

Create Project

Sunday, January 7, 2024 11:50 AM

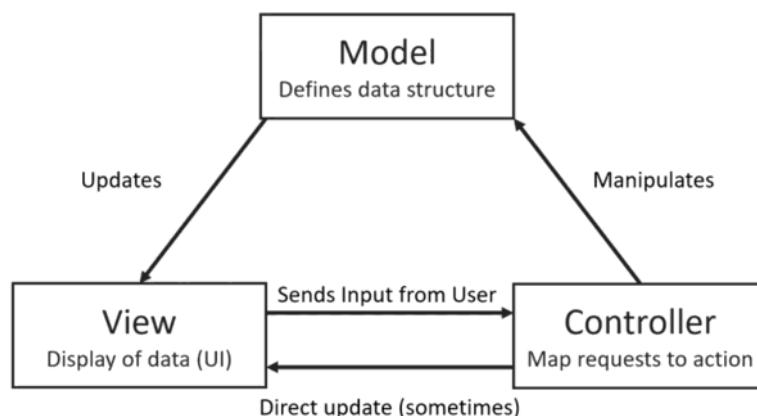
Differences Minimal API & MVC API

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

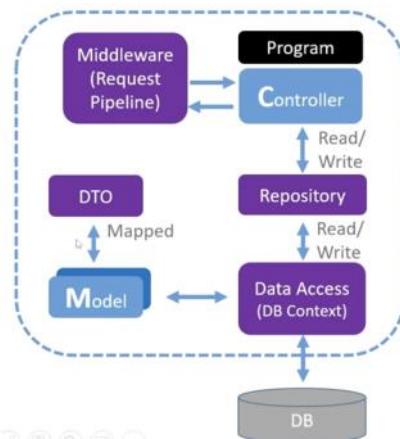
Model – View - Controller

- 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

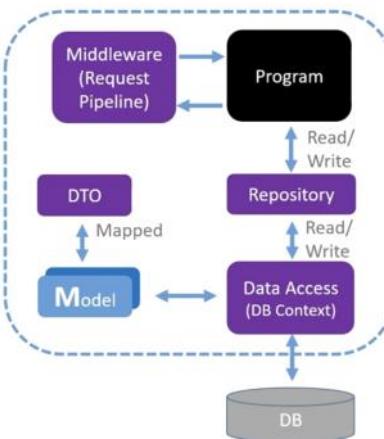
Model – View - Controller



MVC / Controller API



Minimal API



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

Tawasul - Home Moodle™ LMS - Inst... Time Tech... Faker From MVC to Mini... Synology Account LIONSanBox - Syno... Tiger_SanBox - Syn... Access via Synology New Tab JS Bin - Collaborativ...

alhafibarefoot

Overview Repositories Projects Packages Stars

Find a repository... Type Language Sort New

HtmxBlog Public

JavaScript Updated 2 days ago

Vidly Public

JavaScript Updated on Sep 19, 2023

Star

Star

alhafibarefoot

Owner * **Repository name ***

alhafibarefoot / 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](#).

(i) 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 Add file Code

alhafibarefoot Initial commit 1fc1e56 · now 1 Commits

README.md Initial commit now

README

About

No description, website, or topics provided

Readme Activity 0 stars 1 watching 0 forks

Releases

No releases published

A screenshot of a GitHub repository page for 'BareFoot_API'. The top navigation bar includes 'Go to file', 'Add file', 'Code' (selected), and 'About'. Below the navigation is a 'Local' tab and a 'Codespaces' tab. Under the 'Local' tab, there are three cloning options: 'Clone' (disabled), 'HTTPS' (selected and highlighted with a red box), 'SSH', and 'GitHub CLI'. A red box also highlights the copy icon next to the URL 'https://github.com/alhafibarefoot/BareFoot_API'. To the right of the URL are several icons: a bookmark, a download arrow, a star, a eye, and a gear. Below the cloning options are links for 'Open with GitHub Desktop', 'Open with Visual Studio', and 'Download ZIP'.

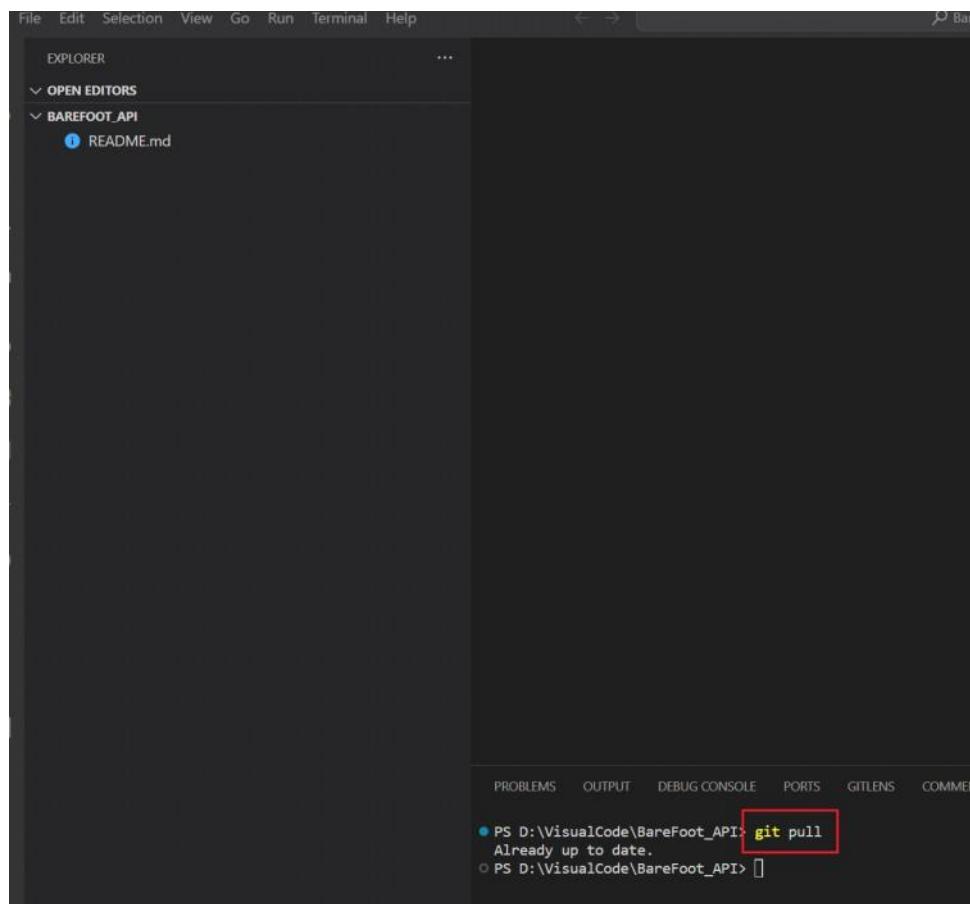
```
git clone https://github.com/alhafibarefoot/BareFoot\_API.git
```

```
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

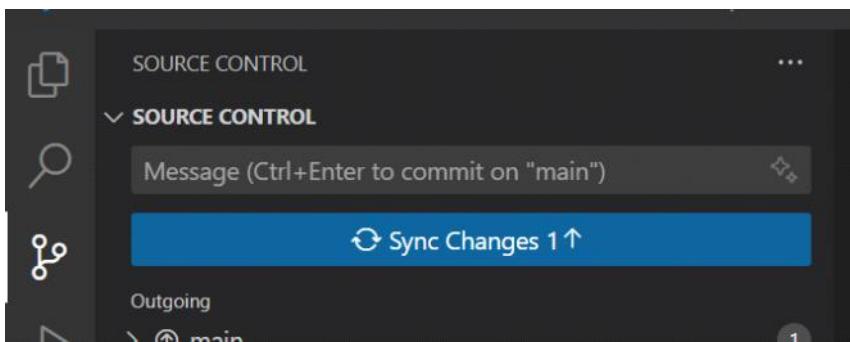
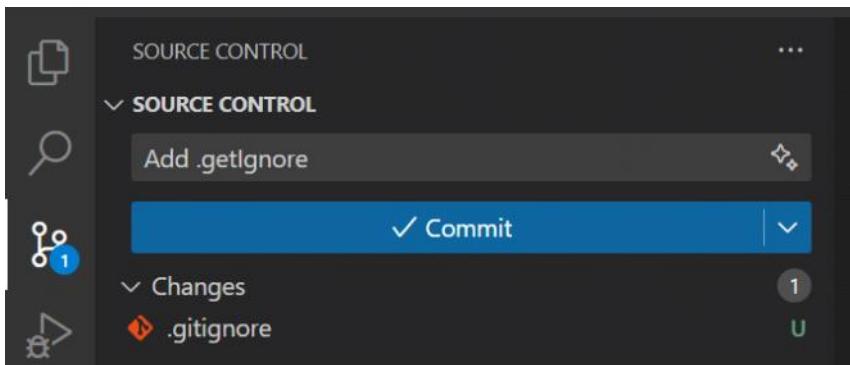
```
D:\VisualCode\BareFoot_API>code .
```



Before Adding Project Let we add ignore

```
① .gitignore ●
② .gitignore
1  #####
2  # This .gitignore file was automatically created by Microsoft(R) Visual Studio.
3  #####
4  *.CASH
5  # User-specific files
6  *.suo
7  *.user
8  *.userosscache
9  *.sln.docstates
10 _ReSharper.*/ 
11 *.sdf
12
13 # User-specific files (MonoDevelop/Xamarin Studio)
14 *.userprefs
15
16
17 # Build/debug/releaze results
18 [Dd]ebug/
19 [Dd]ebugPublic/
20 [Rr]elease/
21 [Rr]eleases/
22 x64/
23 x86/
24 bld/
25 [Bb]in/
26 [Oo]bj/
27 [Ll]og/
28 [Ll]ogs/
29 [Uu]pload/
```

A screenshot of a code editor showing a .gitignore file. The file contains standard ignore patterns for build artifacts and user-specific files. Lines 1 through 13 are comments explaining the file's purpose and creation source. Lines 14 through 29 list specific file and directory patterns to ignore.



Checking Update git hub

The screenshot shows a GitHub repository page for "BareFoot_API" owned by "alhafibarefoot". The repository is public. The commit history shows two commits:

- Commit 1: "Add .getignore" by alhafibarefoot at e736c8e - now. The file ".gitignore" is highlighted with a red border.
- Commit 2: "Initial commit" by alhafibarefoot at now. The file "README.md" is listed.

The screenshot shows the Terminal tab in Visual Studio Code. The command "dotnet --help" is being run, and the output is displayed:

```
PS D:\VC\API_LookUp> dotnet --help
Welcome to .NET 8.0!
-----
SDK Version: 8.0.100
Telemetry
-----
The .NET tools collect usage data in order to help us improve your community. You can opt-out of telemetry by setting the DOTNET_CLT_TELEMETRY environment variable to false.
```

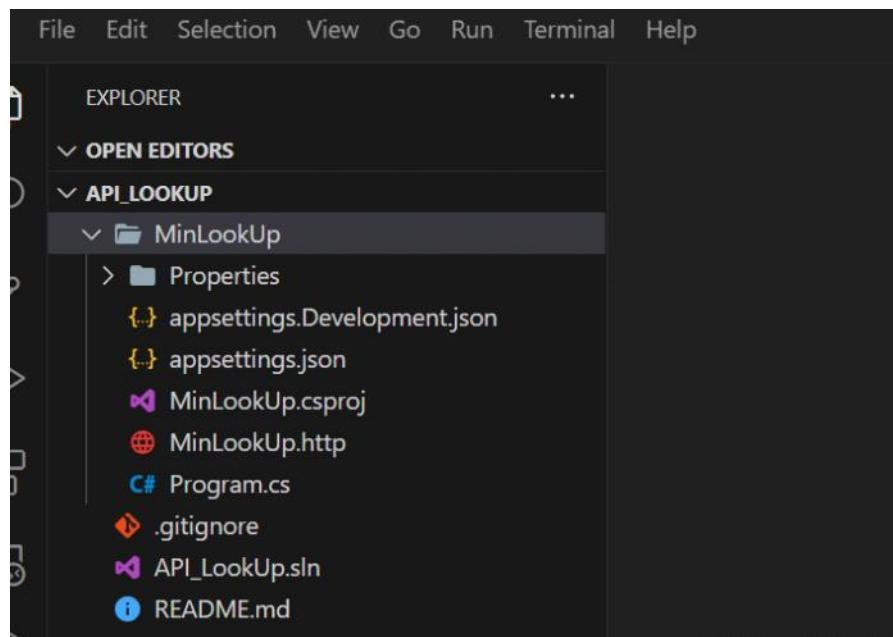
```
● PS D:\VC\API_LookUp> dotnet --list-sdks
8.0.100 [C:\Program Files\dotnet\sdk]
○ PS D:\VC\API_LookUp> []
```

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

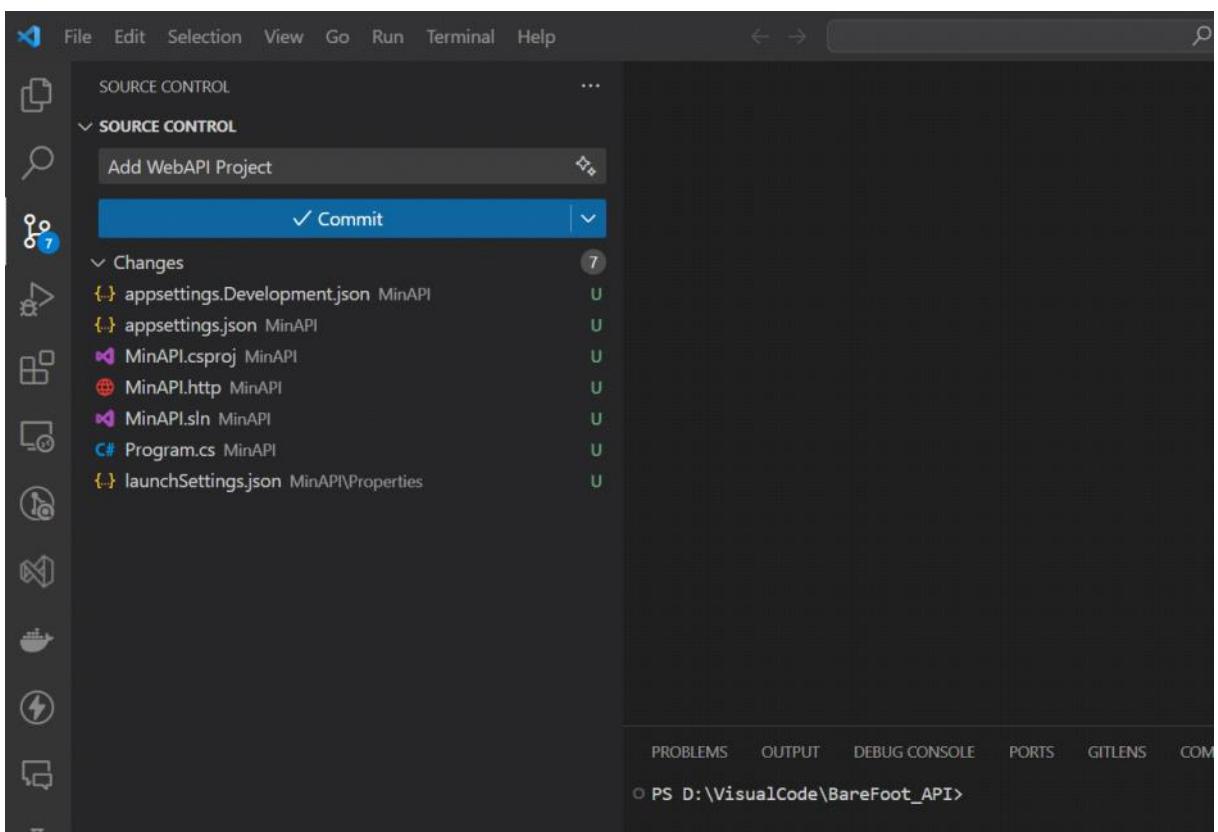
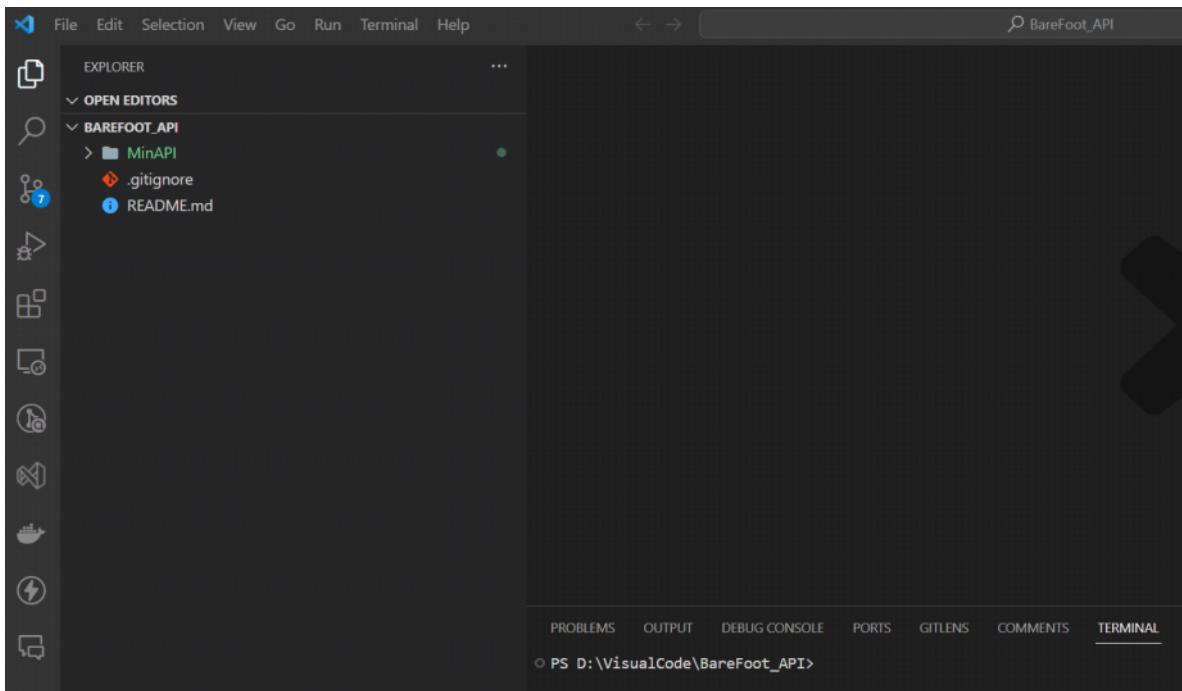
Template Name	Short Name	Language	Tags
.NET MAUI App	maui	[C#]	MAUI/Android/iOS/macOS/M
.NET MAUI Blazor	maui-blazor	[C#]	MAUI/Android/iOS/macOS/M
.NET MAUI Class	maulib	[C#]	MAUI/Android/iOS/macOS/M
.NET MAUI Content	maui-page-csharp	[C#]	MAUI/Android/iOS/macOS/M
.NET MAUI Content	maui-page-xaml	[C#]	MAUI/Android/iOS/macOS/M
.NET MAUI Content	maui-view-csharp	[C#]	MAUI/Android/iOS/macOS/M
.NET MAUI Content	maui-view-xaml	[C#]	MAUI/Android/iOS/macOS/M
.NET MAUI Resource	maui-dict-xaml	[C#]	MAUI/Android/iOS/macOS/M
Android Activity	android-activity	[C#]	Android/Mobile
Android Application	android	[C#]	Android/Mobile
Android Class Library	androidlib	[C#]	Android/Mobile
Android Java Library	android-bindinglib	[C#]	Android/Mobile
Android Layout	android-layout	[C#]	Android/Mobile
Android Wear App	androidwear	[C#]	Android/Mobile
API Controller	apicontroller	[C#]	Web/ASP.NET
ASP.NET Core Empty	web	[C#], F#	Web/Empty
ASP.NET Core gRPC	grpc	[C#]	Web/gRPC/API/Service
ASP.NET Core Web API	webapi	[C#], F#	Web/WebAPI/Web API/API
ASP.NET Core Web API	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

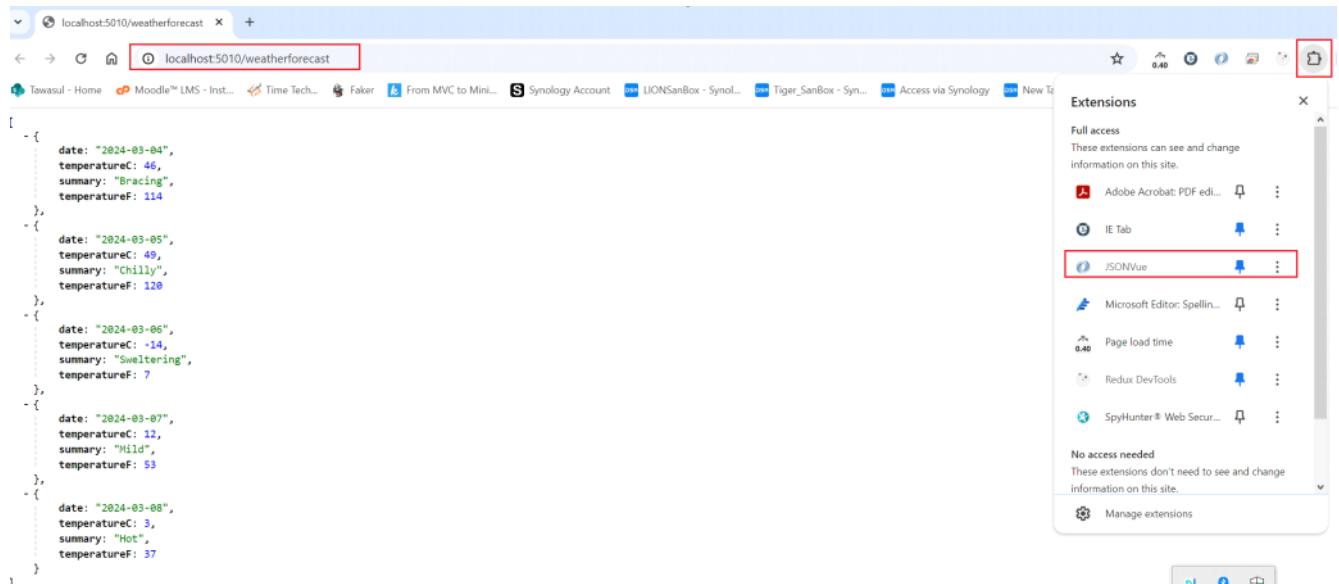
The screenshot shows the MinAPI documentation for a GET request to the endpoint `/weatherforecast`. The interface includes sections for 'Parameters' (No parameters), 'Execute' (a blue button), and 'Responses'. Under 'Responses', there is a 'Curl' section with the command:

```
curl -X 'GET' \
  'http://localhost:5240/weatherforecast' \
  -H 'accept: application/json'
```

Below it is a 'Request URL' field containing `http://localhost:5240/weatherforecast`. The 'Server response' section shows a status code of 200 and a JSON response body:

```
[  
  {  
    "date": "2024-03-04",  
    "temperatureC": 32,  
    "summary": "Freezing",  
    "temperatureF": 89  
  },  
  {  
    "date": "2024-03-05",  
    "temperatureC": 46,  
    "summary": "Bracing",  
    "temperatureF": 114  
  },  
  {  
    "date": "2024-03-06",  
    "temperatureC": 49,  
    "summary": "Chilly",  
    "temperatureF": 102  
  },  
  {  
    "date": "2024-03-07",  
    "temperatureC": 14,  
    "summary": "Sweating",  
    "temperatureF": 7  
  },  
  {  
    "date": "2024-03-08",  
    "temperatureC": 12,  
    "summary": "Mild",  
    "temperatureF": 53  
  },  
  {  
    "date": "2024-03-09",  
    "temperatureC": 3,  
    "summary": "Hot",  
    "temperatureF": 37  
}
```

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



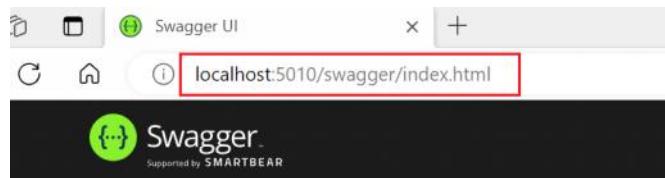
custom out port of http--->5010 and https--->7010

The screenshot shows the VS Code interface with the file `launchSettings.json` open. The code defines profiles for both `http` and `https`. For the `http` profile, the `applicationUrl` is set to `'http://localhost:5010'`. For the `https` profile, the `applicationUrl` is set to `'https://localhost:7010;http://localhost:5010'`. The `sslPort` is also specified as `44346`.

