

Development Tools

Sunday, January 7, 2024 11:48 AM

Download Git

<https://git-scm.com/downloads>

The screenshot shows the official Git website. On the left, there's a sidebar with links for About, Documentation, Downloads (which is highlighted in red), GUI Clients, Logos, and Community. The main content area has a large "Downloads" heading. Below it are three buttons: "macOS" with an Apple icon, "Windows" with a Windows icon, and "Linux/Unix" with a terminal icon. To the right, there's a prominent callout box for the latest source release, version 2.44.0, with a "Download for Windows" button.

Download NodeJS

<https://nodejs.org/en/download>

Downloads

Latest LTS Version: 20.11.1 (includes npm 10.2.4)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

The screenshot shows the Node.js download page. It features two main sections: "LTS" (Recommended For Most Users) and "Current". Under "LTS", there's a "Windows Installer" link with the file "node-v20.11.1-x64.msi". Under "Current", there's a "macOS Installer" link with the file "node-v20.11.1.pkg" and a "Source Code" link with the file "node-v20.11.1.tar.gz".

Download .Net 8

<https://dotnet.microsoft.com/en-us/download/dotnet/8.0>

Download .NET 8.0

Not sure what to download? See recommended downloads for the latest version of .NET.

8.0.0 Security patch

[Release notes](#) Latest release date November 14, 2023

Build apps - SDK

SDK 8.0.100

OS	Installers	Binaries
Linux	Package manager instructions	Arm32 Arm32 Alpine Arm64 Arm64 Alpine x64 x64 Alpine
macOS	Arm64 x64	Arm64 x64
Windows	Arm64 x64 x86 winget instructions	Arm64 x64 x86
All	dotnet-install scripts	

Run apps - Runtime

ASP.NET Core Runtime 8.0.0

The ASP.NET Core Runtime enables you to run existing web/server applications. On Windows, we recommend installing the Hosting Bundle, which includes the .NET Runtime and IIS support.

IIS runtime support (ASP.NET Core Module v2)
18.0.23305.0

OS	Installers	Binaries
Linux	Package manager instructions	Arm32 Arm32 Alpine Arm64

Download postman

<https://www.postman.com/downloads/>



The screenshot shows the Postman website's download page. At the top, there's a navigation bar with links like 'Product', 'Use Cases', 'Pricing', 'Enterprise', 'Explore', 'Learning Center', 'Sign In', and 'Sign Up'. Below the navigation, a large heading says 'Download Postman'. A subtext below it reads: 'Download the app to quickly get started using the Postman API Platform. Or, if you prefer a browser experience, you can try the new web version of Postman.' The page has a dark background with light-colored text and links.

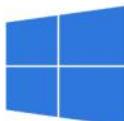
Download Visual Code

<https://code.visualstudio.com/download>

[Version 1.85](#) is now available! Read about the new features and fixes from November.

Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.



↓ Windows

Windows 10, 11



↓ .deb

Debian, Ubuntu

↓ .rpm

Red Hat, Fedora, SUSE



↓ Mac

macOS 10.15+

User Installer	x64	Arm64
System Installer	x64	Arm64
.zip	x64	Arm64
CLI	x64	Arm64

.deb	x64	Arm32	Arm64
.rpm	x64	Arm32	Arm64
.tar.gz	x64	Arm32	Arm64
Snap	Snap Store		
CLI	x64	Arm32	Arm64

.zip	Intel chip	Apple silicon	Universal
CLI	Intel chip	Apple silicon	

Checked All Installed

git --version

dotnet --version

node --version

```
C:\Users\aalhafi>git --version
git version 2.44.0.windows.1

C:\Users\aalhafi>dotnet --version
8.0.200

C:\Users\aalhafi>node --version
v20.11.1

C:\Users\aalhafi>
```

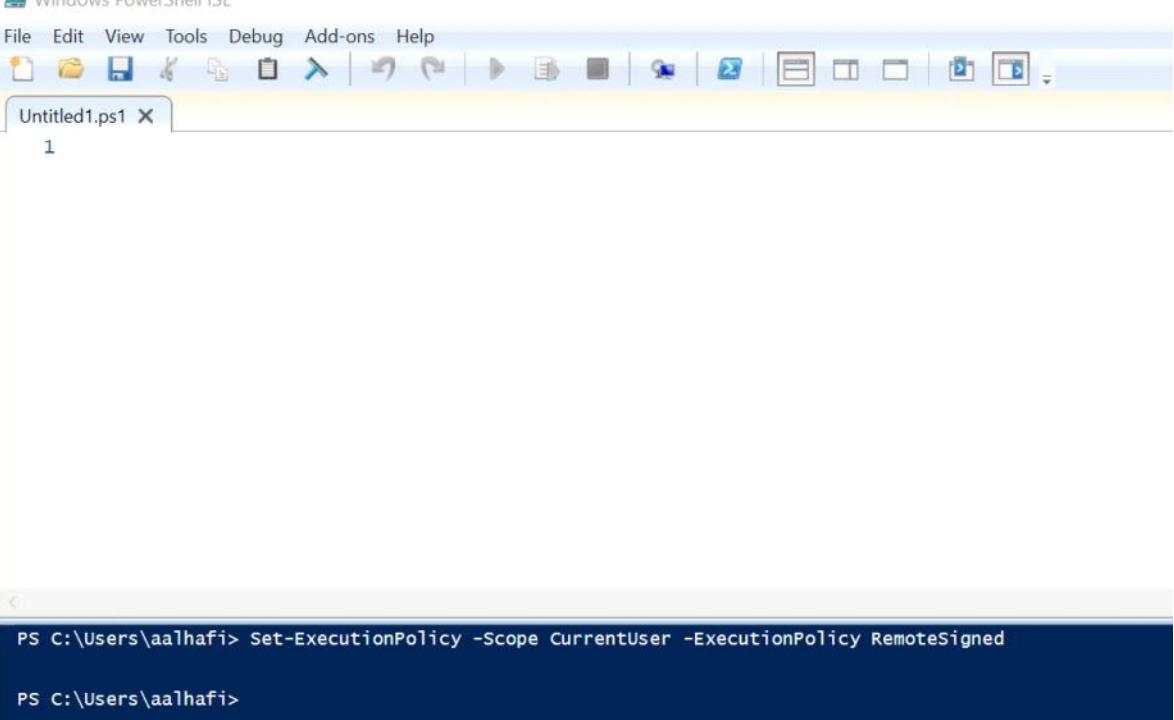
NPM

NPM : Node Package Manager tools to install 3 parties libraries

One of these libraries we attend to install , Angular Command Line Interface (**CLI**)

[powershell](#)

Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned



```
PS C:\Users\aalhafi> Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
```

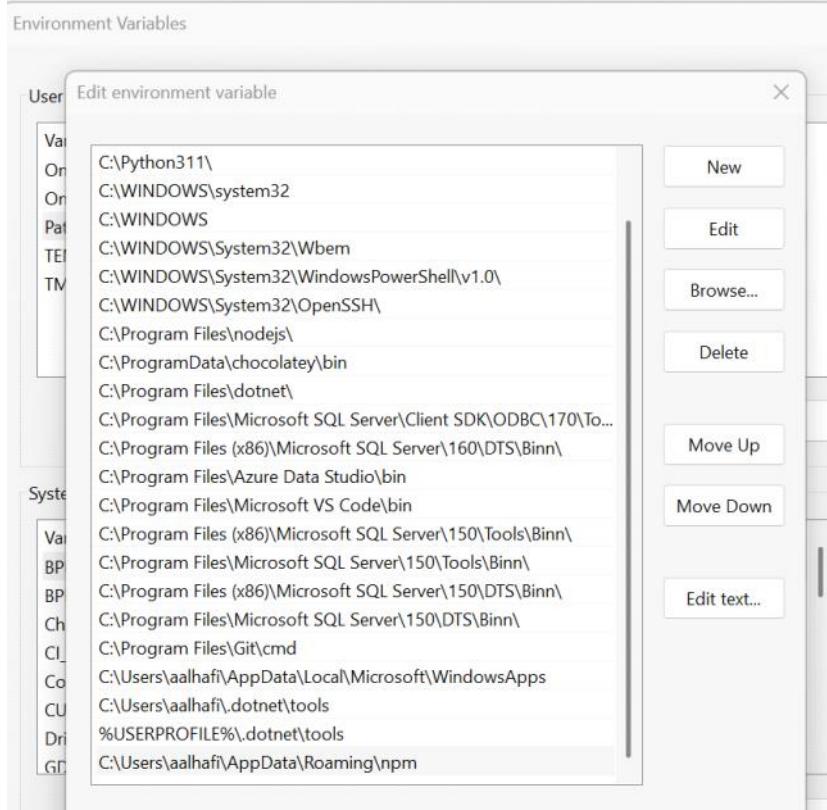
npm install -g npm@latest

npm config get prefix

```
C:\Windows\System32>npm config get prefix
C:\Users\aalhafi\AppData\Roaming\npm
```

npm --version

In case command not working check environment



Install Type script

Cmd

**npm install typescript -g
tsc --version**

```
C:\Users\aalhafi>tsc --version
Version 5.3.3

C:\Users\aalhafi>
```

Install angular

Cmd

npm install -g @angular/cli@latest

ng version

C:\Windows\system32\cmd.exe

```
Global setting: enabled
Local setting: No local workspace configuration file.
Effective status: enabled
```



```
Angular CLI: 17.2.2
Node: 20.11.1
Package Manager: npm 10.5.0
OS: win32 x64
```

```
Angular:
```

```
...
```

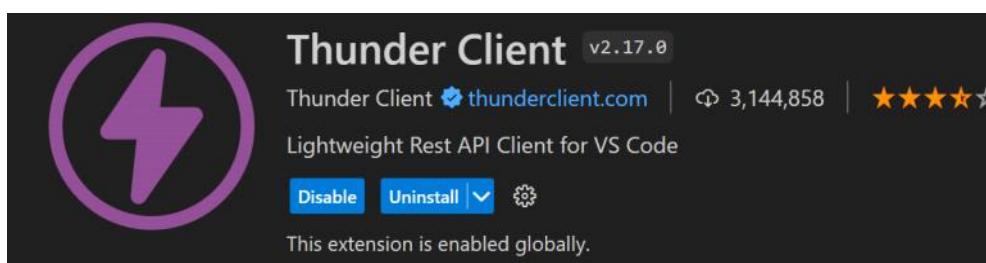
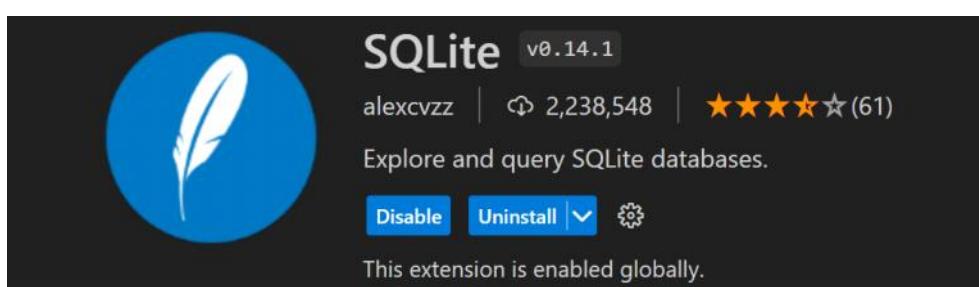
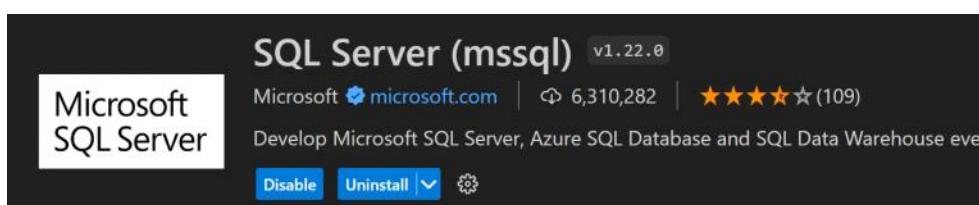
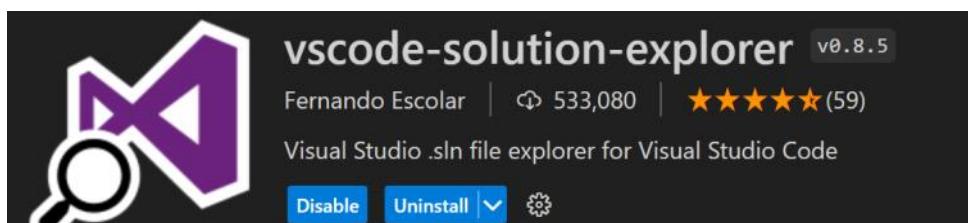
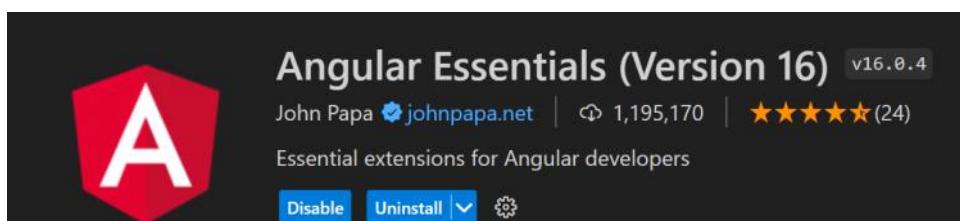
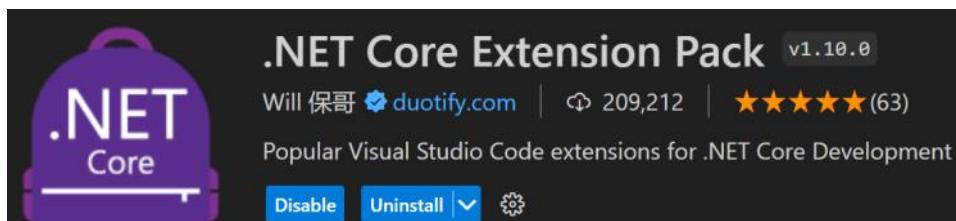
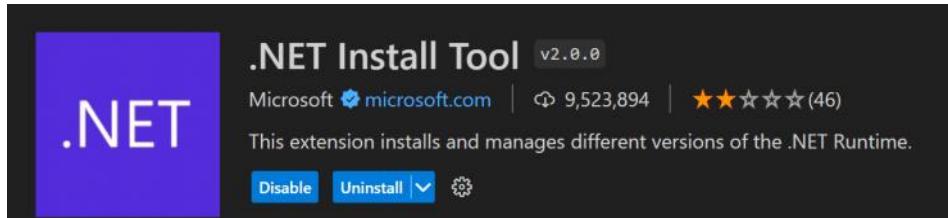
Package	Version
<hr/>	
@angular-devkit/architect	0.1702.2 (cli-only)
@angular-devkit/core	17.2.2 (cli-only)
@angular-devkit/schematics	17.2.2 (cli-only)
@schematics/angular	17.2.2 (cli-only)

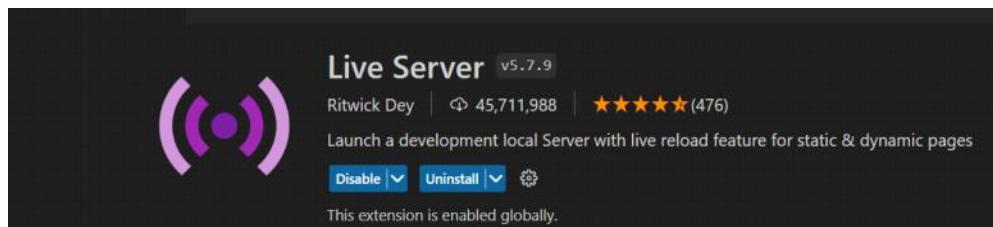
```
C:\Users\aalhafi>
```

Visual Code Extensions

07 January 2024

19:05





Create Project

Sunday, January 7, 2024 11:50 AM

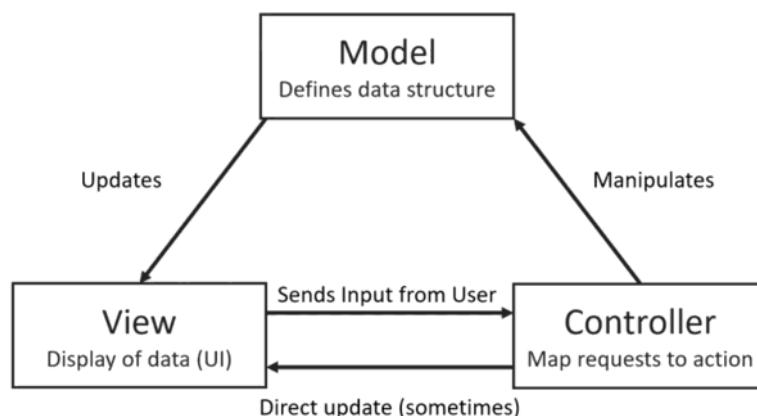
Differences Minimal API & MVC API

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

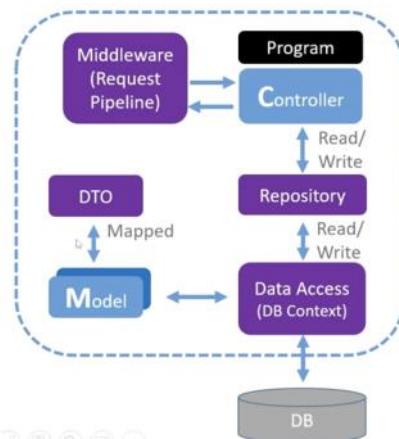
Model – View - Controller

- 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

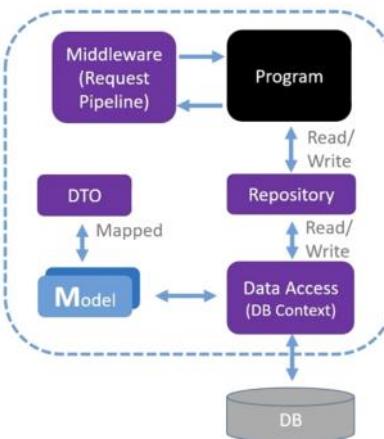
Model – View - Controller



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](#)

Screenshot of GitHub repository creation process:

Step 1: Repository Overview

The user is on their GitHub profile page, showing repositories like `HtmxBlog` and `Vidly`. A new repository button (`New`) is highlighted.

Step 2: Repository Creation Form

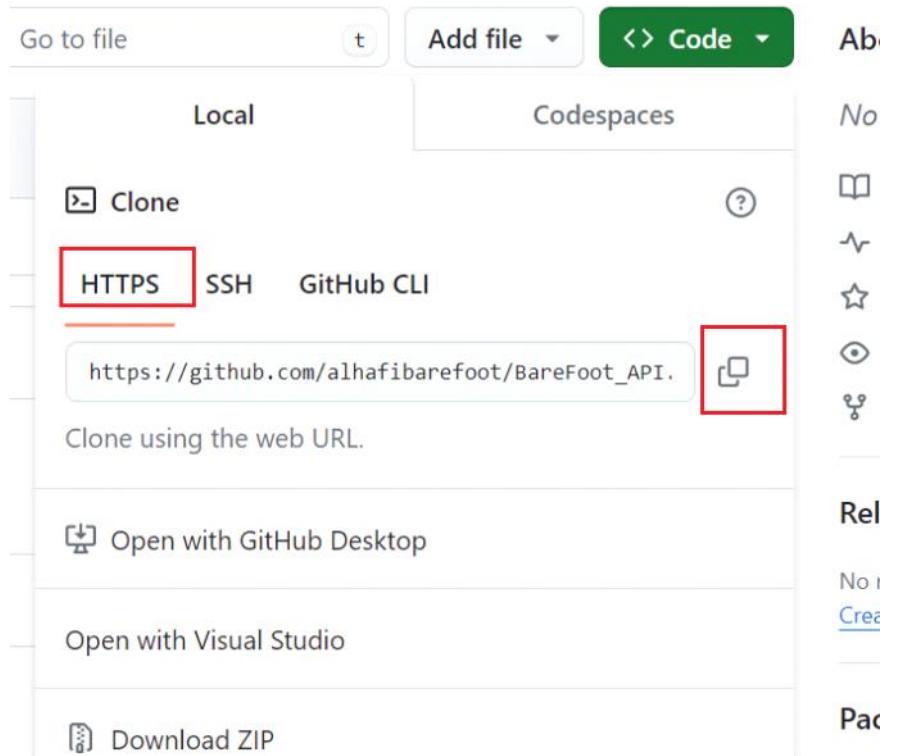
Form fields include:

- Owner ***: alhafibarefoot
- Repository name ***: BareFoot_API
- Description (optional)**: (empty)
- Visibility**:
 - Public** (selected): 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** (checked): 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](#).
- Default Branch**: main
- Notes**: You are creating a public repository in your personal account.

Step 3: Repository Details

The newly created repository `BareFoot_API` is shown in its details view:

- Owner**: alhafibarefoot
- Branches**: main (1 branch)
- Tags**: 0 Tags
- Commits**: 1 Commit (1fc1e56 - now)
- Files**:
 - README.md (Initial commit, now)
- About**:
 - No description, website, or topics provided
 - [Readme](#)
 - [Activity](#)
 - [0 stars](#)
 - [1 watching](#)
 - [0 forks](#)
- Releases**: No releases published



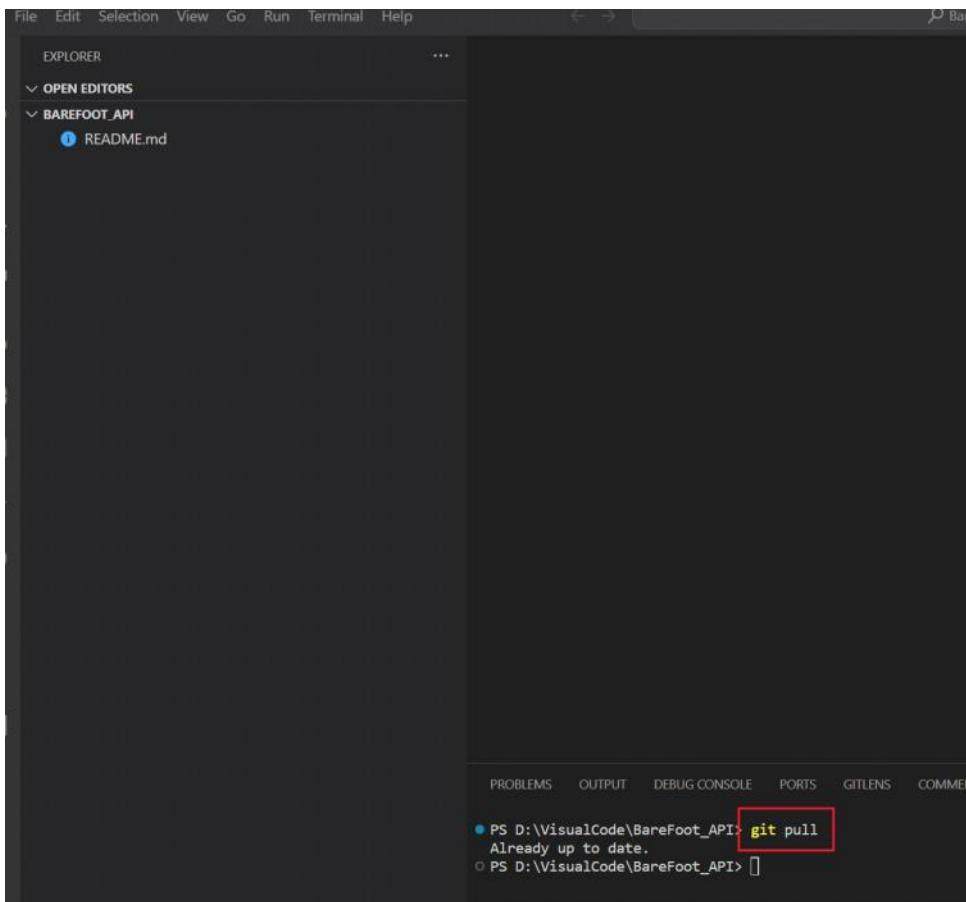
```
git clone https://github.com/alhafibarefoot/BareFoot\_API.git
```

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

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

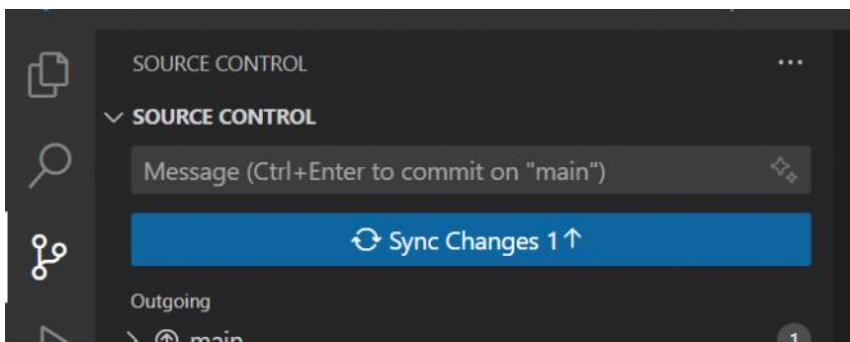
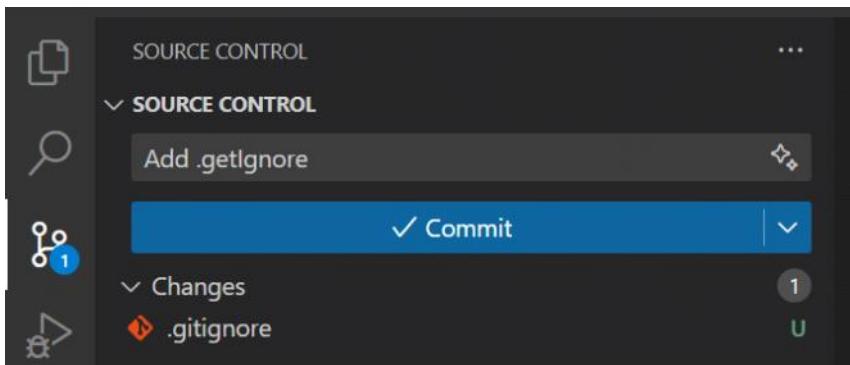
Using Visual Code

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



Before Adding Project Let we add ignore

```
.gitignore
# This .gitignore file was automatically created by Microsoft(R) Visual Studio.
*.cashe
*.userprefs
*.user
*.userosscache
*.sln.docstates
_ReSharper.*/ 
*.sdf
# User-specific files (MonoDevelop/Xamarin Studio)
*.userprefs
# Build/debug/releaze results
[Dd]ebug/
[Dd]ebugPublic/
[Rr]elease/
[Rr]eleases/
x64/
x86/
bld/
[Bb]in/
[Oo]bj/
[Ll]og/
[Ll]ogs/
[Uu]pload/
```



Checking Update git hub

The screenshot shows a GitHub repository page for "BareFoot_API" owned by "alhafibarefoot". The repository is public. The main branch is "main". There is 1 branch and 0 tags. The commit history shows two commits:

- alhafibarefoot Add .gitignore (e736c8e - now) - A file named ".gitignore" was added.
- Initial commit (11 minutes ago) - A file named "README.md" was added.

