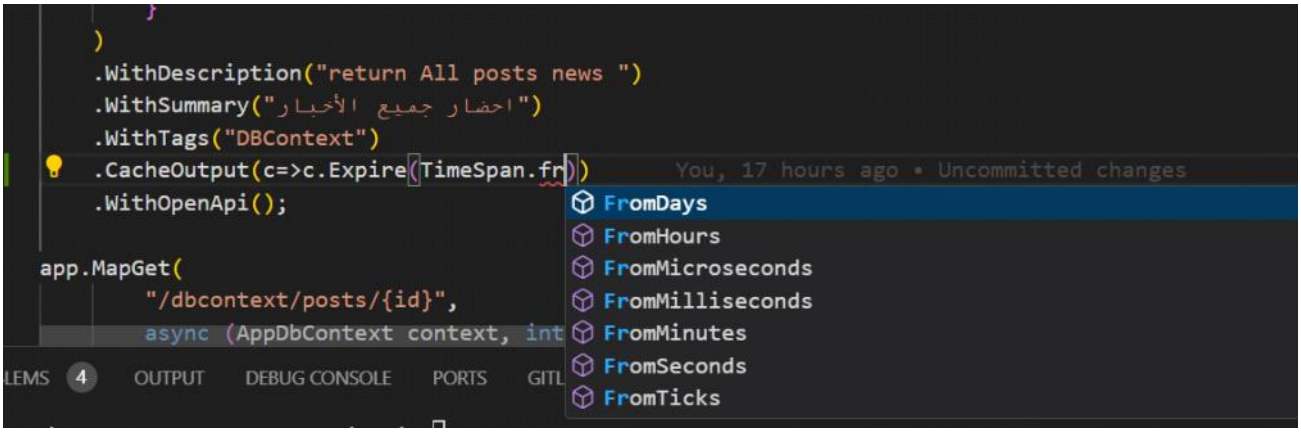
Output cache

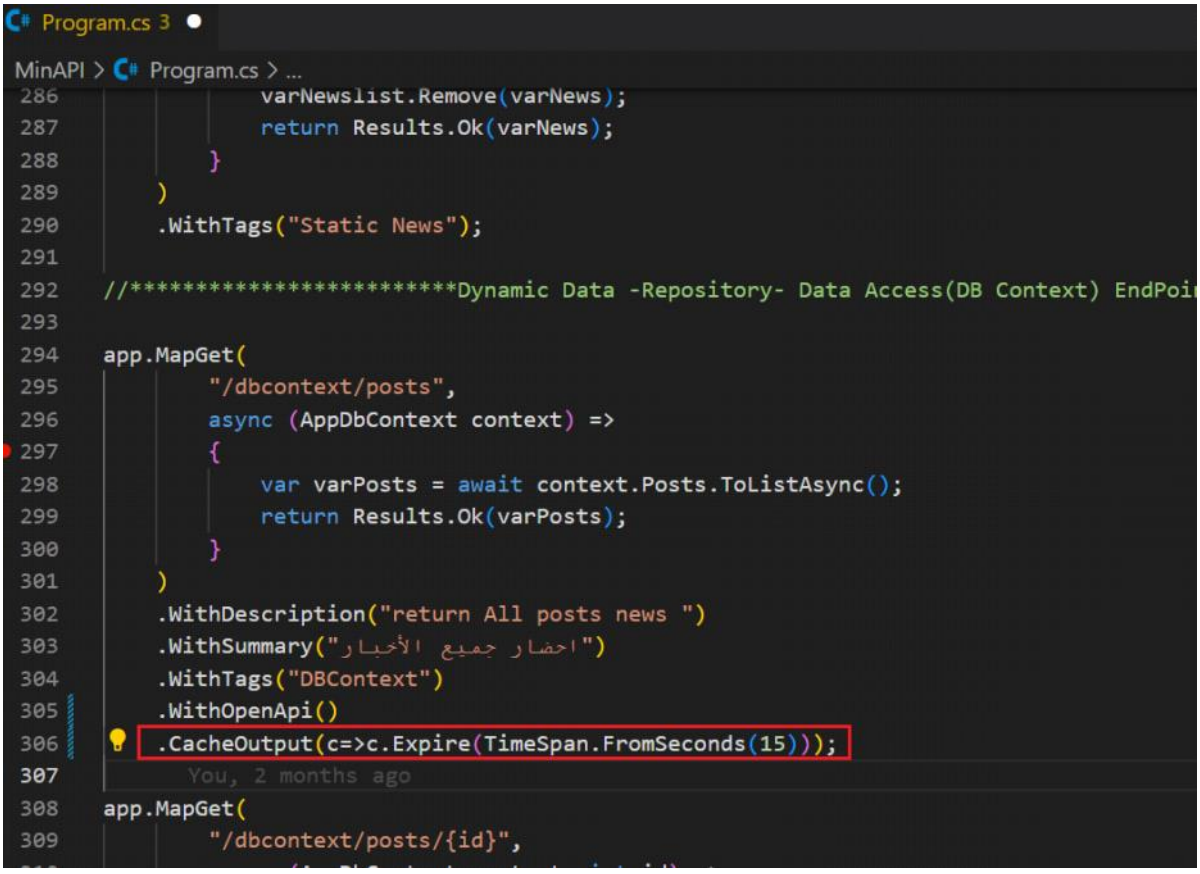


builder.Services.AddOutputCache();