```
{...} launchSettings.json M X
MinAPI > Properties > {...} launchSettings.json > {} profiles > {} https > applicationUrl
8     "sslPort": 44346
9 }
10 },
11 "profiles": {
12     "http": {
13         "commandName": "Project",
14         "dotnetRunMessages": true,
15         "launchBrowser": true,
16         "launchUrl": "swagger",
17         "applicationUrl": "http://localhost:5010",
18         "environmentVariables": {
19             "ASPNETCORE_ENVIRONMENT": "Development"
20         }
21     },
22     "https": {
23         "commandName": "Project",
24         "dotnetRunMessages": true,
25         "launchBrowser": true,
26         "launchUrl": "swagger",
27         "applicationUrl": "https://localhost:7010;http://localhost:5010",
28         "environmentVariables": {
29             "ASPNETCORE_ENVIRONMENT": "Development"
}
PROBLEMS OUTPUT DEBUG CONSOLE PORTS GITLENS COMMENTS TERMINAL
```

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

It work as http



MinAPI

1.0 OAS3

<http://localhost:5010/swagger/v1/swagger.json>

MinAPI

GET /weatherforecast

Schemas

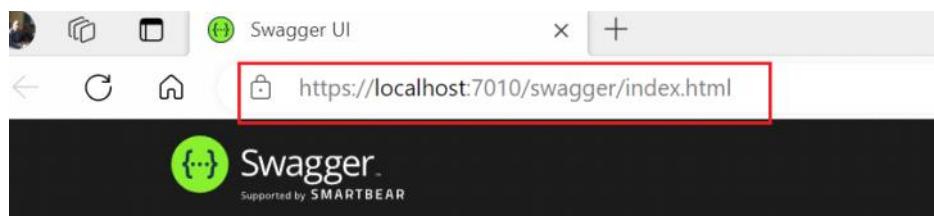
DateOnly >

DayOfWeek >

WeatherForecast >

Let we make default https by swapping https first

```
{...} launchSettings.json M •  
MinAPI > Properties > {...} launchSettings.json > {} profiles  
--  
11  "profiles": {  
12    "https": {  
13      "commandName": "Project",  
14      "dotnetRunMessages": true,  
15      "launchBrowser": true,  
16      "launchUrl": "swagger",  
17      "applicationUrl": "https://localhost:7010;http://127.0.0.1:5010",  
18      "environmentVariables": {  
19        "ASPNETCORE_ENVIRONMENT": "Development"  
20      }  
21    },  
22    "http": {  
23      "commandName": "Project",  
24      "dotnetRunMessages": true,  
25      "launchBrowser": true,  
26      "launchUrl": "swagger",  
27      "applicationUrl": "http://localhost:5010",  
28      "environmentVariables": {  
29        "ASPNETCORE_ENVIRONMENT": "Development"  
30      }  
31    },  
--
```



MinAPI

<https://localhost:7010/swagger/v1/swagger.json>

MinAPI

GET /weatherforecast

Schemas

Adding HTTPS developer trust certification

dotnet dev-certs https --trust

```
PROBLEMS OUTPUT DEBUG CONSOLE PORTS GITLENS COMMENTS TERMINAL
PS D:\VisualCode\BareFoot_API\minapi> dotnet dev-certs https --trust
>>
Trusting the HTTPS development certificate was requested. A confirmation prompt
t to trust the certificate.
Successfully trusted the existing HTTPS certificate.
D PS D:\VisualCode\BareFoot_API\minapi> 
```

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

My Workspace

Overview

GET http://localhost:5001/weatherforecast

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	...
Key	Value	Description	Bulk Edit

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

1 [{"date": "2024-01-12", "temperatureC": 1, "summary": "Balmy", "temperatureF": 33}, {"date": "2024-01-13", "temperatureC": 38, "summary": "Freezing", "temperatureF": 100}, {"date": "2024-01-14", "temperatureC": 17, "summary": "Scorching", "temperatureF": 62}, {"date": "2024-01-15", "temperatureC": -17, "summary": "Cool", "temperatureF": 2}, {"date": "2024-01-16", "temperatureC": 49, "summary": "Sweltering", "temperatureF": 120}]

200 OK 18 ms 541 B Save as example

Or extension in Visual code Thunder Client

File Edit Selection View Go Run ...

THUNDER CLIENT

New Request

Activity Collections Env

filter activity

GET localhost:5001/weatherforecast just now

TC New Request

GET http://localhost:5001/weatherforecast Send

Query Headers 2 Auth Body Tests Pre Run

Query Parameters

parameter value

Status: 200 OK Size: 390 Bytes Time: 4 ms

Response Headers 5 Cookies Results Docs

```
1 [
2   {
3     "date": "2024-01-12",
4     "temperatureC": 31,
5     "summary": "Chilly",
6     "temperatureF": 87
7   },
8   {
9     "date": "2024-01-13",
10    "temperatureC": -5,
11    "summary": "Warm",
12    "temperatureF": 24
13  },
14  {
15    "date": "2024-01-14",
16    "temperatureC": 20,
```

Viewing Web API

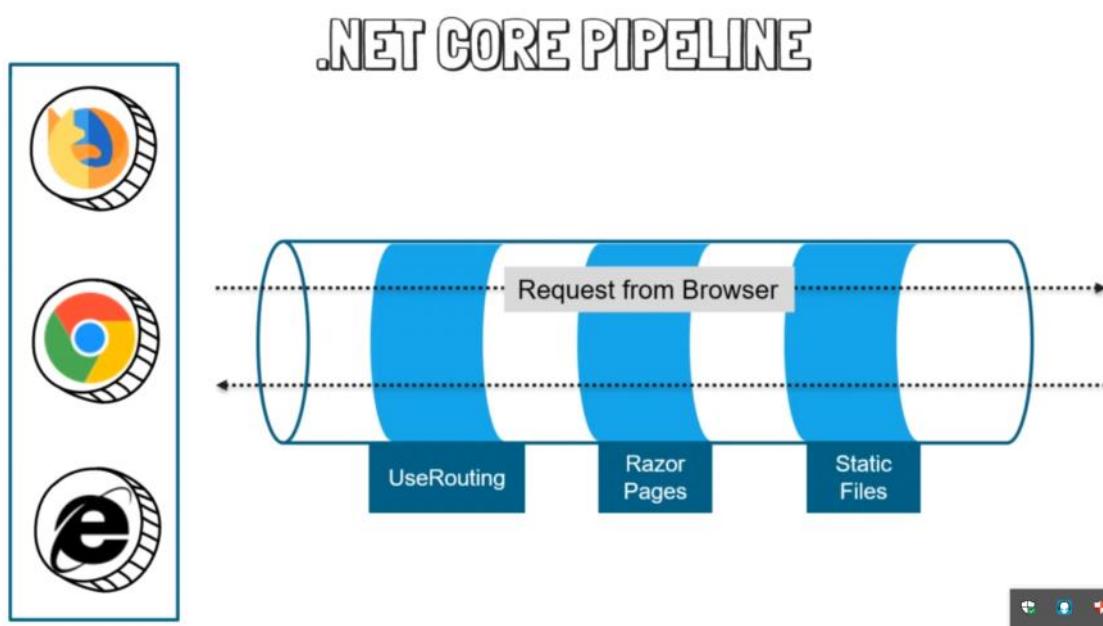
Tuesday, January 9, 2024 12:33 PM

@ 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/swashbuckle
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();
}
```

```
C# Program.cs X
MinAPI > C# Program.cs > ...
8     var app = builder.Build();
9
10    // Configure the HTTP request pipeline.
11    if (app.Environment.IsDevelopment())
12    {
13        app.UseSwagger();
14        app.UseSwaggerUI();
15    }
16
17    app.UseHttpsRedirection();
18
19    var summaries = new[]
20    {
21        "Freezing", "Bracing", "Chilly", "Cool", "Mild",
22    };
23
24    app.MapGet("/weatherforecast", () =>
25    {
26        var forecast = Enumerable.Range(1, 5).Select(ind
```

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

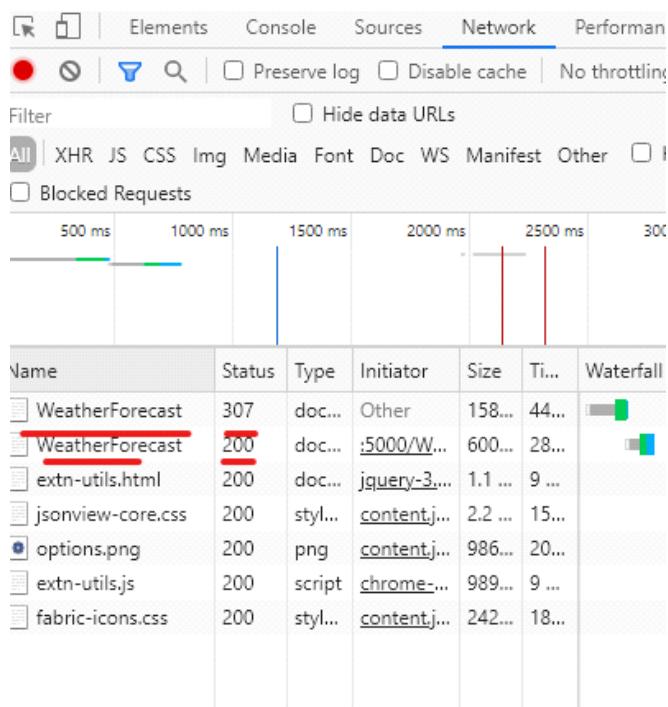
```
app.UseHttpsRedirection();
```

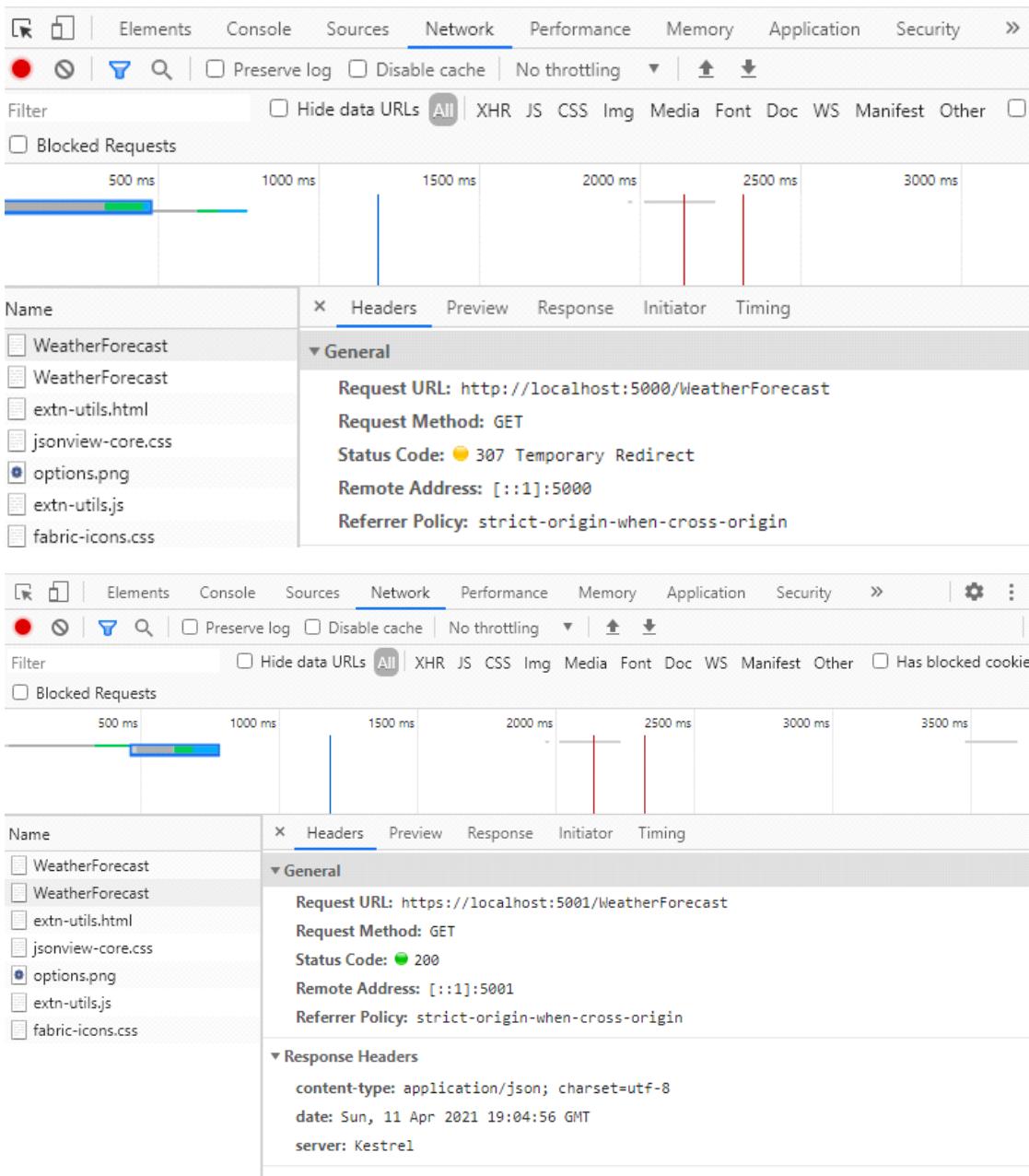
Assume we call HTTP instead of https

<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())
{
    You, 1 minute ago * Uncommitted changes
    // 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
}

```

The screenshot shows two JSON configuration files side-by-side in a code editor:

launchSettings.json

```

"profiles": {
    "https": {
        "commandName": "Project",
        "dotnetRunMessages": true,
        "launchBrowser": true,
        "launchUrl": "swagger",
        "applicationUrl": "https://localhost:7010;http://localhost:5010",
        "environmentVariables": {
            "ASPNETCORE_ENVIRONMENT": "Development"
        }
    },
    "http": {
        "commandName": "Project",
        "dotnetRunMessages": true,
        "launchBrowser": true,
        "launchUrl": "swagger",
        "applicationUrl": "http://localhost:5010",
        "environmentVariables": {
            "ASPNETCORE_ENVIRONMENT": "Development"
        }
    },
    "IIS Express": {
        "commandName": "IISExpress",
        "dotnetRunMessages": true,
        "launchBrowser": true,
        "launchUrl": "swagger",
        "environmentVariables": {
            ...
        }
    }
}

```

appsettings.Development.json

```

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

```

Understanding environments

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

```
appsettings.Development.json ●  
MinAPI > appsettings.Development.json > ...  
You, 2 months ago | 1 author (You)  
1 | {  
2 |   "myEnv": "I'm In Development Environment",  
3 |   "Logging": {  
4 |     "LogLevel": {  
5 |       "Default": "Information",  
6 |       "Microsoft.AspNetCore": "Warning"  
7 |     }  
8 |   },  
9 |   "ConnectionStrings": {  
10 |     "DefaultConnection": "Data source=AlhafiNewsPaper.db"  
11 |   },  
12 |   "AllowedHosts": "*"  
13 | }  
  
"myEnv": "I'm In Development Environment",
```

```
appsettings.json M X  
MinAPI > appsettings.json > myEnv  
You, 1 second ago | 1 author (You)  
1 | {  
2 |   "myEnv": "I'm In Development Production",  
3 |   "Logging": {  
4 |     "LogLevel": {  
5 |       "Default": "Information",  
6 |       "Microsoft.AspNetCore": "Warning"  
7 |     }  
8 |   },  
9 |   "AllowedHosts": "*"  
10 | }  
  
"myEnv": "I'm In Production Environment",
```

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

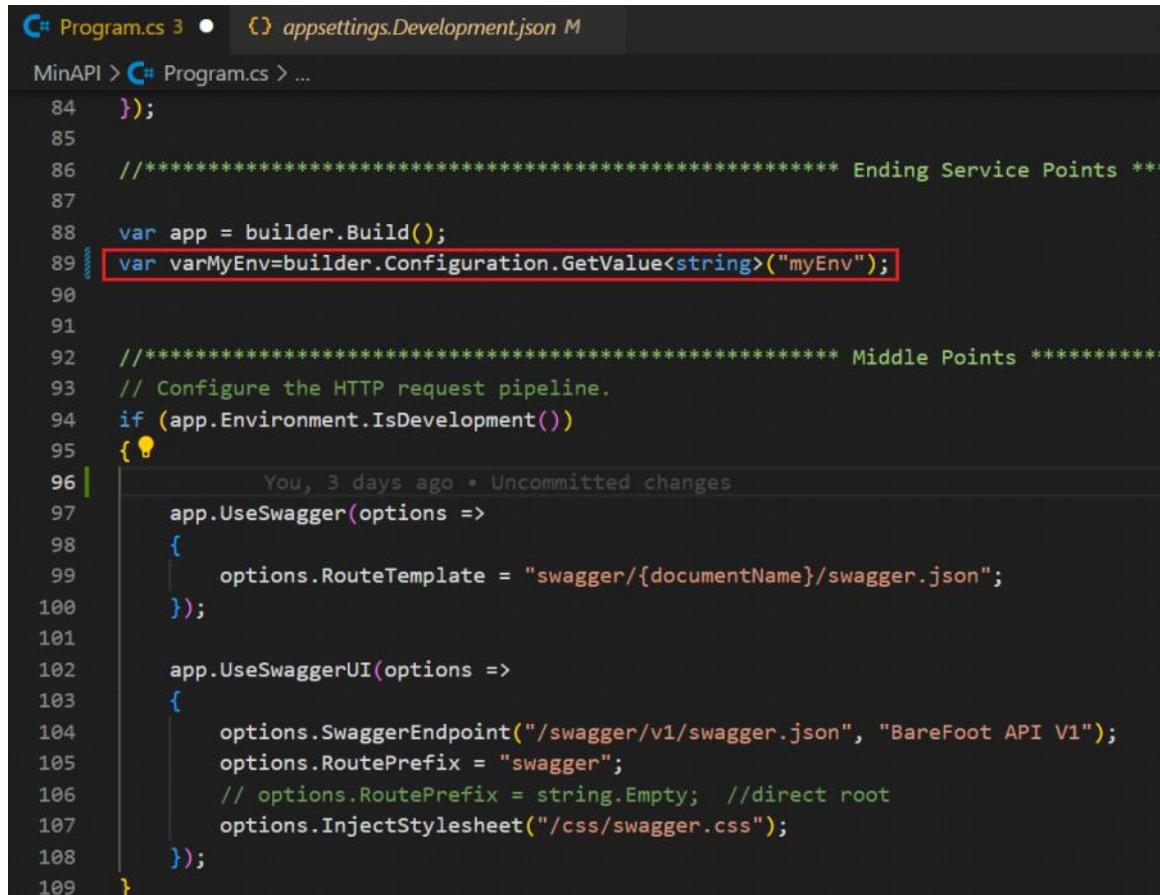
```
launchSettings.json X  
MinAPI > Properties > launchSettings.json > ...  
3 |   "iisSettings": {  
4 |     "iisExpress": {  
5 |       "applicationUrl": "http://localhost:7072",  
6 |       "sslPort": 44346  
7 |     }  
8 |   },  
9 |   "profiles": {  
10 |     "https": {  
11 |       "commandName": "Project",  
12 |       "dotnetRunMessages": true,  
13 |       "launchBrowser": true,  
14 |       "launchUrl": "swagger",  
15 |       "applicationUrl": "https://localhost:7010;http://localhost:5010",  
16 |       "environmentVariables": {  
17 |         "ASPNETCORE_ENVIRONMENT": "Development"  
18 |       }  
19 |     },  
20 |     "http": {  
21 |       "commandName": "Project",  
22 |       "dotnetRunMessages": true,  
23 |       "launchBrowser": true,  
24 |       "launchUrl": "swagger",  
25 |       "applicationUrl": "http://localhost:5010",  
26 |       "environmentVariables": {  
27 |         "ASPNETCORE_ENVIRONMENT": "Development"  
28 |       }  
29 |     },  
30 |   },  
31 | }
```

But Preferred Method While developing

dotnet watch run --environment Production

```
dotnet watch run --environment Development
```

Let we try this



```
C# Program.cs 3 ● ⓘ appsettings.Development.json M  
MinAPI > C# Program.cs > ...  
84     });  
85  
86     //***** Ending Service Points *****  
87  
88     var app = builder.Build();  
89     var varMyEnv=builder.Configuration.GetValue<string>("myEnv");  
90  
91  
92     //***** Middle Points *****  
93     // Configure the HTTP request pipeline.  
94     if (app.Environment.IsDevelopment())  
95     {  
96         You, 3 days ago • Uncommitted changes  
97         app.UseSwagger(options =>  
98         {  
99             options.RouteTemplate = "swagger/{documentName}/swagger.json";  
100        });  
101  
102        app.UseSwaggerUI(options =>  
103        {  
104            options.SwaggerEndpoint("/swagger/v1/swagger.json", "BareFoot API V1");  
105            options.RoutePrefix = "swagger";  
106            // options.RoutePrefix = string.Empty; //direct root  
107            options.InjectStylesheet("/css/swagger.css");  
108        });  
109    }
```

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

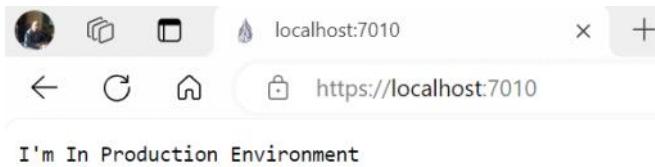
The screenshot shows the Visual Studio code editor with two tabs open: `Program.cs` and `appsettings.Development.json`. The `Program.cs` file contains C# code for a .NET Core application. The code includes environment handling, static file serving, and endpoint mapping. A specific line of code, `app.MapGet("/", ()=>varMyEnv);`, is highlighted with a red rectangle. A tooltip above the code indicates "You, 3 days ago • Uncommitted changes". The `appsettings.Development.json` file is also visible in the background.