The screenshot shows the Terminal tab in Visual Studio Code. The command "dotnet --help" is being run in the terminal. The output is as follows:

```
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 0.
```

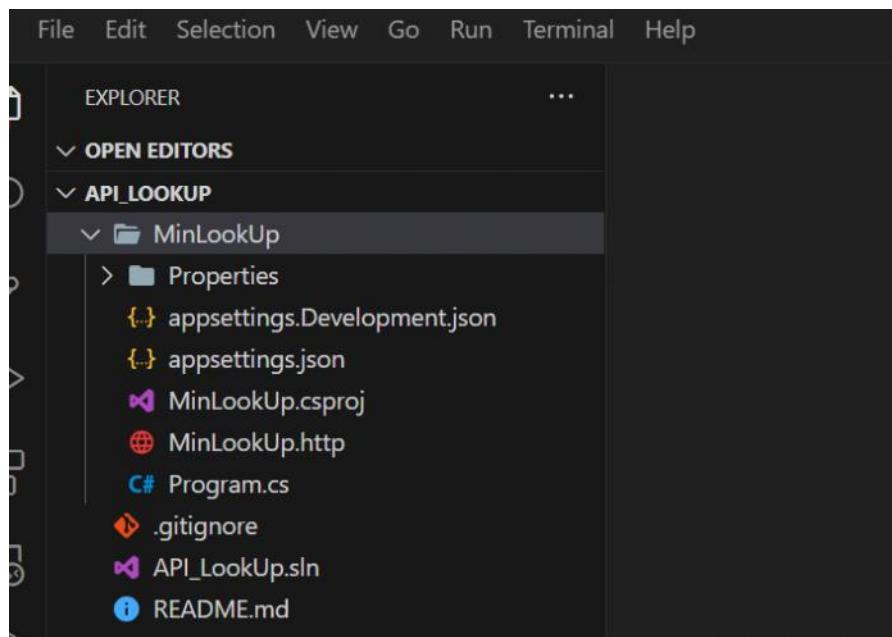
```
● PS D:\VC\API_LookUp> dotnet --list-sdks
8.0.100 [C:\Program Files\dotnet\sdk]
○ PS D:\VC\API_LookUp> []
```

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

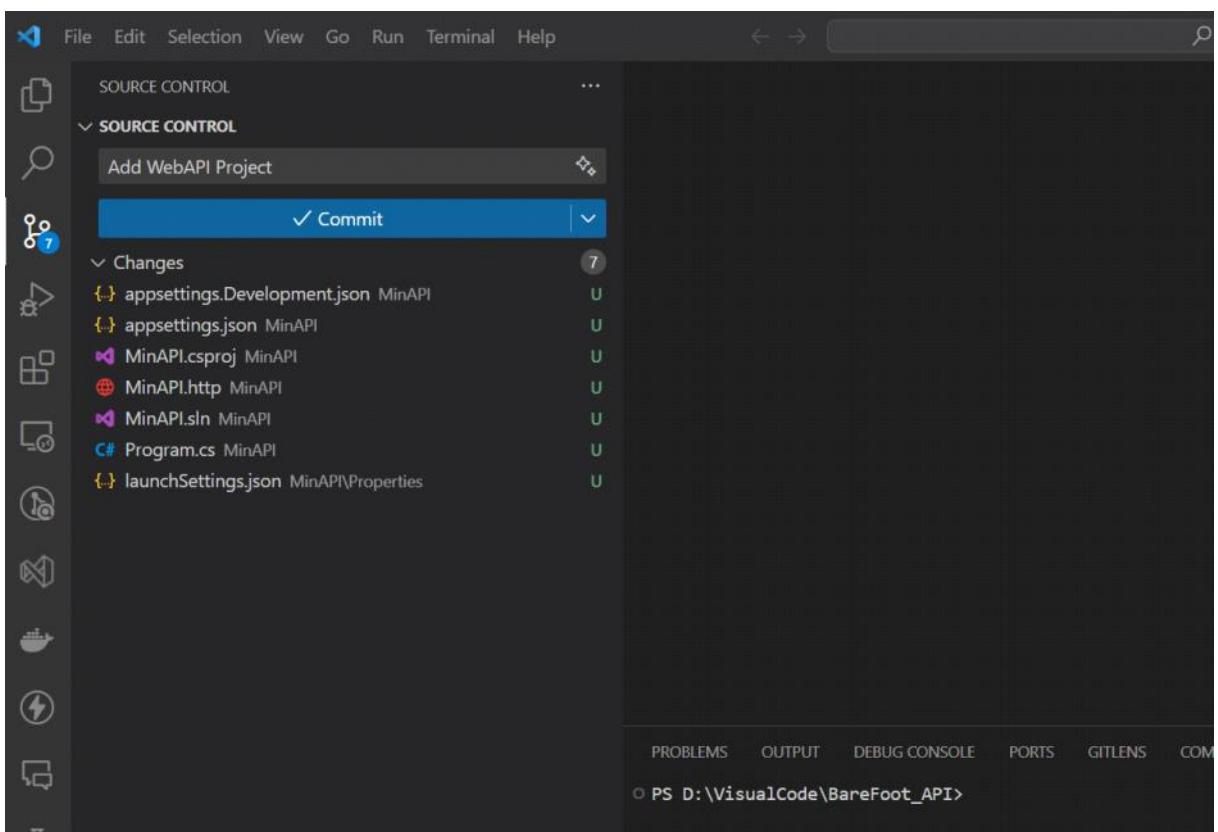
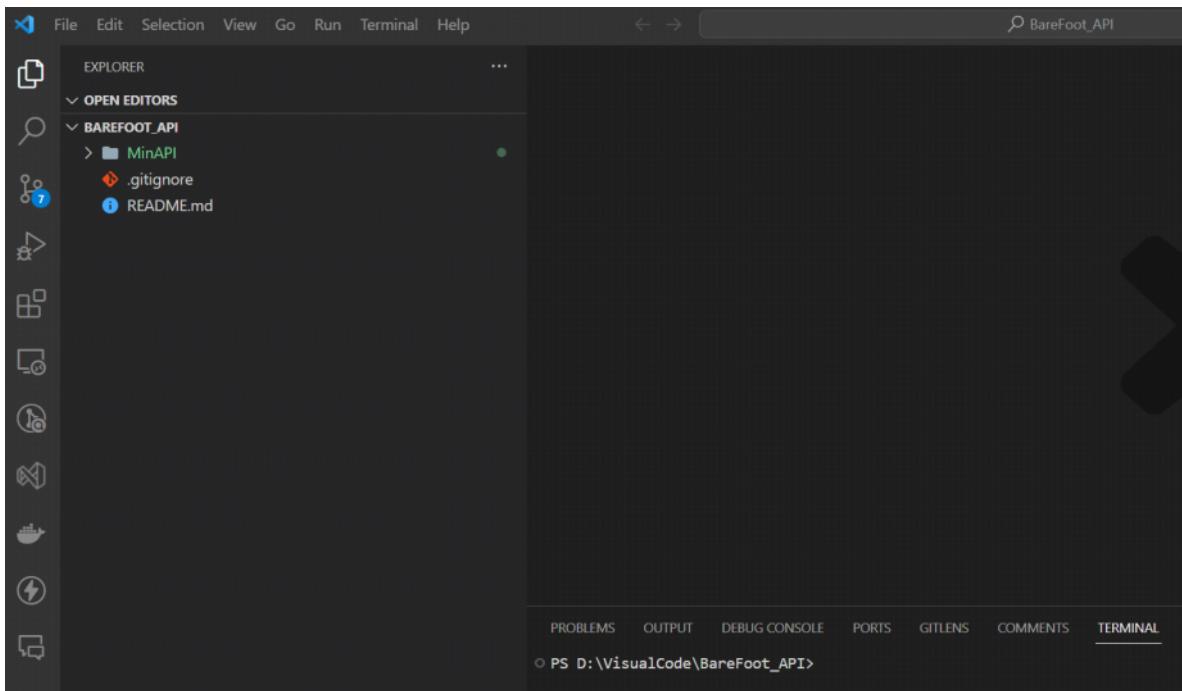
Template Name	Short Name	Language	Tags
.NET MAUI App	maui	[C#]	MAUI/Android/iOS/macOS/M
.NET MAUI Blazor	maui-blazor	[C#]	MAUI/Android/iOS/macOS/M
.NET MAUI Class	maulib	[C#]	MAUI/Android/iOS/macOS/M
.NET MAUI Content	maui-page-csharp	[C#]	MAUI/Android/iOS/macOS/M
.NET MAUI Content	maui-page-xaml	[C#]	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

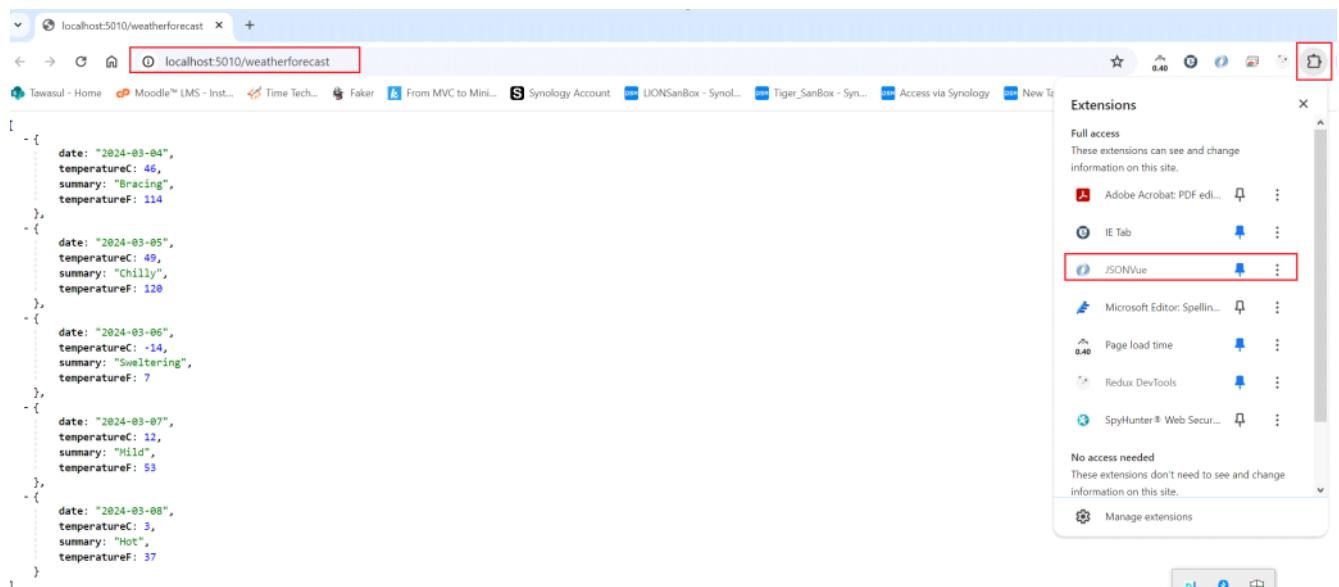
The screenshot shows the MinAPI documentation for a GET request to the endpoint `/weatherforecast`. The interface includes sections for 'Parameters' (which states 'No parameters'), an 'Execute' button, and 'Responses'. Under 'Responses', there is a 'Curl' section with the command:

```
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

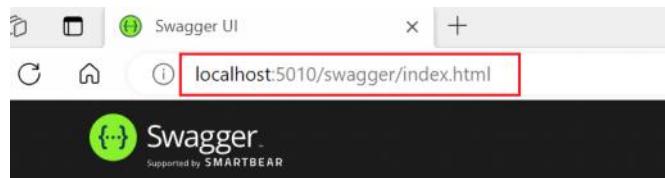
The screenshot shows the VS Code interface with the file `launchSettings.json` open. The code defines profiles for both `http` and `https`. For the `http` profile, the `applicationUrl` is set to `'http://localhost:5010'`. For the `https` profile, the `applicationUrl` is set to `'https://localhost:7010;http://localhost:5010'`.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

```
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"
30      }
31  }
```

Below the code editor, the status bar shows the tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, PORTS, GITLENS, COMMENTS, and TERMINAL. The TERMINAL tab is currently active.

```
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

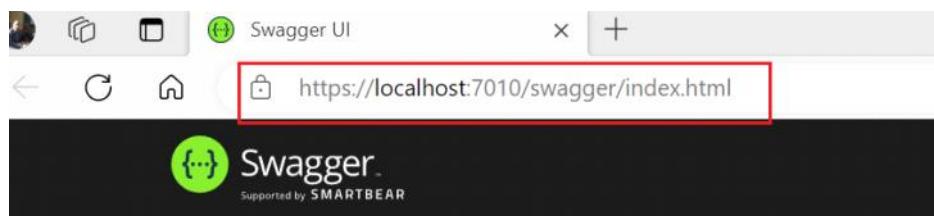
DateOnly >

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

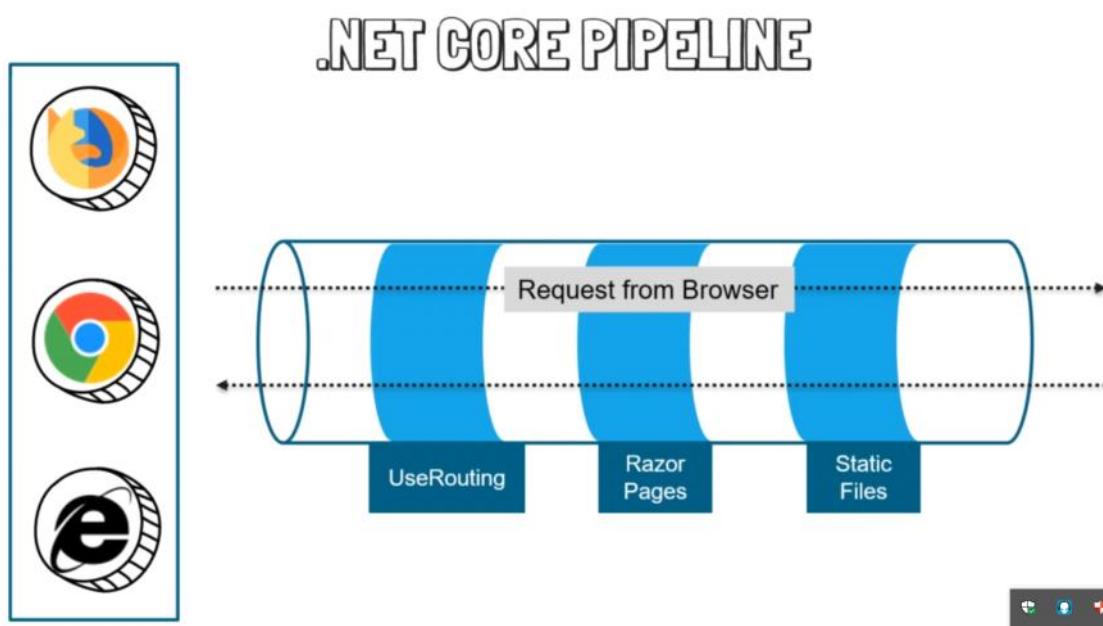
Tuesday, January 9, 2024 12:33 PM

@ 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

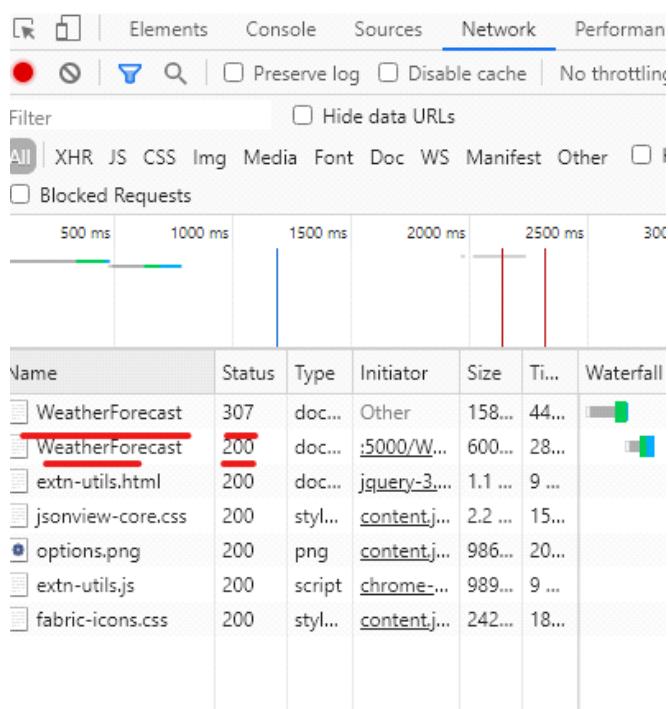
```
app.UseHttpsRedirection();
```

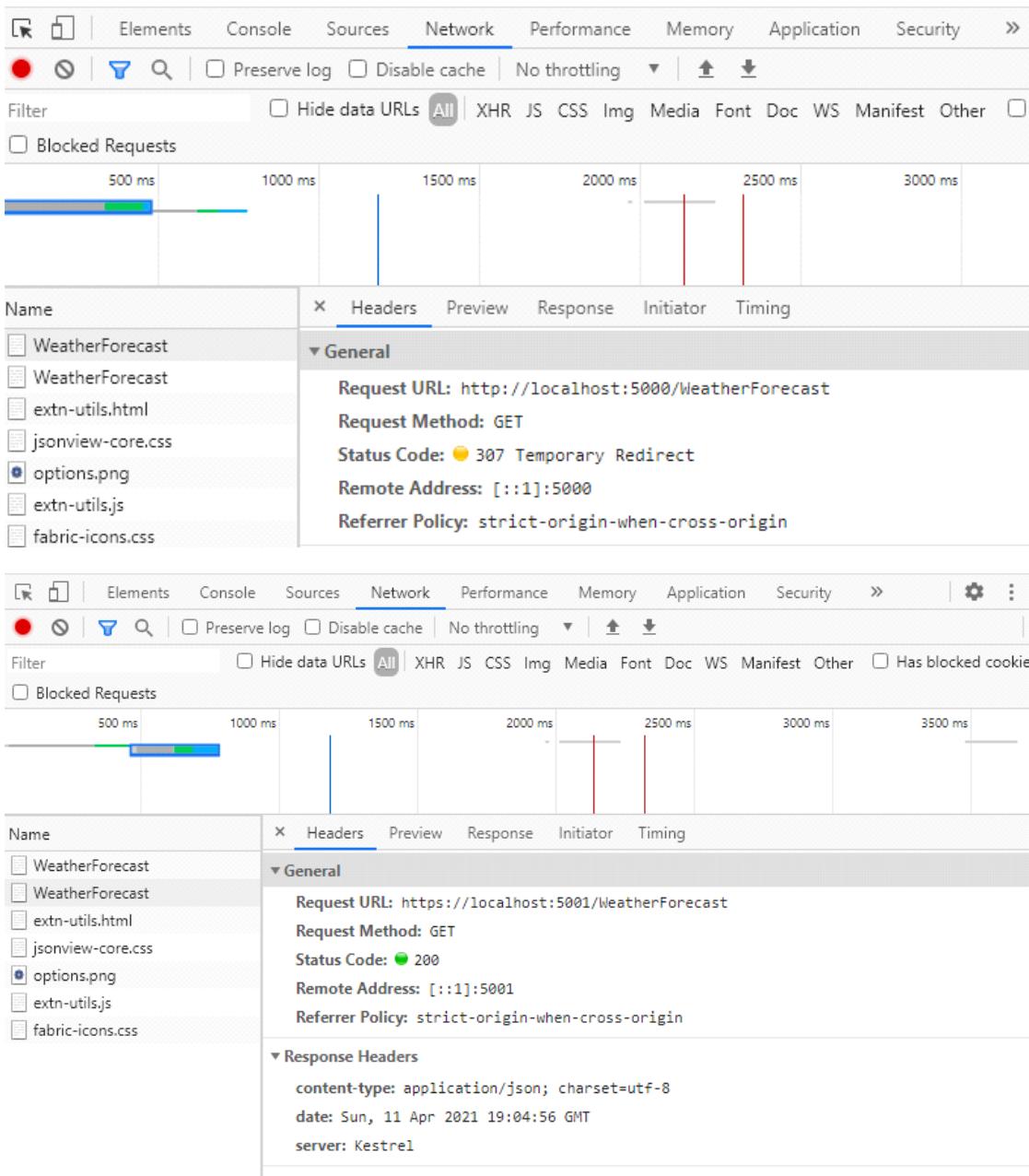
Assume we call HTTP instead of https

<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
}

```

launchSettings.json

```

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: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     },
32
33     "IIS Express": {
34         "commandName": "IISExpress",
35         "launchBrowser": true,
36         "launchUrl": "swagger",
37         "environmentVariables": {

```

appsettings.Development.json

```

MinAPI > appsettings.Development.json > ...
You, 2 months 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 }

```

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

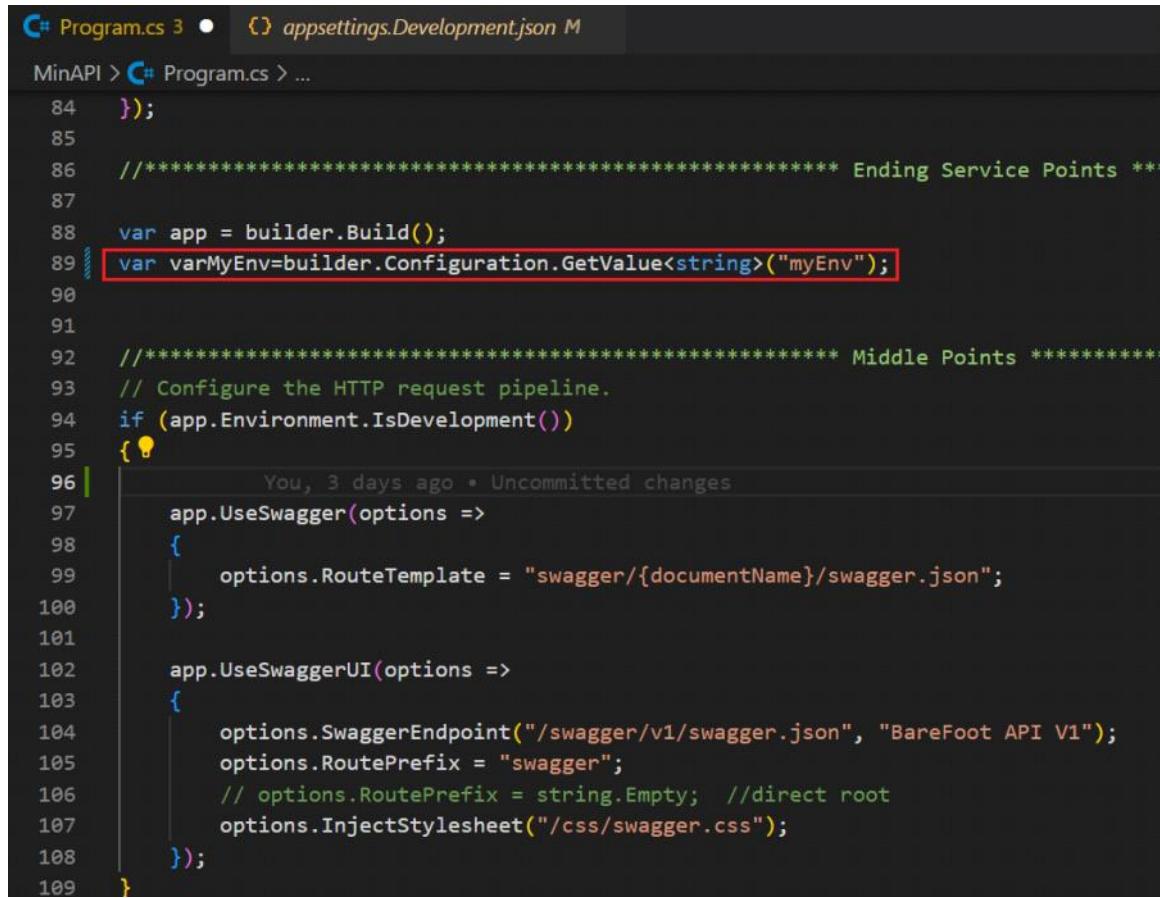
```
launchSettings.json X  
MinAPI > Properties > launchSettings.json > ...  
3 |   "iisSettings": {  
4 |     "iisExpress": {  
5 |       "applicationUrl": "http://localhost:7010",  
6 |       "sslPort": 44346  
7 |     }  
8 |   },  
9 |   "profiles": {  
10 |     "https": {  
11 |       "commandName": "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");
```

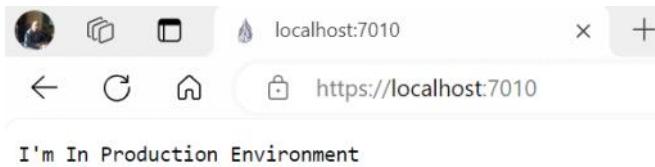
The screenshot shows the Visual Studio code editor with two tabs open: 'Program.cs' and 'appsettings.Development.json'. The 'Program.cs' tab contains C# code for a .NET Core application. The code includes environment handling logic, static file serving, and several endpoint mappings using the `MapGet` method. A specific line of code, `app.MapGet("/", ()=>varMyEnv);`, is highlighted with a red rectangle and has a yellow status bar tip indicating 'You, 3 days ago * Uncommitted changes'. The 'appsettings.Development.json' tab is also visible at the top.