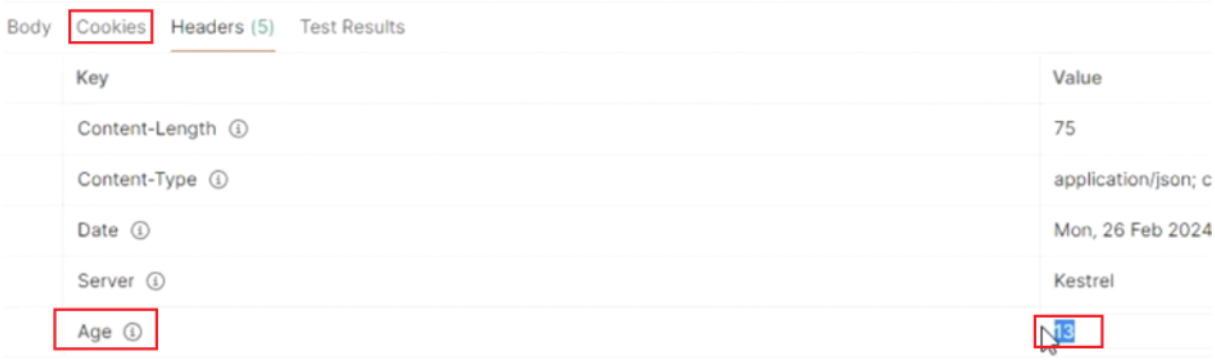
app.UseOutputCache();






```
.CacheOutput(c=>c.Expire(TimeSpan.FromSeconds(15)));
```

Test now , you will found it loading and stop @ break point one time then it will direct get data next time from output cache life will be expire after 15 second , you can find cookies life age @ below



Clean cache

What About if we want make update to table by POST or PUT or Delete , because we put output cache we will not see effect of changes till cache expire.

```
app.MapGet(
        "/dbcontext/posts",
        async (AppDbContext context) =>
        {
            var varPosts = await context.Posts.ToListAsync();
            return Results.Ok(varPosts);
        }
    )
    .WithDescription("return All posts news ")
    .WithSummary("احضار جميع الأخبار")
    .WithTags("DBContext")
    .WithOpenApi()
    .CacheOutput(c=>c.Expire(TimeSpan.FromSeconds(60)));
```

Assume I made by days

```
app.MapGet(
        "/dbcontext/posts",
        async (AppDbContext context) =>
        {
            var varPosts = await context.Posts.ToListAsync();
            return Results.Ok(varPosts);
        }
    )
    .WithDescription("return All posts news ")
    .WithSummary("احضار جميع الأخبار")
    .WithTags("DBContext")
    .WithOpenApi()
    .CacheOutput(c=>c.Expire(TimeSpan.FromDays(360)));
```

So let we solve this by cleaning cache tag

```
app.MapGet(
        "/dbcontext/posts",
        async (AppDbContext context) =>
        {
            var varPosts = await context.Posts.ToListAsync();
            return Results.Ok(varPosts);
        }
    )
    .WithDescription("return All posts news ")
    .WithSummary("احضار جميع الأخبار")
    .WithTags("DBContext")
    .WithOpenApi()
    .CacheOutput(c=>c.Expire(TimeSpan.FromDays(360)).Tag("Post_Get"));
```

```
.CacheOutput(c=>c.Expire(TimeSpan.FromDays(360)).Tag("Post_Get"));
```

Let we get instance injection for output cache

```
app.MapPost(
        "/dbcontext/posts",
        async (AppDbContext context, Post poss, IValidator<Post> validator, IOutputCacheStore outputCacheRestore) =>
        {
            var validationResult = await validator.ValidateAsync(poss);
    You, 3 weeks ago
            if (validationResult.IsValid)
            {
                await context.Posts.AddAsync(poss);
                await context.SaveChangesAsync();
                return Results.Created($"/posts/{poss.Id}", poss);
            }
            return Results.ValidationProblem(
                validationResult.ToDictionary(),
                statusCode: (int)HttpStatusCode.UnprocessableEntity
```

Let we clear cache by tag name , and use default cancelation token (default) after saving and before viewing results

```
app.MapPost(
        "/dbcontext/posts",
        async (AppDbContext context, Post poss, IValidator<Post> validator, IOutputCacheStore outputCacheRes
        {
            var validationResult = await validator.ValidateAsync(poss);

            if (validationResult.IsValid)
            {
                await context.Posts.AddAsync(poss);
                await context.SaveChangesAsync();
                await outputCacheRestore.EvictByTagAsync("Post_Get",default);      You, 8 minutes ago • Unc
                return Results.Created($"/posts/{poss.Id}", poss);
            }
            return Results.ValidationProblem(
                validationResult.ToDictionary(),
                statusCode: (int)HttpStatusCode.UnprocessableEntity
            );
```

```
await outputCacheRestore.EvictByTagAsync("Post_Get",default);
```

Now if you make post , you will find changes by call get , So repeat above in put and delete methods

Map Group

13 May 2024 08:56

Advantages of Using Map Group

Improved Organization: Map Group significantly enhances the organization of endpoints by logically grouping related actions, making the API easier to understand and maintain