```
MinAPI > C# Program.cs > ...
113     // your code here
114 }
115 if (app.Environment.IsProduction())
116 {
117     // your code here
118 }
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(/
```

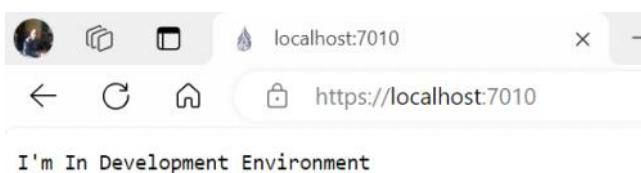
```
app.MapGet("/", ()=>varMyEnv);
```

Let we run this

`dotnet watch run --environment Production`



`dotnet watch run --environment Development`



Or by swagger

GET /

Parameters

No parameters

Responses

Curl

```
curl -X 'GET' \
  'https://localhost:7010/' \
  -H 'accept: text/plain'
```

Request URL

<https://localhost:7010/>

Server response

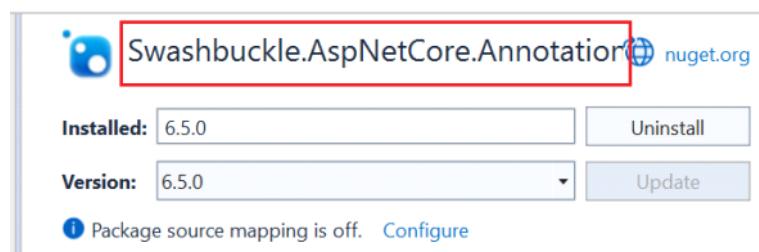
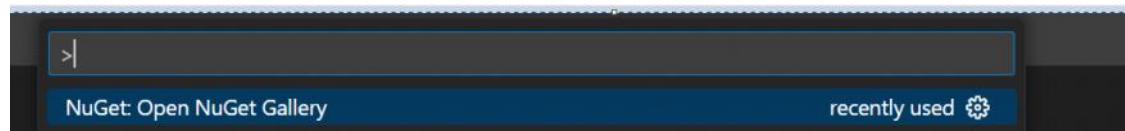
Code	Details
200	Response body

```
I'm In Development Environment
```

[Copy](#) [Download](#)

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='https://github.com/Alhafi/BareFootMinimalAPI'>https://github.com/Alhafi/BareFootMinimalAPI</a> License = new Microsoft.OpenApi.Models.OpenApiLicense(),
        TermsOfService = new("https://www.alhafi.org/")
    });
});
```

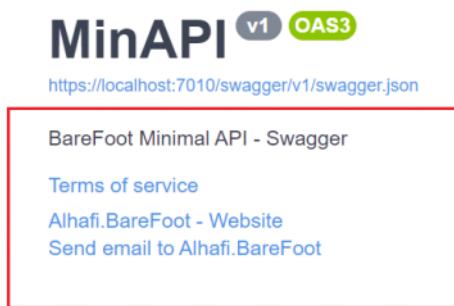
```

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/")
        }
    );
});

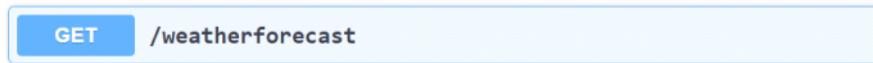
```

Testing

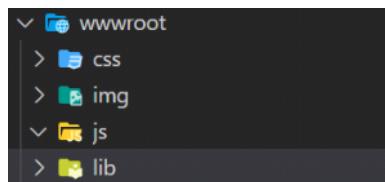
PS D:\VisualCode\BareFoot_API\minapi> **dotnet watch run**



MinAPI



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

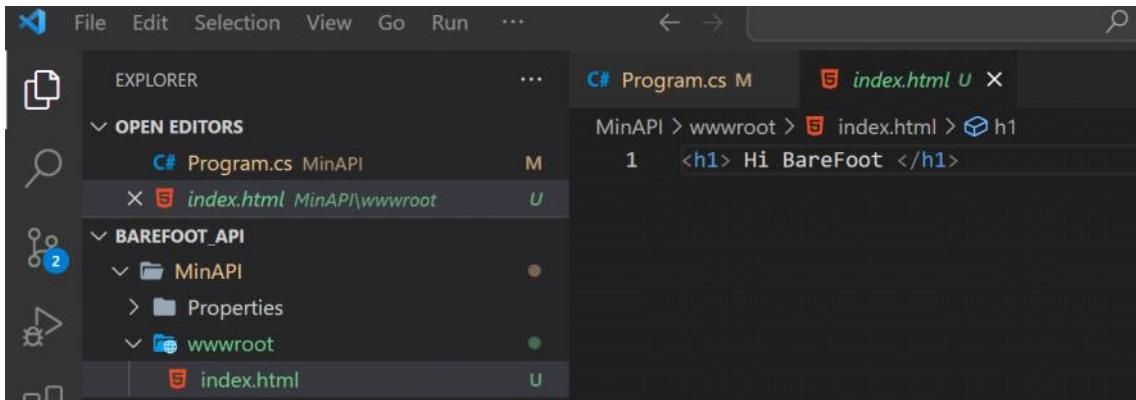


Add middle point

```
C# Program.cs X
MinAPI > C# Program.cs > ...
34
35 // Configure the HTTP request pipeline.
36 if (app.Environment.IsDevelopment())
37 {
38     app.UseSwagger(options =>
39     {
40         options.RouteTemplate =
41     });
42
43     app.UseSwaggerUI(options =>
44     {
45         options.SwaggerEndpoint =
46         options.RoutePrefix = "swagger";
47         options.InjectStylesheet();
48     });
49 }
50 if (app.Environment.IsStaging())
51 {
52     // your code here
53 }
54 if (app.Environment.IsProduction())
55 {
56     // your code here
57 }
58
59 app.UseHttpsRedirection();
60
61 app.UseStaticFiles();
62
63 var summaries = new[]
```

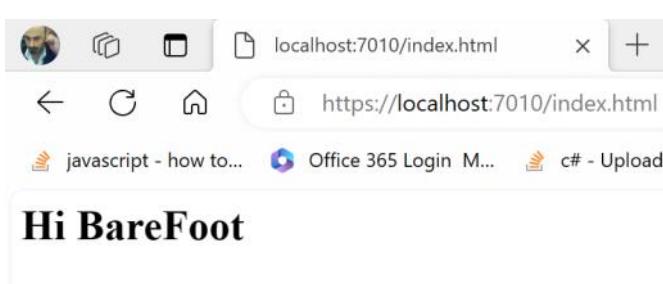
app.UseStaticFiles();

Test the static folder wwwroot by adding static html



Testing

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



Route Update

```

C# Program.cs ●
MinAPI > C# Program.cs > ...
29     }
30   );
31 });
32
33 var app = builder.Build();
34
35 // Configure the HTTP request pipeline.
36 if (app.Environment.IsDevelopment())
37 {
38   app.UseSwagger(options =>
39   {
40     options.RouteTemplate = "swagger/{documentName}/swagger.json";
41   });
42
43   app.UseSwaggerUI(options =>
44   {
45     options.SwaggerEndpoint("/swagger/v1/swagger.json", "BareFoot API V1");
46     options.RoutePrefix = "swagger";
47   });
48 }
49
50 if (app.Environment.IsStaging())
51 {
52   // your code here
53 }
54 if (app.Environment.IsProduction())
55 {
56   // your code here
57 }

```

```

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

```

```

File Edit Selection View Go Run Terminal Help ← → 🔍 BareFoot_API
EXPLORER ... { } launchSettings.json 1 ×
MinAPI > Properties > { } launchSettings.json > () profiles
11   "profiles": {
12     "https": {
13       "commandName": "Project",
14       "dotnetRunMessages": true,
15       "launchBrowser": true,
16       "launchUrl": "swagger",
17       "applicationUrl": "https://localhost:7010;http://localhost:5016",
18       "environmentVariables": {
19         "ASPNETCORE_ENVIRONMENT": "Development"
20       }
21     },
22     "http": {
23       "commandName": "Project",
24       "dotnetRunMessages": true,
25       "launchBrowser": true,
26       "launchUrl": "swagger",
27       "applicationUrl": "http://localhost:5010",
28       "environmentVariables": {
29         "ASPNETCORE_ENVIRONMENT": "Development"
30       }
31     },
32   }
33   "IIS Express": {
34     "commandName": "IISExpress",
35     "launchBrowser": true,
36     "launchUrl": "swagger",
37     "environmentVariables": {
38       "ASPNETCORE_ENVIRONMENT": "Development"
39     }
40   }

```

Testing

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

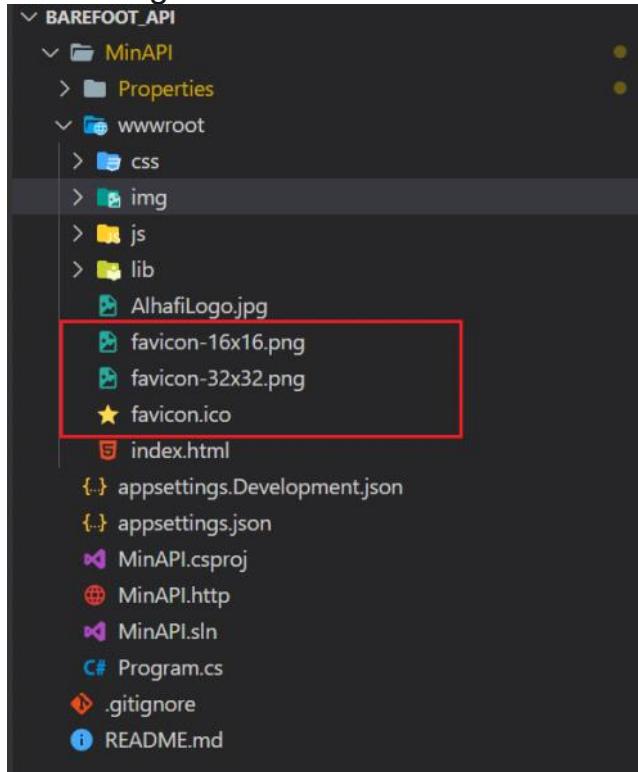
Replace Swagger Icon/Logo

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

The screenshot shows the Visual Studio Code interface. The left sidebar displays the file structure of a .NET Core API project named 'MinAPI'. The 'wwwroot/css' folder is expanded, showing files like 'AlhafiLogo.jpg', 'favicon-16x16.png', 'favicon-32x32.png', 'favicon.ico', 'index.html', and 'swagger.css'. The 'swagger.css' file is selected and open in the main editor area. The code in 'swagger.css' is:

```
.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;  
}
```

```
.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

The screenshot shows the 'Startup.cs' file in Visual Studio Code. The 'Configure' method contains code to configure the HTTP request pipeline. A specific section of the code, which injects a CSS file into the Swagger UI, is highlighted with a red box:

```
// 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";  
        options.InjectStylesheet("/css/swagger.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/swagger.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

```

```

RECENTS
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**

BareFoot Swagger Documentation

Barefoot API v1 OAS3

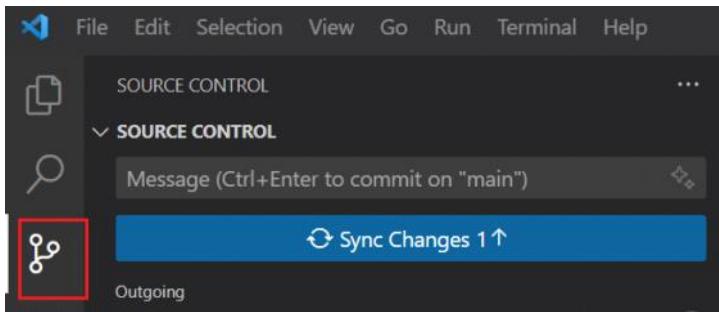
/swagger/v1/swagger.json

BareFoot Minimal API Build in **dotnet new webapi -minimal** Hosted at [github](#)

[Terms of service](#)
[Alhafi.BareFoot - Website](#)
Send email to [Alhafi.BareFoot](#)

No operations defined in spec!

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
```



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

MinAPI

GET /hello

POST /hello

PUT /hello

DELETE /hello

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
```

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

Hello

GET /hello

POST /hello

PUT /hello

DELETE /hello

Creating Static Record

C# Program.cs M X

MinAPI > C# Program.cs > NewsListStatic

```
59     app.UseHttpsRedirection();
60
61     app.UseStaticFiles();
62
63     //*****Static Hello End Point*****
64
65     app.MapGet("/hello", () => "[GET] Hello World");
66     app.MapPost("/hello", () => "[POST] Hello World");
67     app.MapPut("/hello", () => "[PUT] Hello World");
68     app.MapDelete("/hello", () => "[DELETE] Hello World");
69
70     //*****
71
72     app.Run();
73
74
75 0 references | You, 24 seconds ago | 1 author (You)
76 record NewsListStatic
77 {
78     public int Id { get; set; }
79     public string? Title { get; set; }
80     public string? Content { get; set; }
81 }
```

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

Define Variable

C# Program.cs M ●

MinAPI > C# Program.cs > ...

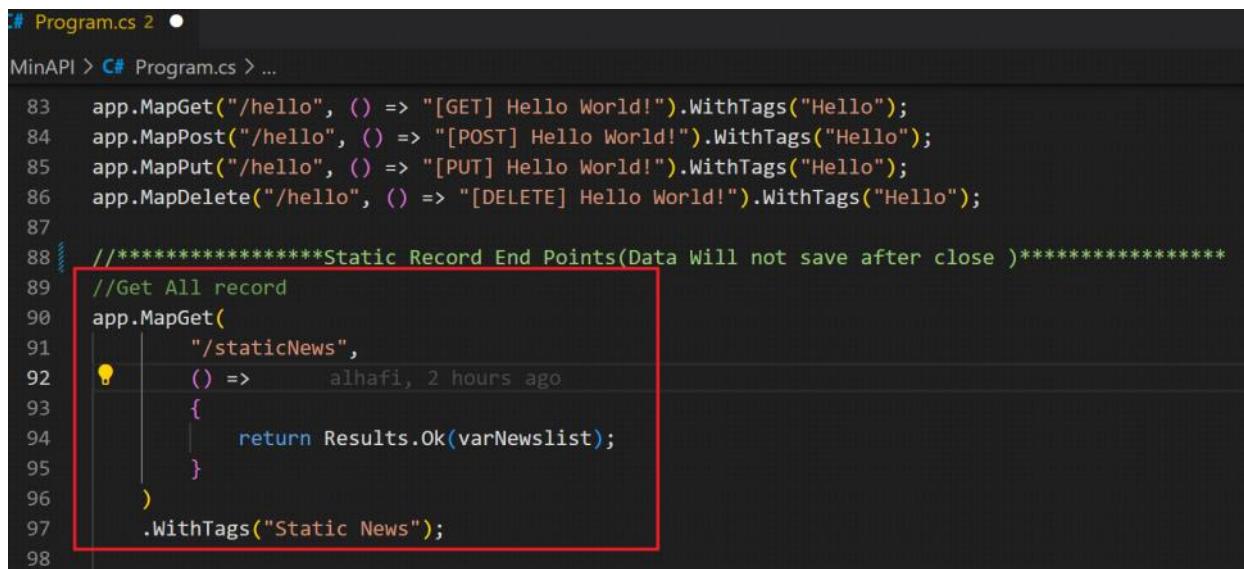
```
You, 7 minutes ago | 2 authors (You and others)
1 using Microsoft.OpenApi.Models;
2
3 var builder = WebApplication.CreateBuilder(args);
4
5 //Creating Variables of Lists
6 var varNewslist = new List<NewsListStatic>
7 {
8     new NewsListStatic
9     {
10         Id = 1,
11         Ttile = "F1 News",
12         Content = "Christian Horner Surprised By Team Strategy At Bahrain GP ."
13     },
14     new NewsListStatic
15     {
16         Id = 2,
17         Ttile = "Haley will win",
18         Content =
19             "Former South Carolina Gov. Nikki Haley will win the Republican presidential primary in Washington, DC"
20     },
21 };
22
```

```

//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 exists");
        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 exists");
        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 exists");

    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 exists");
    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 Exists");
    }

    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 Exists");
    }
    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 exists");
    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 exists");
    varNewslist.Remove(varNews);
    return Results.Ok(varNews);
}
)
.WithTags("Static News");

```

Testing

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

The screenshot shows a browser window with the title 'Swagger UI'. The address bar displays 'https://localhost:7010/swagger/index.html'. Below the address bar, there are several tabs: 'javascript - how to...', 'Office 365 Login M...', 'c# - Upload File wit...', and 'Parameter'. The main content area has a dark header with the text 'BareFoot Swagger Documentation' and a flame icon.

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

Hello

GET	/hello
POST	/hello
PUT	/hello
DELETE	/hello

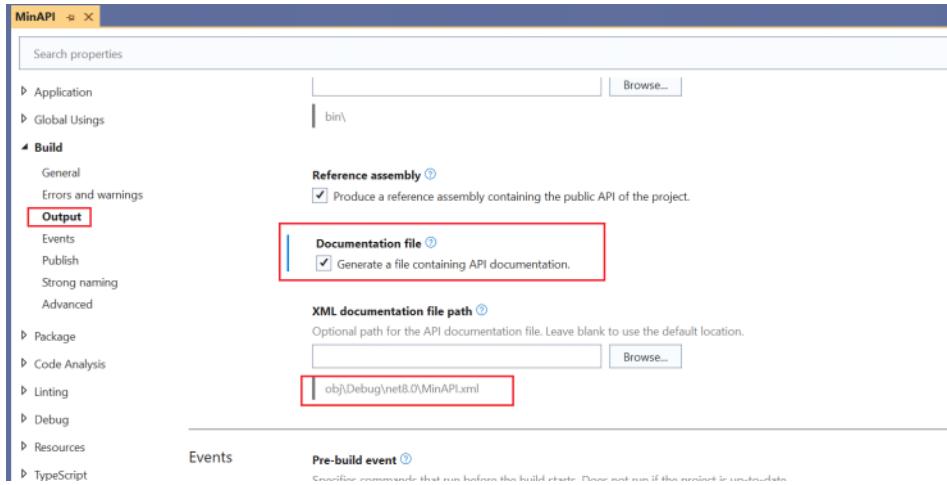
Static News

GET	/staticNews
POST	/staticNews
GET	/staticNews/{id}
PUT	/staticNews/{id}
DELETE	/staticNews/{id}

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

```

<Project Sdk="Microsoft.NET.Sdk.Web">
  ...
  <PropertyGroup>
    <TargetFramework>net8.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
    <GenerateDocumentationFile>true</GenerateDocumentationFile>
  </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>

```

<GenerateDocumentationFile>true</GenerateDocumentationFile>

Reflection will be generate in xml

```
C# Program.cs X
MinAPI > C# Program.cs > ...
31     c.EnableAnnotations();
32     c.SwaggerDoc(
33         "v1",
34         new OpenApiInfo
35         {
36             Title = "Barefoot API",
37             Version = "v1",
38             Contact = new()
39             {
40                 Name = "Alhafi.BareFoot",
41                 Email = "alhafi@hotmail.com",
42                 Url = new Uri("https://www.alhafi.org/")
43             },
44             Description =
45                 " BareFoot Minimal API Build in <b>dotnet new webapi -minimal</b> Hosted at github <a href='https://github.com/
46             License = new Microsoft.OpenApi.Models.OpenApiLicense(),
47             TermsOfService = new("https://www.alhafi.org/")
48         }
49     );
50
51     // using System.Reflection;
52     var fileName = typeof(Program).Assembly.GetName().Name + ".xml";
53     var filePath = Path.Combine(AppContext.BaseDirectory, fileName);
54
55     // integrate xml comments
56     c.IncludeXmlComments(filePath);
57 });
58
59 var app = builder.Build();
60
```

```
// 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's Document Fields in Models

```
C# Program.cs X
MinAPI > C# Program.cs > ...
205 //*****
206
207 app.Run();
208
209 You, 6 minutes ago | 1 author (You) | 5 references
210 record NewsListStatic
211 {
212     4 references
213     public int Id { get; set; }
214
215     /// <summary>
216     /// Title is Any headline news
217     /// </summary>
218     /// <example>Formula One World Championship</example>
219     5 references
220     public string? Title { get; set; }
221
222     /// <summary>
223     /// Contenty contain details of the news
224     /// </summary>
225     /// <example>
226     /// 2024 FIA Formula One World Championship is a motor racing championship
227     /// for Formula One cars and is the 75th running of the Formula One World Championship.
228     /// </example>
229     3 references
230     public string? Content { get; set; }
231 }
```

Testing

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

The screenshot shows a Swagger UI interface with the following sections:

- POST /staticNews**: A green button.
- GET /staticNews/{id}**: A blue button.
- PUT /staticNews/{id}**: An orange button.
- DELETE /staticNews/{id}**: A red button.

Schemas

```
NewsListStatic ▼ {
    id           integer($int32)
    title        string
    nullable: true
    example: Formula One World Championship
    Title is Any headline news

    content      string
    nullable: true
    Contenty contain details of the news
}
```

Now Searching xml in obj or bin folder

This PC > DATA (D:) > VisualCode > BareFoot_API > MinAPI > obj > Debug > net8.0			
	Name	Date modified	Type
Documents	ref	12/03/2024 11:52	File folder
Pictures	refint	12/03/2024 11:52	File folder
DATA (D)	staticwebassets	12/03/2024 11:52	File folder
Git-HtmxBlog	.NETCoreApp.Version=v8.0.AssemblyAttributes	12/03/2024 11:52	C# Source
Pictures	apphost	12/03/2024 11:52	Application
ttt	MinAPI.AssemblyInfo	12/03/2024 11:52	C# Source
OneDrive - Royal I	MinAPI.AssemblyInfoInputs.cache	12/03/2024 11:52	CACHE File
This PC	MinAPI.assets.cache	12/03/2024 11:52	CACHE File
3D Objects	MinAPI.csproj.AssemblyReference.cache	12/03/2024 11:52	CACHE File
Desktop	MinAPI.csproj.CoreCompileInputs.cache	12/03/2024 11:52	CACHE File
Documents	MinAPI.csproj.FileListAbsolute	12/03/2024 11:52	Text Document
Downloads	MinAPI.csproj.Up2Date	12/03/2024 11:52	UP2DATE
Music	MinAPI.dll	12/03/2024 11:52	Application
Pictures	MinAPI.GeneratedMSBuildEditorConfig	12/03/2024 11:52	Editor Configuration
Videos	MinAPI.GeneratedRuntimeConfig.cache	12/03/2024 11:52	CACHE File
Local Disk (C)	MinAPI.GlobalUsings.g	12/03/2024 11:52	C# Source
DATA (D)	MinAPI.MvcApplicationPartsAssemblyInfo.cache	12/03/2024 11:52	CACHE File
Network	MinAPI.MvcApplicationPartsAssemblyInfo	12/03/2024 11:52	C# Source
Linux	MinAPI.pdb	12/03/2024 11:52	Program
	MinAPI.sourcelink	12/03/2024 11:52	JSON Source
	MinAPI	12/03/2024 11:52	Microsoft

This PC > DATA (D:) > VisualCode > BareFoot_API > MinAPI > bin > Debug > net8.0			
	Name	Date modified	Type
Documents	Microsoft.Win32.SystemEvents.dll	23/10/2021 02:40	Application extension
Pictures	MinAPI.deps	11/03/2024 13:01	JSON Source File
DATA (D)	MinAPI.dll	11/03/2024 13:09	Application extension
Git-HtmxBlog	MinAPI	11/03/2024 13:09	Application
Pictures	MinAPI.pdb	11/03/2024 13:09	Program Debug Data...
ttt	MinAPI.runtimeconfig	11/03/2024 12:50	JSON Source File
OneDrive - Royal I	MinAPI.staticwebassets.runtime	10/03/2024 14:55	JSON Source File
This PC	MinAPI	11/03/2024 13:09	Microsoft Edge HTML...
3D Objects	Mono.TextTemplating.dll	23/02/2021 06:04	Application extension
Desktop	Newtonsoft.Json.Json.dll	28/11/2018 02:10	Application extension

Open this XML , So Our generation work well