```
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(/
```

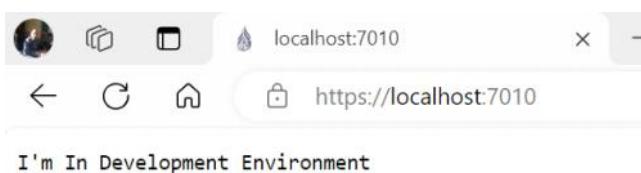
```
app.MapGet("/", ()=>varMyEnv);
```

Let we run this

dotnet watch run --environment Production



dotnet watch run --environment Development

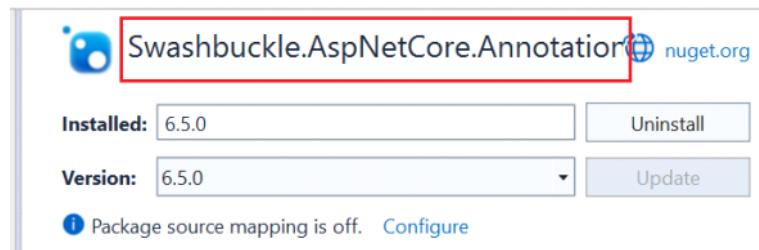
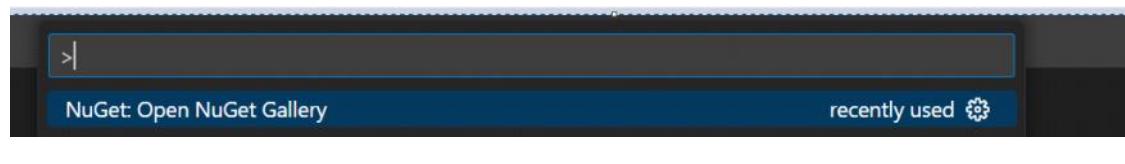


Or by swagger

The screenshot shows the Swagger UI interface for a .NET Core API. At the top, there's a navigation bar with icons for back, forward, search, and refresh. The URL in the address bar is `https://localhost:7010/swagger/index.html`. Below the address bar, a blue header bar says "GET /". Underneath, a "Parameters" section says "No parameters". A large "Execute" button is prominent, followed by a "Clear" button. The "Responses" section is collapsed. In the "Curl" section, there's a code block for a curl command to make a GET request to `https://localhost:7010/` with an accept header of `text/plain`. Below that, the "Request URL" is shown as `https://localhost:7010/`. The "Server response" section shows a status code of 200 and a response body containing the text "I'm In Development Environment". There are "Download" and "Copy" buttons next to the response body.

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

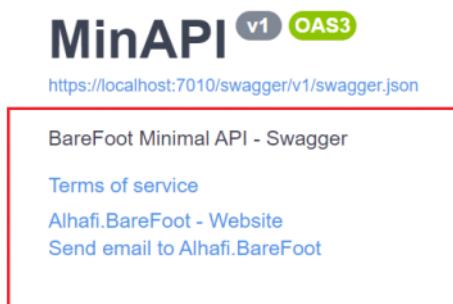
```

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/")
        }
    );
});

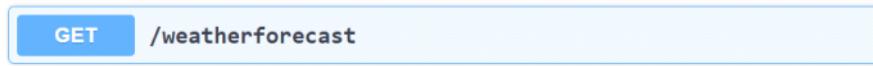
```

Testing

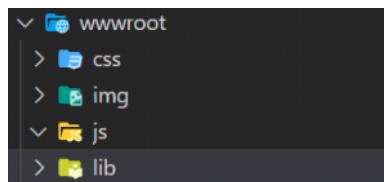
PS D:\VisualCode\BareFoot_API\minapi> **dotnet watch run**



MinAPI



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



Add middle point

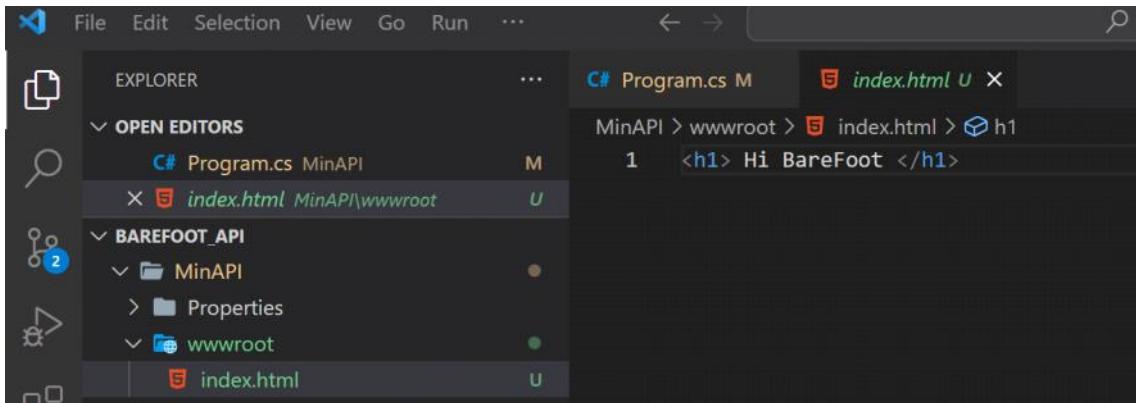
```

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 = "";
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[]

```

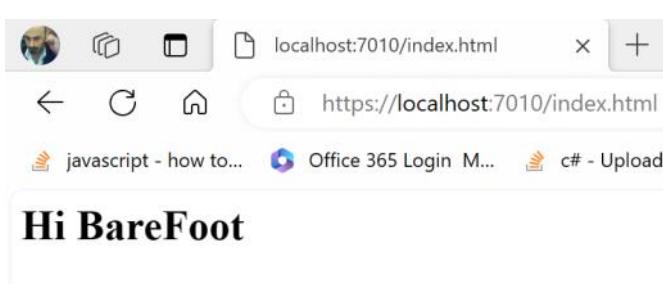
app.UseStaticFiles();

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
```

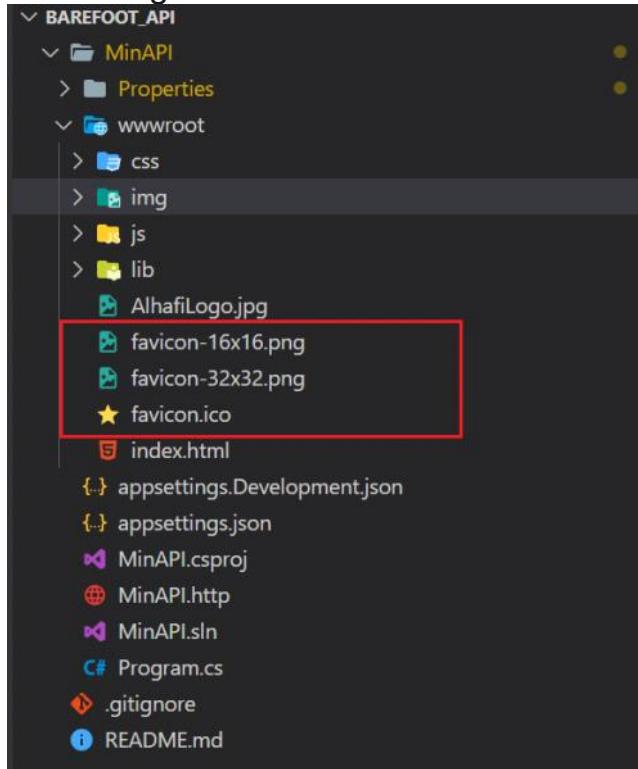
Replace Swagger Icon/Logo

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 under 'OPEN EDITORS'. The 'wwwroot/css' folder contains 'swagger.css', which is currently selected and highlighted with a red box. The right pane shows the code editor with the content of 'swagger.css'. The code defines two CSS rules: '.topbar-wrapper img' and '.topbar-wrapper .link:after'. The first rule sets the background image to 'AlhafiLogo.jpg', the background color to white, and the alt text to 'AlhafiLogo'. The second rule creates a pseudo-element for the link selector, setting its content to 'BareFoot Swagger Documentation', color to white, visibility to visible, display to block, position to absolute, and padding to 60px. A red box highlights the entire code block.

```
.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 code configures the HTTP request pipeline. It starts by checking if the environment is development. If so, it uses the Swagger UI and injects the 'swagger.css' stylesheet from the 'wwwroot/css' directory. The 'options.InjectStylesheet' method 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

```

```

RECORDS
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

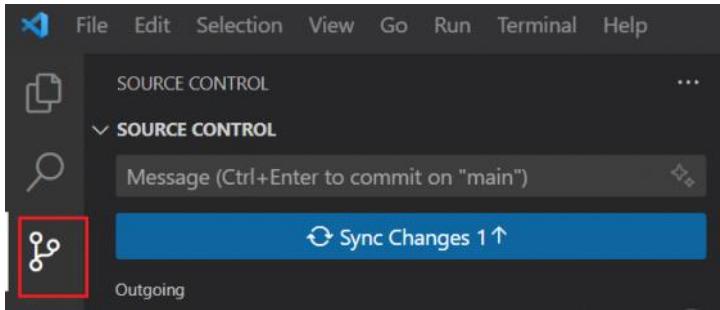
/swagger/v1/swagger.json

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
```

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 }
82
```

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

Define Variable

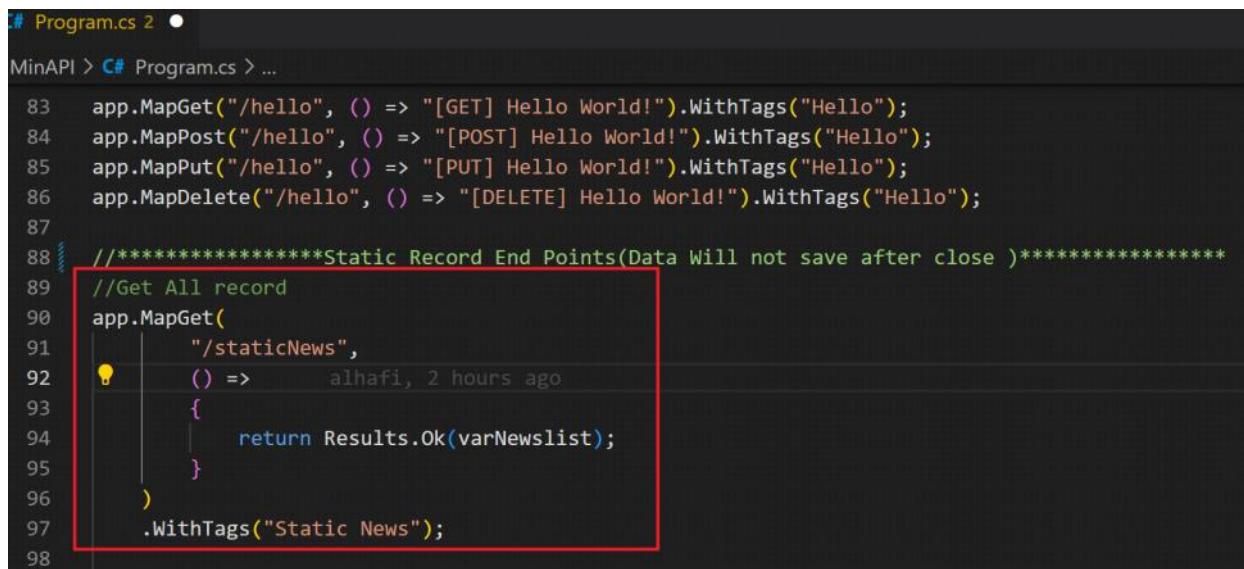
```
C# Program.cs M ●
MinAPI > C# Program.cs > ...
You, 7 minutes ago | 2 authors (You and others)
1 using Microsoft.OpenApi.Models;
2
3 var builder = WebApplication.CreateBuilder(args);
4
5 //Creating Variables of Lists
6 var varNewslist = new List<NewsListStatic>;
7 {
8     new NewsListStatic
9     {
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 navigation icons and links: '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 small 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

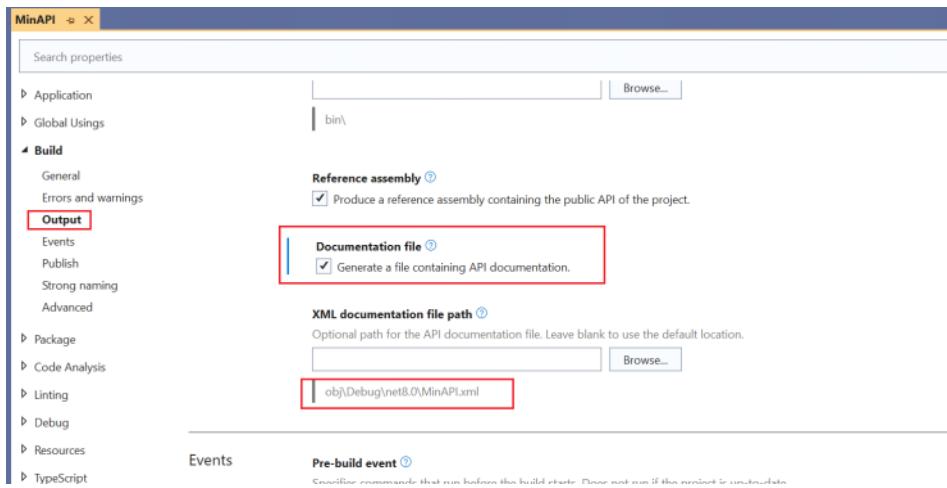
Static News

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 = "إظهار جميع الاخبار",
    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 السجل")
.WithName("GetStaticNewsbyID")
.WithTags("Static News")
.WithOpenApi();

```

```

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

```

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

Dynamic Data -ContextDB

11 February 2024 10:52

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

Adding Packaging

Databases

Microsoft.EntityFrameworkCore by Microsoft

Microsoft.EntityFrameworkCore.Design by Microsoft

SQLite

Microsoft.EntityFrameworkCore.Sqlite.Core by Microsoft

MSSQL

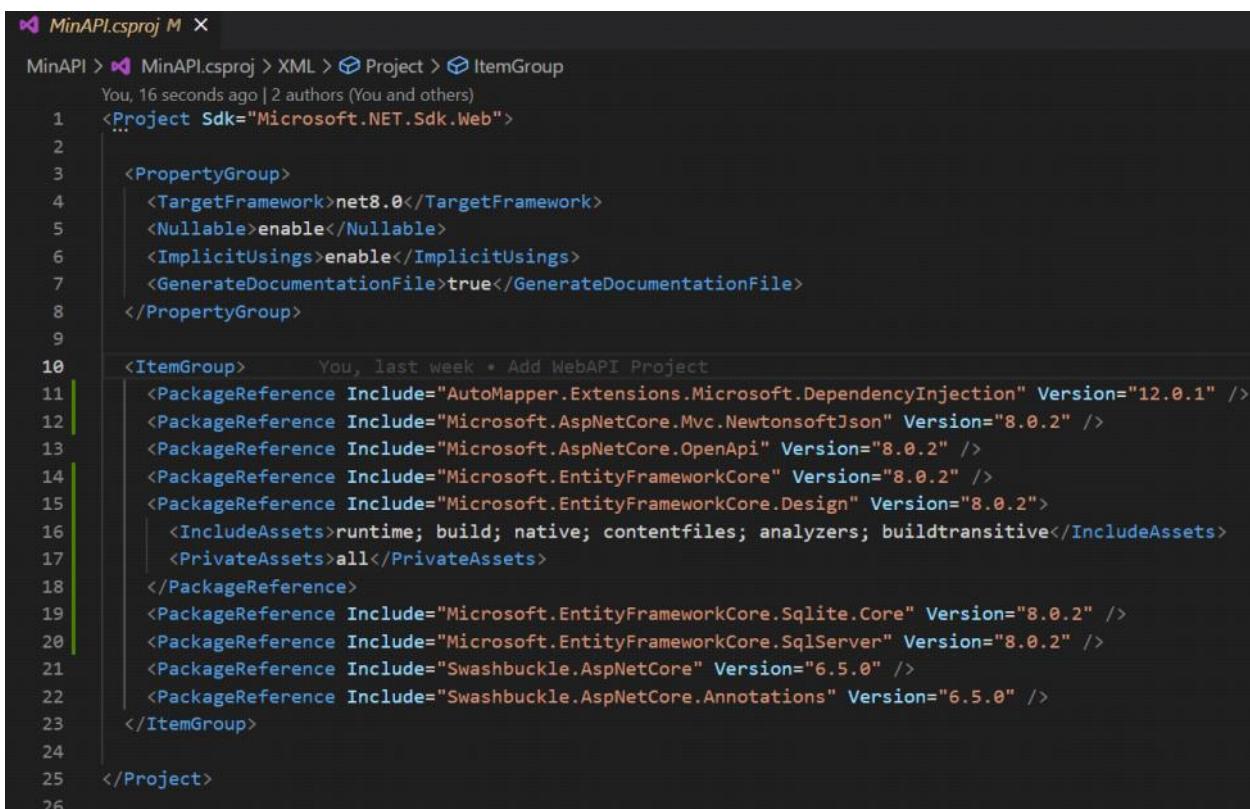
Microsoft.EntityFrameworkCore.SqlServer by Microsoft

Utilities

Microsoft.AspNetCore.Mvc.NewtonsoftJson by Microsoft

AutoMapper.Extensions.Microsoft.DependencyInjection by Microsoft

Our project installed packages



```
MinAPI > MinAPI.csproj M X
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          public AppDbContext(DbContextOptions<AppDbContext> options)
 10             : base(options) { }
 11
 12
 13      }
 14
  
```

```

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

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

Adding Connection String to the Database

The screenshot shows a GitHub commit for the file `appsettings.Development.json`. The commit message is "MinAPI > { } appsettings.Development.json M". It was made by "You" 1 minute ago | 1 author (You). The JSON code is as follows:

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

The line containing the connection string is highlighted with a red rectangle.

```
{  
    "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

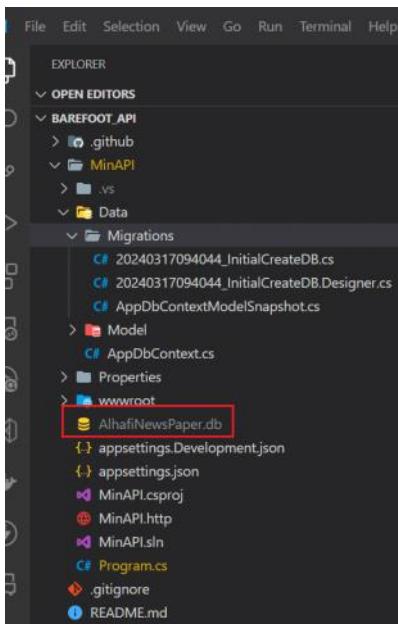
List Migration

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

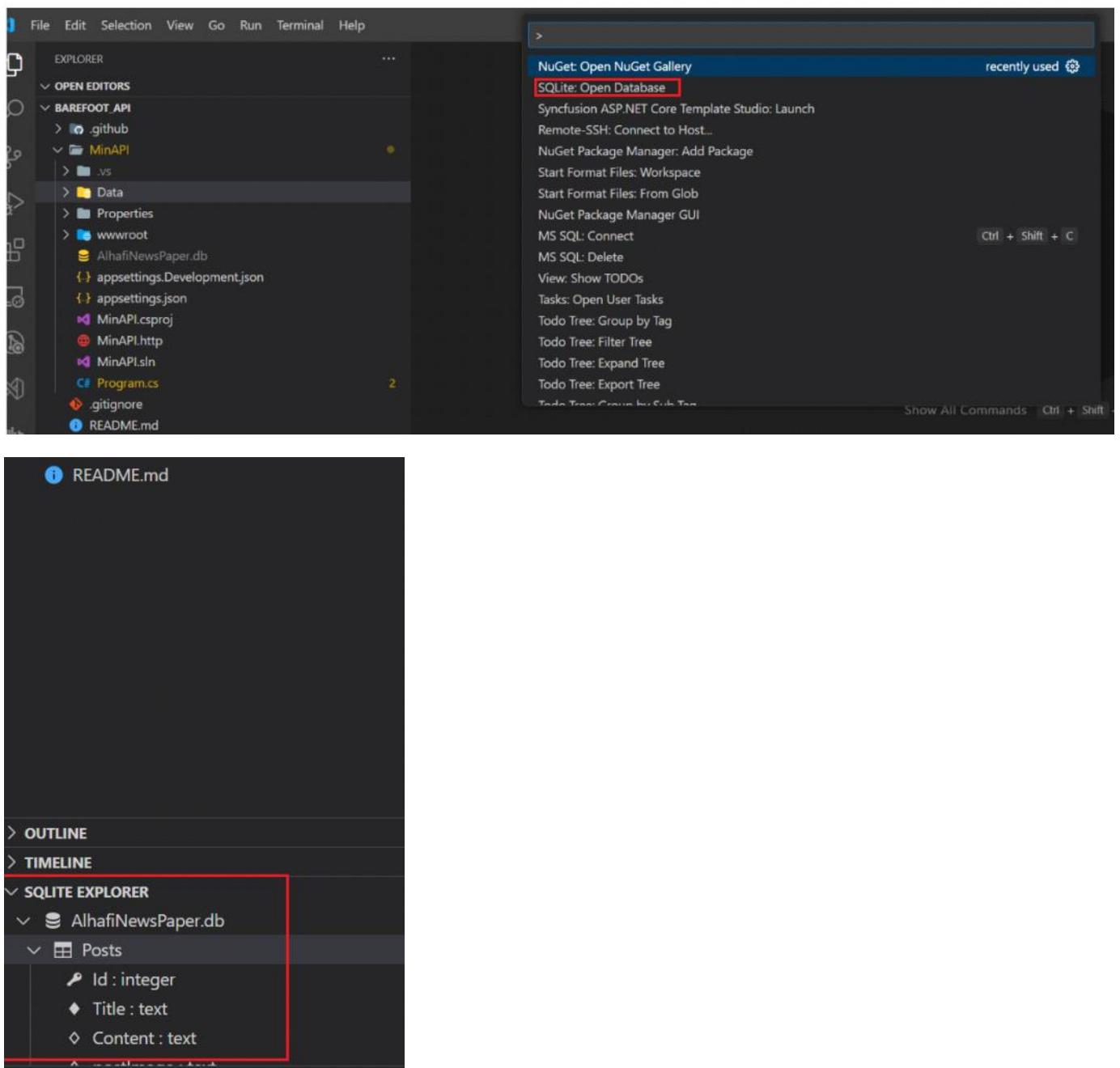
Creating Database

dotnet ef database update

Check your DB

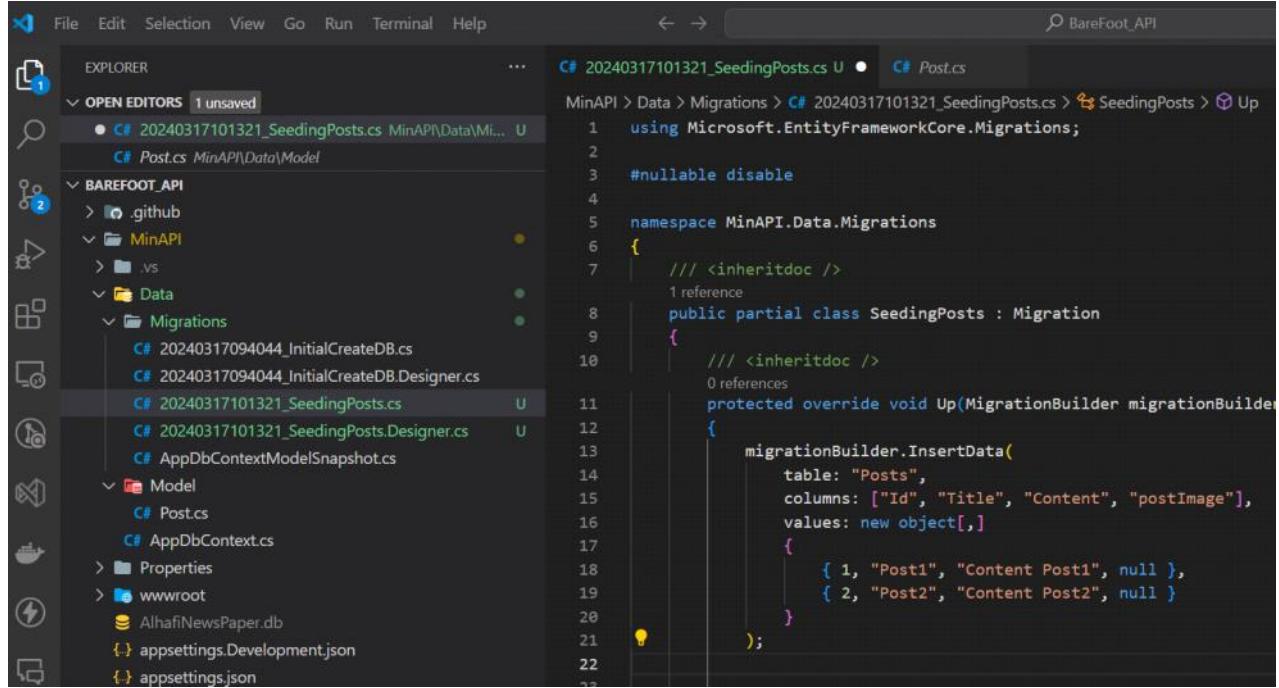


CTRL+Shift+P



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: .github, MinAPI (containing .vs, Data, Migrations, Model, Properties, wwwroot), and AlhafinNewsPaper.db. The "Migrations" folder contains several files: 20240317094044_InitialCreateDB.cs, 20240317094044_InitialCreateDB.Designer.cs, 20240317101321_SeedingPosts.cs (which is selected and has a yellow dot icon next to it), 20240317101321_SeedingPosts.Designer.cs, and ApplicationDbContextModelSnapshot.cs. The right pane shows the code for 20240317101321_SeedingPosts.cs:

```
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

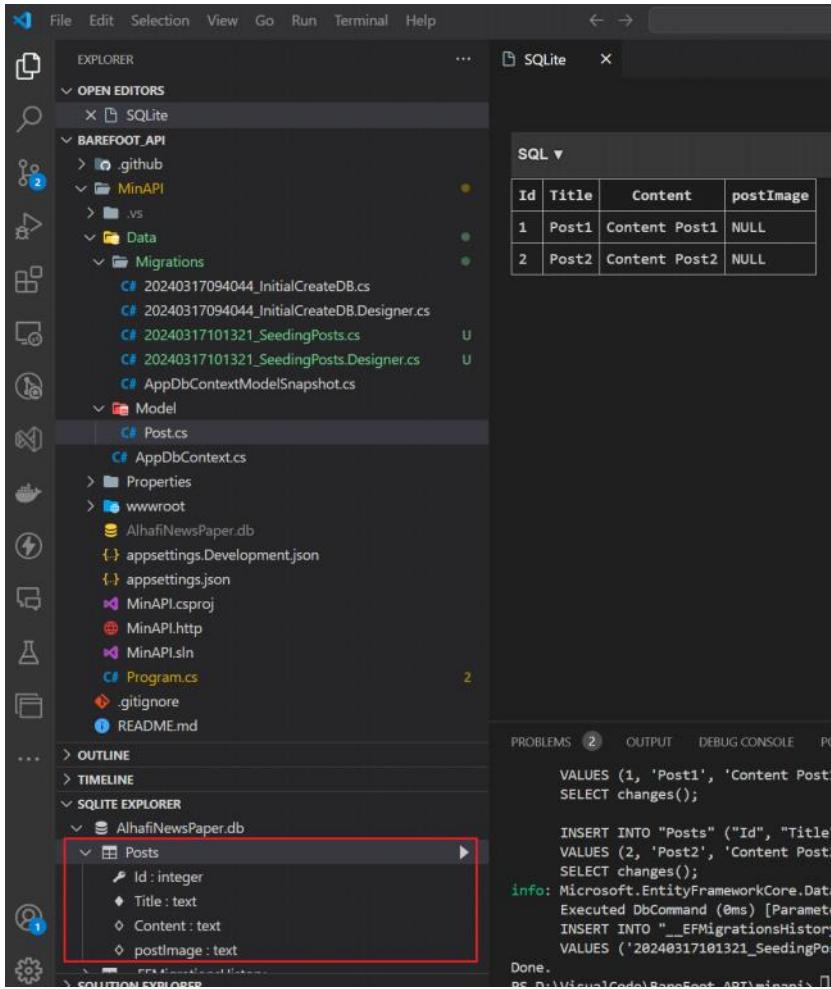
```
migrationBuilder.InsertData(
    table: "Posts",
    columns: ["Id", "Title", "Content", "postImage"],
    values: new object[,]
    {
        { 1, "Post1", "Content Post1", null },
        { 2, "Post2", "Content Post2", null }
    });

```

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_Get"));
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_Get"));
41
42     }
43 }
```

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 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

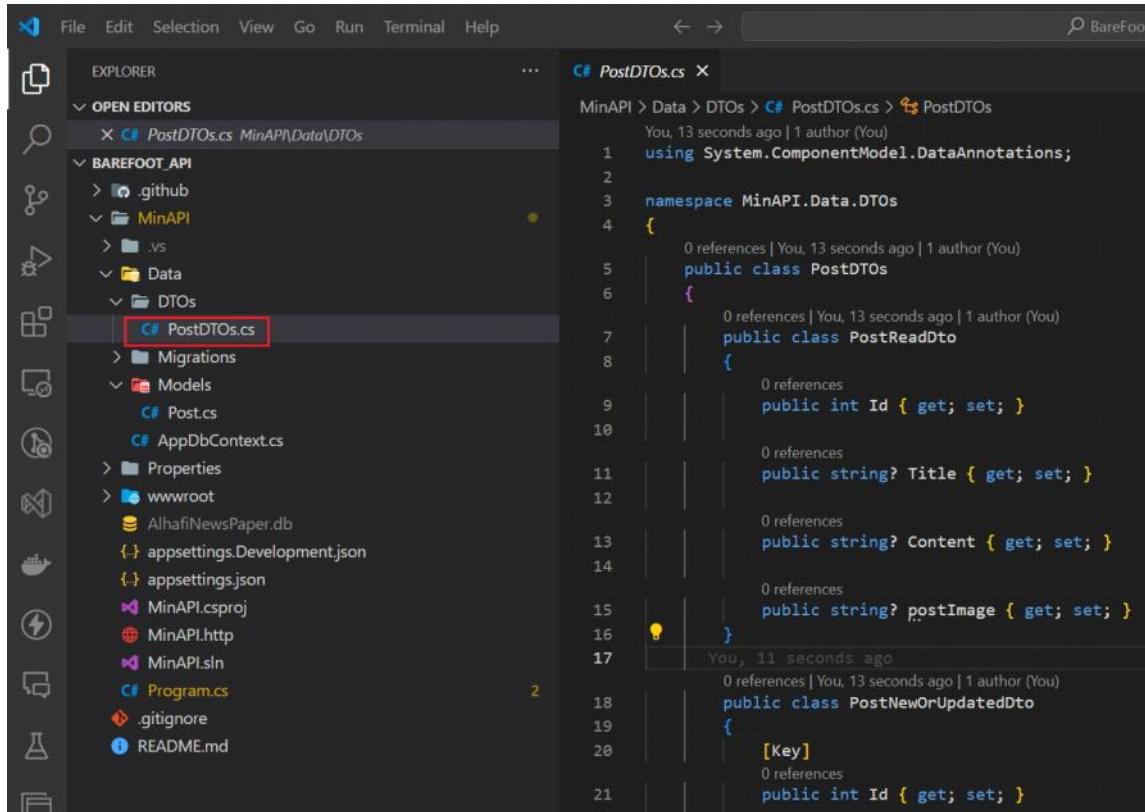
GET /hello

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 IDE interface. The left sidebar is the Explorer pane, which lists the project structure. The 'OPEN EDITORS' section shows 'PostDTOs.cs' is open. The main area is the code editor for 'PostDTOs.cs'. The code defines two classes: 'PostReadDto' and 'PostNewOrUpdatedDto', both annotated with [Required] and [MinLength(3)]. The code is as follows:

```
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; }
        }
    }
}
```

```
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, getting all posts, creating posts, 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` parameter and sets it to the `_context` field. It also includes an implementation for the `CreatePost` method.

```

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
    {

```

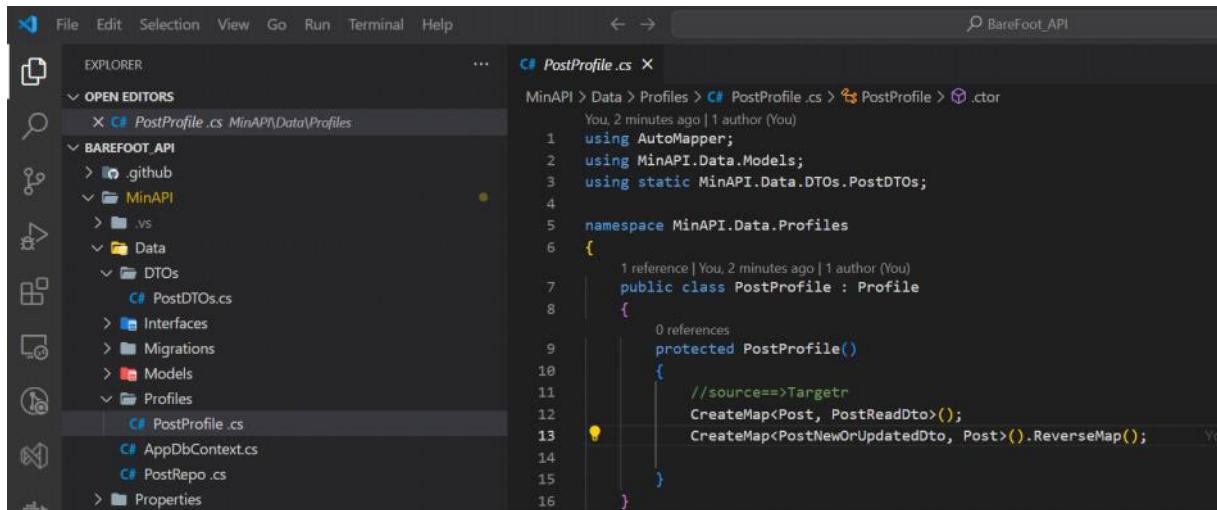
```

private readonly ApplicationDbContext _context;
public PostRepo(ApplicationDbContext 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



```

File Edit Selection View Go Run Terminal Help
EXPLORER OPEN EDITORS PostProfile.cs X
MinAPI > Data > Profiles > PostProfile.cs > PostProfile > .ctor
You, 2 minutes ago | 1 author (You)
1 using AutoMapper;
2 using MinAPI.Data.Models;
3 using static MinAPI.Data.DTOs.PostDTOs;
4
5 namespace MinAPI.Data.Profiles
6 {
7     1 reference | You, 2 minutes ago | 1 author (You)
8     public class PostProfile : Profile
9     {
10         0 references
11         protected PostProfile()
12         {
13             //source==>Targetr
14             CreateMap<Post, PostReadDto>();
15             CreateMap<PostNewOrUpdatedDto, Post>().ReverseMap();
16         }
17     }
18 }

```

```

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

```

```

        }
    }
}

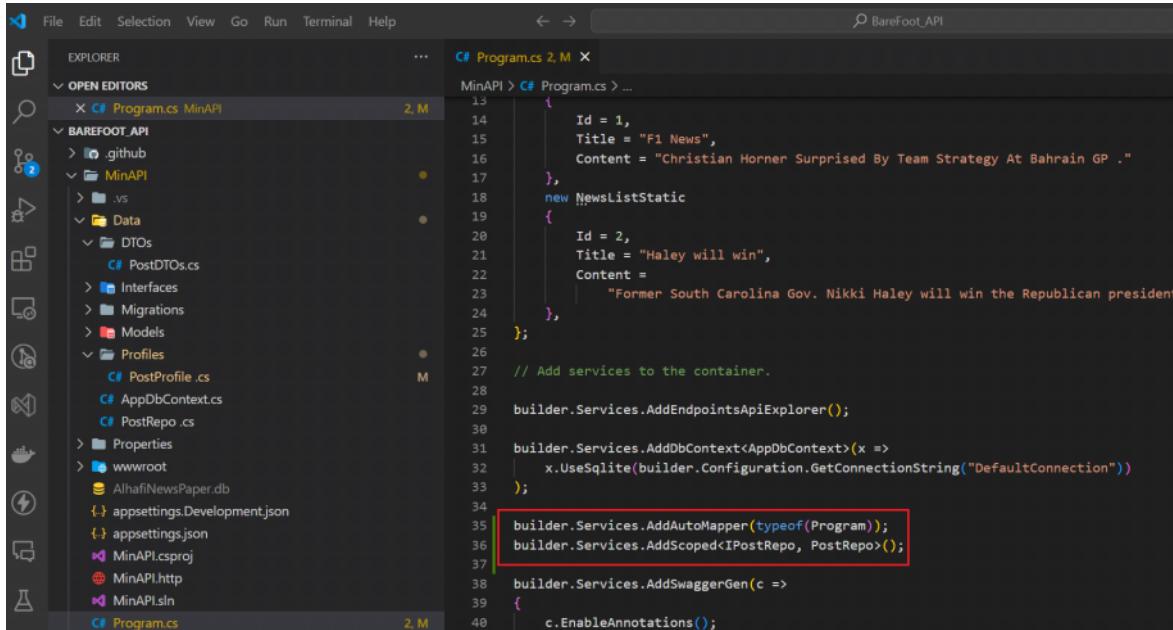
```

Register Automapper

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

Register Repository by Concrete (Class and Interface)

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



Building Endpoints API

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

app.MapPost(
    "/automapper/posts",
    async (IPostRepo repo, IMapper mapper, PostReadDto post) =>
{
    var varPost = mapper.Map<PostCreateDto>(post);
    var result = await repo.CreatePost(varPost);
    return Results.Created($"/automapper/posts/{result.Id}", result);
}
)

```

```

    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();

//*****

```

 BareFoot Swagger Documentation Select a definition BareFoot API V1 ▾

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

AutoMapper

- GET /automapper/posts احصئ جميع الاخبار
- POST /automapper/posts تغيل خبر جديده
- GET /automapper/posts/{id} احصئ خبر واحد
- PUT /automapper/posts/{id} تحليل خبر
- DELETE /automapper/posts/{id} حذف خبر

DBContext

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/>

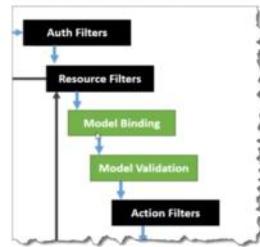
This means*...

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

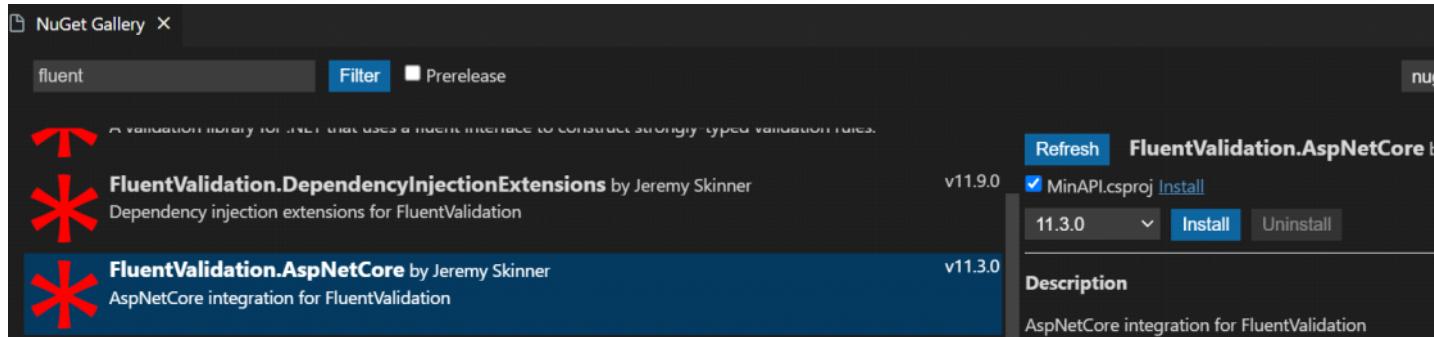
```
curl -X 'GET' \
  'https://localhost:7010/automapper/posts' \
  -H 'accept: */*'
```

Request URL

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

Server response

Let we add Fluent Validation Package



Then Add validation

The screenshot shows a Visual Studio code editor window. The title bar indicates the project is 'MinAPI' and the current file is 'Post.cs'. The code editor displays the following C# code:

```
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

```
C# Program.cs 2 ●
MinAPI > C# Program.cs > ...
26         Content =
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
51             {
52                 Title = "Barefoot APT"
53             }
54         );
55     }
56     .WithDescription("return Only One Post News")
57     .WithSummary("احضار خبر واحد")
58     .WithTags("DbContext")
59     .WithOpenApi();
60
61     app.MapPost(
62         "/dbcontext/posts",
63         async (AppDbContext context, Post poss) =>
64         {
65             await context.Posts.AddAsync(poss);
66             await context.SaveChangesAsync();
67             return Results.Created($"/posts/{poss.Id}", poss);
68         }
69     )
70     .WithDescription("Insert New Post News")
71     .WithSummary("ادخال خبر جديـد")
72     .WithTags("DbContext")
73     .WithOpenApi();
74
75 }
```

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

Let we validate Post

```
C# Program.cs 2, M X
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 جديـد")
266 .WithTags("DbContext")
267 .WithOpenApi();
268
269 }
```

```
app.MapPost<Post>("/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();
```

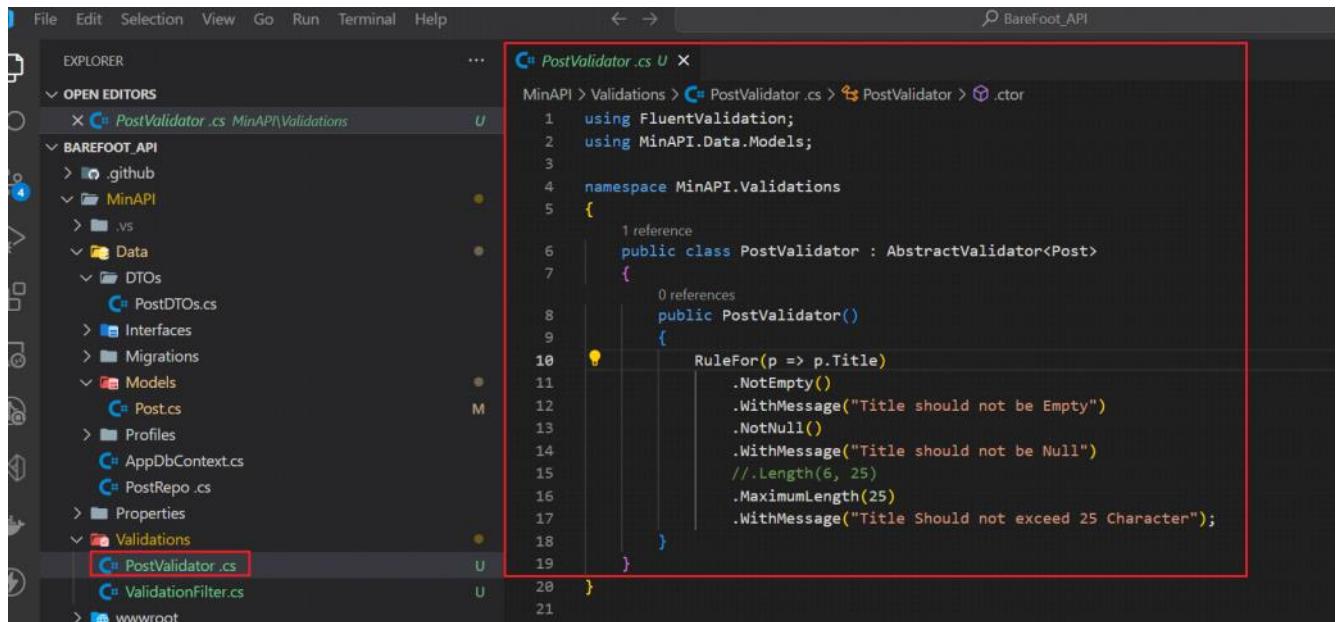
Let we Test

```
{  
  "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
    );
}
)

```

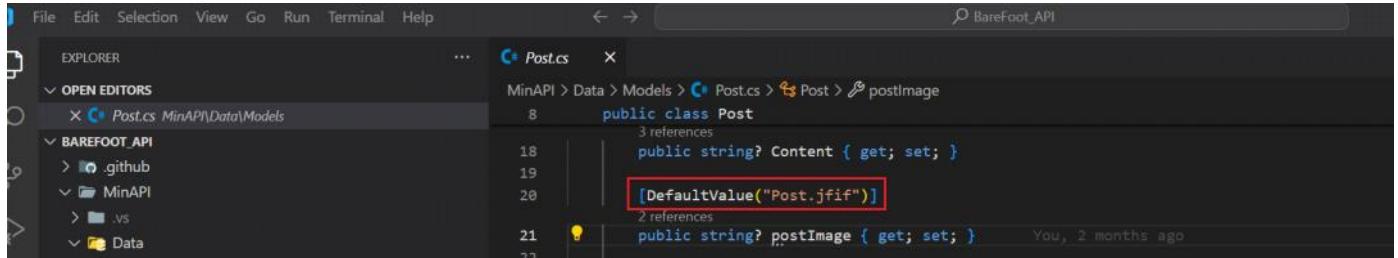
```

.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

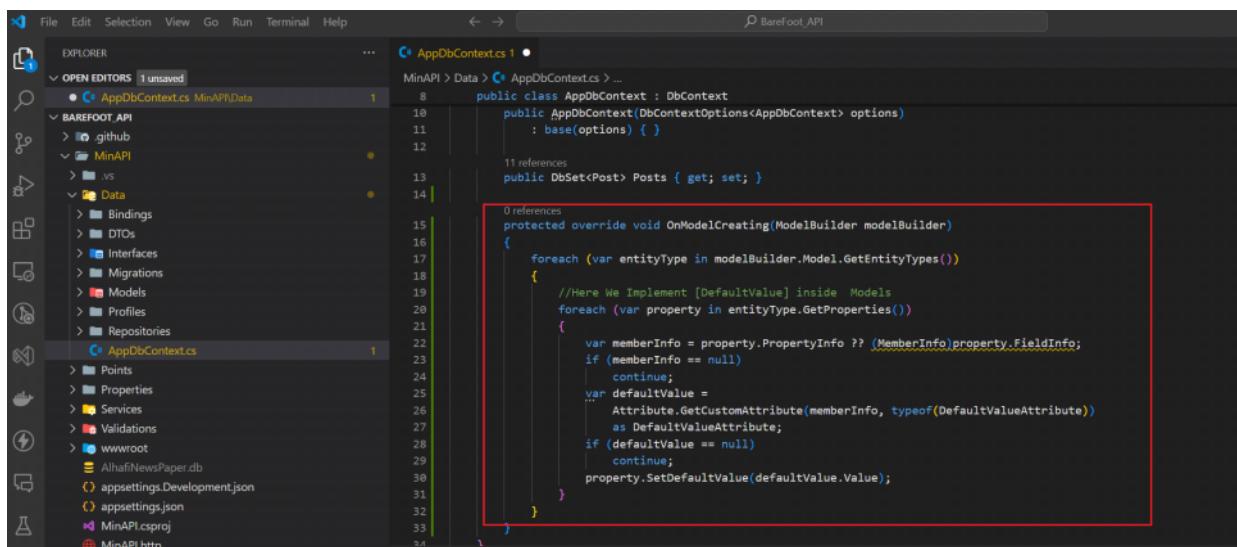


```

File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
Post.cs
MinAPI > Data > Models > Post.cs > Post > postImage
public class Post
{
    public string? Content { get; set; }
    [DefaultValue("Post.jfif")]
    public string? postImage { get; set; }
}

```

Let we add on Model creation



```

File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS 1 unsaved
AppDbContext.cs
MinAPI > Data > AppDbContext.cs ...
public class AppDbContext : DbContext
{
    public AppDbContext(DbContextOptions<AppDbContext> options)
        : base(options) {}

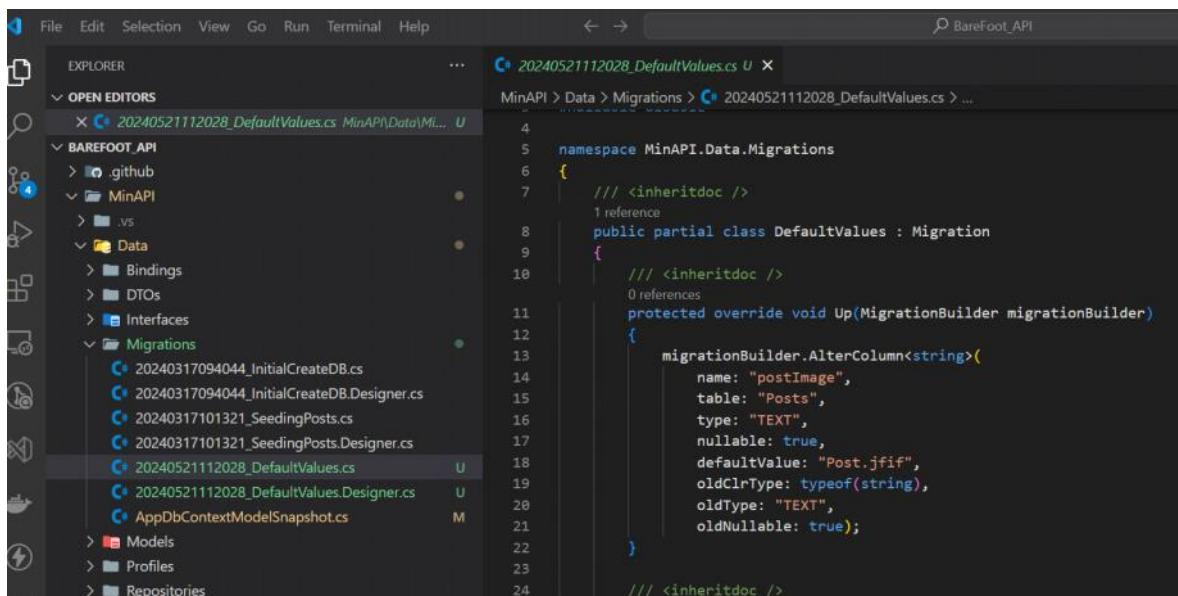
    public DbSet<Post> Posts { get; set; }

    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));
                if (defaultValue == null)
                    continue;
                property.SetDefaultValue(defaultValue.Value);
            }
        }
    }
}

```

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

dotnet ef migrations add DefaultValues



```

File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
20240521112028_DefaultValues.cs
MinAPI > Data > Migrations > 20240521112028_DefaultValues.cs ...
namespace MinAPI.Data.Migrations
{
    /// <inheritdoc />
    public partial class DefaultValues : Migration
    {
        /// <inheritdoc />
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.AlterColumn<string>(
                name: "postImage",
                table: "Posts",
                type: "TEXT",
                nullable: true,
                defaultValue: "Post.jfif",
                oldClrType: typeof(string),
                oldType: "TEXT",
                oldNullable: true);
        }
    }
}

```

```
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; }
```

```
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 code snippet:

```
        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.