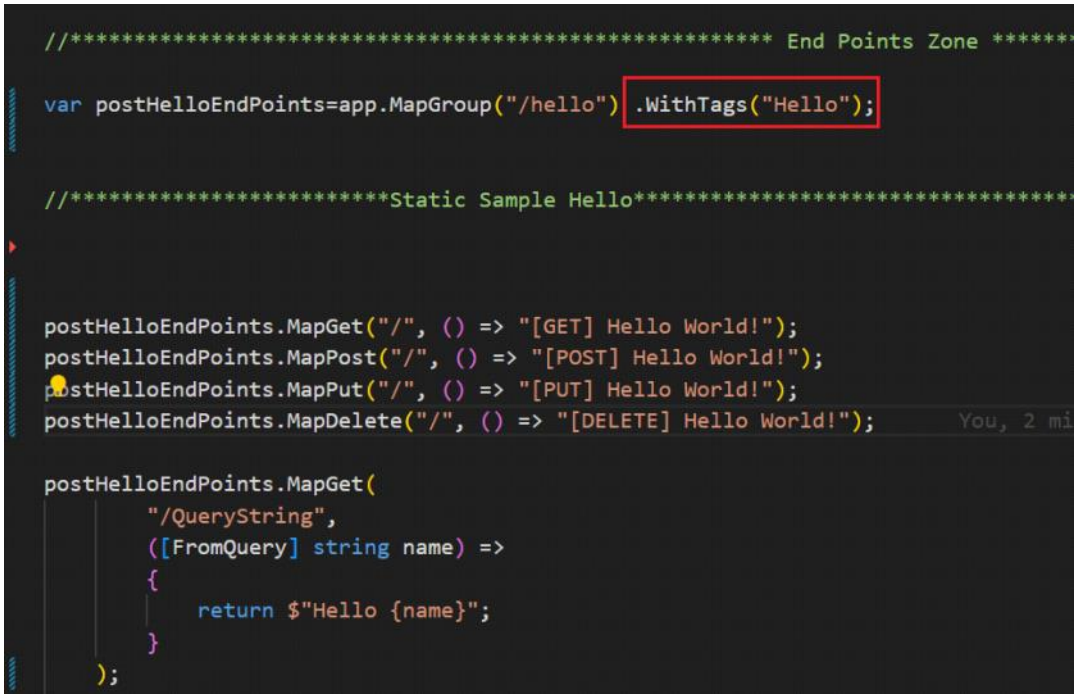
Let we see changes for below

```
//************************Static Sample Hello*******************************

app.MapGet("/", () => varMyEnv);

app.MapGet("/hello", () => "[GET] Hello World!").WithTags("Hello");
app.MapPost("/hello", () => "[POST] Hello World!").WithTags("Hello");
app.MapPut("/hello", () => "[PUT] Hello World!").WithTags("Hello");
app.MapDelete("/hello", () => "[DELETE] Hello World!").WithTags("Hello");

app.MapGet(
        "/hello/QueryString",
        ([FromQuery] string name) =>       You, 2 weeks ago
        {
            return $"Hello {name}";
        }
    )
    .WithTags("Hello");
```

To Map Group

```
Program.cs 3, M
MinAPI > Program.cs > ...
153  //************************************************** End Points Zone ***************************
154
155  var postHelloEndPoints=app.MapGroup("/hello");
156
157  //************************Static Sample Hello*****************************************
158
159  app.MapGet("/", () => varMyEnv);
160
161  postHelloEndPoints.MapGet("/", () => "[GET] Hello World!").WithTags("Hello");
162  postHelloEndPoints.MapPost("/", () => "[POST] Hello World!").WithTags("Hello");
163  postHelloEndPoints.MapPut("/", () => "[PUT] Hello World!").WithTags("Hello");
164  postHelloEndPoints.MapDelete("/", () => "[DELETE] Hello World!").WithTags("Hello");
165
166  postHelloEndPoints.MapGet(
167          "/QueryString",
168          ([FromQuery] string name) =>
169          {
170              return $"Hello {name}";
171          }       You, 2 weeks ago
172      )
173      .WithTags("Hello");
```

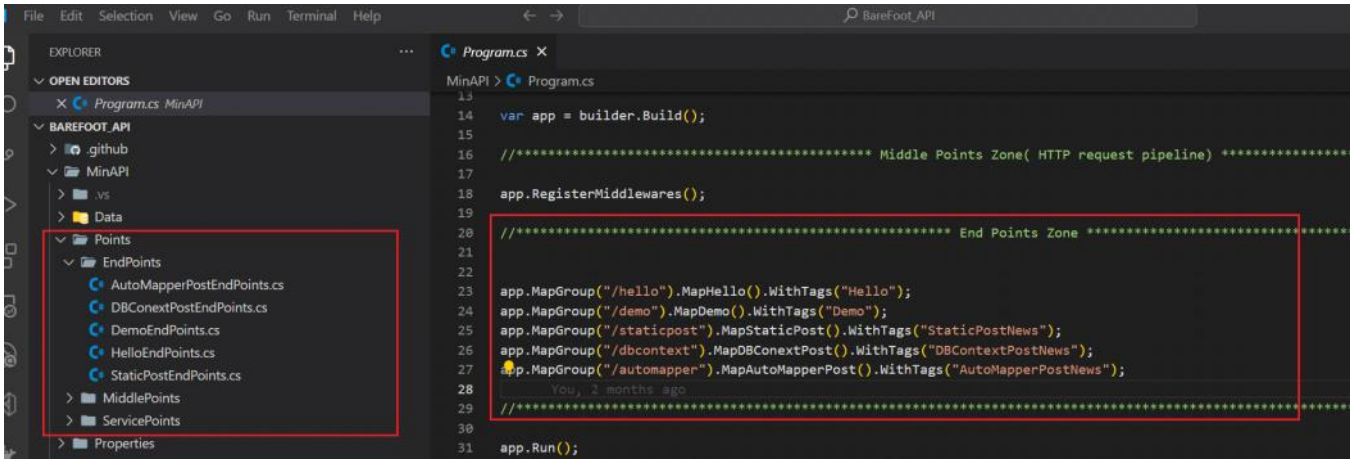
Even we can handle extended such as tag



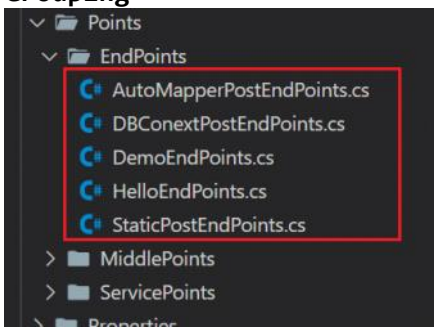
```
//************************************************** End Points Zone *******

var postHelloEndPoints=app.MapGroup("/hello") .WithTags("Hello");


//************************Static Sample Hello*********************************


postHelloEndPoints.MapGet("/", () => "[GET] Hello World!");
postHelloEndPoints.MapPost("/", () => "[POST] Hello World!");
postHelloEndPoints.MapPut("/", () => "[PUT] Hello World!");
postHelloEndPoints.MapDelete("/", () => "[DELETE] Hello World!");       You, 2 mi

postHelloEndPoints.MapGet(
        "/QueryString",
        ([FromQuery] string name) =>
        {
            return $"Hello {name}";
        }
    );
```