```

MinAPI.xml
1  <?xml version="1.0"?>
2  <doc>
3    <assembly>
4      <name>MinAPI</name>
5    </assembly>
6    <members>
7      <member name="P:NewsListStatic.Title">
8        <summary>
9          Title is Any headline news
10         </summary>
11         <example>Formula One World Championship</example>
12       </member>
13       <member name="P:NewsListStatic.Content">
14         <summary>
15           Contenty contain details of the news
16         </summary>
17         <example>
18           2024 FIA Formula One World Championship is a motor racing championship
19             for Formula One cars and is the 75th running of the Formula One World Championship
20           </example>
21         </member>
22       </members>
23     </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 exists");
        varNewslist.Remove(varNews);
        return Results.Ok(varNews);
    }
)
.WithTags("Static News");
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE PORTS GITLENS COMMENTS TERMINAL

Some error shows

C# Program.cs 3, M X

```
MinAPI > C# Program.cs > ...
152     {
153         return Results.BadRequest("HowTo Exsists");
154     }
155
156     NewNewsListStatic.Id = varNewslist.OrderByDescending(c => c.I
157     varNewslist.Add(NewNewsListStatic);
158     return Results.Ok(varNewslist);
159 }
160 }
Alhafisurface, 7 days ago
161 .WithTags("Static News");
162 XML comment is not placed on a valid language element (CS1587)
View Problem (Alt+F8) No quick fixes available
163 // <summary>
164 //Delete Specific News.
165 // </summary>
166 // <param name="id"></param>
167 // <returns></returns>
168 app.MapDelete(
169     "/staticNews/{id}",
170     (int id) =>
171     {
172         var varNews = varNewslist.Find(c => c.Id == id);
173         if (varNews == null)
174             return Results.NotFound("Sorry this News doesn't exists"
175             varNewslist.Remove(varNews);
176             return Results.Ok(varNews);
177         }
178     }
)
181 .WithTags("Static News");
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE PORTS GITLENS COMMENTS TERMINAL

Let we disable errors 1587

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

```

1 <Project Sdk="Microsoft.NET.Sdk.Web">
2
3     <PropertyGroup>
4         <TargetFramework>net8.0</TargetFramework>
5         <Nullable>enable</Nullable>
6         <ImplicitUsings>enable</ImplicitUsings>
7         <GenerateDocumentationFile>true</GenerateDocumentationFile>
8         <NoWarn>$(NoWarn);1591;1587</NoWarn>
9     </PropertyGroup>
10
11     <ItemGroup>
12         <PackageReference Include="AutoMapper.Extensions.Microsoft.DependencyInjection" Version="12.0.1" />
13         <PackageReference Include="Microsoft.AspNetCore.Mvc.NewtonsoftJson" Version="8.0.2" />
14         <PackageReference Include="Microsoft.AspNetCore.OpenApi" Version="8.0.2" />
15         <PackageReference Include="Microsoft.EntityFrameworkCore" Version="8.0.2" />
16         <PackageReference Include="Microsoft.EntityFrameworkCore.Design" Version="8.0.2">
17             <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
18             <PrivateAssets>all</PrivateAssets>
19         </PackageReference>
20         <PackageReference Include="Microsoft.EntityFrameworkCore.Sqlite.Core" Version="8.0.2" />
21         <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="8.0.2" />
22         <PackageReference Include="Swashbuckle.AspNetCore" Version="6.5.0" />
23         <PackageReference Include="Swashbuckle.AspNetCore.Annotations" Version="6.5.0" />
24     </ItemGroup>
25
26 </Project>
27

```

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

Error Notes gone

```

// <summary>
///Delete Specific News.
/// </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" } }
});

```

https://localhost:7070/swagger/index.html

DELETE /hello

Static News

GET /staticNews احصل على جميع الأخبار

Returns information about all the available news from the Alhafi Blog.

Parameters

No parameters

Request body

Execute

Responses

Code	Description
200	OK

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 exists");
        return Results.Ok(varNews);
    }
)
.WithDescription("return one news ")
.WithSummary("إحضار خبر واحد بناء على قيمة رقم المدخل")
.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 exists");
        return Results.Ok(varNews);
    }
)
.WithDescription("return one news ")
.WithSummary("إحضار خبر واحد بناء على قيمة رقم المدخل")

```

```
.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

```
MinAPI > MinAPI.csproj > XML > Project > ItemGroup
You, 16 seconds ago | 2 authors (You and others)
1 <Project Sdk="Microsoft.NET.Sdk.Web">
2
3   <PropertyGroup>
4     <TargetFramework>net8.0</TargetFramework>
5     <Nullable>enable</Nullable>
6     <ImplicitUsings>enable</ImplicitUsings>
7     <GenerateDocumentationFile>true</GenerateDocumentationFile>
8   </PropertyGroup>
9
10  <ItemGroup>    You, last week * Add WebAPI Project
11    <PackageReference Include="AutoMapper.Extensions.Microsoft.DependencyInjection" Version="12.0.1" />
12    <PackageReference Include="Microsoft.AspNetCore.Mvc.NewtonsoftJson" Version="8.0.2" />
13    <PackageReference Include="Microsoft.AspNetCore.OpenApi" Version="8.0.2" />
14    <PackageReference Include="Microsoft.EntityFrameworkCore" Version="8.0.2" />
15    <PackageReference Include="Microsoft.EntityFrameworkCore.Design" Version="8.0.2" />
16      <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
17      <PrivateAssets>all</PrivateAssets>
18    </PackageReference>
19    <PackageReference Include="Microsoft.EntityFrameworkCore.Sqlite.Core" Version="8.0.2" />
20    <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="8.0.2" />
21    <PackageReference Include="Swashbuckle.AspNetCore" Version="6.5.0" />
22    <PackageReference Include="Swashbuckle.AspNetCore.Annotations" Version="6.5.0" />
23  </ItemGroup>
24
25  </Project>
26
```

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

```

File Edit Selection View Go Run Terminal Help
EXPLORER ... C# Post.cs U X
OPEN EDITORS MinAPI > Data > Model > C# Post.cs > Post > Id
BAREFOOT_API
  .github
  MinAPI
    .vs
      Data
        Model
          C# Post.cs
          C# DbContext.cs
        Properties
        wwwroot
        appsettings.Development.json
        appsettings.json
      MinAPI.csproj
      MinAPI.http
      MinAPI.sln
      Program.cs
      .gitignore
      README.md
  1  using System.ComponentModel.DataAnnotations;
  2
  3  namespace MinAPI.Data.Model
  4  {
  5
  6      public class Post
  7  {
  8          [Key]
  9          public int Id { get; set; }
 10
 11         [Required]
 12         public string? Title { get; set; }
 13
 14         public string? Content { get; set; }
 15
 16         public string? postImage { get; set; }
 17
 18     }
  
```

```

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)

```

File Edit Selection View Go Run Terminal Help
EXPLORER ... C# DbContext.cs U X
OPEN EDITORS MinAPI > Data > C# DbContext.cs ...
BAREFOOT_API
  .github
  MinAPI
    .vs
      Data
        Model
          C# Post.cs
          C# DbContext.cs
        Properties
        wwwroot
        appsettings.Development.json
        appsettings.json
      MinAPI.csproj
      MinAPI.http
      MinAPI.sln
      Program.cs
      .gitignore
      README.md
  1  using Microsoft.EntityFrameworkCore;
  2  using MinAPI.Data.Model;
  3
  4  namespace MinAPI.Data
  5  {
  6
  7      public class AppDbContext : DbContext
  8  {
  9
 10         public AppDbContext(DbContextOptions<AppDbContext> options)
 11             : base(options) { }
 12
 13         public DbSet<Post> posts { get; set; }
 14     }
  
```

```

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

MinAPI > appsettings.Development.json M ●

You, 1 minute ago | 1 author (You)

```

1  {
2      "Logging": {
3          "LogLevel": {
4              "Default": "Information",
5              "Microsoft.AspNetCore": "Warning"
6          }
7      },
8      "ConnectionStrings": {
9          "DefaultConnection": "Data source=AlhafiNewsPaper.db"
10     },
11     "AllowedHosts": "*"
12 }

```

```

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

```

Define Connection & Register DBConext as Service

```
Program.cs 2 ●
MinAPI > C# Program.cs > ...
8 //Creating Variables of Lists
9 var varNewslist = new List<NewsListStatic>
10 {
11     new NewsListStatic
12     {
13         Id = 1,
14         Title = "F1 News",
15         Content = "Christian Horner Surprised By Team Strategy At Bahrain GP ."
16     },
17     new NewsListStatic
18     {
19         Id = 2,
20         Title = "Haley will win",
21         Content =
22             "Former South Carolina Gov. Nikki Haley will win the Republican presid
23     },
24 };
25
26 // Add services to the container.
27
28 builder.Services.AddEndpointsApiExplorer();
29 You, 1 hour ago • Uncommitted changes
30 builder.Services.AddDbContext<AppDbContext>(x =>
31     x.UseSqlite(builder.Configuration.GetConnectionString("DefaultConnection"))
32 );
33
34 builder.Services.AddSwaggerGen(c =>
35 {
36     c.EnableAnnotations();
```

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

Migration Database

```

C# 20240312114530_InitialCreateDB.cs U X
MinAPI > Data > Migrations > C# 20240312114530_InitialCreateDB.cs ...
1  using Microsoft.EntityFrameworkCore.Migrations;
2
3  #nullable disable
4
5  namespace MinAPI.Data.Migrations
6  {
7      /// <inheritdoc />
8      public partial class InitialCreateDB : Migration
9      {
10         /// <inheritdoc />
11         protected override void Up(MigrationBuilder migrationBuilder)
12         {
13             migrationBuilder.CreateTable(
14                 name: "Posts",
15                 columns: table => new
16                 {
17                     Id = table.Column<int>(type: "INTEGER", nullable: false)
18                         .Annotation("Sqlite:Autoincrement", true),
19                     Title = table.Column<string>(type: "TEXT", nullable: false),
20                     Content = table.Column<string>(type: "TEXT", nullable: true),
21                     postImage = table.Column<string>(type: "TEXT", nullable: true)
22                 },
23                 constraints: table =>
24                 {
25                     table.PrimaryKey("PK_Posts", x => x.Id);
26                 });
27         }
28     }

```

PROBLEMS (2) OUTPUT DEBUG CONSOLE PORTS GITLENS COMMENTS TERMINAL

PS D:\VisualCode\BareFoot_API\minapi> dotnet ef migrations add InitialCreateDB -o Data/Migrations
Build started...

[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

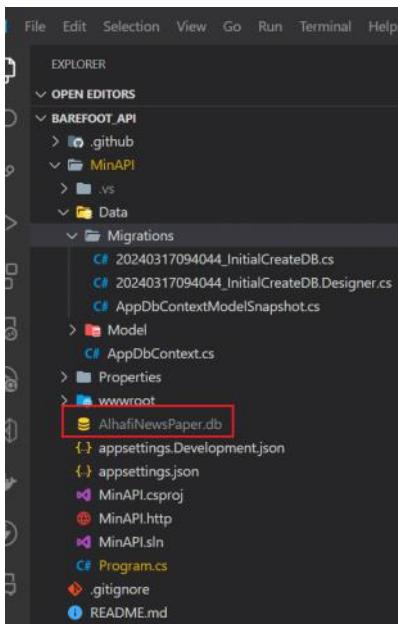
List Migration

```
PS D:\VisualCode\BareFoot_API\minapi> dotnet ef migrations list
Build started...
Build succeeded.
20240312114530_InitialCreateDB (Pending)
20240312114530_SeedingPosts (Pending)
```

Creating Database

dotnet ef database update

Check your DB

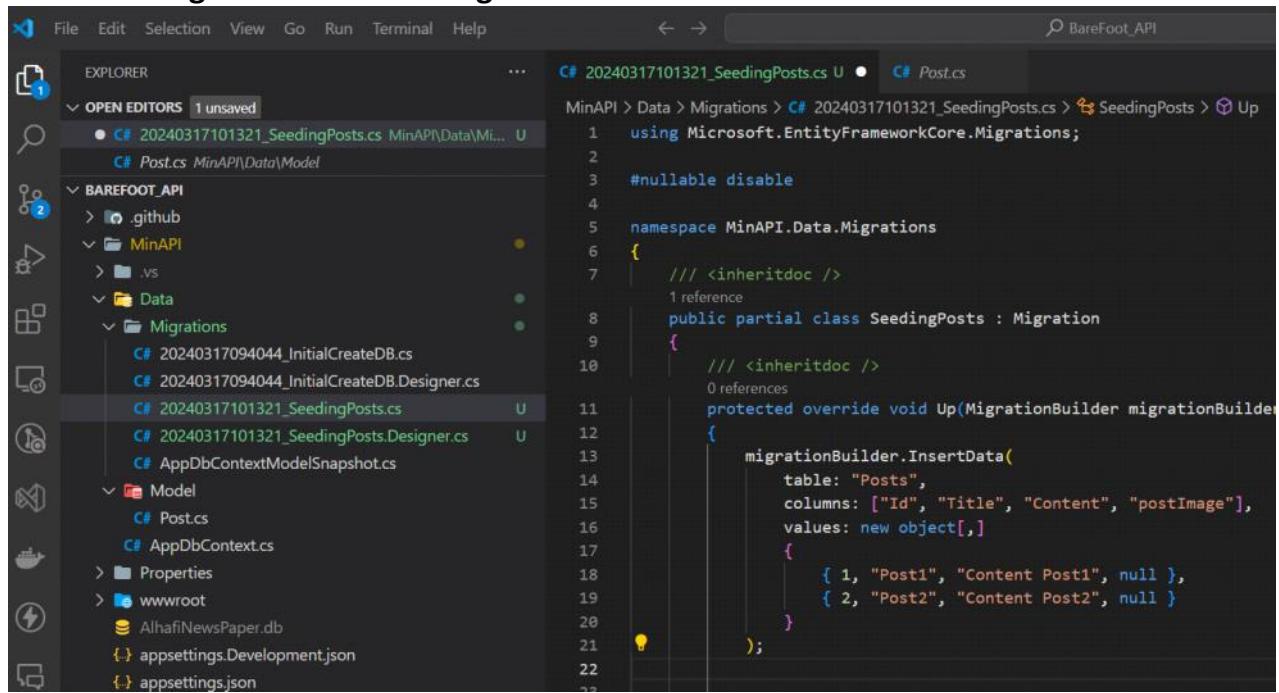


CTRL+Shift+P

The screenshot shows the Visual Studio interface with the Command Palette open. The 'SQLite: Open Database' option is selected and highlighted with a red box. The main workspace shows the 'README.md' file. In the bottom left corner, the SQLite Explorer is visible, displaying the structure of the 'AlhafiNewsPaper.db' database, specifically the 'Posts' table with columns 'Id', 'Title', and 'Content'.

Seeding stage migration

dotnet ef migrations add SeedingPosts



The screenshot shows the Visual Studio IDE interface. The title bar says "BareFoot_API". The left sidebar is the "EXPLORER" view, showing the project structure under "BAREFOOT_API". The "Migrations" folder contains several files: "20240317094044_InitialCreateDB.cs", "20240317094044_InitialCreateDB.Designer.cs", "20240317101321_SeedingPosts.cs" (which is selected), "20240317101321_SeedingPosts.Designer.cs", and "AppDbContextModelSnapshot.cs". The main editor window displays the code for "20240317101321_SeedingPosts.cs". The code defines a partial class "SeedingPosts" that inherits from "Migration". The "Up" method uses "migrationBuilder.InsertData" to insert two posts into the "Posts" table.

```
1  using Microsoft.EntityFrameworkCore.Migrations;
2
3  #nullable disable
4
5  namespace MinAPI.Data.Migrations
6  {
7      //<inheritdoc />
8      public partial class SeedingPosts : Migration
9      {
10         //<inheritdoc />
11         protected override void Up(MigrationBuilder migrationBuilder)
12         {
13             migrationBuilder.InsertData(
14                 table: "Posts",
15                 columns: ["Id", "Title", "Content", "postImage"],
16                 values: new object[,]
17                 {
18                     { 1, "Post1", "Content Post1", null },
19                     { 2, "Post2", "Content Post2", null }
20                 });
21         }
22     };
23 }
```

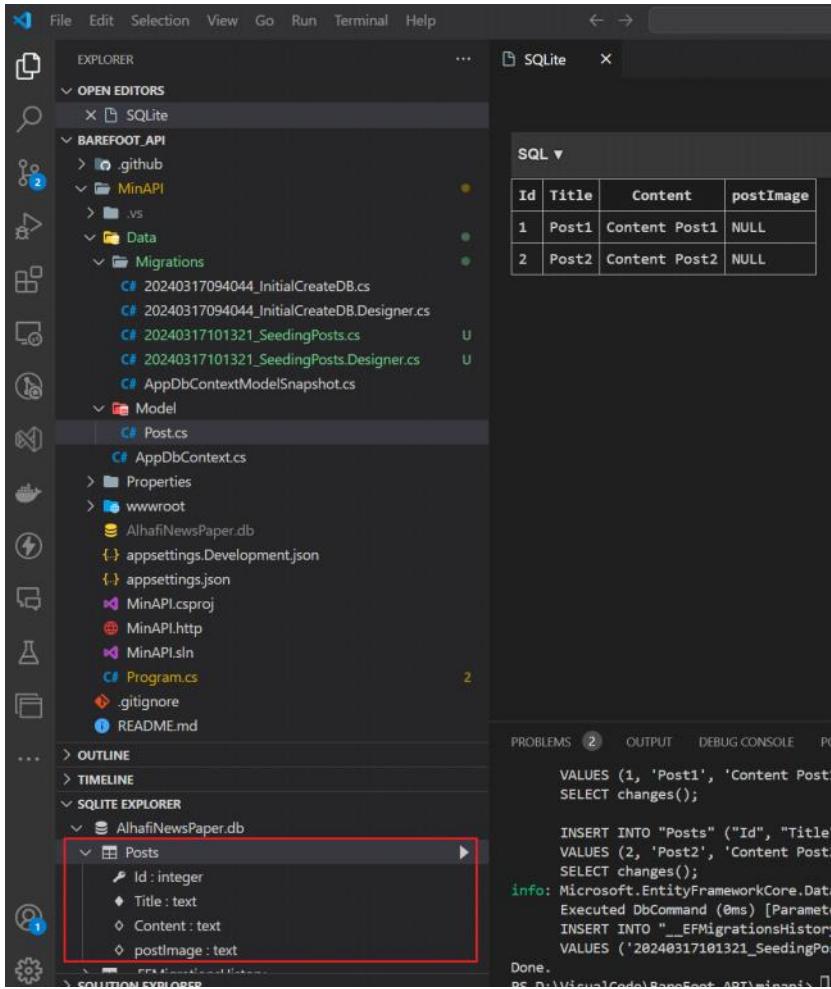
Seeding Data

```
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 ")
    ."احفظ جميع الاخبار"
    .WithSummary("DBContext")
    .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")
    ."احفظ خبر واحد"
    .WithSummary("DBContext")
    .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 جدید")
.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

DBConextPostEndPoints copy.cs 1 ●

MinAPI > EndPoints > DBConextPostEndPoints copy.cs > DBConext > MapDBConextPost

0 references | You, 7 days ago | 1 author (You)

```
14     public static class DBConext
15     {
16         1 reference
17         public static RouteGroupBuilder MapDBConextPost(this RouteGroupBuilder group)
18         {
19             group.MapGet(
20                 "/posts",
21                 async (AppDbContext context) =>
22                 {
23                     var varPosts = await context.Posts.OrderBy(p=>p.Title).ToListAsync();
24                     return Results.Ok(varPosts);
25                 }
26             ).WithDescription("return All posts news ")
27             .WithSummary("احضار جميع الاخبار")
28             .WithOpenApi()
29             .CacheOutput(c => c.Expire(TimeSpan.FromDays(360)).Tag("Post_Get"));
30
31         group.MapGet(
32             "/posts/{id}",
33             async (AppDbContext context, int id) =>
34             {
35                 var varPost = await context.Posts.FirstOrDefaultAsync(c => c.Id == id);
36             }
37         ).WithDescription("return post by id {id} ")
38         .WithSummary("احضار الاخبار من اجل المعرفة")
39         .WithOpenApi()
40         .CacheOutput(c => c.Expire(TimeSpan.FromDays(360)).Tag("Post_By_Id"));
41
42     }
43 }
```

Descending

DBConextPostEndPoints copy.cs 1 ●

MinAPI > EndPoints > DBConextPostEndPoints copy.cs > DBConext > MapDBConextPost

0 references | ...

```
14     public static class DBConext
15     {
16         1 reference
17         public static RouteGroupBuilder MapDBConextPost(this RouteGroupBuilder group)
18         {
19             group.MapGet(
20                 "/posts",
21                 async (AppDbContext context) =>
22                 {
23                     var varPosts = await context.Posts.OrderByDescending(p=>p.Title).ToListAsync();
24                     return Results.Ok(varPosts);
25                 }
26             ).WithDescription("return All posts news ")
27             .WithSummary("احضار جميع الاخبار")
28             .WithOpenApi()
29             .CacheOutput(c => c.Expire(TimeSpan.FromDays(360)).Tag("Post_Get"));
30
31         group.MapGet(
32             "/posts/{id}",
33             async (AppDbContext context, int id) =>
34             {
35                 var varPost = await context.Posts.FirstOrDefaultAsync(c => c.Id == id);
36             }
37         ).WithDescription("return post by id {id} ")
38         .WithSummary("احضار الاخبار من اجل المعرفة")
39         .WithOpenApi()
40         .CacheOutput(c => c.Expire(TimeSpan.FromDays(360)).Tag("Post_By_Id"));
41
42     }
43 }
```

Testing

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

The screenshot shows a web browser window with the title bar "Swagger UI". The address bar displays the URL "https://localhost:7010/swagger/index.html". The main content area is titled "BareFoot Swagger Documentation" and features a logo of a bare foot. Below the title, it says "Barefoot API v1 OAS3". A link to "/swagger/v1/swagger.json" is provided. A note states: "BareFoot Minimal API Build in dotnet new webapi -minimal Hosted at github [here](#)". It also includes links to "Terms of service" and "Alhafi.BareFoot - Website" along with an email link to "Send email to Alhafi.BareFoot".