GET /hello/QueryString

Parameters

Name	Description
name * required	AbdullRaheim Alhafi (query)

Responses

Curl

```
curl -X 'GET' \
  'https://localhost:7010/hello/QueryString?name=AbdullRaheim%20Alhafi' \
  -H 'accept: text/plain'
```

Request URL

<https://localhost:7010/hello/QueryString?name=AbdullRaheim%20Alhafi>

Server response

Code	Details
200	Response 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 > Program.cs ...
Program.cs 3, 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);
```

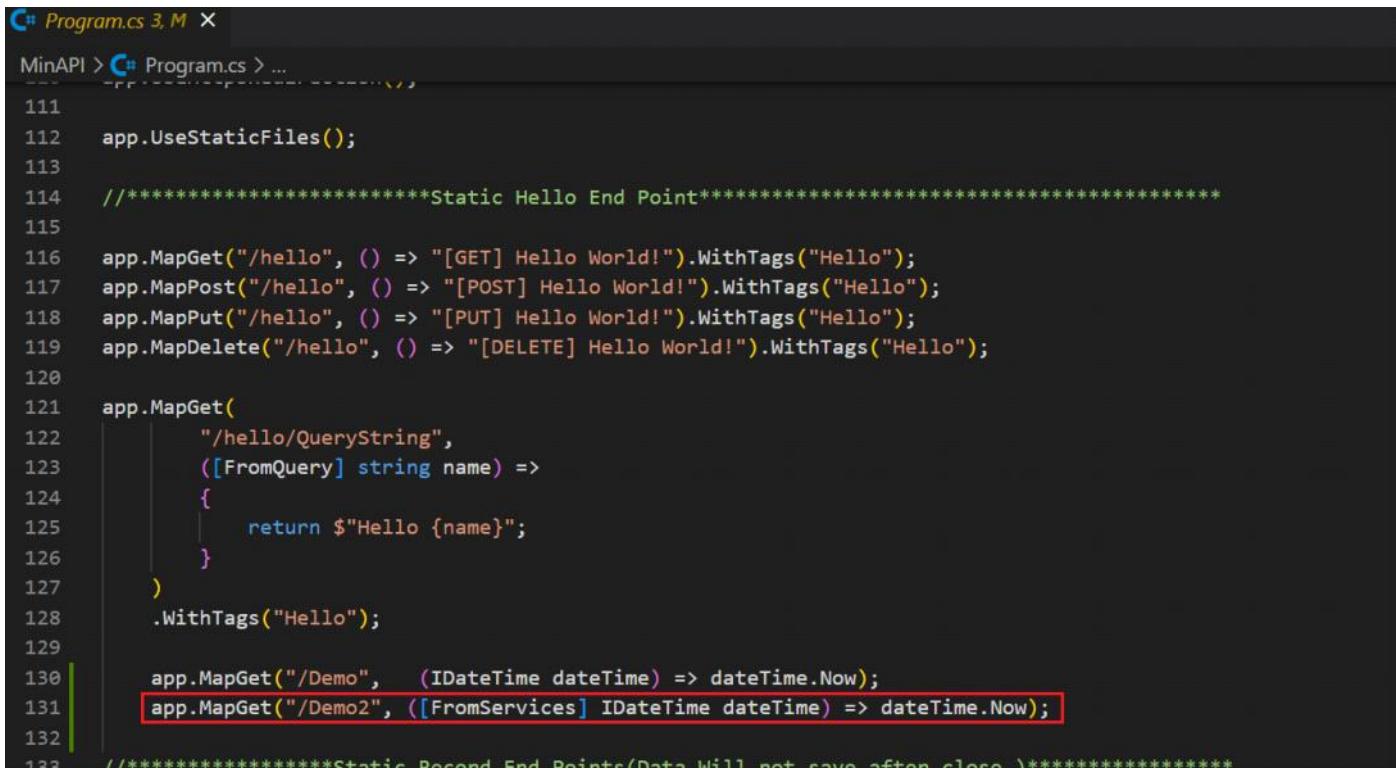
MinAPI

The screenshot shows the MinAPI documentation interface. At the top, there is a navigation bar with 'GET /Demo'. Below it, under 'Parameters', it says 'No parameters'. In the center, there is a large blue button labeled 'Execute'. Under 'Responses', there is a 'Curl' section with a code snippet:

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

Below the curl section, there is a 'Request URL' field containing <https://localhost:7010/Demo>. Under 'Server response', there is a table with two columns: 'Code' and 'Details'. The 'Code' column has a row for '200'. The 'Details' column for '200' has a sub-section 'Response body' which contains the text '30/04/2024 14:46:55'. This 'Response body' section is highlighted with a red box.

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();
```

<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)

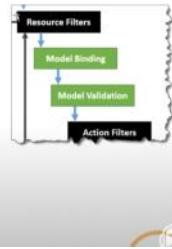
Custom model binding allows developers to define their own binding logic for complex data types or scenarios that cannot be handled by the default binding mechanisms. Custom binding is particularly useful when working with APIs that require data transformation or normalization before the data can be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

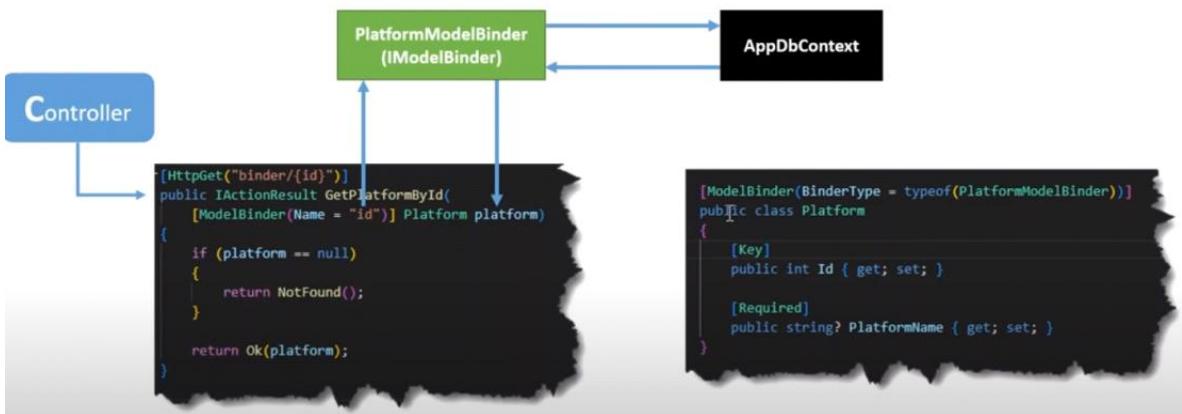
In the `BindModelAsync` method, you can perform any necessary data transformation or validation before returning the bound model.

Custom Model Binding (IModelBinder)

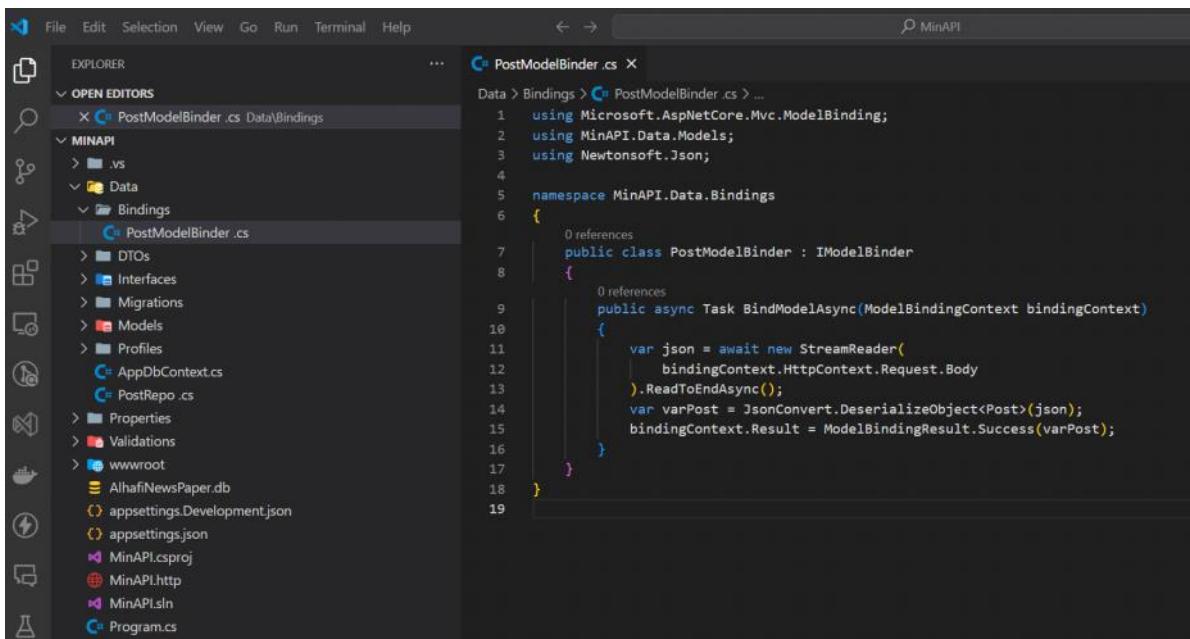
- Allow Controller Actions to work directly with Model Types
 - (Rather than HTTP requests)
- Can be used in more complex “binding scenarios”
 - E.g. if you need to transform data
- Default model binders support most common .NET types
 - They meet most developers needs!
- Model Binding Occurs before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

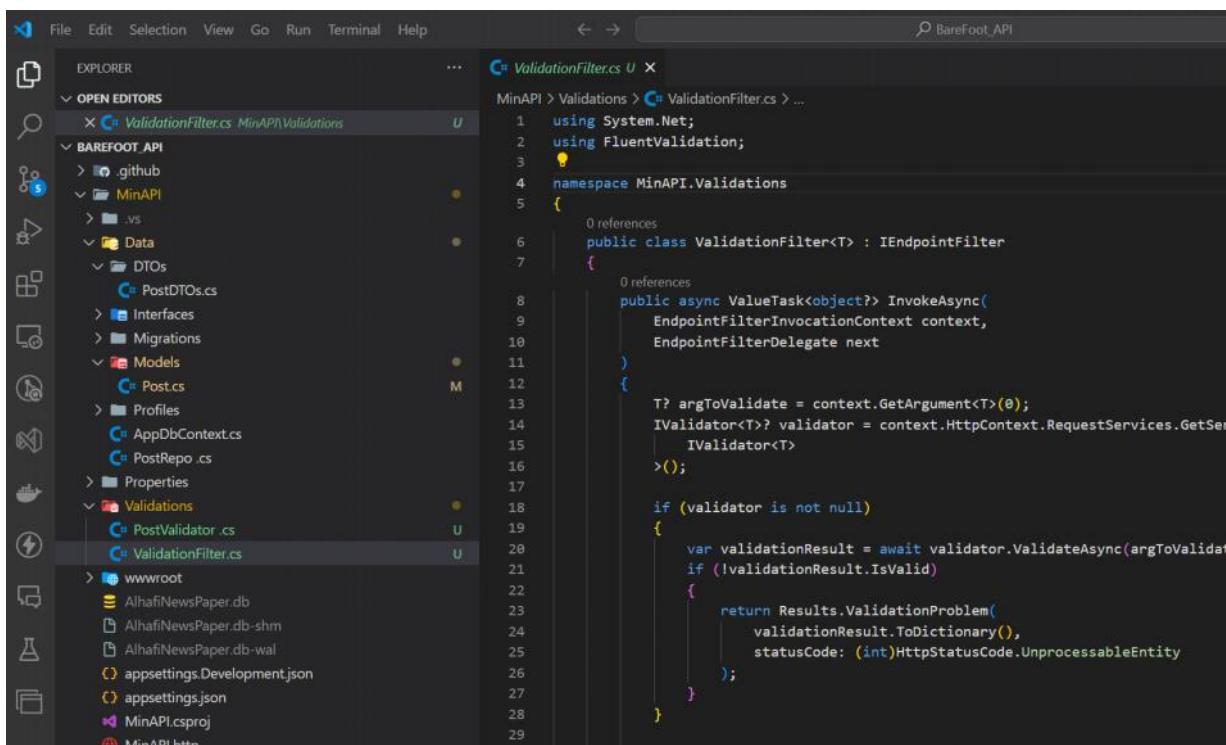
This means*...

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);
}

```

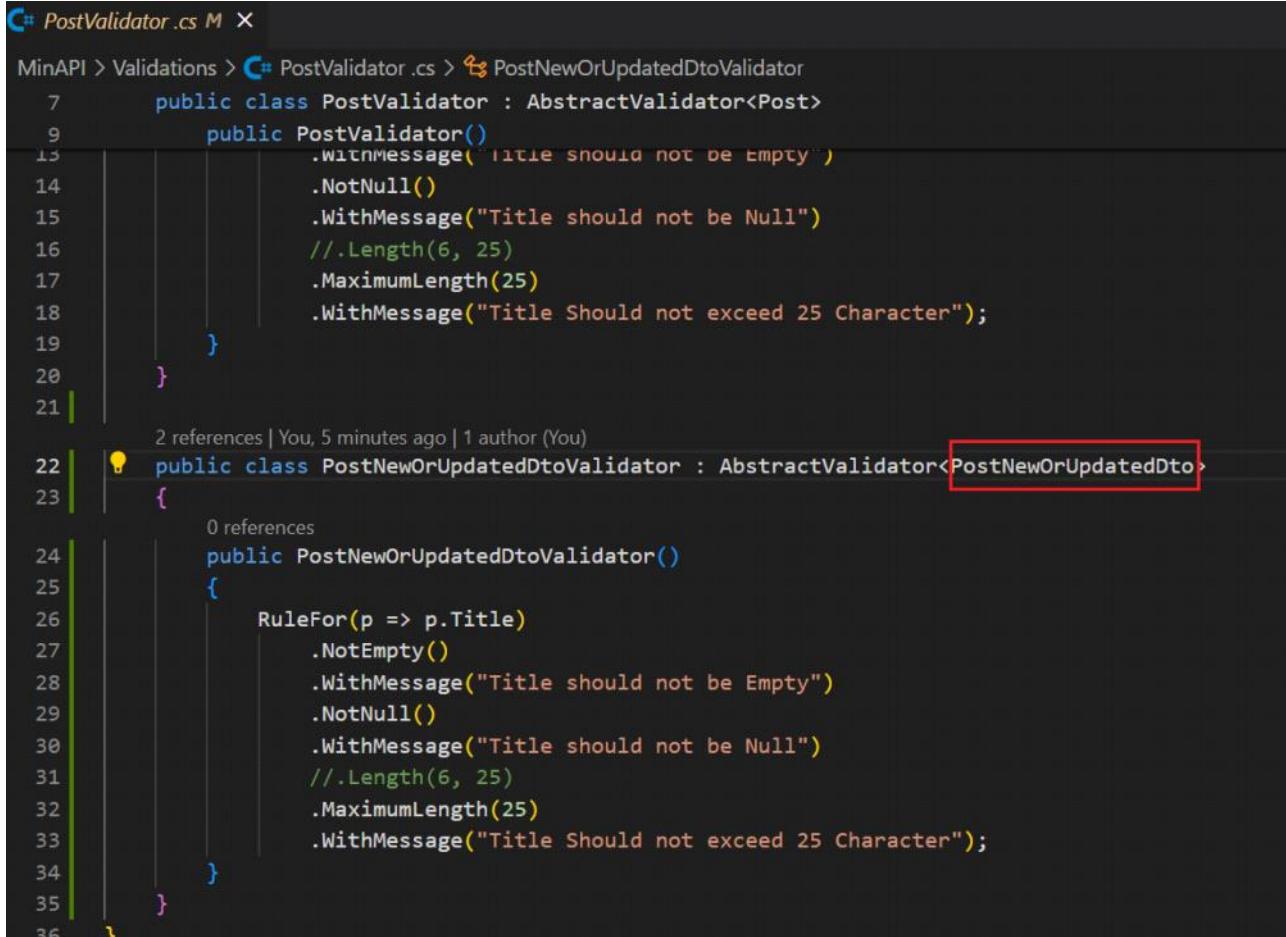
```

        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
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 {
25     0 references
26     public PostNewOrUpdatedDtoValidator()
27     {
28         RuleFor(p => p.Title)
29             .NotEmpty()
30             .WithMessage("Title should not be Empty")
31             .NotNull()
32             .WithMessage("Title should not be Null")
33             //.Length(6, 25)
34             .MaximumLength(25)
35             .WithMessage("Title Should not exceed 25 Character");
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 جديد ")
'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 جديد ")
.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

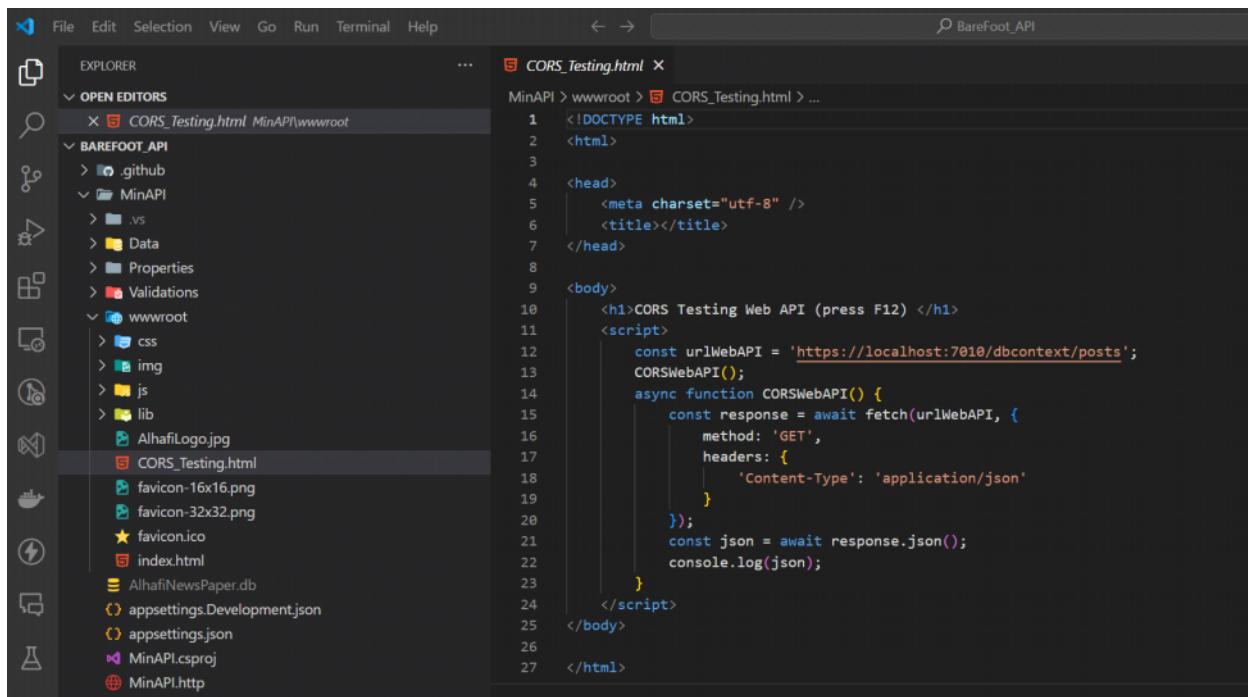
12 May 2024 10:29

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>

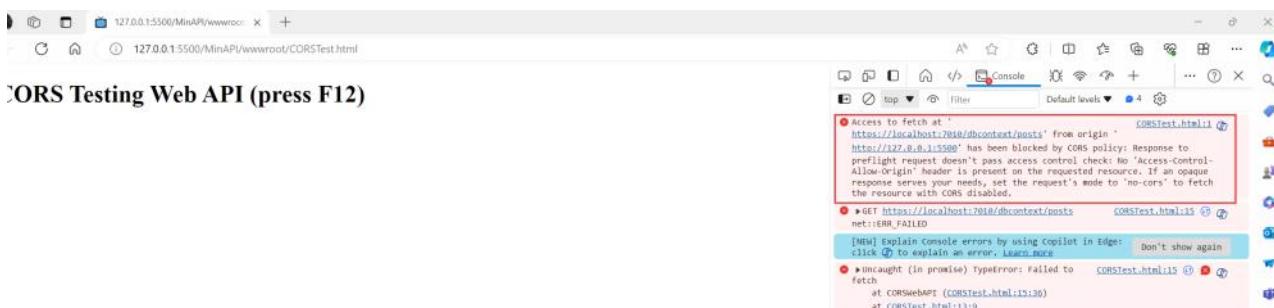
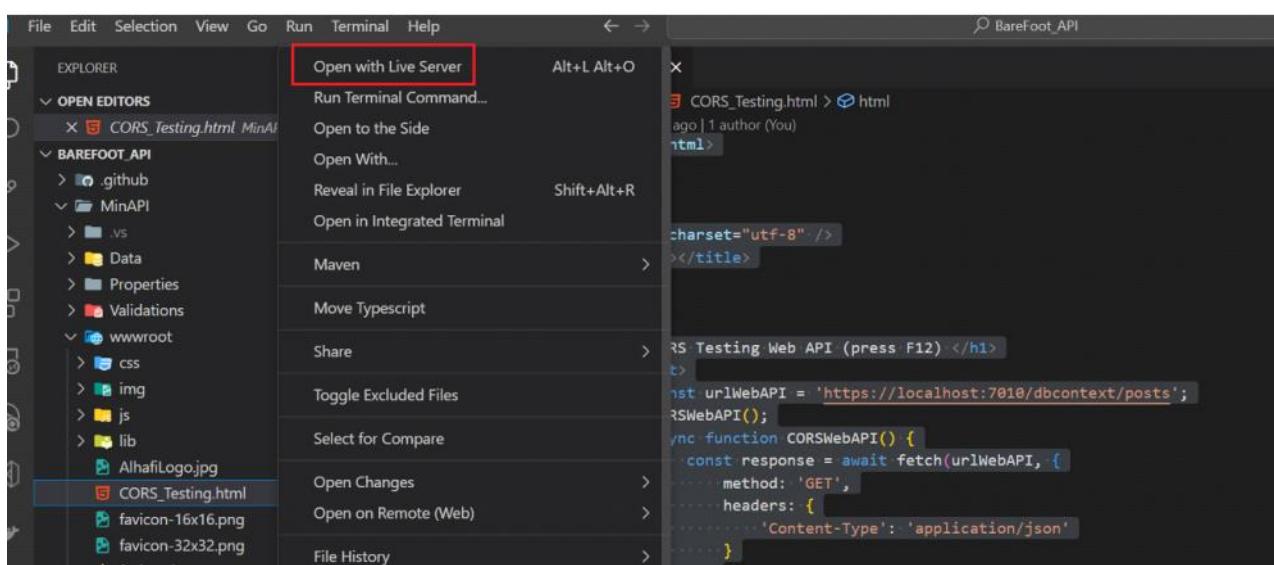
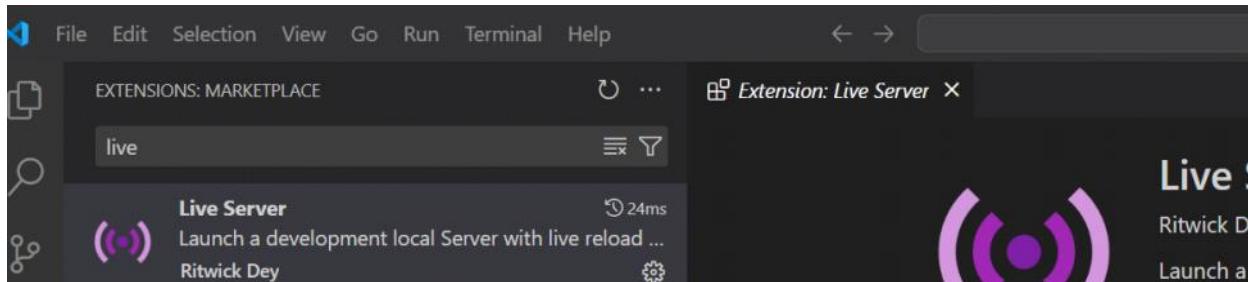
```

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 *****
```

```
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 two tabs: "Swagger UI" and "127.0.0.1:5500/MinAPI/wwwroot/CORS_Testing.html". The second tab displays the results of a POST request to "/api/posts". The response body is an array of 8 objects, each representing a post with fields: id, title, content, and postImage. All fields are set to null or empty strings. The browser's developer tools are open, specifically the "Console" tab, which shows the JSON data being logged.