Grouping End Points

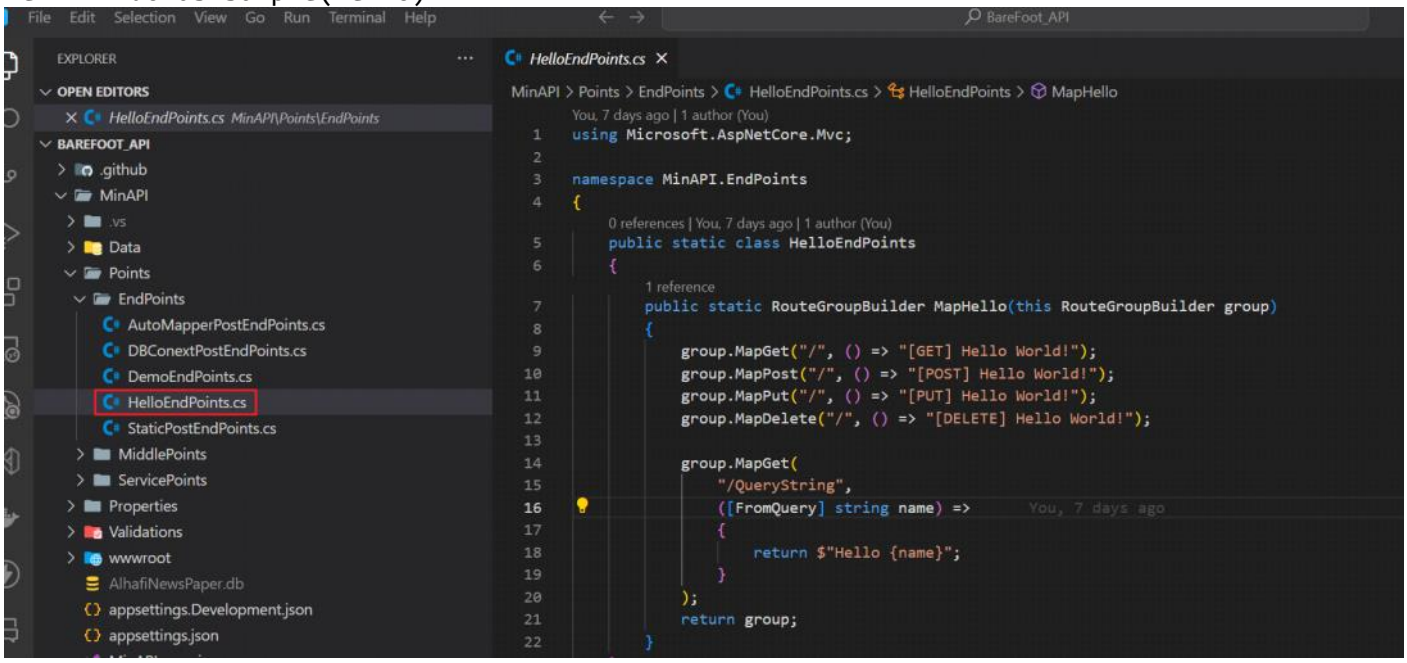
13 May 2024 13:12



Grouping

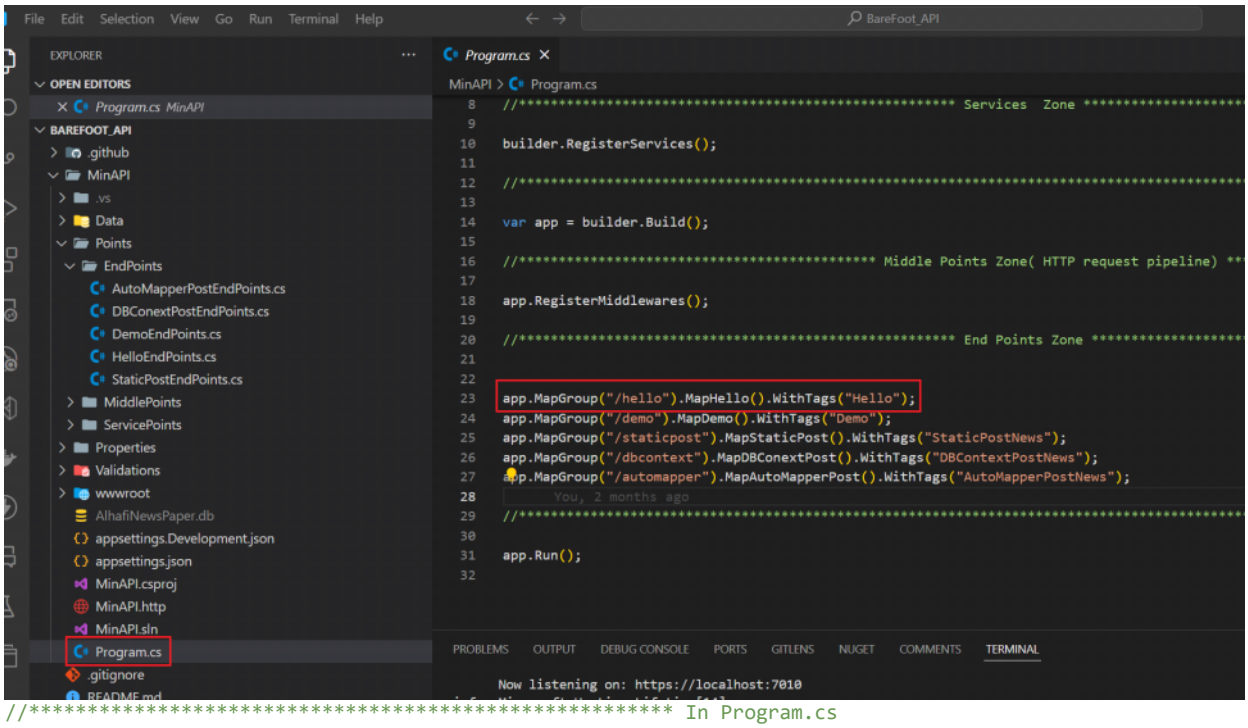


We Will do as sample(Hello)