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} إدخال خبر جديد

PUT /posts/{id} تعديل خبر

DELETE /posts/{id} حذف خبر

Hello

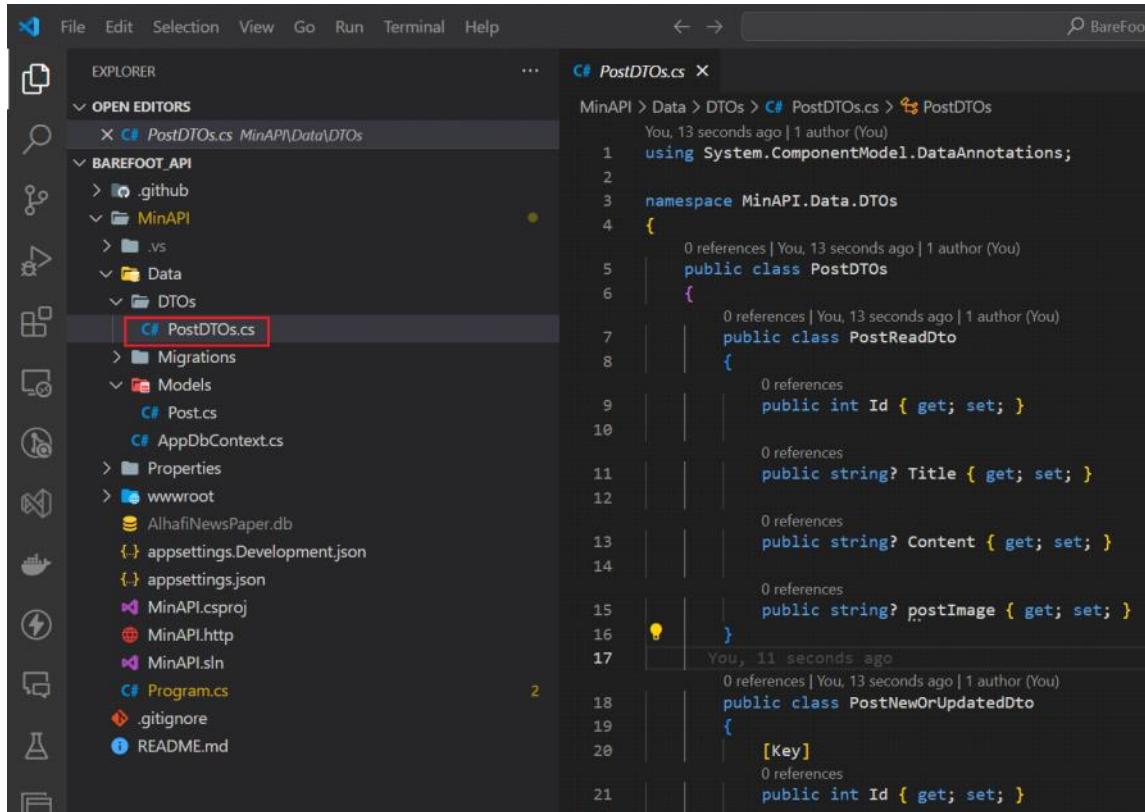
GET /hello

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



The screenshot shows the Visual Studio interface with the 'PostDTOs.cs' file open in the code editor. The file is located in the 'MinAPI\Data\DTOs' folder. The code defines two classes: 'PostReadDto' and 'PostNewOrUpdatedDto'. Both classes inherit from 'PostDTOs'. The 'PostReadDto' class has properties for Id, Title, Content, and postImage. The 'PostNewOrUpdatedDto' class has a [Key] attribute on the Id property and properties for Title, Content, and postImage.

```
File Edit Selection View Go Run Terminal Help
EXPLORER OPEN EDITORS BAREFOOT_API .github MinAPI .vs Data DTOs PostDTOs.cs
C# PostDTOs.cs MinAPI\Data\DTOs
C# PostDTOs.cs
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; }
        }
    }
}
```

```
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

The screenshot shows the Visual Studio IDE with the code editor open to the `IPostRepo.cs` file. The file is located in the `MinAPI\Interfaces` folder under the `BAREFOOT_API` project. The code defines a public interface `IPostRepo` with methods for saving changes, getting posts by ID or all posts, creating, and deleting posts.

```

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);
    }
}

```

```

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

The screenshot shows the Visual Studio IDE with the code editor open to the `PostRepo.cs` file. The file is located in the `MinAPI\Data` folder under the `BAREFOOT_API` project. The code implements the `IPostRepo` interface using Entity Framework Core. It includes a constructor that takes an `AppDbContext` and initializes a private field `_context`. It also includes implementations for the `CreatePost` method, which checks if the post is null and throws an exception if so, then adds it to the context.

```

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);
        }
    }
}

```

```

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

The screenshot shows the Visual Studio IDE interface. The left sidebar (Explorer) lists files in the 'BAREFOOT_API' project, including 'PostProfile.cs' under 'MinAPI\Profiles'. The right pane shows the code for 'PostProfile.cs'.

```

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

```

```

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

```

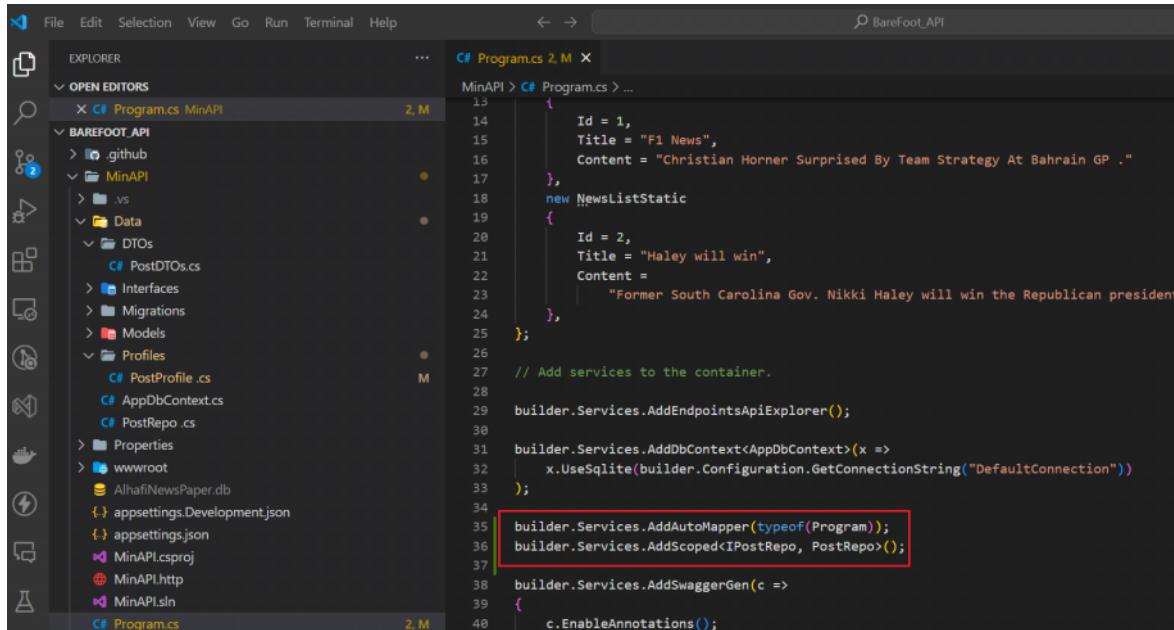
```
    }  
}
```

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("احفظ جميع الاخبار")  
.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("احفظ 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 جدید")
.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();

//*****

```

The screenshot shows the Barefoot Swagger Documentation interface. At the top, there's a navigation bar with a flame icon, the text "BareFoot Swagger Documentation", and a dropdown menu "Select a definition" set to "BareFoot API V1".

Below the navigation bar, the main content area displays two sections:

- AutoMapper** (highlighted in green):
 - GET /automapper/posts** (green button): "ادخال خبر جدید"
 - POST /automapper/posts** (green button): "تخلیل خبر جدید"
 - GET /automapper/posts/{id}** (blue button): "احصیر خبر واحد"
 - PUT /automapper/posts/{id}** (orange button): "تعديل خبر"
 - DELETE /automapper/posts/{id}** (red button): "حذف خبر"
- DBContext** (highlighted in blue):
 - GET /dbcontext** (blue button): "احصیر دیتابیس"

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/>

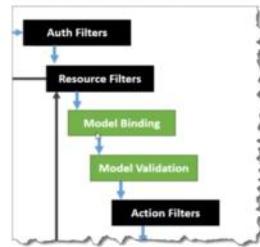
This means*...

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`



The screenshot shows a code editor with the following details:

- File Path:** MinAPI > Data > Models > Post.cs
- Annotation:** [MaxLength(25)] is highlighted with a red box.
- Annotations List:** A sidebar on the right lists the following annotations:
 - [Key]
 - [Required]
 - [MaxLength(25)]
- Code Snippet:**

```
1  using System.ComponentModel.DataAnnotations;
2
3  namespace MinAPI.Data.Models
4  {
5      public class Post
6      {
7          [Key]
8          public int Id { get; set; }
9
10         [Required]
11         [MaxLength(25)]
12         public string? Title { get; set; }
13         You, last month
14         public string? Content { get; set; }
15
16         public string? postImage { get; set; }
17     }
18 }
```

Let We test

Accepted Not Validate

Execute

Responses

Curl

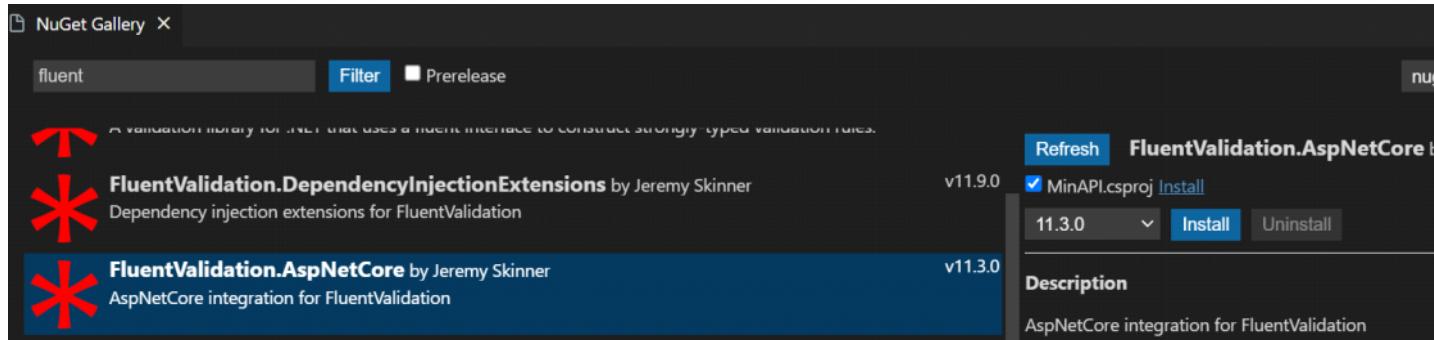
```
curl -X 'GET' \
  'https://localhost:7010/automapper/posts' \
  -H 'accept: */*'
```

Request URL

<https://localhost:7010/automapper/posts>

Server response

Let we add Fluent Validation Package



Then Add validation

The screenshot shows the Visual Studio code editor with the file `Post.cs` open. The code defines a `Post` class with properties `Title`, `Content`, and `postImage`. It also contains a `Validator` class that implements `AbstractValidator<Post>`. The `Validator` class has a constructor that calls `RuleFor(p => p.Title)`, which includes validation rules for `NotEmpty()`, `NotNull()`, and `MaximumLength(25)`.

```
public class Post
{
    public string? Title { get; set; }
    public string? Content { get; set; }
    public string? postImage { get; set; }
}

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");
    }
}
```

```
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

```
# Program.cs 2 ●

MinAPI > C# Program.cs > ...
Content =
26
27         "Former South Carolina Gov. Nikki Haley will win the Republican presidential primary in Washington"
28     },
29 };
30
31 // Add services to the container.
32
33 builder.Services.AddEndpointsApiExplorer();
34
35 builder.Services.AddDbContext<AppDbContext>(x =>
36     x.UseSqlite(builder.Configuration.GetConnectionString("DefaultConnection"))
37 );
38
39 builder.Services.AddAutoMapper(typeof(Program));
40 builder.Services.AddScoped<IPostRepo, PostRepo>();
41
42 //builder.Services.AddValidatorsFromAssemblyContaining(typeof(PostValidator));
43 builder.Services.AddValidatorsFromAssemblyContaining<Program>(ServiceLifetime.Singleton);
44
45 builder.Services.AddSwaggerGen(c =>
46 {
47     c.EnableAnnotations();
48     c.SwaggerDoc(
49         "v1",
50             new OpenApiInfo      You, 2 months ago
51                 {
52                     Title = "Barefoot API"
```

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

Let we validate Post

```
MinAPI > C# Program.cs > ...
246         }
247         return Results.NotFound();
248     }
249 }
250 .WithDescription("return Only One Post News")
251 .WithSummary("احضار خبر واحد")
252 .WithTags("DbContext")
253 .WithOpenApi();
254
255 app.MapPost(
256     "/dbContext/posts",
257     async (AppDbContext context, Post poss) =>
258     {
259         await context.Posts.AddAsync(poss);
260         await context.SaveChangesAsync();
261         return Results.Created($"/posts/{poss.Id}", poss);
262     }
263 )
264 .WithDescription("Insert New Post News")
265 .WithSummary("ادخال خبر جديد")
266 .WithTags("DbContext")
267 .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 جدید")  
.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 جدید")
.WithTags("DbContext")
.WithOpenApi();
```

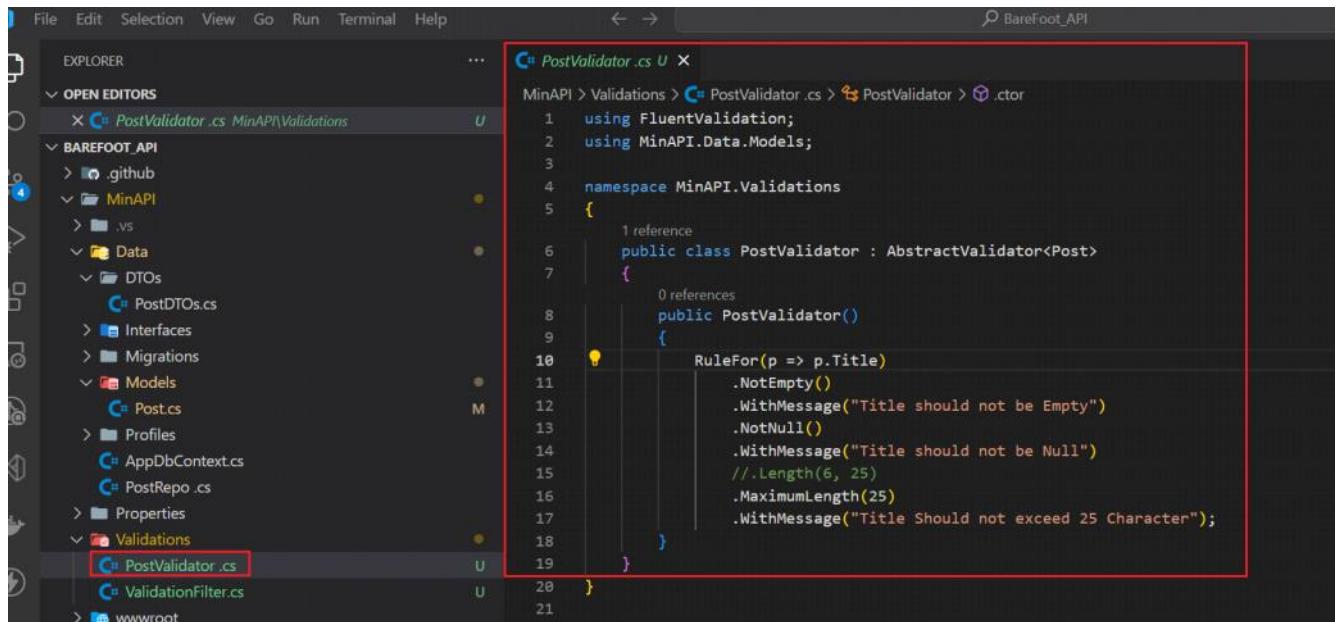
Let we Test

```
{  
  "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

```

C# Program.cs 2 ●
MinAPI > C# Program.cs > ...
...
24     {
25         Id = 2,
26         Title = "Haley will win",
27         Content =
28             "Former South Carolina Gov. Nikki Haley will win the Republican presidential primary in Washington, DC"
29     },
30 };
31
32 // Add services to the container.
33
34 builder.Services.AddEndpointsApiExplorer();
35
36 builder.Services.AddDbContext<AppDbContext>(x =>
37     x.UseSqlite(builder.Configuration.GetConnectionString("DefaultConnection"))
38 );
39
40 builder.Services.AddAutoMapper(typeof(Program));
41 builder.Services.AddScoped<IPostRepo, PostRepo>();
42
43 //builder.Services.AddValidatorsFromAssemblyContaining(typeof(PostValidator));
44 [builder.Services.AddScoped<IValidator<Post>, PostValidator>()];
45 //builder.Services.AddValidatorsFromAssemblyContaining<Program>(ServiceLifetime.Singleton); You, 23 minutes ago + Undo
46
47 builder.Services.AddSwaggerGen(c =>
48 {
49     c.EnableAnnotations();
50     c.SwaggerDoc(
51         "v1",
52         new OpenApiInfo

```

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("إدخال 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
    );
}
)

```

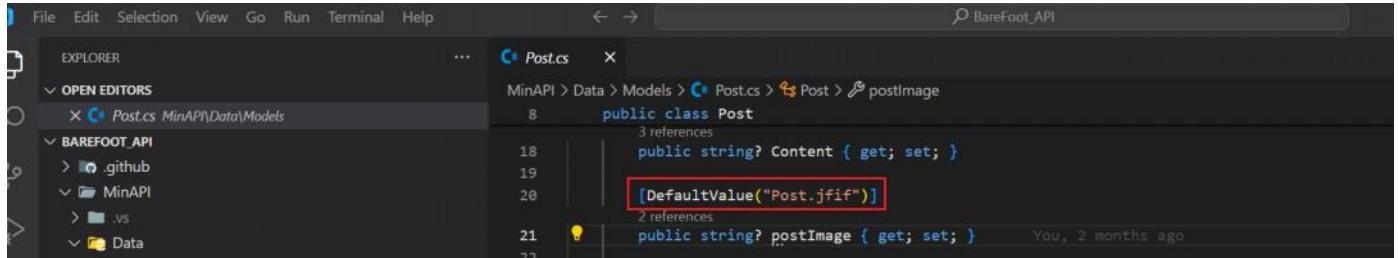
```

.WithDescription("Insert New Post News")
.WithSummary("ادخال خبر جدید")
.WithTags("DbContext")
.WithOpenApi();

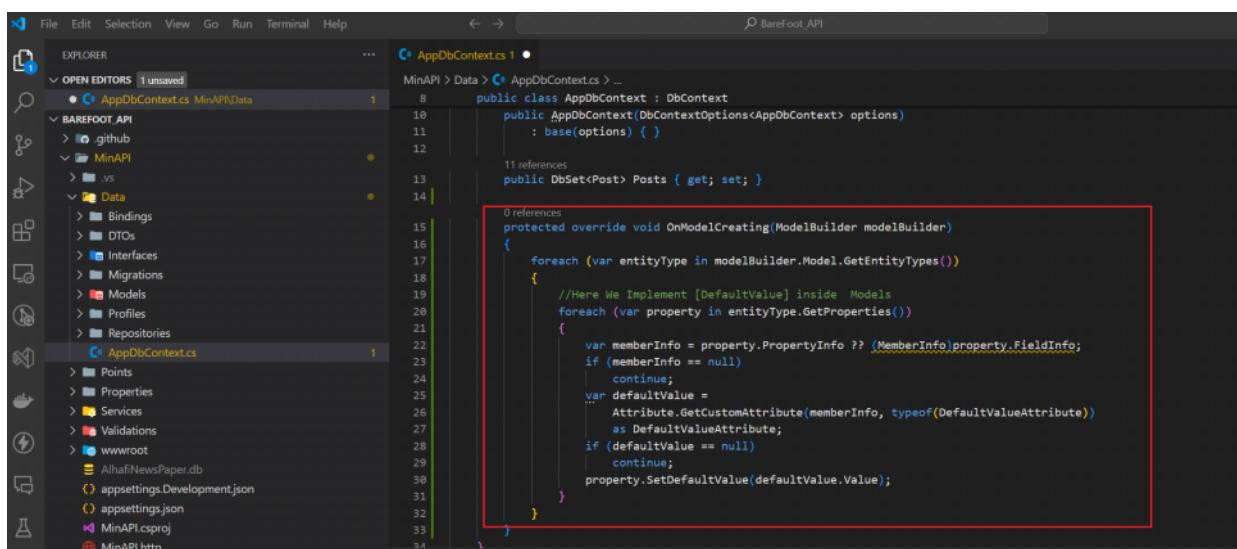
```

Default Values

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

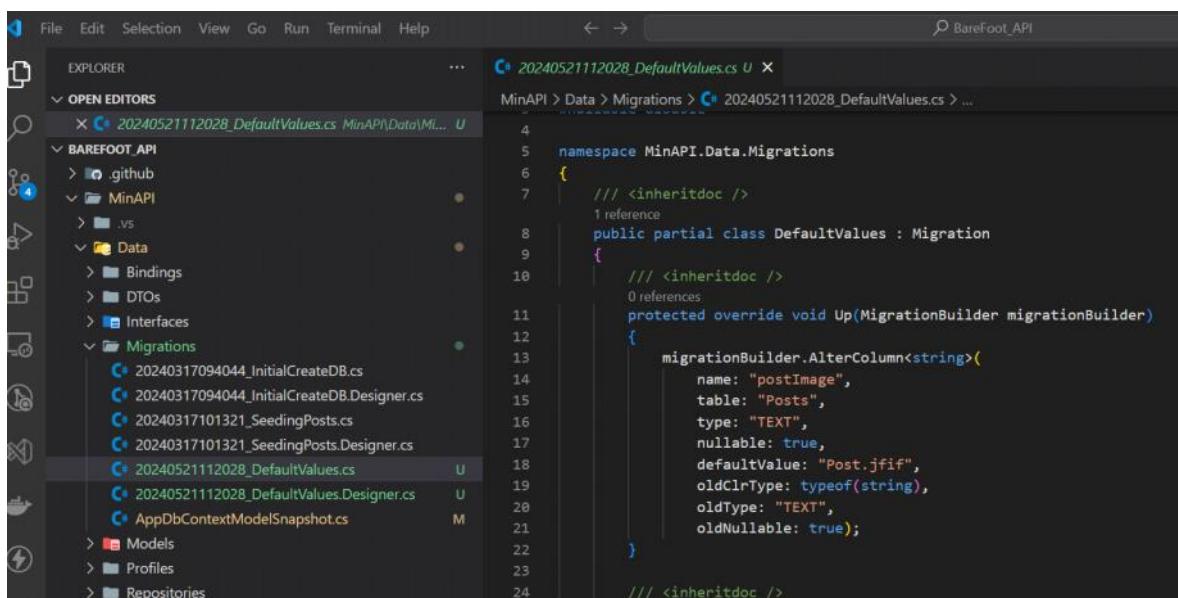


Let we add on Model creation



```
PS D:\VisualCode\BareFoot_API\minaPI> dotnet ef migrations add DefaultValues
```

dotnet ef migrations add DefaultValues



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

dotnet ef database update

POST /automapper/automapper/posts ادخل خبر جديد

Insert New Post News

Parameters

No parameters

Request body required

```
{
  "title": "post3",
  "content": "post3",
  "postImage": null
}
```

Curl

```
curl -X 'POST' \
'https://localhost:7010/automapper/automapper/posts' \
-H 'accept: */*' \
-H 'Content-Type: application/json' \
-d '{
  "title": "post3",
  "content": "post3",
  "postImage": null
}'
```

Request URL

```
https://localhost:7010/automapper/automapper/posts
```

Server response

Code	Details
201 <small>Undocumented</small>	Response body
	{ "id": 3, "title": "post3", "content": "post3", "postImage": "Post.jfif" }

Default Values for Date & Time

```
C# Post.cs
MinAPI > Data > Models > C# Post.cs > ...
  9   public class Post
16     public string? Title { get; set; }
17
18     [SwaggerParameter(Description = "Property description", Required = false)]
19       3 references
20         public string? Content { get; set; }
21
22         [DefaultValue("Post.jfif")]
23           2 references
24             public string? postImage { get; set; }
25
26         [Column("CreatedOn", TypeName = "SmallDateTime")]
27           [DisplayFormat(DataFormatString = "{0:dd-MM-yyyy}", ApplyFormatInEditMode = true)]
28             0 references
29               public DateTime? CreatedOn { get; set; }
```

The screenshot shows the `AppDbContext.cs` file in Visual Studio. The code implements the `OnModelCreating` method to set default values for properties. A specific line of code, which sets the `CreatedOn` property to the current date, is highlighted with a red box.