```
CORS_Testing.html:22
Array(8)
▶ 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

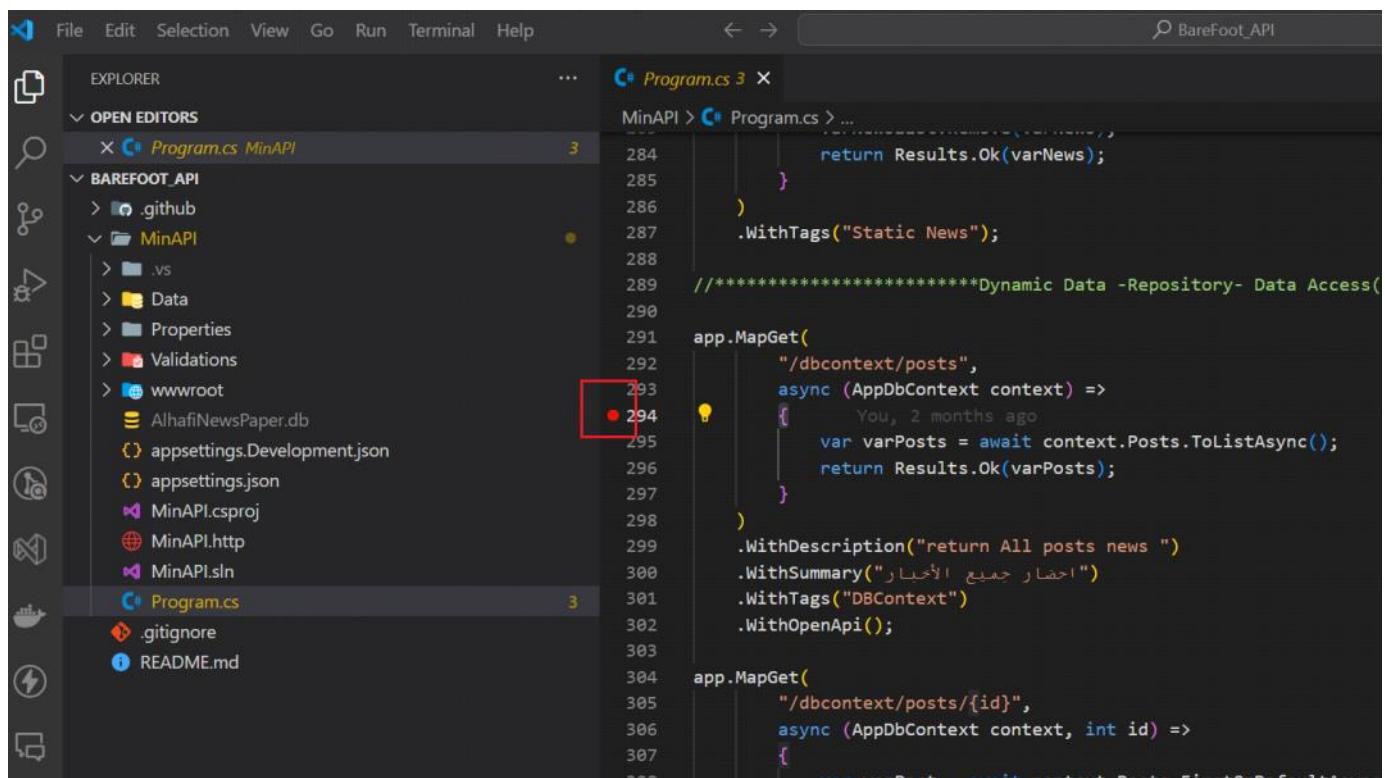
Cache

- 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. A red box highlights a break point at line 294, which contains the code 'app.MapGet("/dbcontext/posts", async (AppDbContext context) =>'. The code also includes comments in Arabic and English.

```
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 الأختبارات")
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         if (varPost != null)
310             return Results.Ok(varPost);
311         else
312             return Results.NotFound();
313     }
314     .WithDescription("return post by id")
315     .WithSummary("أختبار جميع الأختبارات")
316     .WithTags("DBContext")
317     .WithOpenApi();
318 }
```

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     {

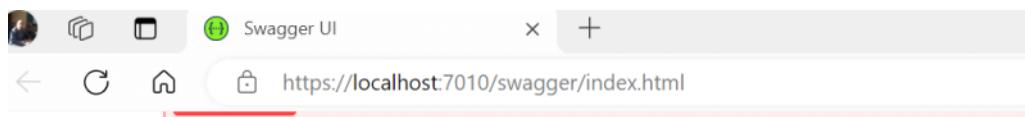
```

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

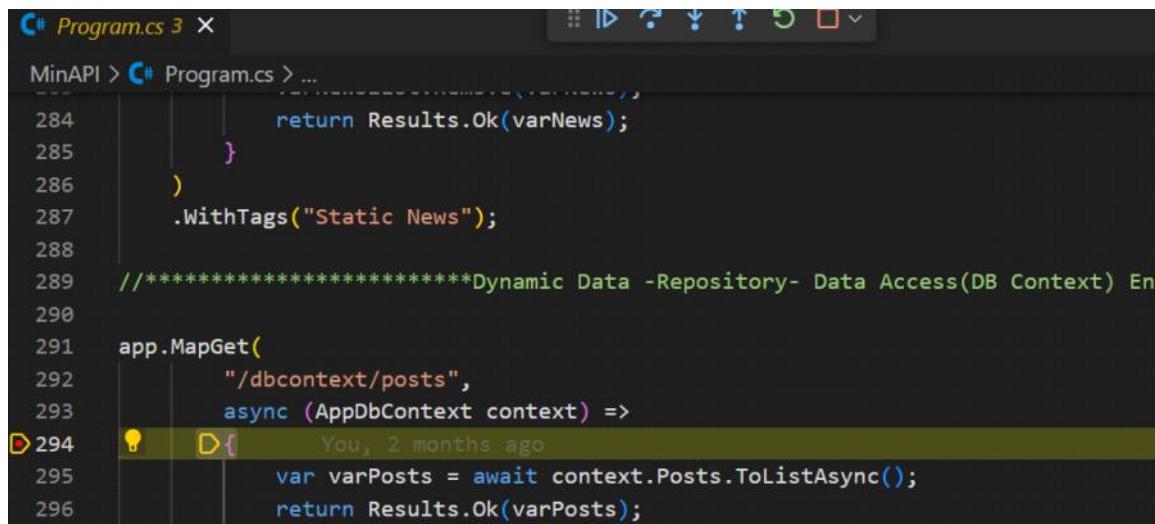
Parameters

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 code editor showing a tooltip for the method `TimeSpan.FromSeconds(15)`. The tooltip lists various static methods for creating `TimeSpan` objects based on different time units: FromDays, FromHours, FromMicroseconds, FromMilliseconds, FromMinutes, FromSeconds, and FromTicks. The tooltip also includes the current user information: "You, 17 hours ago • Uncommitted changes".

```
    )
    .WithDescription("return All posts news ")
    .WithSummary("إحضار جميع الأخبار")
    .WithTags("DbContext")
    .CacheOutput(c=>c.Expire(TimeSpan.FromSeconds(15))) You, 17 hours ago • Uncommitted changes
    .WithOpenApi();

app.MapGet(
    "/dbcontext/posts/{id}",
    async (AppDbContext context, int id) =>
```

A screenshot of a code editor showing a tooltip for the method `TimeSpan.FromSeconds(15)` in a different part of the code. The tooltip lists the same set of static methods for creating `TimeSpan` objects. The tooltip also includes the current user information: "You, 2 months ago".

```
286         varNewslist.Remove(varNews);
287         return Results.Ok(varNews);
288     }
289 }
290 .WithTags("Static News");
291
292 //*****Dynamic Data -Repository- Data Access(DB Context) EndPoint*****
293
294 app.MapGet(
295     "/dbcontext/posts",
296     async (AppDbContext context) =>
297     {
298         var varPosts = await context.Posts.ToListAsync();
299         return Results.Ok(varPosts);
300     }
301 )
302 .WithDescription("return All posts news ")
303 .WithSummary("إحضار جميع الأخبار")
304 .WithTags("DbContext")
305 .WithOpenApi()
306 .CacheOutput(c=>c.Expire(TimeSpan.FromSeconds(15)));
307 You, 2 months ago
308 app.MapGet(
309     "/dbcontext/posts/{id}",
```

```
.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 .WithTags("Hello");  
171  
172  
173  
174
```

Even we can handle extended such as tag

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

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

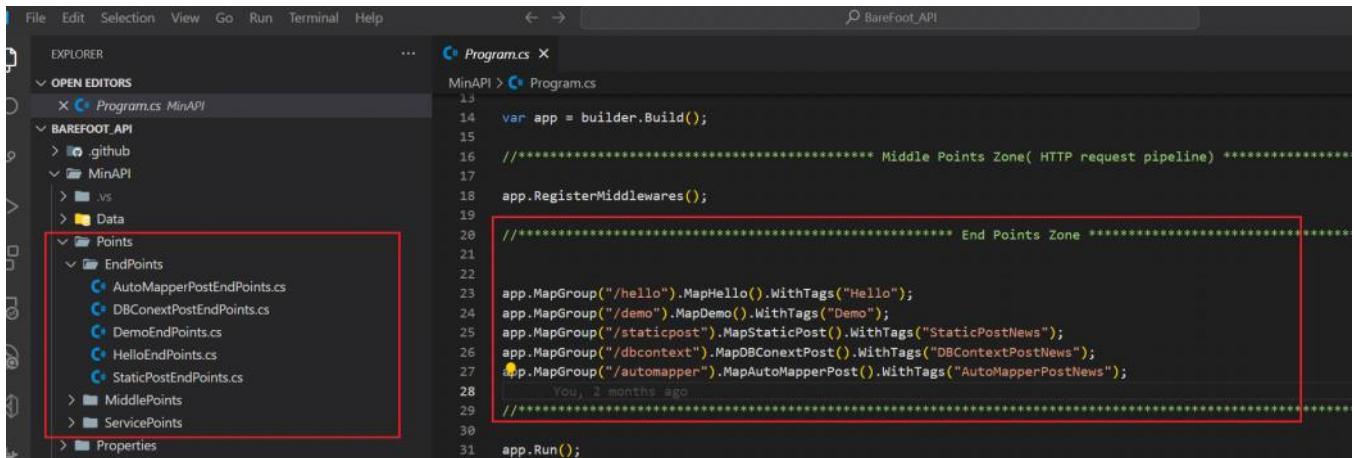
//*****Static Sample Hello*****


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

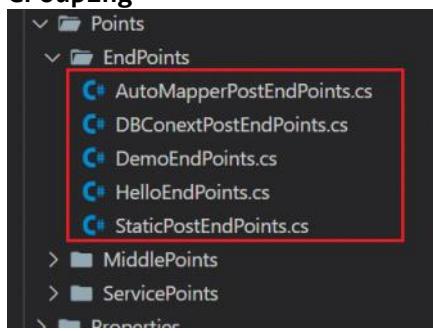
13 May 2024 13:12



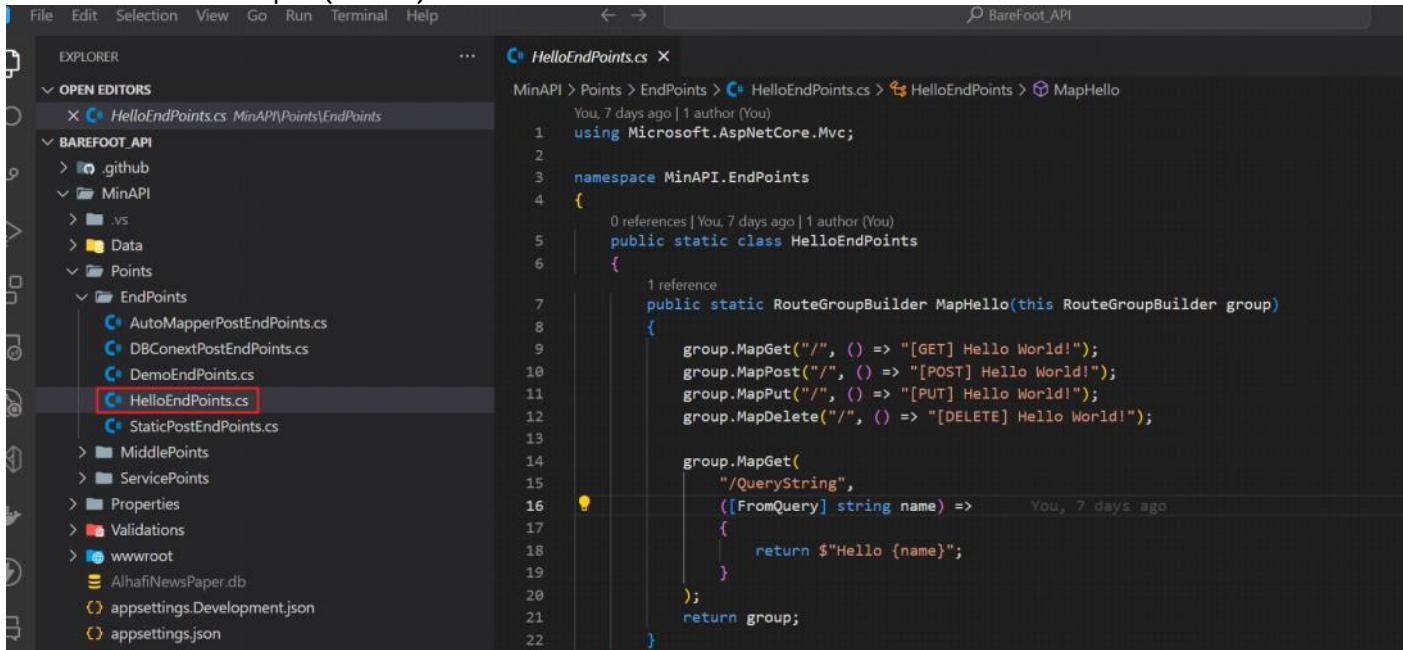
The screenshot shows the Visual Studio interface. On the left is the Explorer pane, which lists the project structure. A red box highlights the 'EndPoints' folder under 'Points'. Inside 'EndPoints', there are five files: AutoMapperPostEndPoints.cs, DBConextPostEndPoints.cs, DemoEndPoints.cs, HelloEndPoints.cs, 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 being 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. On the left is the Explorer pane, which lists the project structure. A red box highlights the 'HelloEndPoints.cs' file under 'Points\EndPoints'. On the right is the code editor showing the 'HelloEndPoints.cs' file. The code defines a static class 'HelloEndPoints' with a single static method 'MapHello' that maps various HTTP verbs to a 'Hello World!' response.

```
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 Program.cs
MinAPI > C# Program.cs
8 //***** Services Zone *****
9
10 builder.RegisterServices();
11
12 //***** Middle Points Zone( HTTP request pipeline ) *****
13
14 var app = builder.Build();
15
16 //***** End Points Zone *****
17
18 app.RegisterMiddlewares();
19
20
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 You, 2 months ago
29
30
31 app.Run();
32
PROBLEMS OUTPUT DEBUG CONSOLE PORTS GITLENS NUGET COMMENTS TERMINAL
Now listening on: https://localhost:7010
***** In Program.cs
*****
```

```

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

We Will do as sample(Demo)

```

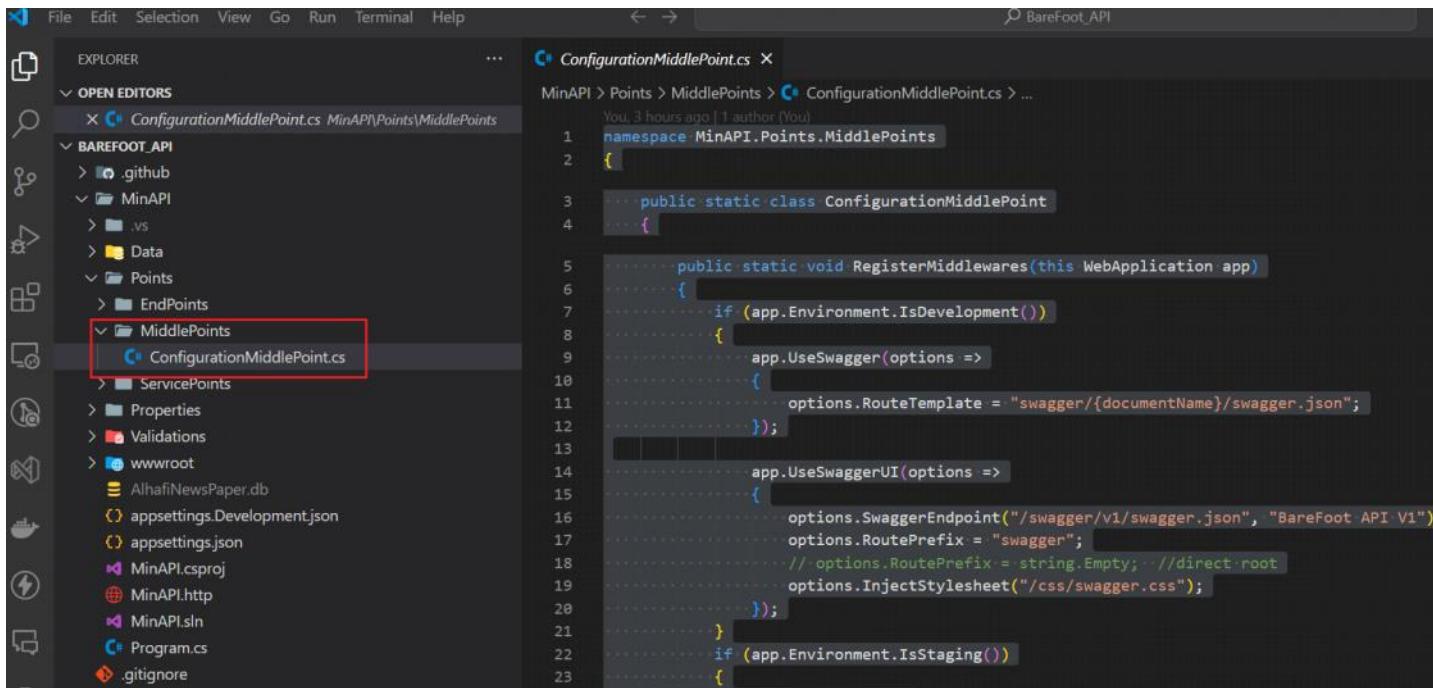
File Edit Selection View Go Run Terminal Help BareFoot_API
EXPLORER OPEN EDITORS Program.cs
MinAPI > C# Program.cs
8 //***** Services Zone *****
9
10 builder.RegisterServices();
11
12 //***** Middle Points Zone( HTTP request pipeline ) *****
13
14 var app = builder.Build();
15
16 //***** End Points Zone *****
17
18 app.RegisterMiddlewares();
19
20
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 You, 2 months ago
29
30
31 app.Run();
32

```

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

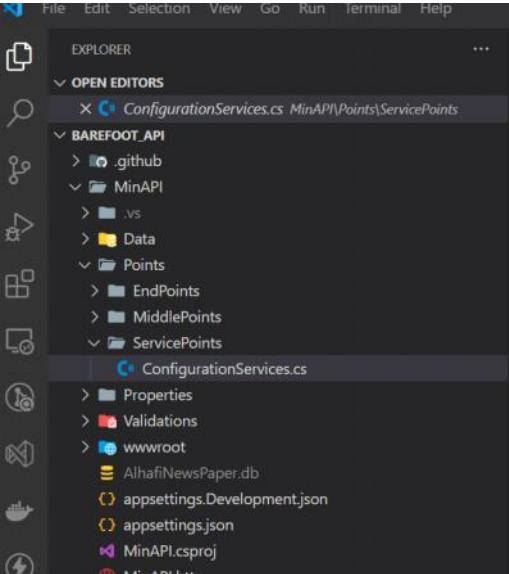
The screenshot shows a code editor window with the file `Program.cs` open. The code is written in C# and defines a `Program` class that inherits from `WebApplication`. The code is organized into several sections:

- Services Zone:** Contains the line `builder.RegisterServices();`
- Middle Points Zone (HTTP request pipeline):** Contains the line `app.RegisterMiddlewares();`, which is highlighted with a red rectangular selection.
- End Points Zone:** Contains route mappings for various endpoints like `/hello`, `/demo`, etc., using `MapGroup` and `Map` methods.

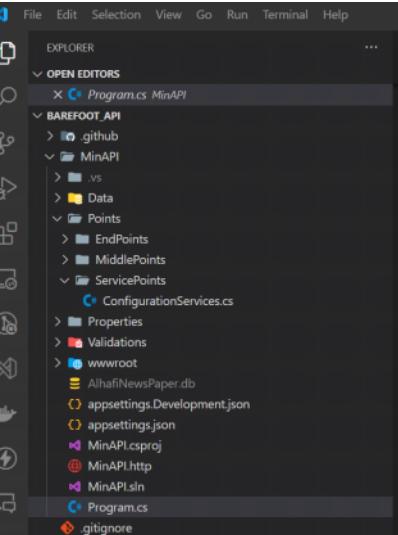
```
MinAPI > Program.cs
  5
  6 var builder = WebApplication.CreateBuilder(args);
  7
  8 //***** Services Zone *****
  9
 10 builder.RegisterServices();
 11
 12 //***** Middle Points Zone( HTTP request pipeline) *****
 13
 14 var app = builder.Build();
 15
 16 //***** End Points Zone *****
 17
 18 app.RegisterMiddlewares();
 19
 20 //*****
 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 //*****
 30
 31
 32
 33
 34
```

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



Mastering-
Minimal-...

[Create WebAPI with GitHub Copilot and .NET Core 8 | Development with GitHub Copilot](#)



[Develop a .NET API with GitHub Copilot in SECONDS!](#)



[Tutorial: Create a minimal API with ASP.NET Core | Microsoft Learn](#)

<https://github.com/sswietoniowski/learning-aspnetcore-webapi-7-building-minimal-apis>

<https://medium.com/@fahrican.kcn/building-your-first-minimal-api-with-net-7-f4864495ea8a>

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

<https://benfoster.io/blog/mvc-to-minimal-apis-aspnet-6/#model-binding>

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

<https://benfoster.io/blog/minimal-apis-custom-model-binding-aspnet-6/>

<https://learn.microsoft.com/en-us/answers/questions/1348193/jsonpatchdocument-problems-with-minimal-api-in-net>

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

<https://learn.microsoft.com/en-us/aspnet/core/controllers/dependency-injection?view=aspnetcore-8.0>

<https://inwedo.com/blog/when-not-use-json-patch-in-asp-net-core/>

<https://medium.com/@kasunzerothvision/implementing-dependency-injection-using-c-in-net-6-0-4071a9eee8bc>

<https://github.com/dotnet/AspNetCore.Docs/blob/main/aspnetcore/web-api/index/samples/7.x/ApiController/SystemDateTime.cs>

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

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

Increase Your Query Performance Up to 4 Times in EF Core with AsNoTracking

<https://medium.com/@m.mobasher.z/a-deep-dive-into-entity-framework-8-asnotrackingwithidentityresolution-vs-asnotracking-54faf454f332>

<https://www.devtrends.co.uk/blog/avoid-asnotracking-and-include-when-querying-using-entity-framework-in-asp.net>

<https://medium.com/@satyaprakashsahoo.blog/what-is-the-use-of-asnotracking-in-entity-framework-cb6c453d1f1>

<https://www.linkedin.com/pulse/increase-your-query-performance-up-4-times-ef-core-karan-pargaien-6zqkf>

Fluent API

<https://dev.to/fabriziobagala/fluentvalidation-in-net-2hg7>

[DevOps In 5 Minutes | What Is DevOps? | DevOps Explained | DevOps Tutorial For Beginners | Simplilearn](#)



[The Complete DevOps Roadmap \[2024\]](#)



Introduction

Monday, June 21, 2021 9:13 PM

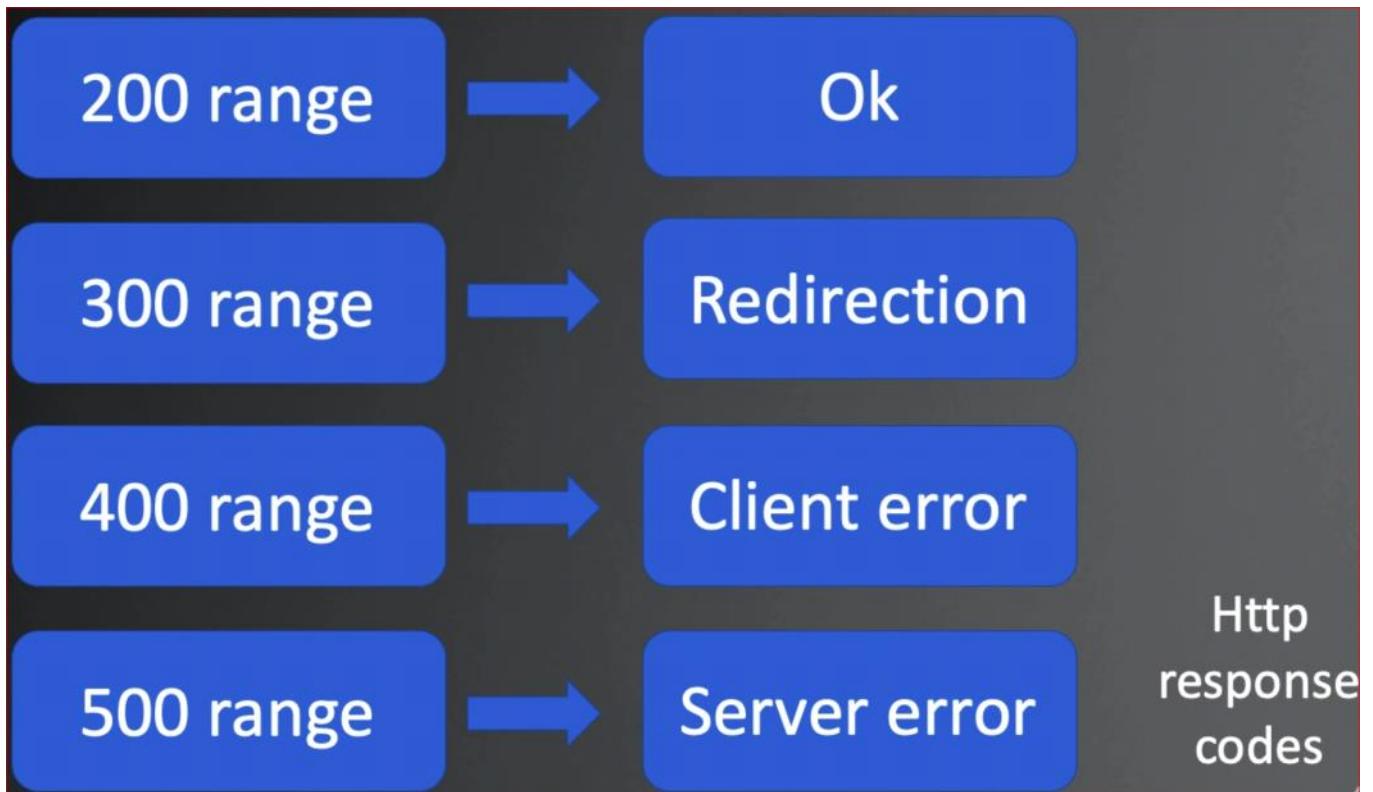
Here

[Global Error Handling in C# Minimal APIs](#)

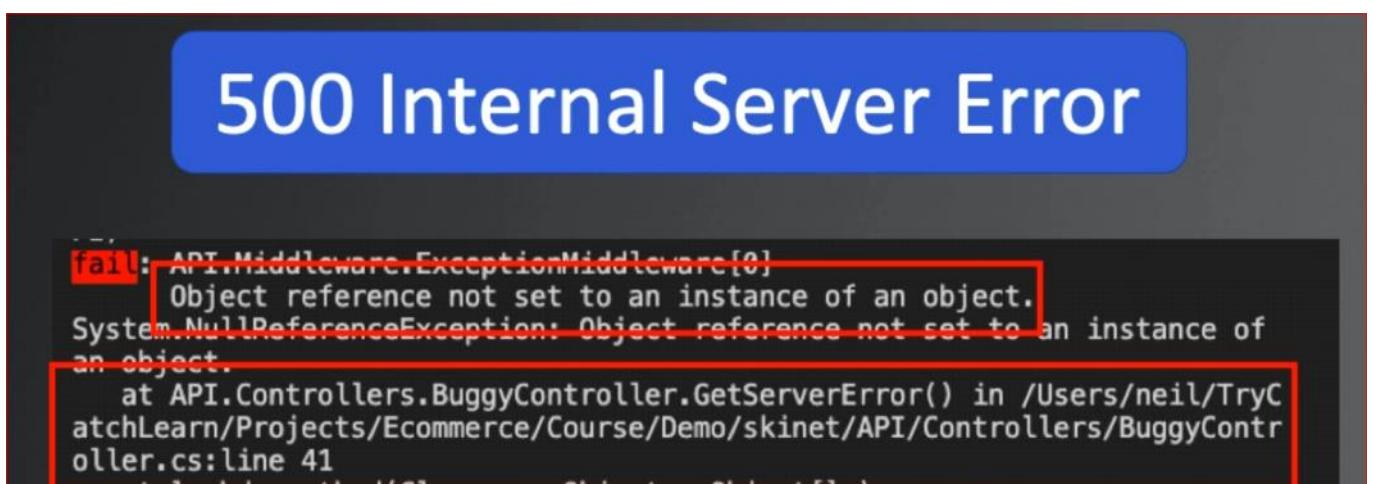


- Error handling and exceptions
- Developer exception page
- Validation errors
- Http response errors
- Customising the error handling
- Middleware
- Swagger