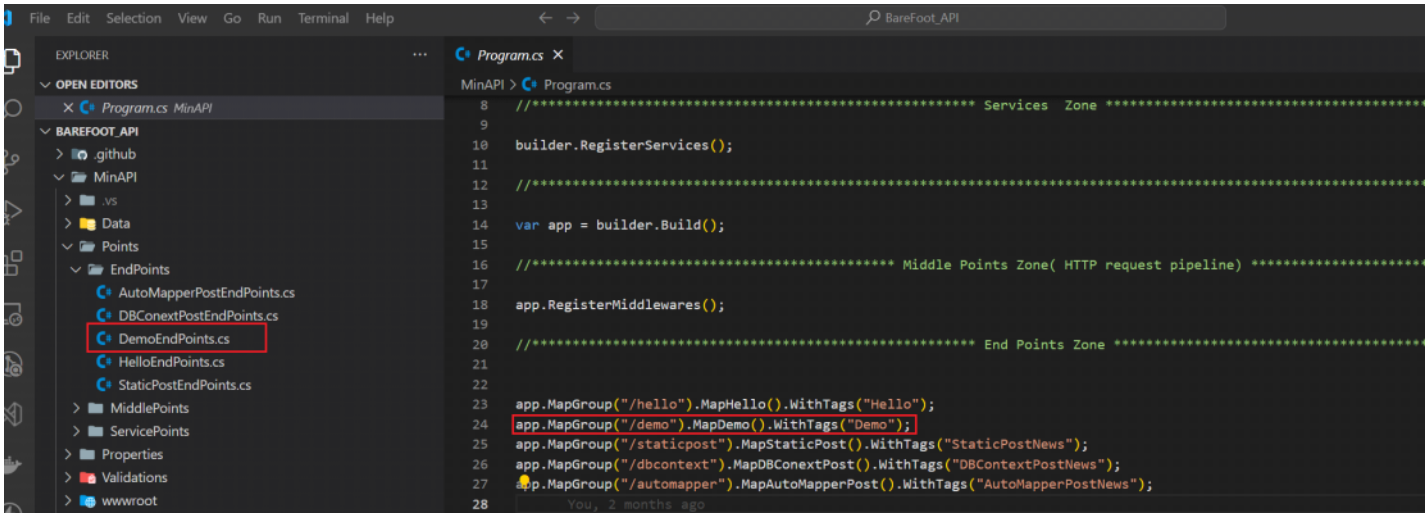

```
using Microsoft.AspNetCore.Mvc;
namespace MinAPI.EndPoints
{
    public static class HelloEndPoints
    {
        public static RouteGroupBuilder MapHello(this RouteGroupBuilder
group)
        {
            group.MapGet("/", () => "[GET] Hello World!");
            group.MapPost("/", () => "[POST] Hello World!");
```

```
            group.MapPut("/", () => "[PUT] Hello World!");
            group.MapDelete("/", () => "[DELETE] Hello World!");
            group.MapGet(
                "/QueryString",
                ([FromQuery] string name) =>
                {
                    return $"Hello {name}";
                }
            );
            return group;
        }
    }
}
```




```
//********** In Program.cs
**************************************************

app.MapGroup("/hello").MapHello().WithTags("Hello");
app.MapGroup("/demo").MapDemo().WithTags("Demo");
app.MapGroup("/staticpost").MapStaticPost().WithTags("StaticPostNews");
app.MapGroup("/dbcontext").MapDBConextPost().WithTags("DBContextPostNews");
app.MapGroup("/automapper").MapAutoMapperPost().WithTags("AutoMapperPostNews
");
//
**************************************************************************
**************
```