```
public class AppDbContext : DbContext
{
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        foreach (var entityType in modelBuilder.Model.GetEntityTypes())
        {
            //Here We Implement [DefaultValue] inside Models
            foreach (var property in entityType.GetProperties())
            {
                var memberInfo = property.PropertyInfo ?? (MemberInfo)property.FieldInfo;
                if (memberInfo == null)
                    continue;
                var defaultValue =
                    Attribute.GetCustomAttribute(memberInfo, typeof(DefaultValueAttribute))
                    as DefaultValueAttribute;
                if (defaultValue == null)
                    continue;
                property.SetDefaultValue(defaultValue.Value);
            }
        }

        //Fixed defaultValue for datetime in SQL for All Entities as one
        entityType.FindProperty("CreatedOn")?.SetDefaultValueSql("GETDATE()");
    }
}
```

```
dotnet ef migrations add DefaultValuesDateTime
```

A screenshot of a code editor showing a list of migration files in the left pane and a partial code snippet in the right pane. The migration files listed are:

- 20240521142020_DefaultValues.Designer.cs
- 2024052113144_DefaultValuesDateTime.cs
- 2024052113144_DefaultValuesDateTime.Des...
- AppDbContextModelSnapshot.cs

The right pane shows the following C# code:

```
        type: "smalldatetime",
        nullable: true,
        defaultValueSql: "GETDATE()");
```

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.

```

C# Program.cs 3, M X
MinAPI > C# Program.cs > ...
108 }
109
110 app.UseHttpsRedirection();
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 //*****Static Record End Points(Data Will not save after close )*****
131

```

```

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.

The screenshot shows a Swagger UI interface for a `GET /hello/QueryString` endpoint. The `Parameters` section displays a single parameter named `name` with a description of `AbdullRaheim Alhafi`. The `Responses` section shows a successful `200` response with the body `Hello AbdullRaheim Alhafi`.

Parameter binding with dependency injection

```

File Edit Selection View Go Run Terminal Help
EXPLORER OPEN EDITORS SystemDateTime.cs MinAPI.Data.Models
BAREFOOT_API > .github > MinAPI > Data > Models > SystemDateTime.cs ...
SystemDateTime.cs U X
MinAPI > Data > Models > SystemDateTime.cs ...
1 namespace MinAPI.Data.Models
2 {
3     0 references
4         public class SystemDateTime : IDateTime
5     {
6         1 reference
7             public string Now => DateTime.Now.ToString();
8     }
9     1 reference
10    public interface IDateTime
11    {
12        1 reference
13            string Now { get; }
14    }
15 }

```

```

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.

```

File Edit Selection View Go Run Terminal Help
EXPLORER OPEN EDITORS Program.cs MinAPI
BAREFOOT_API > .github > MinAPI > Data > Models > SystemDateTime.cs ...
Program.cs M X
MinAPI > Program.cs ...
29     Content =
30         "Former South Carolina Gov. Nikki Haley will win the Republican presidential primary in Washington,
31     },
32 };
33
34 // Add services to the container.
35
36 builder.Services.AddEndpointsApiExplorer();
37
38 builder.Services.AddDbContext<AppDbContext>(x =>
39     x.UseSqlite(builder.Configuration.GetConnectionString("DefaultConnection")));
40 );
41
42 builder.Services.AddAutoMapper(typeof(Program));
43 builder.Services.AddScoped<IPostRepo, PostRepo>();
44
45 //builder.Services.AddValidatorsFromAssemblyContaining(typeof(PostValidator));
46 //builder.Services.AddScoped<IValidator<Post>, PostValidator>();
47 //builder.Services.AddScoped<IValidator<PostNewOrUpdatedDto>, PostNewOrUpdatedDtoValidator>();
48
49
50 builder.Services.AddSingleton<IDateTime, SystemDateTime>(); You, 1 second ago * Uncommitted changes
51 builder.Services.AddValidatorsFromAssemblyContaining<Program>(ServiceLifetime.Singleton);
52

```

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

```
C# Program.cs 3, M X
MinAPI > C# 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

The screenshot shows the MinAPI documentation interface for the `/Demo` endpoint. At the top, there is a navigation bar with a blue button labeled "GET" and a URL field containing `/Demo`. Below this is a "Parameters" section which states "No parameters". Under the "Responses" section, there is a "Curl" block with the command:

```
curl -X 'GET' \
'https://localhost:7010/Demo' \
-H 'accept: text/plain'
```

Below the curl command is a "Request URL" block showing `https://localhost:7010/Demo`. Under the "Server response" section, there is a table with two columns: "Code" and "Details". The "Code" column contains "200" and the "Details" column contains "Response body". A red box highlights the "Response body" section, which displays the current date and time: `30/04/2024 14:46:55`. Below the response body is a "Response headers" section.

It's not necessary to explicitly apply it as below

```

C# Program.cs 3, M ×
MinAPI > C# 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("/Demo2", ([FromServices] IDateTime dateTime) => dateTime.Now);
132
133     //*****Static Record End Points(Data will not save after close *****

```

```
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.

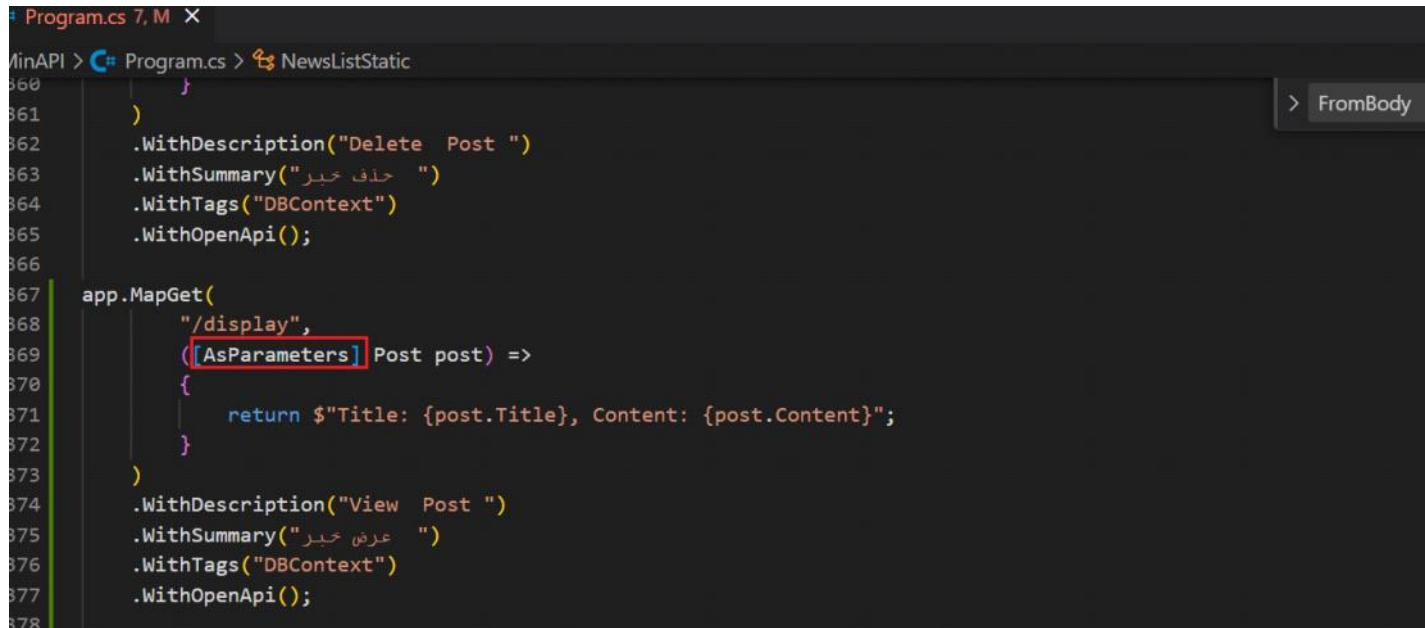
```

app.MapPost(
    "/dbcontext/posts/v2",
    async (AppDbContext context, [FromBody] Post poss) => You, last week
    {
        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 جدید")
.WithTags("DbContext")
.WithOpenApi();

```

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.



```
# Program.cs 7, M X
MinAPI > C# Program.cs > NewsListStatic
360     }
361     .WithDescription("Delete Post ")
362     .WithSummary(" حذف خبر ")
363     .WithTags("DBContext")
364     .WithOpenApi();
365
366
367 app.MapGet(
368     "/display",
369     ([AsParameters] Post post) =>
370     {
371         return $"Title: {post.Title}, Content: {post.Content}";
372     }
373 )
374     .WithDescription("View Post ")
375     .WithSummary("عرض خبر ")
376     .WithTags("DBContext")
377     .WithOpenApi();
378
app.MapGet(
    "/display",
    ([AsParameters] Post post) =>
{
    return $"Title: {post.Title}, Content: {post.Content}";
}
)
    .WithDescription("View Post ")
    .WithSummary("عرض خبر ")
    .WithTags("DBContext")
    .WithOpenApi();
```

View Post

Parameters

Name	Description
Id <small>required</small>	integer(\$int32) (query) 1
Title	string (query) aaa
Content	string (query) bbb
postImage	string (query) postImage

Execute

Responses

Curl

```
curl -X 'GET' \
  "https://localhost:7010/display?Id=1&Title=aaa&Content=bbb" \
  -H 'accept: text/plain'
```

Request URL

https://localhost:7010/display?id=1&title=aaa&content=bbb

Server response

Code	Details
200	Response body Title: aaa, Content: bbb

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

Custom model binding in minimal APIs

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)

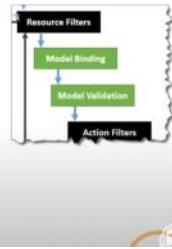
Custom model binding allows developers to define their own binding logic for complex data types or scenarios that cannot be handled by the default binding mechanisms. Custom binding is particularly useful when working with APIs that require data transformation or normalization before the data can be 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.

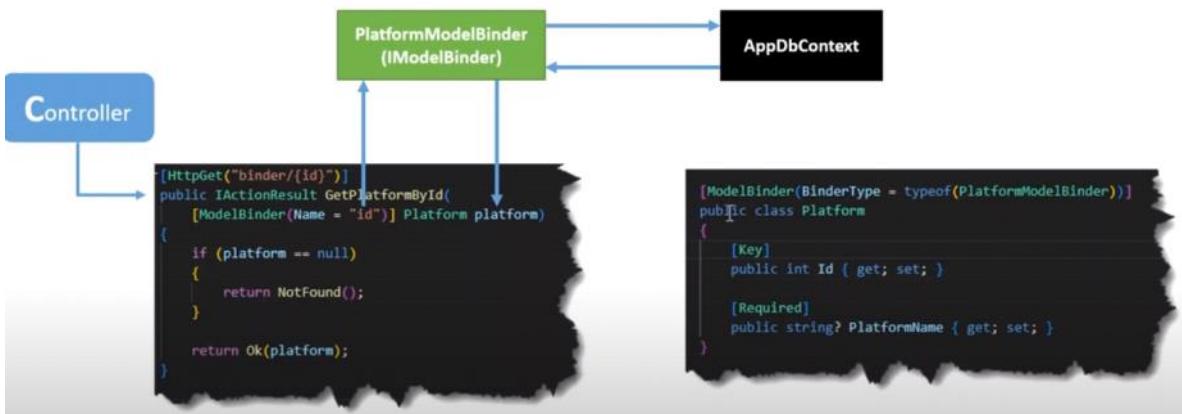
In the `BindModelAsync` method, you can perform any necessary data transformation or validation before returning the bound model.

Custom Model Binding (IModelBinder)

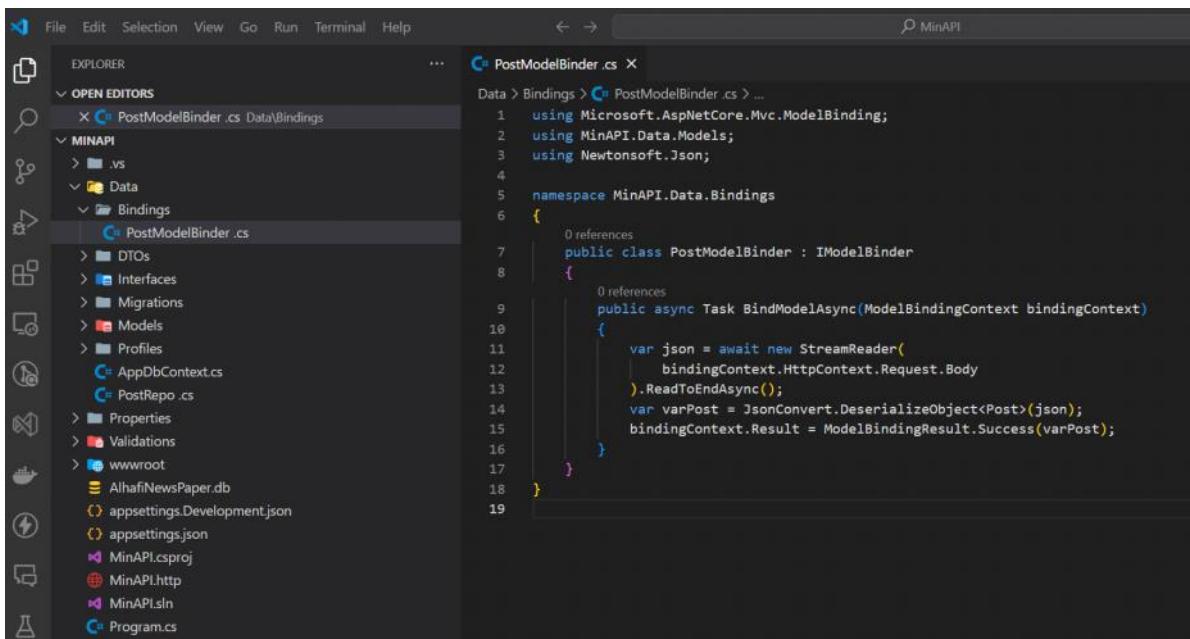
- Allow Controller Actions to work directly with Model Types
 - (Rather than HTTP requests)
- Can be used in more complex “binding scenarios”
 - E.g. if you need to transform data
- Default model binders support most common .NET types
 - They meet most developers needs!
- Model Binding Occurs before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

```

```

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 جدید")
.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 \(c-sharpcorner.com\)](https://c-sharpcorner.com)

Endpoint Filters

21 April 2024 13:55

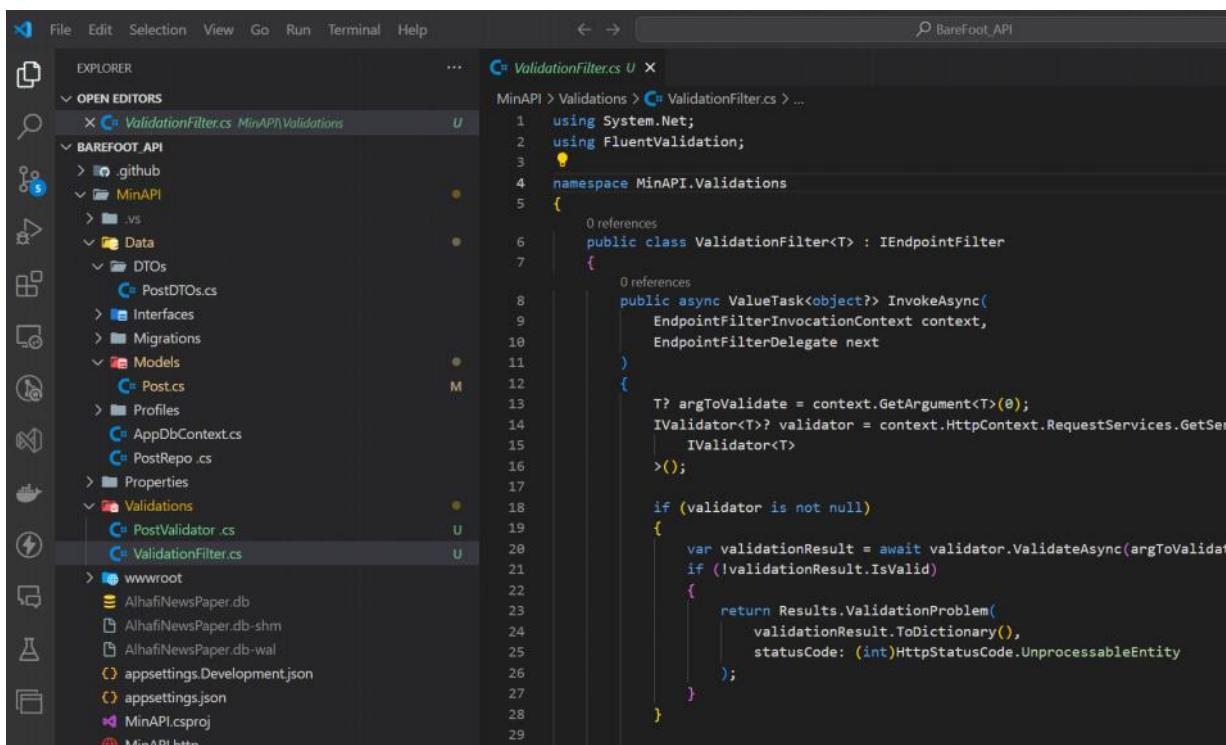
This means*...

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:



```
File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
BAREFOOT_API
> github
> MinAPI
> .vs
> Data
> DTOs
> PostDTOs.cs
> Interfaces
> Migrations
> Models
> Post.cs
> Profiles
> AppDbContext.cs
> PostRepo.cs
> Properties
> Validations
> PostValidator.cs
> ValidationFilter.cs
> wwwroot
> AlhafiNewsPaper.db
> AlhafiNewsPaper.db-shm
> AlhafiNewsPaper.db-wal
> appsettings.Development.json
> appsettings.json
> MinAPI.csproj
> MinAPI.hhp

C ValidationFilter.cs U x
MinAPI > Validations > ValidationFilter.cs ...
1 using System.Net;
2 using FluentValidation;
3
4 namespace MinAPI.Validations
5 {
6     public class ValidationFilter<T> : IEndpointFilter
7     {
8         public async ValueTask<object?> InvokeAsync(
9             EndpointFilterInvocationContext context,
10            EndpointFilterDelegate next
11        )
12        {
13            T? argToValidate = context.GetArgument<T>(0);
14            IValidator<T>? validator = context.HttpContext.RequestServices.GetService<IValidator<T>>();
15            if (validator is not null)
16            {
17                var validationResult = await validator.ValidateAsync(argToValidate);
18                if (!validationResult.IsValid)
19                {
20                    return Results.ValidationProblem(
21                        validationResult.ToDictionary(),
22                        statusCode: (int) HttpStatusCode.UnprocessableEntity
23                    );
24                }
25            }
26        }
27    }
28}
29
```

```
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);
}
}
}

```

C# Program.cs 2 ●

MinAPI > C# Program.cs > ...