In this
module



Sampling



Let We removed Unnecessary controller used for training prupose by comments

```
C# WeatherForecastController.cs M X
Api > Controllers > C# WeatherForecastController.cs > {} Api.Controllers
You, 35 seconds ago | 1 author (You)
1 using Microsoft.AspNetCore.Mvc;
2
3 You, 35 seconds ago | 1 author (You)
4 namespace Api.Controllers;
5
6 // [ApiController]
7 // [Route("[controller]")]
8 public class WeatherForecastController : ControllerBase
9 {
10     private static readonly string[] Summaries = new[]
11     {
12         "Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering", "Scorching"
13     };
}
```

And all methods

```
// [HttpGet]
0 references
public async Task<ActionResult<List<Product>>> getProductsContextAsync(){

    var products=await _repo.GetProductsAsync();
    return Ok(products);
}

// [HttpGet("{id}")]
0 references
public async Task<ActionResult<Product>> getProductContextAsync(int id){
    return await _repo.GetProductByIDAsync(id);
}
```

We kept only ProductController

Product

GET /api/Product/{id}

GET /api/Product

GET /api/Product/brands

GET /api/Product/types

Schemas

[ProductBrand >](#)

[ProductToReturnDto >](#)

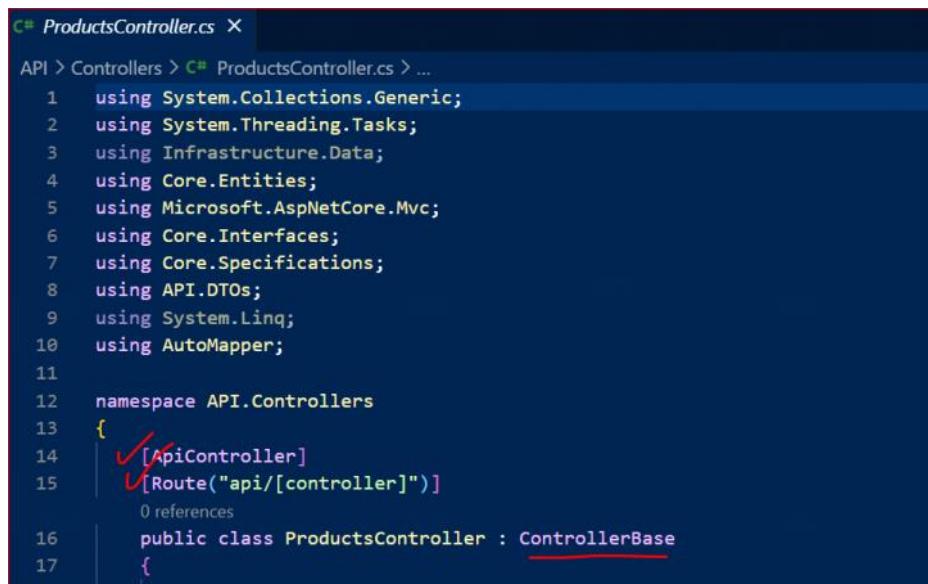
[ProductType >](#)

Creating a test controller for errors

22 June 2021 11:38

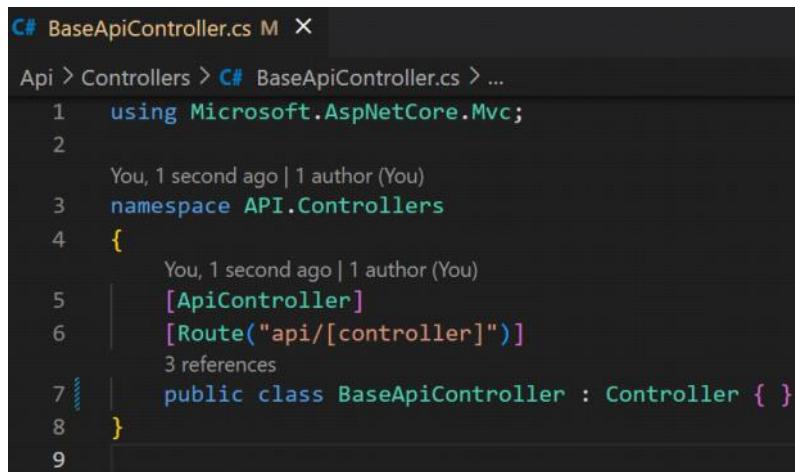
Here what are we going to do ,is we are going to test things up , so we can easily test our ever responses that we send back from our server and what we will do to achieve this

first of all Create new controller acting as base controller for API controller or normal controller , for example in **ProductController** as API controller



```
C# ProductsController.cs X
API > Controllers > C# ProductsController.cs > ...
1  using System.Collections.Generic;
2  using System.Threading.Tasks;
3  using Infrastructure.Data;
4  using Core.Entities;
5  using Microsoft.AspNetCore.Mvc;
6  using Core.Interfaces;
7  using Core.Specifications;
8  using API.DTOs;
9  using System.Linq;
10 using AutoMapper;
11
12 namespace API.Controllers
13 {
14     [ApiController]
15     [Route("api/[controller]")]
16     public class ProductsController : ControllerBase
17     {
```

Instead I derive from **ControllerBase** and repeat **[ApiController]** and **[Route("api/[Controller]")]** for each controller I will create in the future it better to create new controller acting as base controller we will name assume= **BaseApiController**

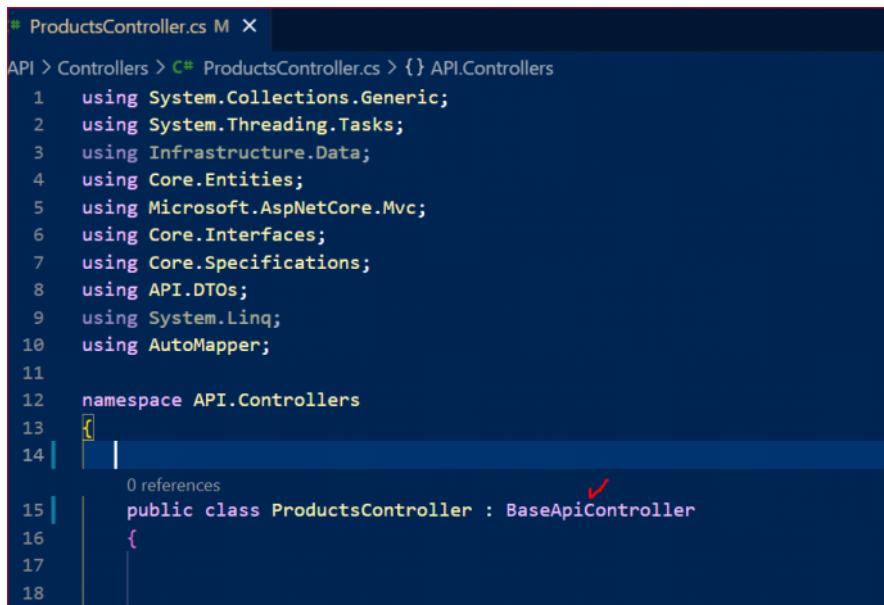


```
C# BaseApiController.cs M X
Api > Controllers > C# BaseApiController.cs > ...
1  using Microsoft.AspNetCore.Mvc;
2
3  You, 1 second ago | 1 author (You)
4  namespace API.Controllers
5  {
5    You, 1 second ago | 1 author (You)
6    [ApiController]
6    [Route("api/[controller]")]
6    3 references
7    public class BaseApiController : ControllerBase { }
8  }
9
```

```
using Microsoft.AspNetCore.Mvc;
namespace API.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class BaseApiController : ControllerBase { }
```

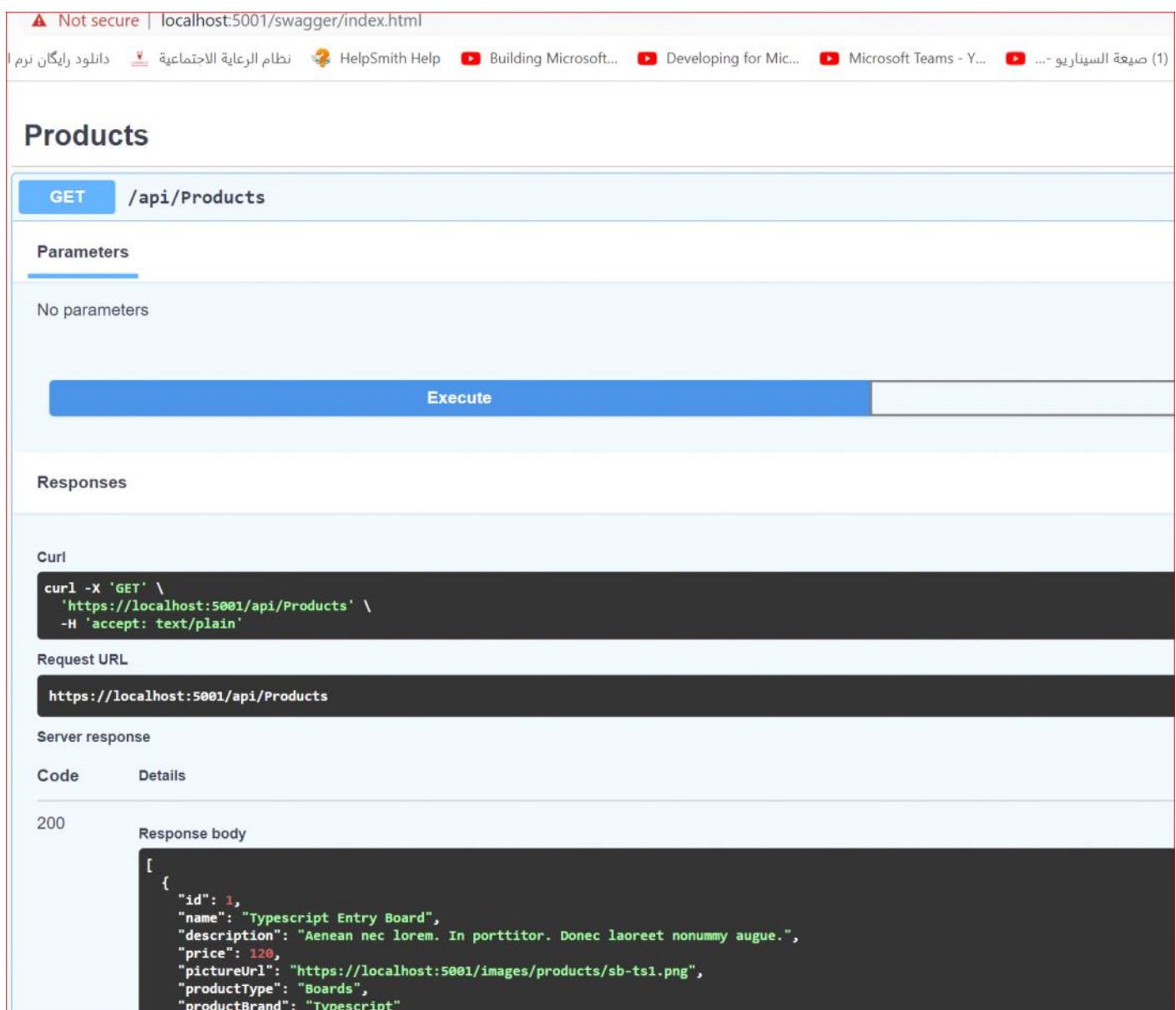
back to **ProductController** and derive for **BaseApiController** instead of **ControllerBase** and remove

[ApiController] and [Route("api/[Controller]")]



```
# ProductsController.cs M X
API > Controllers > C# ProductsController.cs > {} API.Controllers
1  using System.Collections.Generic;
2  using System.Threading.Tasks;
3  using Infrastructure.Data;
4  using Core.Entities;
5  using Microsoft.AspNetCore.Mvc;
6  using Core.Interfaces;
7  using Core.Specifications;
8  using API.DTOs;
9  using System.Linq;
10 using AutoMapper;
11
12 namespace API.Controllers
13 {
14     public class ProductsController : BaseApiController
15     {
16     }
17 }
18
```

Let we test this **dotnet watch run**



⚠ Not secure | localhost:5001/swagger/index.html

دانلود رایگان نرم افزار نظام الرعایة الاجتماعیة HelpSmith Help Building Microsoft... Developing for Microsoft... Microsoft Teams - YouTube ... (1) صيغة السيناريو -

Products

GET /api/Products

Parameters

No parameters

Execute

Responses

Curl

```
curl -X 'GET' \
'https://localhost:5001/api/Products' \
-H 'accept: text/plain'
```

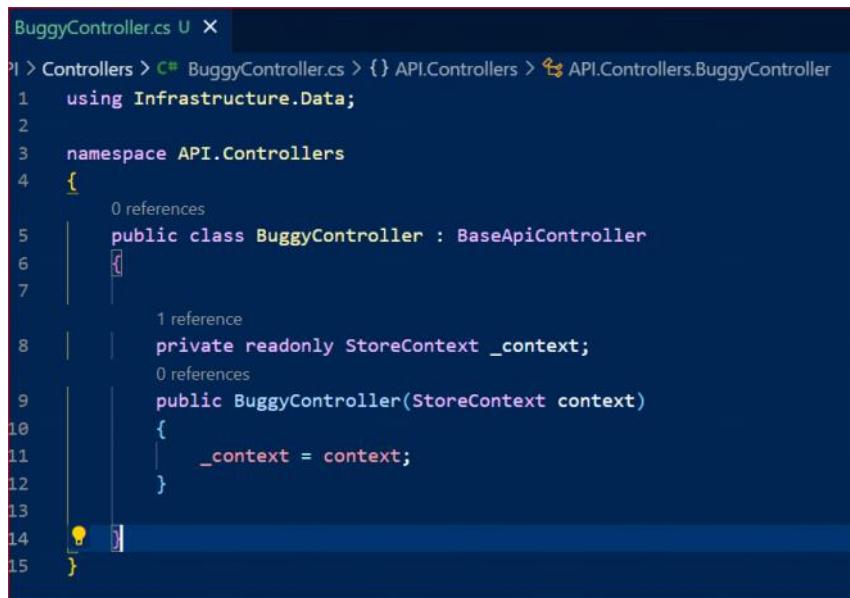
Request URL

```
https://localhost:5001/api/Products
```

Server response

Code	Details
200	Response body <pre>[{ "id": 1, "name": "TypeScript Entry Board", "description": "Aenean nec lorem. In porttitor. Donec laoreet nonummy augue.", "price": 120, "pictureUrl": "https://localhost:5001/images/products/sb-ts1.png", "productType": "Boards", "productBrand": "TypeScript"</pre>

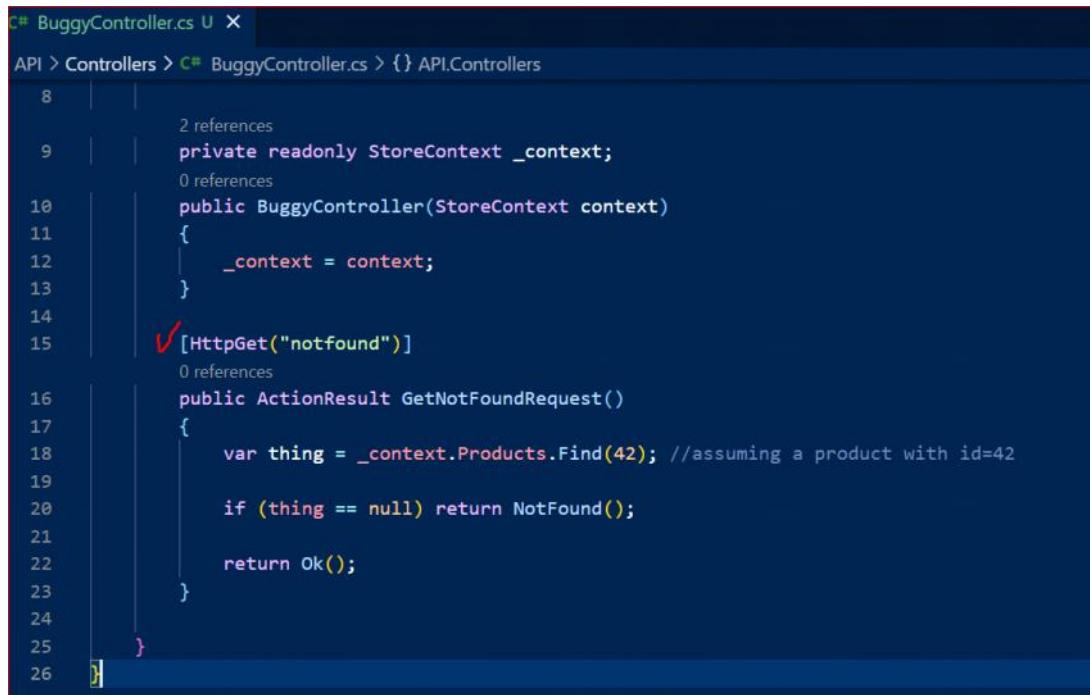
It's working fine, let's now create a controller to set up some errors to see different kinds of responses, let's name it **BuggyController**



```
BuggyController.cs U X
API > Controllers > C# BuggyController.cs > {} API.Controllers > API.Controllers.BuggyController
1  using Infrastructure.Data;
2
3  namespace API.Controllers
4  {
5      0 references
6      public class BuggyController : BaseApiController
7      {
8          1 reference
9          private readonly StoreContext _context;
10         0 references
11         public BuggyController(StoreContext context)
12         {
13             _context = context;
14         }
15     }
}
```

Now let's add error types and responses

Not found



```
C# BuggyController.cs U X
API > Controllers > C# BuggyController.cs > {} API.Controllers
8
9      2 references
10     private readonly StoreContext _context;
11     0 references
12     public BuggyController(StoreContext context)
13     {
14         _context = context;
15     }
16
17     [HttpGet("notfound")]
18     0 references
19     public ActionResult GetNotFoundRequest()
20     {
21         var thing = _context.Products.Find(42); //assuming a product with id=42
22
23         if (thing == null) return NotFound();
24
25         return Ok();
26     }
}
```

Server Error

```
[HttpGet("servererror")]
0 references
public ActionResult GetServerError()
{
    var thing = _context.Products.Find(42);

    var thingToReturn = thing.ToString(); //In case null , we will get error because I can't convert null to string

    return Ok();
}
```

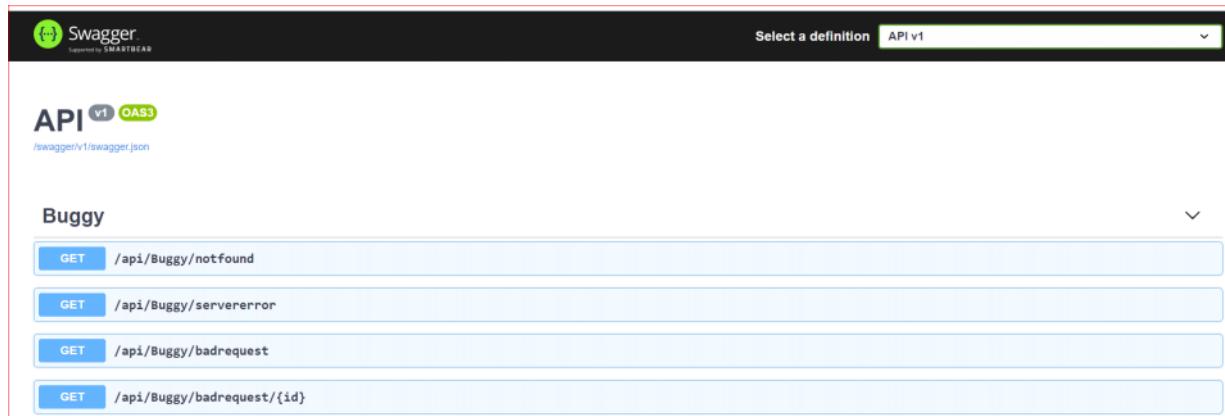
Bad request

```
[HttpGet("badrequest")]
0 references
public ActionResult GetBadRequest()
{
    return BadRequest(); //this will return 400 bad request
}
```

Bad request with validation

```
public ActionResult GetNotFoundRequest(int id) //simply we get error if we pass string not INT
{
    return Ok();
}
```

Now let's test the controller **dotnet watch run**



The screenshot shows the Swagger UI interface for a .NET Core API. At the top, there is a navigation bar with the 'Swagger' logo, a dropdown menu 'Select a definition' set to 'API v1', and a dropdown menu for selecting an OAS version (v1 or OAS3). Below the header, the word 'API' is displayed with a 'v1' badge and an 'OAS3' badge. A link to '/swagger/v1/swagger.json' is shown. The main content area is titled 'Buggy'. It lists four GET requests:

- GET /api/Buggy/notfound
- GET /api/Buggy/servererror
- GET /api/Buggy/badrequest
- GET /api/Buggy/badrequest/{id}

Buggy

GET /api/Buggy/notfound

Parameters

No parameters

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
  'https://localhost:5001/api/Buggy/notfound' \
  -H 'accept: */*'
```

Request URL

<https://localhost:5001/api/Buggy/notfound>

Server response

Code Details

404

Undocumented Error:

Response body

```
{
  "type": "https://tools.ietf.org/html/rfc7231#section-6.5.4",
  "title": "Not Found",
  "status": 404,
  "traceId": "00-42c0b0bcf05e741acd0fb786e8798a2-1089d148dd357c40-00"
}
```



Download

GET /api/Buggy/servererror

Parameters

No parameters

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
  'https://localhost:5001/api/Buggy/servererror' \
  -H 'accept: */*'
```

Request URL

<https://localhost:5001/api/Buggy/servererror>

Server response

Code Details

500

Undocumented Error:

Response body

```
System.NullReferenceException: Object reference not set to an instance of an object.
  at API.Controllers.BuggyController.GetServerError() in C:\Dev_011\VisualCode\skinet\api\Controllers\BuggyController.cs:line 31
  at lambda_method03(Closure , Object , Object[])
  at Microsoft.AspNetCore.Mvc.Infrastructure.ActionMethodExecutor.SynchronousActionResultExecutor.Execute(IActionResultTypeMapper mapper, ObjectMethodExecutor executor, Object controller, Object[] arguments)
  at Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.InvokeActionMethodAsync()
  at Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.Next(State& next, Scope& scope, Object& state, Boolean& isCompleted)
  at Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.TraverseNextActionFilterAsync()
```

GET /api/Buggy/badrequest

Parameters

No parameters

Execute Clear

Responses

Curl

```
curl -X 'GET' \
'https://localhost:5001/api/Buggy/badrequest' \
-H 'accept: */*'
```

Request URL

<https://localhost:5001/api/Buggy/badrequest>

Server response

Code Details

400 Undocumented Error:

Response body

```
{
  "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",
  "title": "Bad Request",
  "status": 400,
  "traceId": "00-15ab7591b0fa5c49bf517a7ce7583a95-ecd00bf3fc162141-00"
}
```

Download

GET /api/Buggy/badrequest/{id}

Parameters

Name	Description
id <small>required</small>	1
integer(\$int32)	(path)

Execute

Responses

Curl

```
curl -X 'GET' \
'https://localhost:5001/api/Buggy/badrequest/1' \
-H 'accept: */*'
```

Request URL

<https://localhost:5001/api/Buggy/badrequest/1>

Server response

Code Details

200 Response headers

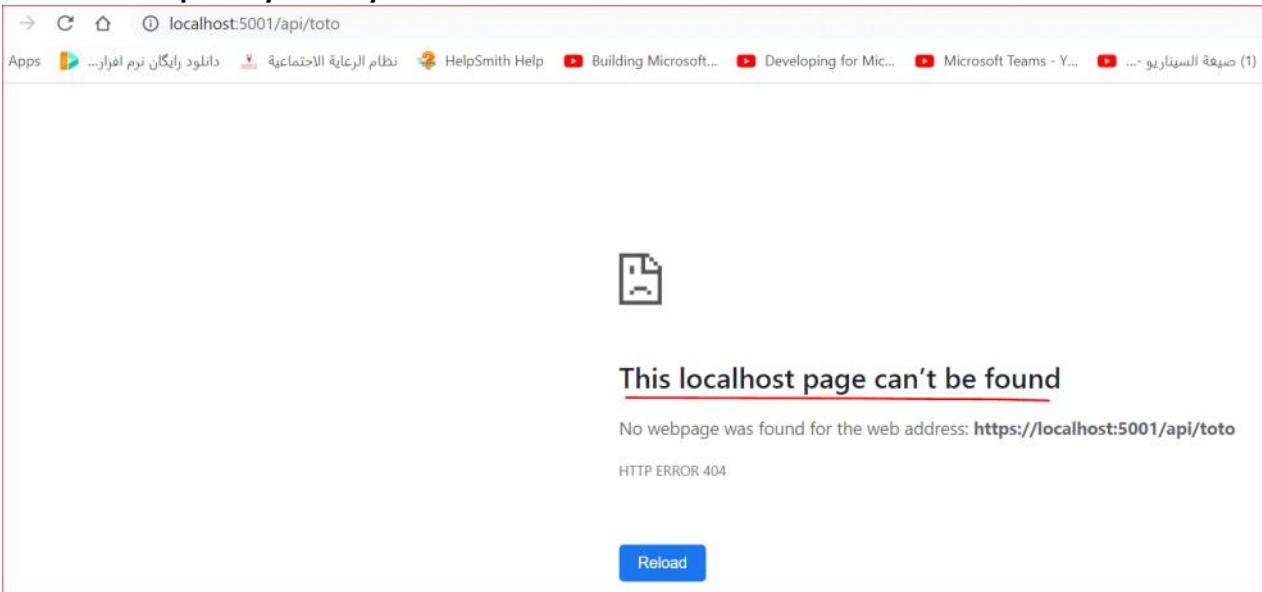
```
content-length: 0
date: Tue,22 Jun 2021 09:44:16 GMT
server: Kestrel
```



```
Not secure | localhost:5001/api/Buggy/badrequest/400
Apps نظام الرعاية الاجتماعية دانلود رایگان نرم افزار... HelpSmith Help Building
```

```
{ "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1", "title": "One or more validation errors occurred.", "status": 400, "traceId": "00-f7470c3e7ddebe46b54394b24a5f24cd-a1d2424aa31a9942-00", "errors": { "id": [ "The value 'A' is not valid." ] } }
```