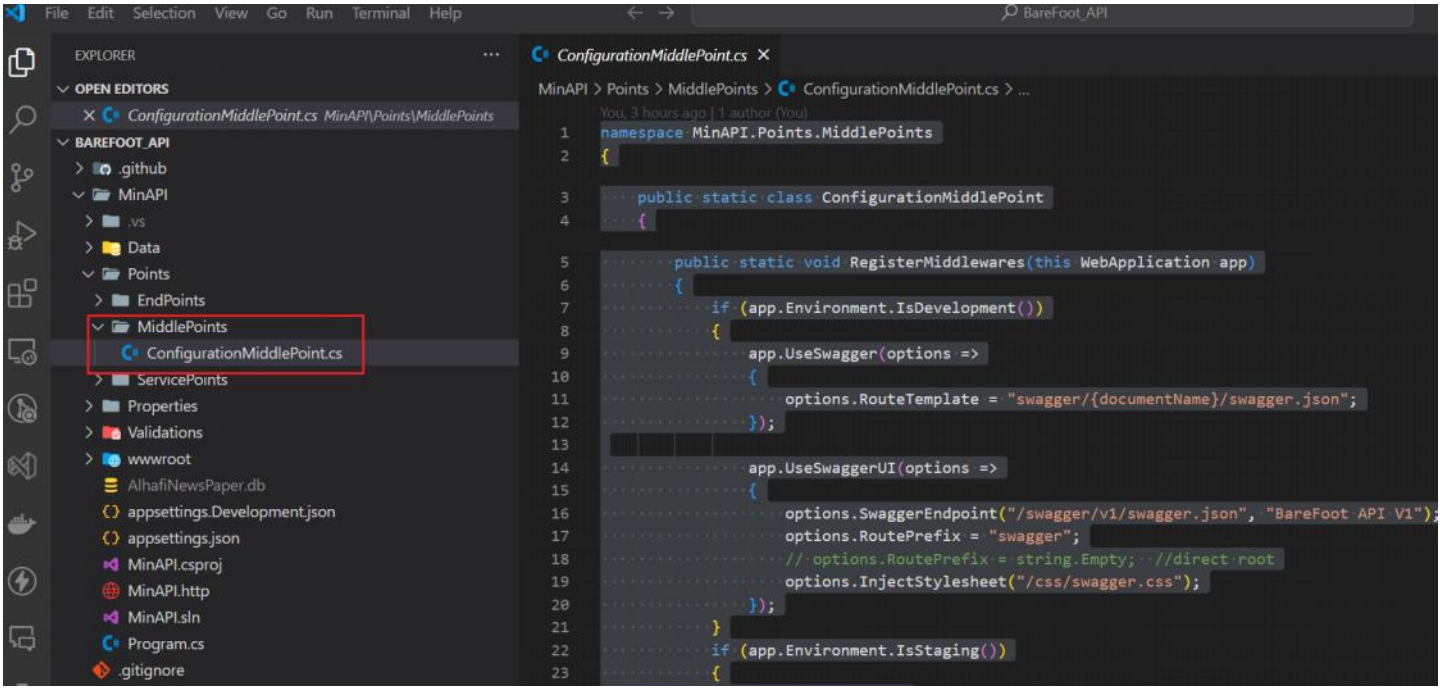
We Will do as sample(Demo)



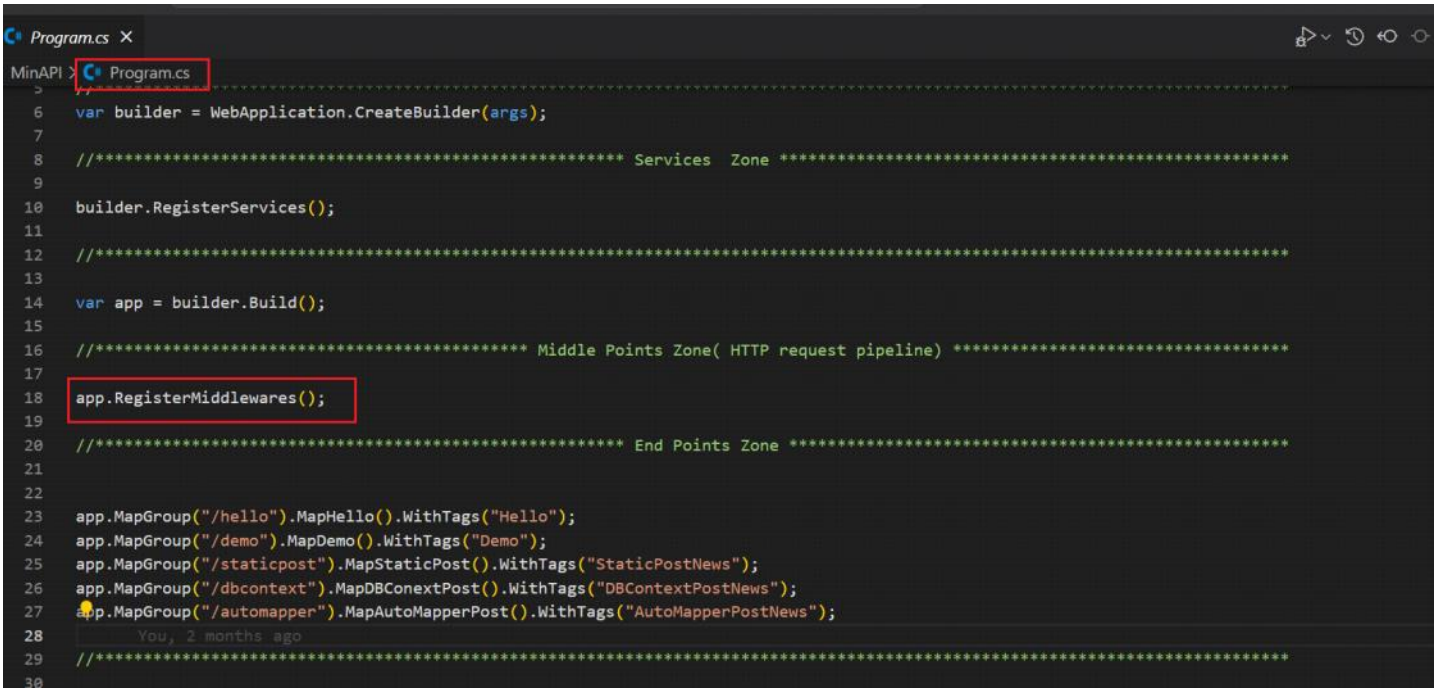
```
using Microsoft.AspNetCore.Mvc;
using MinAPI.Data.Models;
namespace MinAPI.EndPoints
{
    public static class DemoEndPoints
    {
        public static RouteGroupBuilder MapDemo(this RouteGroupBuilder
group)
        {
            var builder = WebApplication.CreateBuilder();
            var varmyenv = builder.Configuration.GetValue<string>("myenv");
            group.MapGet("/", () => varmyenv);
            group.MapGet("/FromModel", (IDateTime dateTime) =>
dateTime.Now);
            group.MapGet("/FromRegisteredService", ([FromServices] IDateTime
dateTime) => dateTime.Now);
            return group;
        }
    }
}
```