```

258
259     app.MapPost(
260         "/dbcontext/posts",
261         async (AppDbContext context, Post poss, IValidator<Post> validator) =>
262     {
263         var validationResult = await validator.ValidateAsync(poss);
264
265         if (validationResult.IsValid)
266         {
267             await context.Posts.AddAsync(poss);
268             await context.SaveChangesAsync();
269             return Results.Created($"/posts/{poss.Id}", poss);
270         }
271         return Results.ValidationProblem(
272             validationResult.ToDictionary(),
273             statusCode: (int) HttpStatusCode.UnprocessableEntity
274         );
275     }
276 )
277 .WithDescription("Insert New Post News")
278 .WithSummary("ادخال خبر جدید")
279 .WithTags("DbContext")
280 .WithOpenApi();
281
282     app.MapPost(
283         "/dbcontext/posts/v2",
284         async (AppDbContext context, [FromBody] Post poss) =>
285     {
286         await context.Posts.AddAsync(poss);
287         await context.SaveChangesAsync();
288         return Results.Created($"/posts/{poss.Id}", poss);
289     }
290 )
291     .AddEndpointFilter<ValidationFilter<Post>>()
292     .WithDescription("Insert New Post News")
293     .WithSummary("ادخال خبر جدید")
294     .WithTags("DbContext")
295     .WithOpenApi();
296

```

You, 16 minutes ago * Uncommit

```

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

```

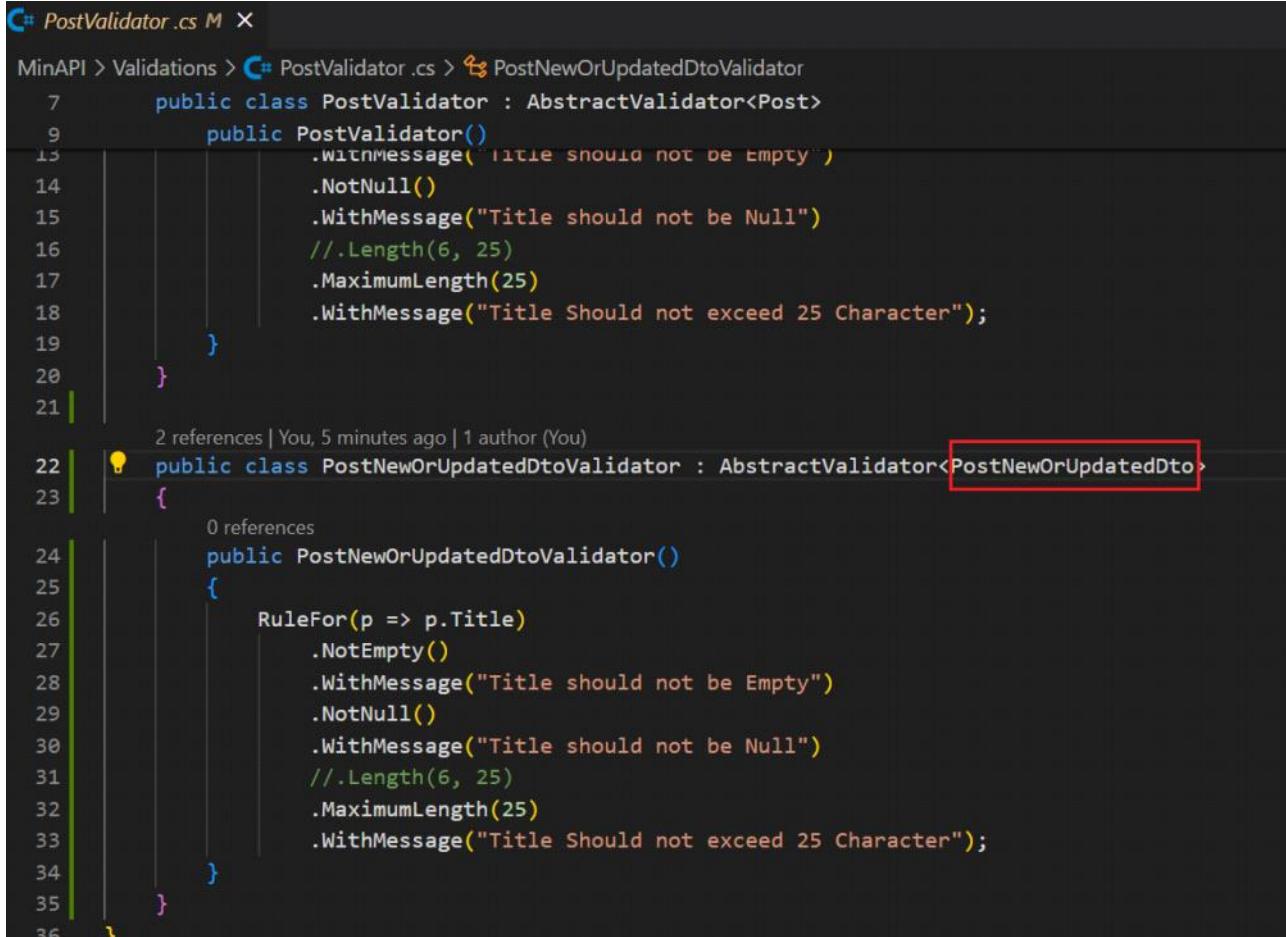
```

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

```

Filtration in Automapping

Adding rule to DTO



```

MinAPI > Validations > PostValidator.cs M X
MinAPI > Validations > PostValidator.cs > PostNewOrUpdatedDtoValidator
7     public class PostValidator : AbstractValidator<Post>
9         public PostValidator()
13             .WithMessage("Title should not be Empty")
14             .NotNull()
15             .WithMessage("Title should not be Null")
16             //.Length(6, 25)
17             .MaximumLength(25)
18             .WithMessage("Title Should not exceed 25 Character");
19     }
20 }
21
22 2 references | You, 5 minutes ago | 1 author (You)
22  public class PostNewOrUpdatedDtoValidator : AbstractValidator<PostNewOrUpdatedDto>
23 {
24     0 references
25         public PostNewOrUpdatedDtoValidator()
26         {
27             RuleFor(p => p.Title)
28                 .NotEmpty()
29                 .WithMessage("Title should not be Empty")
30                 .NotNull()
31                 .WithMessage("Title should not be Null")
32                 //.Length(6, 25)
33                 .MaximumLength(25)
34                 .WithMessage("Title Should not exceed 25 Character");
35         }
36 }

```

```

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

```

C# Program.cs 2, M X
MinAPI > C# Program.cs > ...
40
41     builder.Services.AddAutoMapper(typeof(Program));
42     builder.Services.AddScoped<IPostRepo, PostRepo>();
43
44     //builder.Services.AddValidatorsFromAssemblyContaining(typeof(PostValidator));
45     builder.Services.AddScoped<IValidator<Post>, PostValidator>();
46     builder.Services.AddScoped<IValidator<PostNewOrUpdatedDto>, PostNewOrUpdatedDtoValidator>();
47
48
49
50     //builder.Services.AddValidatorsFromAssemblyContaining<Program>(ServiceLifetime.Singleton);
51
52     builder.Services.AddSwaggerGen(c =>
53     {
54         c.EnableAnnotations();
55         c.SwaggerDoc(

```

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

```

'1
'2     app.MapPost(
'3         "/automapper/posts",
'4         async (IPostRepo repo, IMapper mapper, [FromBody] PostNewOrUpdatedDto postCreateDto) =>
'5     {
'6         var postModel = mapper.Map<Post>(postCreateDto);
'7         await repo.CreatePost(postModel);
'8         await repo.SaveChanges();
'9         var postReadDto = mapper.Map<PostReadDto>(postModel);
'10        return Results.Created($"/automapper/posts/{postReadDto.Id}", postReadDto);
'11    }
'12 )
'13     .AddEndpointFilter<ValidationFilter<PostNewOrUpdatedDto>>()
'14     .WithDescription("Insert New Post News")
'15     .WithSummary("ادخال خبر جدید")
'16     .WithTags("AutoMapper")
'17     .WithOpenApi();

```

```

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 جدید")
.WithTags("AutoMapper")
.WithOpenApi();

```

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

C# Program.cs 2, M X

```
MinAPI > C# Program.cs > ...
29     },
30     "Former South Carolina Gov. Nikki Haley will win the Republican presidential primary in Washington, DC"
31   };
32
33   // Add services to the container.
34
35   builder.Services.AddEndpointsApiExplorer();
36
37   builder.Services.AddDbContext<AppDbContext>(x =>
38     x.UseSqlite(builder.Configuration.GetConnectionString("DefaultConnection"))
39   );
40
41   builder.Services.AddAutoMapper(typeof(Program));
42   builder.Services.AddScoped<IPostRepo, PostRepo>();
43
44   //builder.Services.AddValidatorsFromAssemblyContaining(typeof(PostValidator));
45   //builder.Services.AddScoped<IValidator<Post>, PostValidator>();
46   //builder.Services.AddScoped<IValidator<PostNewOrUpdatedDto>, PostNewOrUpdatedDtoValidator>();
47
48
49   builder.Services.AddValidatorsFromAssemblyContaining<Program>(ServiceLifetime.Singleton);
50
```

You, 1 second ago • L

Client-Server Separation - CORS

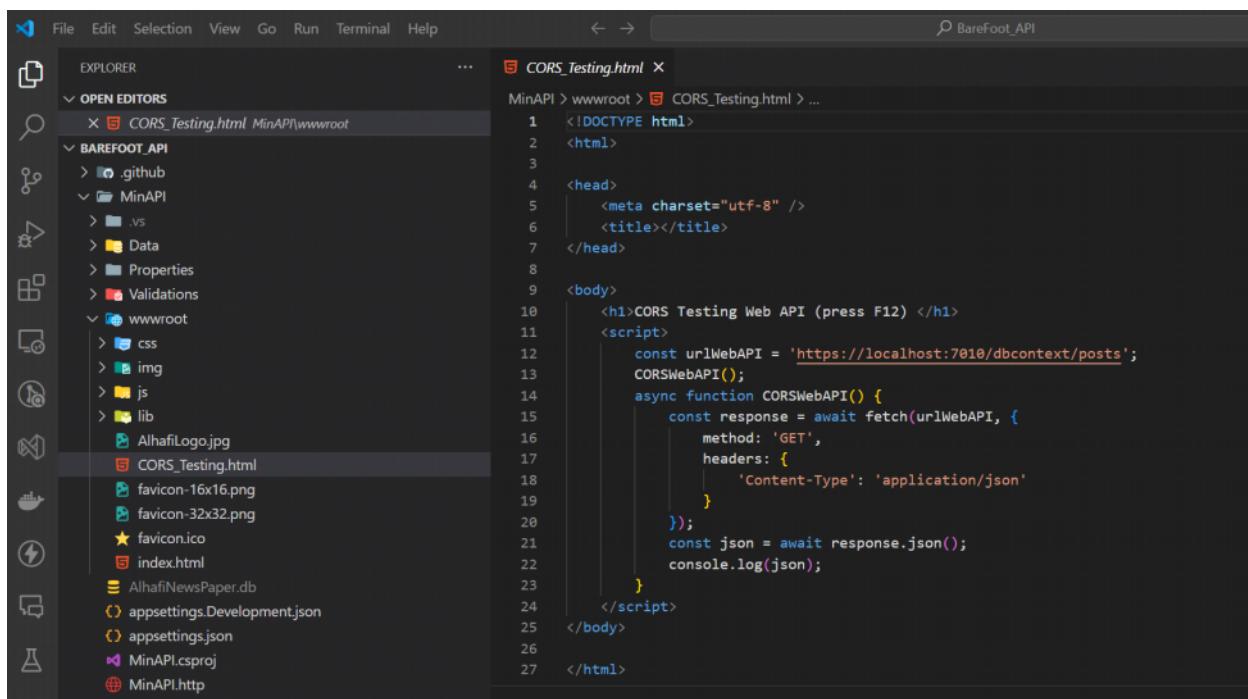
12 May 2024 10:29

Client-Server Separation

- One of the REST principles we saw is client-server separation.
- Where will our clients be found?
- CORS
- When we talk about origin we refer to the combination of different aspects of a URL, such as **the domain, port and protocol**.

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>
```

```
<!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'
                }
            });
        }
    </script>
</body>
</html>
```

```

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

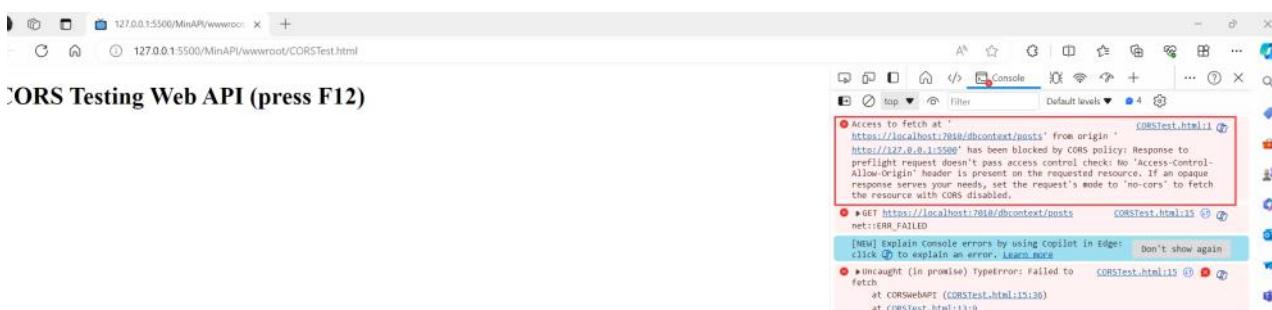
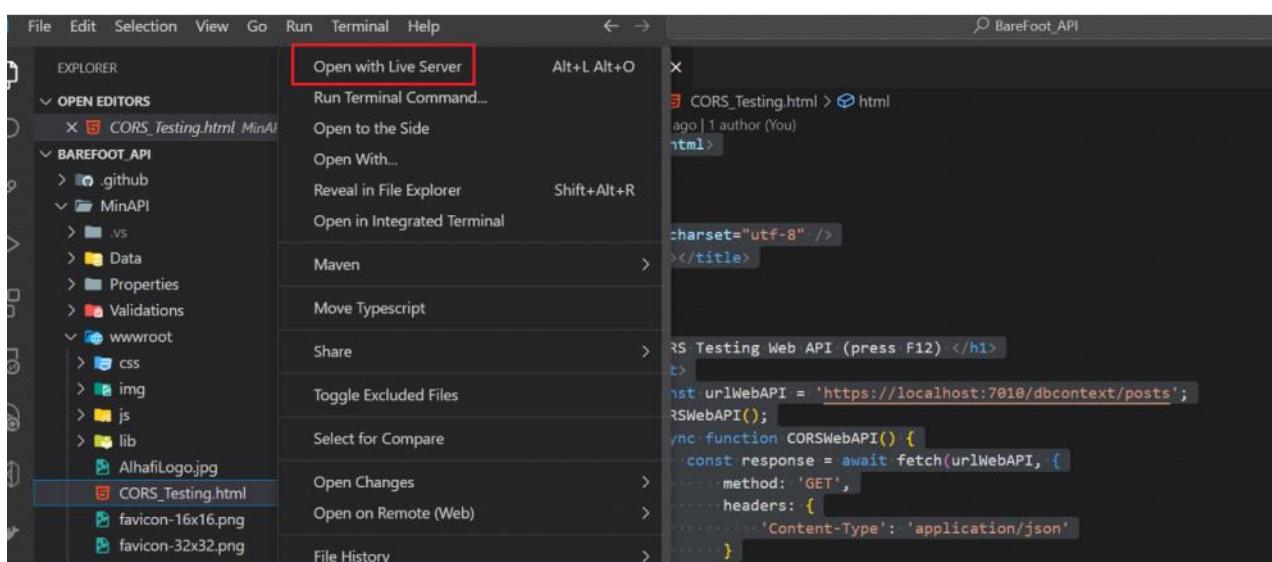
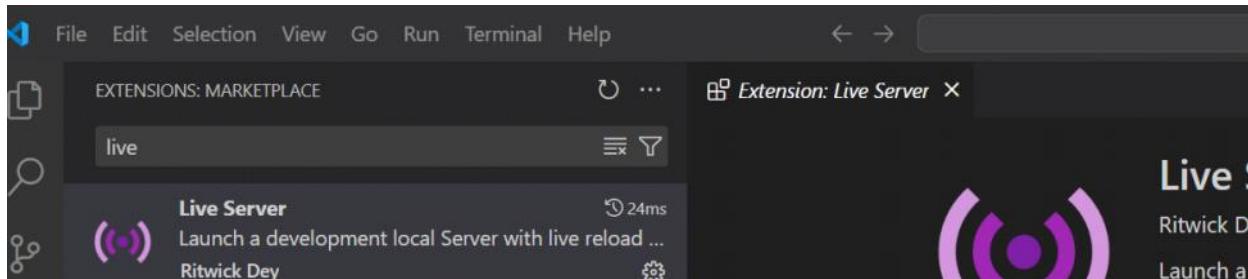
```

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();
    });
});
```

```
C# Program.cs 3. M X
MinAPI > C# 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

The screenshot shows a browser window with the title "CORS Testing Web API (press F12)". The address bar indicates the URL is 127.0.0.1:5500/MinAPI/wwwroot/CORS_Testing.html. To the right of the browser is a developer tools console window. The console output shows an array of 8 objects, each representing a post with properties: id, title, content, and postImage. All post titles and contents are set to 'string'.

```
CORS_Testing.html:22
▼ Array(8) 1
▶ 0: {id: 1, title: 'Post1', content: 'Content Post1', postImage: null}
▶ 1: {id: 2, title: 'Post2', content: 'Content Post2', postImage: null}
▶ 2: {id: 3, title: '', content: 'string', postImage: 'string'}
▶ 3: {id: 4, title: '', content: 'string', postImage: 'string'}
▶ 4: {id: 5, title: '', content: 'string', postImage: 'string'}
▶ 5: {id: 6, title: '', content: 'string', postImage: 'string'}
▶ 6: {id: 7, title: 'string', content: 'string', postImage: 'string'}
▶ 7: {id: 8, title: '', content: 'string', postImage: 'string'}
```

Cache

13 May 2024 07:23

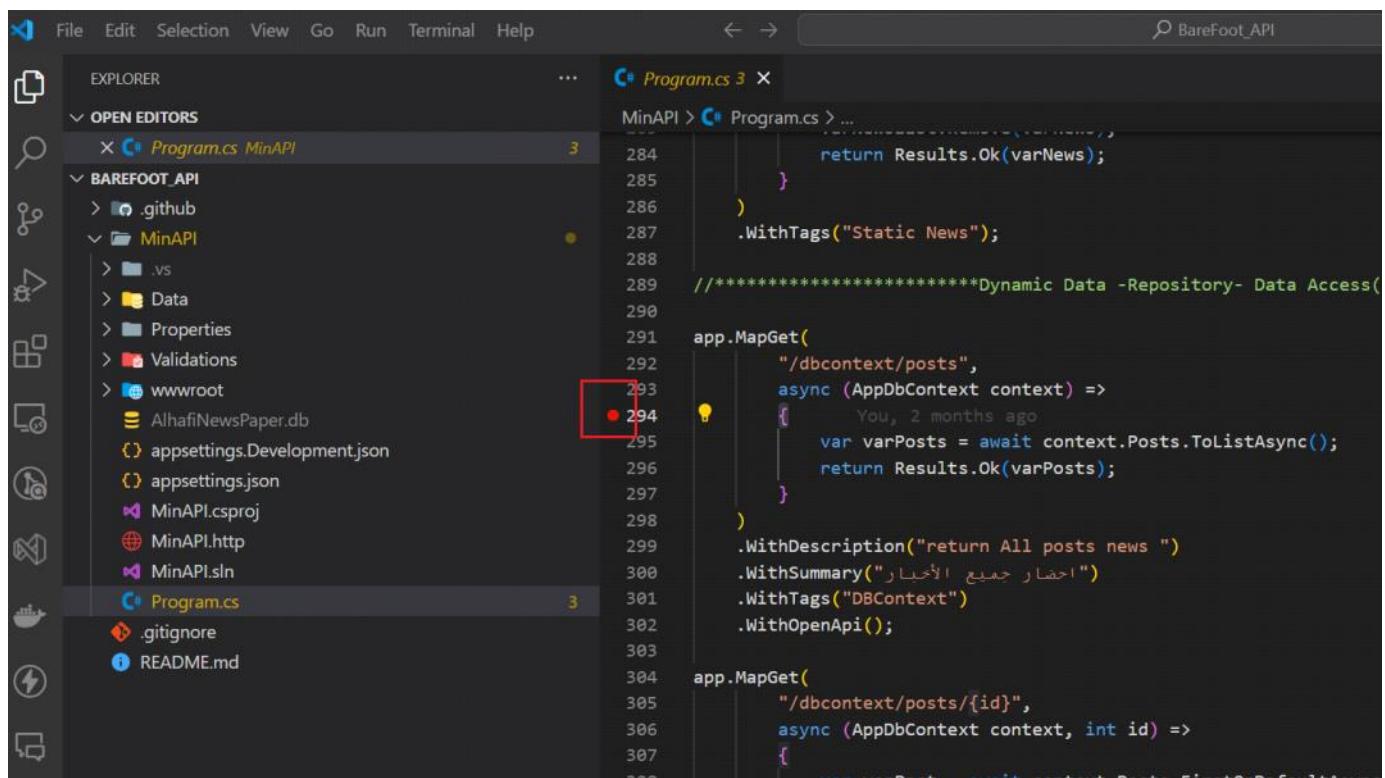
Cache

- Cache refers to mechanisms to serve information with less effort
- Consuming a database is a relatively expensive operation
- We can save this list in a quickly accessible place

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



The screenshot shows the Visual Studio IDE interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The left sidebar has icons for Explorer, Open Editors, GitHub, MinAPI, .vs, Data, Properties, Validations, wwwroot, AlhafiNewsPaper.db, appsettings.Development.json, appsettings.json, MinAPI.csproj, MinAPI.http, MinAPI.sln, Program.cs, .gitignore, and README.md. The right pane displays the code editor for 'Program.cs' under the 'MinAPI' project. The code is as follows:

```
284         return Results.Ok(varNews);
285     }
286 }
287 .WithTags("Static News");
288
289 //*****Dynamic Data -Repository- Data Access(*****
290
291 app.MapGet(
292     "/dbcontext/posts",
293     async (AppDbContext context) =>
294     {
295         You, 2 months ago
296         var varPosts = await context.Posts.ToListAsync();
297         return Results.Ok(varPosts);
298     }
299     .WithDescription("return All posts news ")
300     .WithSummary("أختبار جميع")
301     .WithTags("DBContext")
302     .WithOpenApi();
303
304 app.MapGet(
305     "/dbcontext/posts/{id}",
306     async (AppDbContext context, int id) =>
307     {
308         var varPost = await context.Posts.FirstOrDefaultAsync();
309         return Results.Ok(varPost);
310     }
311     .WithDescription("return post by id")
312     .WithSummary("أختبار جميع")
313     .WithTags("DBContext")
314     .WithOpenApi();
315 }
```

A red square box highlights the break point at line 294, which is annotated with a yellow lightbulb icon.

Press F5

```

284         return Results.Ok(varNews);
285     }
286   )
287   .WithTags("Static News");
288
289 //*****Dynamic Data -Repository- Data Access(DB Context) EndPoint**
290
291 app.MapGet(
292   "/dbcontext/posts",
293   async (AppDbContext context) =>
294   {
295     You, 2 months ago
296     var varPosts = await context.Posts.ToListAsync();
297     return Results.Ok(varPosts);
298   }
299   .WithDescription("return All posts news ")
300   .WithSummary("أحضر جميع الأخبار")
301   .WithTags("DBContext")
302   .WithOpenApi();
303
304 app.MapGet(
305   "/dbcontext/posts/{id}",
306   async (AppDbContext context, int id) =>
307   {
308     var varPost = await context.Posts.FirstOrDefaultAsync(c => c.Id == id);
309     if (varPost != null)
310     {

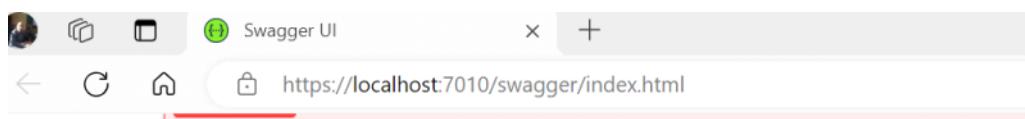
```

PROBLEMS OUTPUT DEBUG CONSOLE PORTS GITLENS NUGET COMMENTS TERMINAL

```

Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.4\System.Xml.XPath.dll'. Skipped because 'Just My Code' is enabled.
Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.4\System.Private.Xml.dll'. Skipped because 'Just My Code' is enabled.

```



DBContext

GET /dbcontext/posts احضار جميع الأخبار

DBContext

GET /dbcontext/posts احضار جميع الأخبار

return All posts news

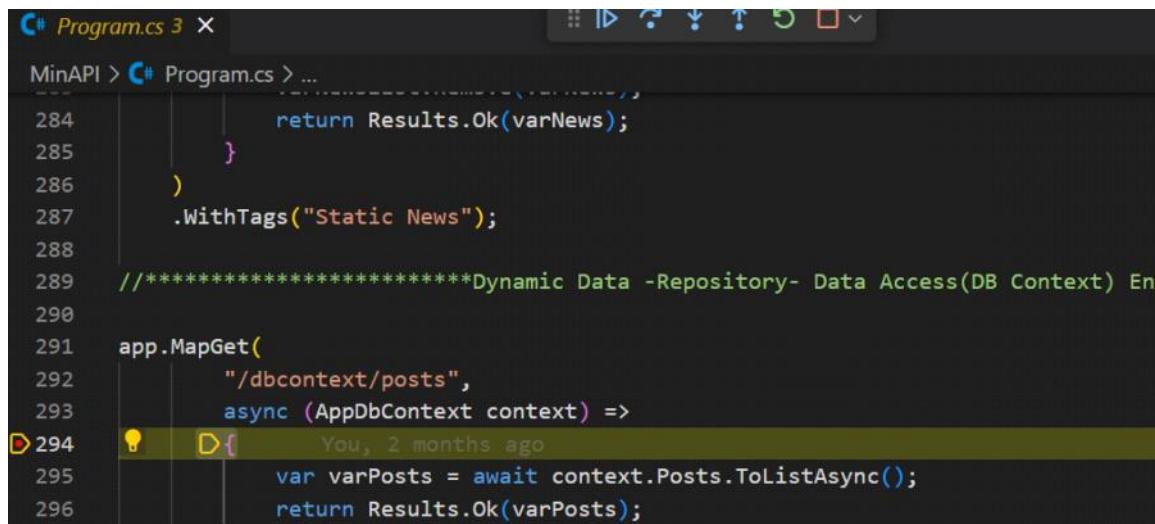
Parameters

No parameters

Execute

LOADING

It will stop @ break point till we press F5



```
284     return Results.Ok(varNews);
285   }
286 }
287 .WithTags("Static News");
288
289 //*****Dynamic Data -Repository- Data Access(DB Context) End*****
290
291 app.MapGet(
292   "/dbcontext/posts",
293   async (AppDbContext context) =>
294   {
295     You, 2 months ago
296     var varPosts = await context.Posts.ToListAsync();
297     return Results.Ok(varPosts);
298   }
299 );
```

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

```
C# Program.cs 3 ●  
MinAPI > C# Program.cs > ...  
47     x.UseSqlite(builder.Configuration.GetConnectionString("DefaultConnection"))  
48 );  
49  
50 builder.Services.AddAutoMapper(typeof(Program));  
51 builder.Services.AddScoped<IPostRepo, PostRepo>();  
52  
53 //builder.Services.AddValidatorsFromAssemblyContaining(typeof(PostValidator));  
54 //builder.Services.AddScoped<IValidator<Post>, PostValidator>();  
55 //builder.Services.AddScoped<IValidator<PostNewOrUpdatedDto>, PostNewOrUpdatedDtoValidator>();  
56  
57 You, 3 weeks ago  
58 builder.Services.AddSingleton<IDateTime, SystemDateTime>();  
59 builder.Services.AddValidatorsFromAssemblyContaining<Program>(ServiceLifetime.Singleton);  
60 builder.Services.AddOutputCache();  
61  
62 builder.Services.AddCors(options =>  
63 {  
64     options.AddPolicy(  
65         "MyAllowedOrigins",  
66         policy =>  
67         {  
68             policy  
69                 .WithOrigins(  
70                     "https://localhost/*",  
71                     "http://localhost/*",  
72                     "http://127.0.0.1/*",
```

builder.Services.AddOutputCache();

```
C# Program.cs 3 ●  
MinAPI > C# Program.cs > ...  
136 }  
137 if (app.Environment.IsStaging())  
138 {  
139     // your code here  
140 }  
141 if (app.Environment.IsProduction())  
142 {  
143     // your code here  
144 }  
145  
146 app.UseHttpsRedirection();  
147  
148 app.UseStaticFiles();  
149  
150 app.UseCors("MyAllowedOrigins"); You, 17 hours ago  
151  
152 app.UseOutputCache();  
153  
154 app.UseOutputCache();  
155  
156 //***** Ending Middle Points ***  
157  
158 //***** End Points Zone *****  
159
```

app.UseOutputCache();

A screenshot of a .NET IDE showing code completion for the `.CacheOutput` method. The code is as follows:

```
    )
    .WithDescription("return All posts news ")
    .WithSummary("إحضار جميع الأخبار")
    .WithTags("DbContext")
    .CacheOutput(c=>c.Expire(TimeSpan.fr()))
    .WithOpenApi();
```

The cursor is on the `.fr` part of `TimeSpan.From`, and a dropdown menu shows the following options:

- FromDays
- FromHours
- FromMicroseconds
- FromMilliseconds
- FromMinutes
- FromSeconds
- FromTicks

A screenshot of a .NET IDE showing code completion for the `.CacheOutput` method. The code is as follows:

```
    )
    .WithTags("Static News");

//*****Dynamic Data -Repository- Data Access(DB Context) EndPoint*****
app.MapGet(
    "/dbcontext/posts",
    async (AppDbContext context) =>
{
    var varPosts = await context.Posts.ToListAsync();
    return Results.Ok(varPosts);
}

    )
    .WithDescription("return All posts news ")
    .WithSummary("إحضار جميع الأخبار")
    .WithTags("DbContext")
    .WithOpenApi()
    .CacheOutput(c=>c.Expire(TimeSpan.FromSeconds(15)));
    You, 2 months ago
```

The cursor is on the `FromSeconds(15)` part, and a dropdown menu shows the following options:

- FromDays
- FromHours
- FromMicroseconds
- FromMilliseconds
- FromMinutes
- FromSeconds
- FromTicks

```
.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

Body	Cookies	Headers (5)	Test Results
Key			Value
Content-Length	75	①	
Content-Type	application/json; c	①	
Date	Mon, 26 Feb 2024	①	
Server	Kestrel	①	
Age	N/A	①	48

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("احفظ جميع الاخبار")
.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("احفظ جميع الاخبار")
.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 ")
.احصاء جميع الاخبار
.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);
    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 outputCacheRestore =>
{
    var validationResult = await validator.ValidateAsync(poss);

    if (validationResult.IsValid)
    {
        await context.Posts.AddAsync(poss);
        await context.SaveChangesAsync();
        await outputCacheRestore.EvictByTagAsync("Post_Get", default);
        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's 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 X  
MinAPI > C# 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(  
    "/QueryString",  
    [FromQuery] string name) =>  
167    {  
        return $"Hello {name}";  
    }  
168    You, 2 weeks ago  
169  
170    }  
171    .WithTags("Hello");  
172  
173
```

Even we can handle extended such as tag

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

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

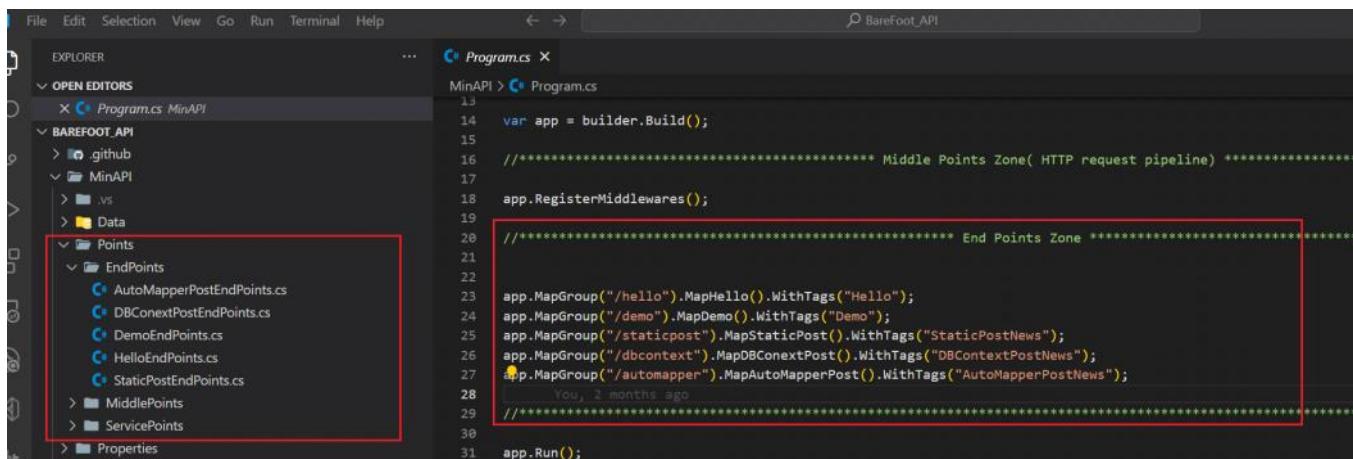
//*****Static Sample Hello*****


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

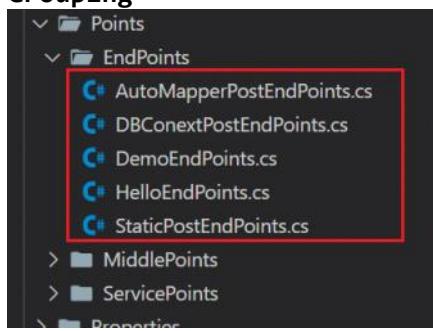
13 May 2024 13:12



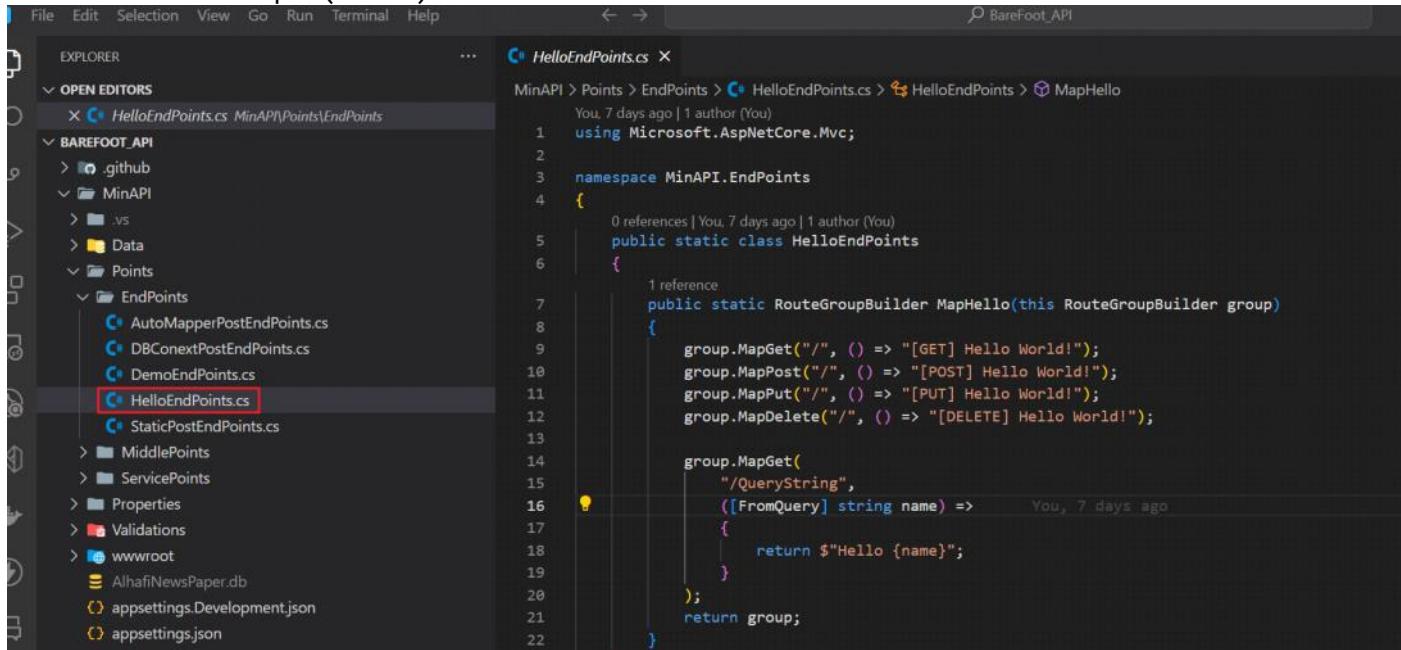
The screenshot shows the Visual Studio interface. On the left is the Explorer pane, which lists the project structure. A red box highlights the 'EndPoints' folder under 'Points'. Inside 'EndPoints', there are five files: AutoMapperPostEndPoints.cs, DBConextPostEndPoints.cs, DemoEndPoints.cs, HelloEndPoints.cs, and StaticPostEndPoints.cs. On the right is the code editor showing the 'Program.cs' file. A red box highlights the section of code where endpoints are registered. The code uses the RouteGroupBuilder to map various HTTP methods to specific actions.

```
13
14     var app = builder.Build();
15
16     //***** Middle Points Zone( HTTP request pipeline) *****
17
18     app.RegisterMiddlewares();
19
20     //***** End Points Zone *****
21
22
23     app.MapGroup("/hello").MapHello().WithTags("Hello");
24     app.MapGroup("/demo").MapDemo().WithTags("Demo");
25     app.MapGroup("/staticpost").MapStaticPost().WithTags("StaticPostNews");
26     app.MapGroup("/dbcontext").MapDBConextPost().WithTags("DbContextPostNews");
27     app.MapGroup("/automapper").MapAutoMapperPost().WithTags("AutoMapperPostNews");
28
29     You, 2 months ago
30
31     app.Run();
```

Grouping



We Will do as sample(Hello)



The screenshot shows the Visual Studio interface. The Explorer pane on the left has 'HelloEndPoints.cs' selected in the 'Points\EndPoints' folder. The code editor on the right shows the 'HelloEndPoints.cs' file. The 'MapHello' method is highlighted, showing route mappings for GET, POST, PUT, and DELETE requests to the root path. A yellow lightbulb icon is shown over the 'MapGet' call at line 16, indicating a potential issue or suggestion.

```
1 using Microsoft.AspNetCore.Mvc;
2
3 namespace MinAPI.EndPoints
4 {
5     public static class HelloEndPoints
6     {
7         public static RouteGroupBuilder MapHello(this RouteGroupBuilder group)
8         {
9             group.MapGet("/", () => "[GET] Hello World!");
10            group.MapPost("/", () => "[POST] Hello World!");
11            group.MapPut("/", () => "[PUT] Hello World!");
12            group.MapDelete("/", () => "[DELETE] Hello World!");
13
14            group.MapGet(
15                "/QueryString",
16                [FromQuery] string name) => You, 7 days ago
17                {
18                    return $"Hello {name}";
19                }
20            );
21        }
22    }
```

```
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;
}
}
}

```

```

File Edit Selection View Go Run Terminal Help BareFoot_API
EXPLORER OPEN EDITORS
  X Program.cs MinAPI
  BAREFOOT_API
    .github
    MinAPI
      .vs
      Data
      Points
        EndPoints
          AutoMapperPostEndPoints.cs
          DBConextPostEndPoints.cs
          DemoEndPoints.cs
          HelloEndPoints.cs
          StaticPostEndPoints.cs
        MiddlePoints
        ServicePoints
      Properties
      Validations
      wwwroot
        AlhafiNewsPaper.db
        appsettings.Development.json
        appsettings.json
        MinAPI.csproj
        MinAPI.http
        MinAPI.sln
  Program.cs
  .gitignore
  README.md
  ***** In Program.cs *****
//***** 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");
*/
*****
***** In Program.cs *****

```

We Will do as sample(Demo)

```

File Edit Selection View Go Run Terminal Help BareFoot_API
EXPLORER OPEN EDITORS
  X Program.cs MinAPI
  BAREFOOT_API
    .github
    MinAPI
      .vs
      Data
      Points
        EndPoints
          AutoMapperPostEndPoints.cs
          DBConextPostEndPoints.cs
          DemoEndPoints.cs
          HelloEndPoints.cs
          StaticPostEndPoints.cs
        MiddlePoints
        ServicePoints
      Properties
      Validations
      wwwroot
        AlhafiNewsPaper.db
        appsettings.Development.json
        appsettings.json
        MinAPI.csproj
        MinAPI.http
        MinAPI.sln
  Program.cs
  .gitignore
  README.md
  ***** In Program.cs *****
//***** In Program.cs *****

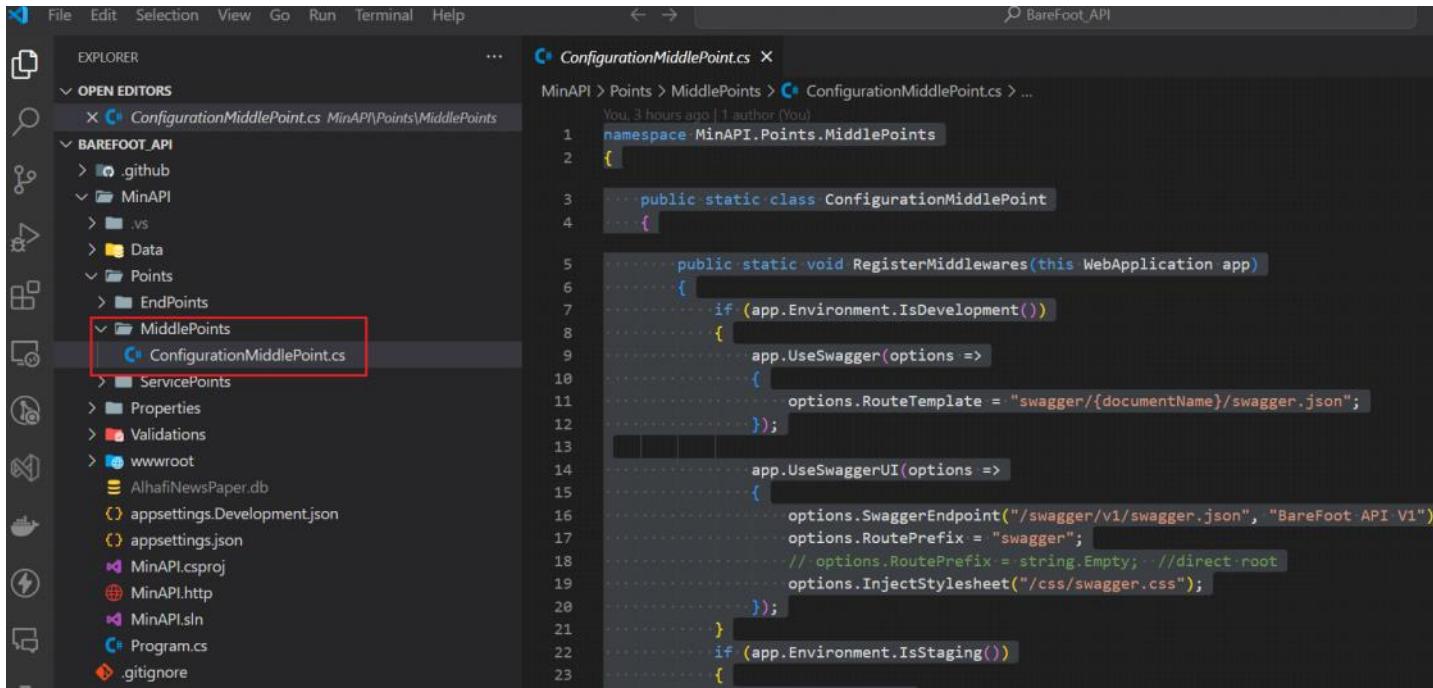
app.MapGroup("/hello").MapHello().WithTags("Hello");
app.MapGroup("/demo").MapDemo().WithTags("Demo");
app.MapGroup("/staticpost").MapStaticPost().WithTags("StaticPostNews");
app.MapGroup("/dbcontext").MapDBConextPost().WithTags("DBContextPostNews");
app.MapGroup("/automapper").MapAutoMapperPost().WithTags("AutoMapperPostNews");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

```

```
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



The screenshot shows the Visual Studio interface with the 'BAREFOOT_API' project selected in the left sidebar. The 'ConfigurationMiddlePoint.cs' file is open in the main editor window. The code implements middleware registration logic based on the application environment.

```
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();
        }
    }
}
```

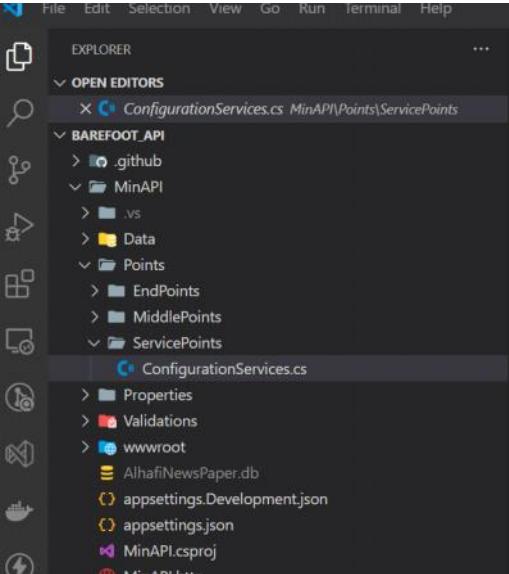
```
namespace MinAPI.Points.MiddlePoints
{
    public static class ConfigurationMiddlePoint
    {
        public static void RegisterMiddlewares(this WebApplication app)
        {
            if (app.Environment.IsDevelopment())
            {
                app.UseSwagger(options =>
                {
                    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();
        }
    }
}
```

The screenshot shows a code editor window with the file `Program.cs` open. The code is written in C# and defines a `Program` class that inherits from `WebApplication`. The code is organized into three main sections: `Services Zone`, `Middle Points Zone (HTTP request pipeline)`, and `End Points Zone`.

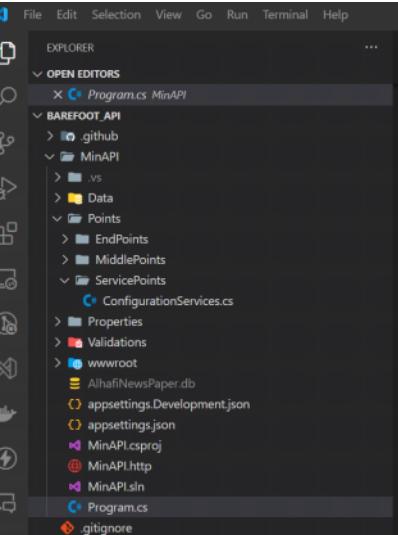
```
MinAPI > Program.cs
1  var builder = WebApplication.CreateBuilder(args);
2
3  //***** Services Zone *****
4
5  builder.RegisterServices();
6
7  //***** Middle Points Zone( HTTP request pipeline) *****
8
8  var app = builder.Build();
9
10 //***** End Points Zone *****
11
12 app.RegisterMiddlewares();
13
14 //*****
15
16 app.MapGroup("/hello").MapHello().WithTags("Hello");
17 app.MapGroup("/demo").MapDemo().WithTags("Demo");
18 app.MapGroup("/staticpost").MapStaticPost().WithTags("StaticPostNews");
19 app.MapGroup("/dbcontext").MapDBConextPost().WithTags("DbContextPostNews");
20 app.MapGroup("/automapper").MapAutoMapperPost().WithTags("AutoMapperPostNews");
21
22 You, 2 months ago
23 //*****
24
25
26
27
28
29 //*****
29
```

Grouping Service Points

20 May 2024 14:43



```
MinAPI > Points > ServicePoints > ConfigurationServices.cs > ConfigurationServices > RegisterServices
You, 3 hours ago | 1 author (You)
1  using FluentValidation;
2  using Microsoft.EntityFrameworkCore;
3  using Microsoft.OpenApi.Models;
4  using MinAPI.Data;
5  using MinAPI.Data.Interfaces;
6  using MinAPI.Data.Models;
7
8  namespace MinAPI.Points.ServicePoints
9  {
10    public static class ConfigurationServices
11    {
12      public static void RegisterServices(this WebApplicationBuilder builder)
13      {
14        builder.Services.AddEndpointsApiExplorer();
15
16        builder.Services.AddDbContext<AppDbContext>(x =>
17          x.UseSqlite(builder.Configuration.GetConnectionString("DefaultConnection"))
18        );
19      }
20    }
21
22  }
```



```
MinAPI > Program.cs
5  //*****
6  var builder = WebApplication.CreateBuilder(args);
7
8  //***** Services Zone *****
9  builder.RegisterServices();
10
11  //***** Middle Points Zone( HTTP request pipeline) *****
12
13  var app = builder.Build();
14
15  //***** End Points Zone *****
16
17  app.RegisterMiddlewares();
18
19
20  app.MapGroup("/hello").MapHello().WithTags("Hello");
21  app.MapGroup("/demo").MapDemo().WithTags("Demo");
22  app.MapGroup("/staticpost").MapStaticPost().WithTags("StaticPostNews");
23  app.MapGroup("/dbcontext").MapDbContextPost().WithTags("DbContextPostNews");
24  app.MapGroup("/automapper").MapAutoMapperPost().WithTags("AutoMapperPostNews");
25
26
27
28
29  //*****
30
```

```
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(ApplicationContext.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>