Grouping Middle Points

20 May 2024 14:39

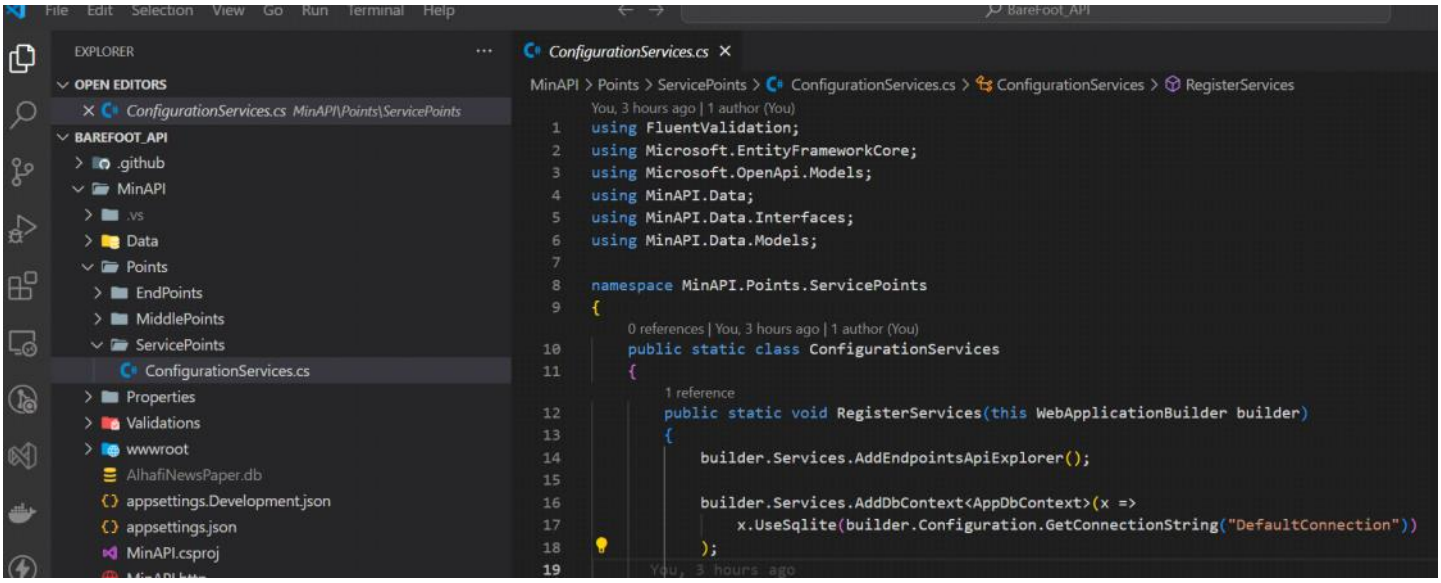


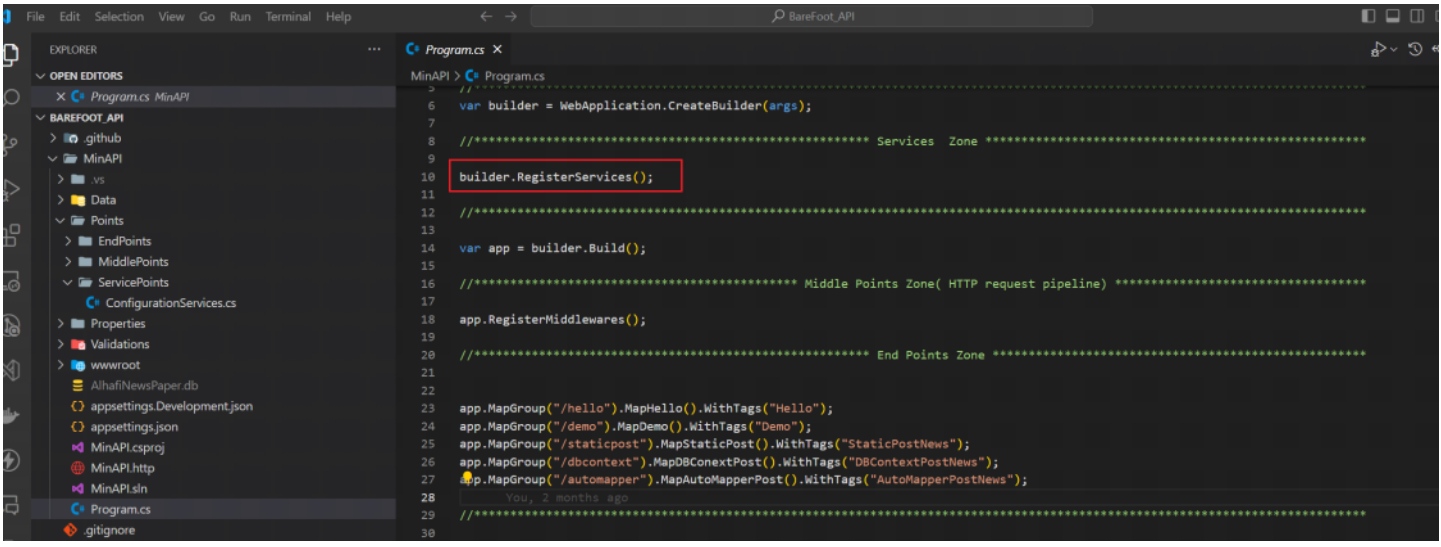

```
namespace MinAPI.Points.MiddlePoints
{
    public static class ConfigurationMiddlePoint
    {
        public static void RegisterMiddlewares(this WebApplication app)
        {
            if (app.Environment.IsDevelopment())
            {
                app.UseSwagger(options =>
                {
                    options.RouteTemplate =
"swagger/{documentName}/swagger.json";
                });
                app.UseSwaggerUI(options =>
                {
                    options.SwaggerEndpoint("/swagger/v1/swagger.json",
"BareFoot API V1");
                    options.RoutePrefix = "swagger";
                    // options.RoutePrefix = string.Empty;  //direct root
                    options.InjectStylesheet("/css/swagger.css");
                });
            }
            if (app.Environment.IsStaging())
            {
                // your code here
            }
            if (app.Environment.IsProduction())
            {
                // your code here
            }
            app.UseHttpsRedirection();
            app.UseStaticFiles();
            app.UseCors("MyAllowedOrigins");
            app.UseOutputCache();
        }
    }
}
```


Program.cs
MinAPI
Program.cs
var builder = WebApplication.CreateBuilder(args);
//********** Services Zone **********
builder.RegisterServices();
var app = builder.Build();
//********** Middle Points Zone(HTTP request pipeline) **********
app.RegisterMiddlewares();
//********** End Points Zone **********
app.MapGroup("/hello").MapHello().WithTags("Hello");
app.MapGroup("/demo").MapDemo().WithTags("Demo");
app.MapGroup("/staticpost").MapStaticPost().WithTags("StaticPostNews");
app.MapGroup("/dbcontext").MapDBConextPost().WithTags("DBContextPostNews");
app.MapGroup("/automapper").MapAutoMapperPost().WithTags("AutoMapperPostNews");
You, 2 months ago

Grouping Service Points

20 May 2024 14:43






```
using FluentValidation;
using Microsoft.EntityFrameworkCore;
using Microsoft.OpenApi.Models;
using MinAPI.Data;
using MinAPI.Data.Interfaces;
using MinAPI.Data.Models;
namespace MinAPI.Points.ServicePoints
{
    public static class ConfigurationServices
    {
        public static void RegisterServices(this WebApplicationBuilder
builder)
        {
            builder.Services.AddEndpointsApiExplorer();
            builder.Services.AddDbContext<AppDbContext>(x =>
                x.UseSqlite(builder.Configuration.GetConnectionString("Default
Connection"))
            );
            builder.Services.AddAutoMapper(typeof(Program));
            builder.Services.AddScoped<IPostRepo, PostRepo>();
            //builder.Services.AddValidatorsFromAssemblyContaining(typeof(Post
Validator));
            //builder.Services.AddScoped<IValidator<Post>, PostValidator>();
            //builder.Services.AddScoped<IValidator<PostNewOrUpdatedDto>,
PostNewOrUpdatedDtoValidator>();
```

```
            builder.Services.AddSingleton<IDateTime, SystemDateTime>();
            builder.Services.AddValidatorsFromAssemblyContaining<Program>(
                ServiceLifetime.Singleton
            );
            builder.Services.AddOutputCache();
            builder.Services.AddCors(options =>
            {
                options.AddPolicy(
                    "MyAllowedOrigins",
                    policy =>
                    {
                        policy
                            .WithOrigins(
                                "https://localhost/*",
                                "http://localhost/*",
                                "http://127.0.0.1/*",
                                "https://www.alhafi.org"
                            )
                            .AllowAnyHeader()
                            .AllowAnyOrigin()
                            .AllowAnyMethod()
                            .AllowAnyHeader();
                    }
                );
            });
            builder.Services.AddSwaggerGen(c =>
            {
                c.EnableAnnotations();
                c.SwaggerDoc(
                    "v1",
                    new OpenApiInfo
                    {
                        Title = "Barefoot API",
                        Version = "v1",
                        Contact = new()
                        {
                            Name = "Alhafi.BareFoot",
                            Email = "alhafi@hotmail.com",
                            Url = new Uri("https://www.alhafi.org/")
                        },
                        Description =
                            " BareFoot Minimal API Build in <b>dotnet new
webapi -minimal</b>  Hosted at github  <a
href='https://github.com/alhafibarefoot/BareFoot_API'>here</a>",
                        License = new
Microsoft.OpenApi.Models.OpenApiLicense(),
                        TermsOfService = new("https://www.alhafi.org/")
                    }
                );
                // using System.Reflection;
                var fileName = typeof(Program).Assembly.GetName().Name +
".xml";
                var filePath = Path.Combine(AppContext.BaseDirectory,
fileName);
                // integrate xml comments
                c.IncludeXmlComments(filePath);
            });
        }
    }
}
```

Error Handling

Wednesday, December 25, 2024 12:59 PM

https://www.infoworld.com/article/2338147/how-to-handle-errors-in-minimal-apis-in-aspnet-core.html
https://medium.com/@umairg404/global-exception-handling-with-net-core-8-minimal-api-ef32ca60fca0
https://codewithmukesh.com/blog/global-exception-handling-in-aspnet-core/

https://www.weekenddive.com/dotnet/gracefully-handling-exceptions-minimal-api
https://andrewhalil.com/2024/09/19/how-to-handle-errors-within-a-net-core-minimal-web-api/

https://www.weekenddive.com/dotnet/gracefully-handling-exceptions-minimal-api
https://dev.to/leandroveiga/mastering-error-handling-in-net-8-strategies-for-minimal-apis-and-controllers-3eb
https://dotnettutorials.net/lesson/error-handling-and-logging-in-asp-net-core-minimal-api/

Default Values

21 May 2024 14:06

Storage File Locally

21 May 2024 11:44

References

03 March 2024 13:02

API Sample Fake Data
https://fakerjs.dev/

Full Sample
MinimalApisMoviesEFCore/ASP.NET Core 8 at main · gavilanch/MinimalApisMoviesEFCore · GitHub

https://notebooklm.google.com/notebook/b1e53748-2a0c-42e8-9fca-36210af3f31b?pli=1