Assume if we put any dummy name such as below



Creating a consistent error response from the API

22 June 2021 13:13

Here we are going to make error responses consistent for front end application such as angular to handle error responses , So let we create folder naming **Errors** under **API** folder then create a class naming **ApiResponse** and create 2 properties as below

```
C# ApiResponse.cs U ●
API > Errors > C# ApiResponse.cs > {} API.Errors
1  namespace API.Errors
2  {
3      0 references
4      public class ApiResponse
5      {
6          0 references
7          public int StatusCode { get; set; }
8          0 references
9          public string Message { get; set; }
10     }
11 }
```

Generate constructor

```
ApiResponse.cs U ●
I > Errors > C# ApiResponse.cs > {} API.Errors > API.Errors.ApiResponse
1  namespace API.Errors
2  {
3      0 references
4      public class ApiResponse
5      {
6          0 references
7          public ApiResponse(int statusCode, string message)
8          {
9              StatusCode = statusCode;
10             Message = message;
11         }
12
13         1 reference
14         public int StatusCode { get; set; }
15         1 reference
16         public string Message { get; set; }
17     }
18 }
```

Then we pass to null ,because we may not have a message attached to message code

```
public ApiResponse(int statusCode, string message=null)
{
```

Then check if message=null then return default code

```
> Errors > C# ApiResponse.cs > {} API.Errors > API.Errors.ApiResponse > API.Errors.ApiResponse(int statusCode, string message = null)
1  namespace API.Errors
2  {
3      0 references
4      public class ApiResponse
5      {
6          0 references
7          public ApiResponse(int statusCode, string message=null)
8          {
9              StatusCode = statusCode;
10             Message = message ?? GetDefaultMessageForStatusCode(statusCode);
11         }
12
13         1 reference
14         public int StatusCode { g Initialize ctor from properties...
15             Wrapping -> Wrap expression
16         1 reference
17         public string Message { g Generate method 'ApiResponse.GetDefaultMessageForStatusCode'
18             Wrapping -> Wrap and align expression
19     }
20 }
```

```
# ApiResponse.cs U ●
API > Errors > C# ApiResponse.cs > {} API.Errors
1  using System;
2
3  namespace API.Errors
4  {
5      0 references
6      public class ApiResponse
7      {
8          0 references
9          public ApiResponse(int statusCode, string message=null)
10         {
11             StatusCode = statusCode;
12             Message = message ?? GetDefaultMessageForStatusCode(statusCode);
13         }
14
15         1 reference
16         private string GetDefaultMessageForStatusCode(int statusCode)
17         {
18             throw new NotImplementedException();
19         }
20
21     }
22 }
```

```
ApiResponse.cs U ●
API > Errors > C# ApiResponse.cs > {} API.Errors > ↗ API.Errors.ApiResponse > ⓘ GetDefaultMessageForStatusCode(int statusCode)
1  using System;
2
3  namespace API.Errors
4  {
5      0 references
6      public class ApiResponse
7      {
8          0 references
9          public ApiResponse(int statusCode, string message=null)
10         {
11             StatusCode = statusCode;
12             Message = message ?? GetDefaultMessageForStatusCode(statusCode);
13         }
14
15         1 reference
16         public int StatusCode { get; set; }
17         1 reference
18         public string Message { get; set; }
19
20         1 reference
21         private string GetDefaultMessageForStatusCode(int statusCode)
22         {
23             return statusCode switch
24             {
25                 400 => "A bad request, you have made",
26                 401 => "Authorized, you are not",
27                 404 => "Resource found, it was not",
28                 500 => "Errors are the path to the dark side. Errors lead to anger. Anger leads to hate. Hate leads to career change."
29                 _ => null // mean default case
30             };
31         }
32     }
33 }
```

Let we implement call 400,404 in our **BuggyController** Just for

```
private string GetDefaultMessageForStatusCode(int statusCode)
{
    return statusCode switch
    [
        400 => "A bad request, you have made",
        401 => "Authorized, you are not",
        404 => "Resource found, it was not",
        500 => "Errors are the path to the dark side. Errors lead to anger. Anger leads to hate. Hate leads to career change."
        _ => null // mean default case
    ];
}
```

In BuggyController

```
[HttpGet("notfound")]
0 references
public ActionResult GetNotFoundRequest()
{
    var thing = _context.Products.Find(42); //assuming a product with id=42

    if (thing == null) return NotFound(new ApiResponse(404));

    return Ok();
}
```

```
[HttpGet("badrequest")]
0 references
public ActionResult GetBadRequest()
{
    return BadRequest(new ApiResponse(400)); //this will return 400 bad request
}
```

```
using API.Errors;
using Base.Contexts;
using Microsoft.AspNetCore.Mvc;
namespace API.Controllers
{
    public class BuggyController : BaseApiController
    {
        private readonly StoreContext _context;
        public BuggyController(StoreContext context)
        {
            _context = context;
        }
        [HttpGet("notfound")]
        public ActionResult GetNotFoundRequest()
        {
            var thing = _context.Products.Find(42); //assuming a product with id=42
            if (thing == null) return NotFound(new ApiResponse(404));
            return Ok();
        }

        [HttpGet("servererror")]
        public ActionResult GetServerError()
        {
            var thing = _context.Products.Find(42);
            var thingToString = thing.ToString(); //In case null , we will get error because I cant convert null to string
            return Ok();
        }

        [HttpGet("badrequest")]
        public ActionResult GetBadRequest()
        {
            return BadRequest(new ApiResponse(400)); //this will return 400 bad request
        }
        [HttpGet("badrequest/{id}")]
        public ActionResult GetNotFoundRequest(int id) //simply we got error if we pass string not INT
        {
            return Ok();
        }
    }
}
```

Now let's test this **dotnet watch run**

GET /api/Buggy/notfound

Parameters

No parameters

Execute **Clear**

Responses

Curl

```
curl -X 'GET' \
'https://localhost:5001/api/Buggy/notfound' \
-H 'accept: */*'
```

Request URL

```
https://localhost:5001/api/Buggy/notfound
```

Server response

Code **Details**

404
Undocumented Error:

Response body

```
{
  "statusCode": 404,
  "message": "Resource found, it was not"
}
```

Download

GET /api/Buggy/badrequest

Parameters

No parameters

Execute **Clear**

Responses

Curl

```
curl -X 'GET' \
'https://localhost:5001/api/Buggy/badrequest' \
-H 'accept: */*'
```

Request URL

```
https://localhost:5001/api/Buggy/badrequest
```

Server response

Code **Details**

400
Undocumented Error:

Response body

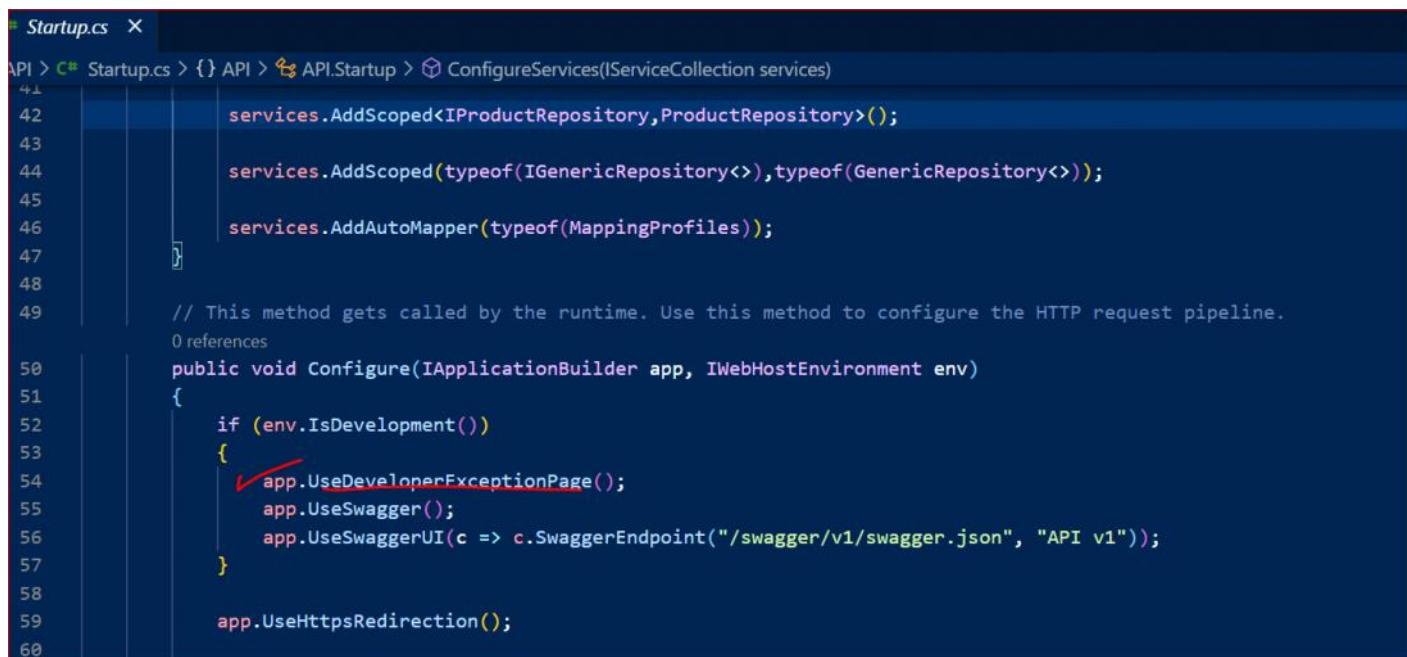
```
{
  "statusCode": 400,
  "message": "A bad request, you have made"
}
```

Download

Adding a not found endpoint error handler

22 June 2021 13:50

Where the default exception error handler ?

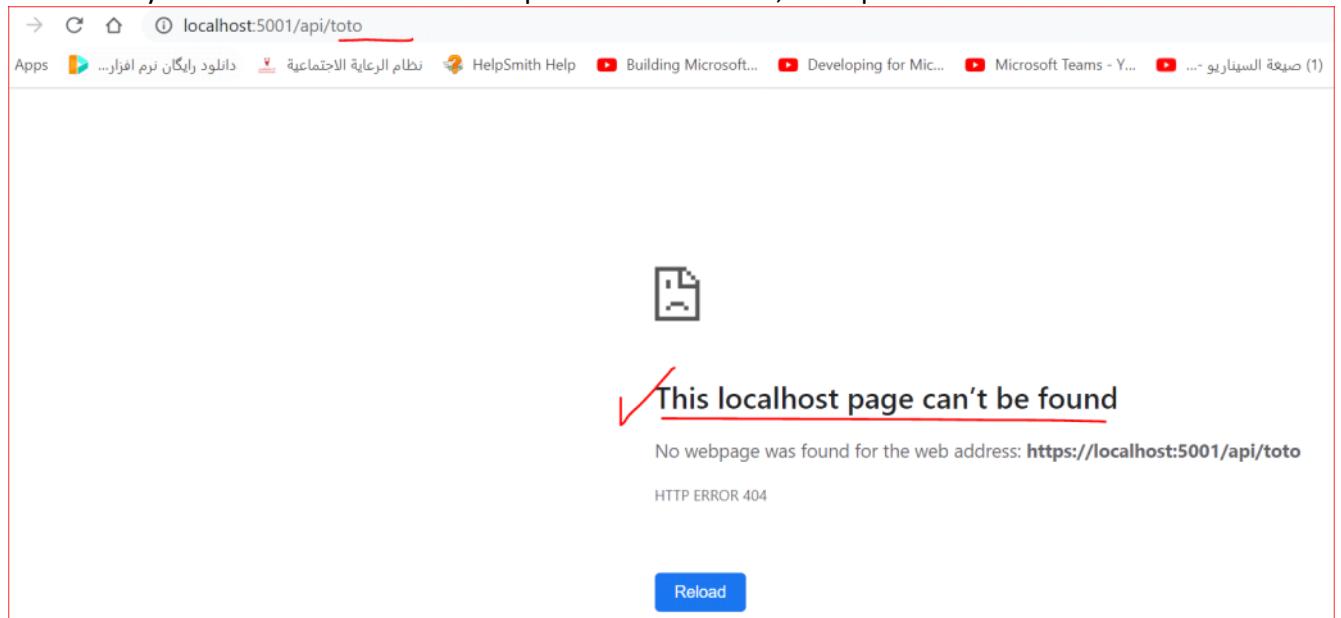


```
API > C# Startup.cs > {} API > API.Startup > ConfigureServices(IServiceCollection services)
41
42     services.AddScoped<IPrductRepository,ProductRepository>();
43
44     services.AddScoped(typeof(IGenericRepository<>),typeof(GenericRepository<>));
45
46     services.AddAutoMapper(typeof(MappingProfiles));
47
48
49     // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
50     0 references
51     public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
52     {
53         if (env.IsDevelopment())
54         {
55             app.UseDeveloperExceptionPage();
56             app.UseSwagger();
57             app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "API v1"));
58
59         }
60
61         app.UseHttpsRedirection();
62     }
63
64 }
```

Sampling

```
1 System.NullReferenceException: Object reference not set to an instance of an object.
2   at API.Controllers.BuggyController.GetServerError() in /Users/neil/TryCatchLearn/Projects/Ecommerce/Course/Demo/skinet/API/
   Controllers/BuggyController.cs:line 32
3   at lambda_method(Closure , Object , Object[] )
4   at Microsoft.Extensions.Internal.ObjectMethodExecutor.Execute(Object target, Object[] parameters)
5   at Microsoft.AspNetCore.Mvc.Infrastructure.ActionMethodExecutor.SyncActionResultExecutor.Execute(IActionResultTypeMapper
   mapper, ObjectMethodExecutor executor, Object controller, Object[] arguments)
```

Ok in our system till now don't have endpoint error handler , example as below



Let's create our endpoint error handler naming **ErrorController** which derives from **BaseApiController** , But we need to override the route

```
API > Controllers > C# ErrorController.cs > {} API.Controllers
1
2     using API.Errors;
3     using Microsoft.AspNetCore.Mvc;
4
5     namespace API.Controllers
6     {
7         [Route("errors/{code}")]
8             0 references
9             public class ErrorController : BaseApiController
10            {
11                0 references
12                public IActionResult Error(int code)
13                {
14                    return new ObjectResult(new ApiResponse(code));
15                }
16            }
17        }
18    }
```

So how to get a request that's come into our API and get it passed to this particular controller "**ErrorController**" ?

```
API > C# Startup.cs > {} API > API.Startup > Configure(IApplicationBuilder app, IWebHostEnvironment env)
41
42     services.AddScoped<IPrductRepository,ProductRepository>();
43
44     services.AddScoped(typeof(IGenericRepository<>),typeof(GenericRepository<>));
45
46     services.AddAutoMapper(typeof(MappingProfiles));
47 }
48
49 // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
50 public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
51 {
52     if (env.IsDevelopment())
53     {
54         app.UseDeveloperExceptionPage();
55         app.UseSwagger();
56         app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "API v1"));
57     }
58
59     app.UseStatusCodePagesWithReExecute("/errors/{0}");
60
61     app.UseHttpsRedirection();
62
63     app.UseRouting();
64
65     app.UseStaticFiles();
66
67     app.UseAuthorization();
68
69     app.UseEndpoints(endpoints =>
```

Let we test this **dotnet watch run**

A screenshot of a web browser window. The address bar shows 'localhost:5001/swagger/index.html'. The page title is 'Swagger' with 'Supported by SMARTBEAR'. A dropdown menu says 'Select a definition' with 'API v1' selected. Below the header, a large error message reads 'Failed to load API definition.' A red-bordered box highlights an 'Errors' section containing 'Fetch error undefined /swagger/v1/swagger.json'. The main content area has a dark background with white text, showing a stack trace:

```
fail: Microsoft.AspNetCore.DiagnosticsDeveloperExceptionPageMiddleware[1]
An unhandled exception has occurred while executing the request.
Swashbuckle.AspNetCore.SwaggerGen.SwaggerGeneratorException: Ambiguous HTTP method for action - API.Controllers.Error (API). Actions require an explicit HttpMethod binding for Swagger/OpenAPI 3.0
at Swashbuckle.AspNetCore.SwaggerGen.SwaggerGenerator.GenerateOperations(IEnumerable`1 apiDescriptions, SchemaRepository schemaRepository)
at Swashbuckle.AspNetCore.SwaggerGen.SwaggerGenerator.GeneratePaths(IEnumerable`1 apiDescriptions, SchemaRepository schemaRepository)
at Swashbuckle.AspNetCore.SwaggerGen.SwaggerGenerator.GetSwagger(String documentName, String host, String basePath)
at Swashbuckle.AspNetCore.Swagger.SwaggerMiddleware.Invoke(HttpContext httpContext, ISwaggerProvider swaggerProvider)
at Microsoft.AspNetCore.DiagnosticsDeveloperExceptionPageMiddleware.Invoke(HttpContext context)
```

Fixing Problem

When applied to a public method on a controller, it prevents that method from appearing in the swagger ui.
Solve

A screenshot of a code editor showing the 'ErrorController.cs' file. The code is as follows:

```
# ErrorController.cs U X
API > Controllers > C# ErrorController.cs > {} API.Controllers
1  using API.Errors;
2  using Microsoft.AspNetCore.Mvc;
3
4  namespace API.Controllers
5  {
6      [Route("errors/{code}")]
7      [ApiExplorerSettings(IgnoreApi = true)]
8      public class ErrorController : BaseApiController
9      {
10          public IActionResult Error(int code)
11          {
12              return new ObjectResult(new ApiResponse(code));
13          }
14      }
15 }
```

Let we test now by putting any dummy endpoint error handler name such as below

A screenshot of a browser window. The address bar shows 'localhost:5001/api/toto'. The page content is a JSON object:

```
{
  statusCode: 404,
  message: "Resource found, it was not"
}
```

So even that request comes into our API server but we don't have an end point that match that particular request then we are going to hit our middleware

```
# Program.cs 2 X
Api > C# Program.cs
58     catch (Exception ex)
59     {
60         var logger = loggerFactory.CreateLogger<Program>();
61         logger.LogError(ex, "An error occurred during migration");
62     }
63
64     // Configure the HTTP request pipeline.
65     if (app.Environment.IsDevelopment())
66     {
67         app.UseSwagger();
68         app.UseSwaggerUI();
69     }
70
71     //Added By Alhafi
72     app.UseMiddleware<ExceptionMiddleware>();
73     app.UseStaticFiles();
74     app.UseStatusCodePagesWithReExecute("/errors/{0}"); You, 13 min
75
76     //
77     app.UseHttpsRedirection();
78
79     app.UseAuthorization();
```

```
app.UseStatusCodePagesWithReExecute("/errors/{0}");
```

and going to redirect to our error controller

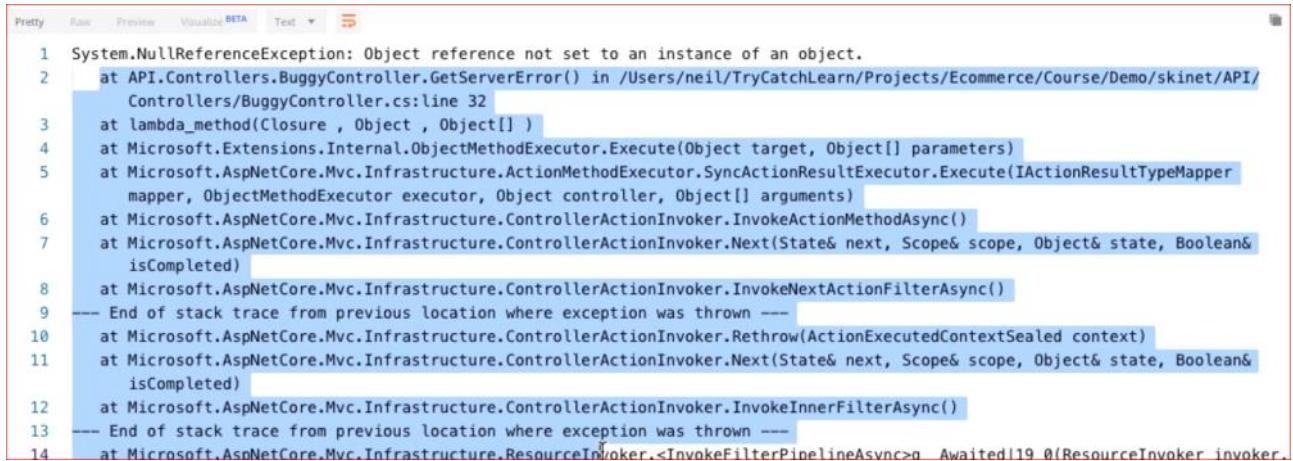
```
# ErrorController.cs U X
API > Controllers > C# ErrorController.cs > {} API.Controllers
1  using API.Errors;
2  using Microsoft.AspNetCore.Mvc;
3
4  namespace API.Controllers
5  {
6      [Route("errors/{code}")]
7      [ApiExplorerSettings(IgnoreApi = true)]
8      public class ErrorController : BaseApiController
9      {
10          public IActionResult Error(int code)
11          {
12              return new ObjectResult(new ApiResponse(code));
13          }
14      }
15  }
```

```
using API.Errors;
using Microsoft.AspNetCore.Mvc;
namespace API.Controllers
{
    [Route("errors/{code}")]
    [ApiExplorerSettings(Ignore = true)]
    public class ErrorController : BaseApiController
    {
        public IActionResult Error(int code)
        {
            return new ObjectResult(new ApiResponse(code));
        }
    }
}
```

Creating Exception handler middleware

22 June 2021 15:09

Here we need handle exception message response for example as below



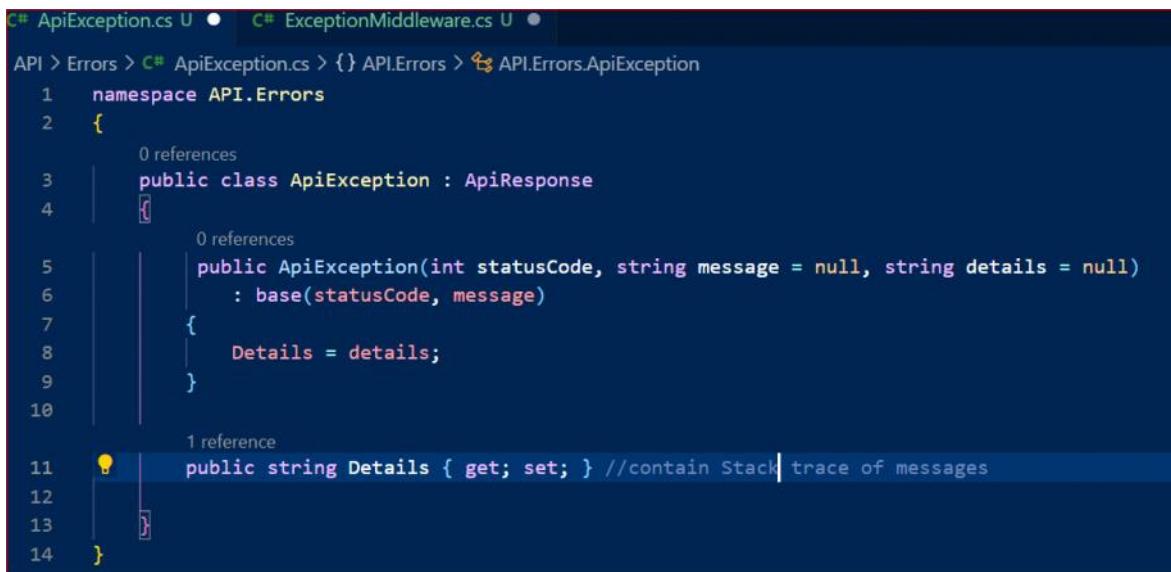
```
Pretty Raw Preview Visualize BETA Text ⚙️
1 System.NullReferenceException: Object reference not set to an instance of an object.
2   at API.Controllers.BuggyController.GetServerError() in /Users/neil/TryCatchLearn/Projects/Ecommerce/Course/Demo/skinet/API/
3     Controllers/BuggyController.cs:line 32
4   at lambda_method(Closure , Object , Object[] )
5   at Microsoft.Extensions.Internal.ObjectMethodExecutor.Execute(Object target, Object[] parameters)
6   at Microsoft.AspNetCore.Mvc.Infrastructure.ActionMethodExecutor.SynchronousActionExecutor.Execute(IActionResultTypeMapper
7     mapper, ObjectMethodExecutor executor, Object controller, Object[] arguments)
8   at Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.InvokeActionMethodAsync()
9   at Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.Next(State& next, Scope& scope, Object& state, Boolean&
10    isCompleted)
11  at Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.InvokeNextActionFilterAsync()
12  --- End of stack trace from previous location where exception was thrown ---
13  at Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.Rethrow(ActionExecutedContextSealed context)
14  at Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.Next(State& next, Scope& scope, Object& state, Boolean&
15    isCompleted)
16  at Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.InvokeInnerFilterAsync()
17  --- End of stack trace from previous location where exception was thrown ---
18  at Microsoft.AspNetCore.Mvc.Infrastructure.ResourceInvoker.<InvokeFilterPipelineAsync>d__19_0(ResourceInvoker invoker,
19    Object result)
```

Most what we need code and return message , which mostly be the first 2 lines and remains line no needs , to be simplify as below



```
← → ⌂ ⌂ Not secure | localhost:5001/api/toto
_apps دانلود رایگان نرم افزار... نظام الرعاية الاجتماعية HelpSmith Help Building Mi
{
  statusCode: 404,
  message: "Resource found, it was not"
}
```

So we go to folder errors and add class present middleware to handle exceptions naming it **ApiException** which will derive from **ApiResponse**



```
C# ApiException.cs U • C# ExceptionMiddleware.cs U •
API > Errors > C# ApiException.cs > {} API.Errors > API.Errors.ApiException
1  namespace API.Errors
2  {
3      0 references
4      public class ApiException : ApiResponse
5      {
6          0 references
7          public ApiException(int statusCode, string message = null, string details = null)
8              : base(statusCode, message)
9          {
10              Details = details;
11          }
12
13      }
14 }
```

Then we create a folder under API we name it **Middleware** and inside it we create a class **ExceptionMiddleware**

ExceptionMiddleware.cs

```
API > Middleware > C# ExceptionMiddleware.cs
1 namespace API.Middleware
2 {
3     public class ExceptionMiddleware
4     {
5     }
6 }
7 }
```

0 references

Generate overrides...
✓Generate constructor 'ExceptionMiddleware()'

Then injected with

ExceptionMiddleware.cs

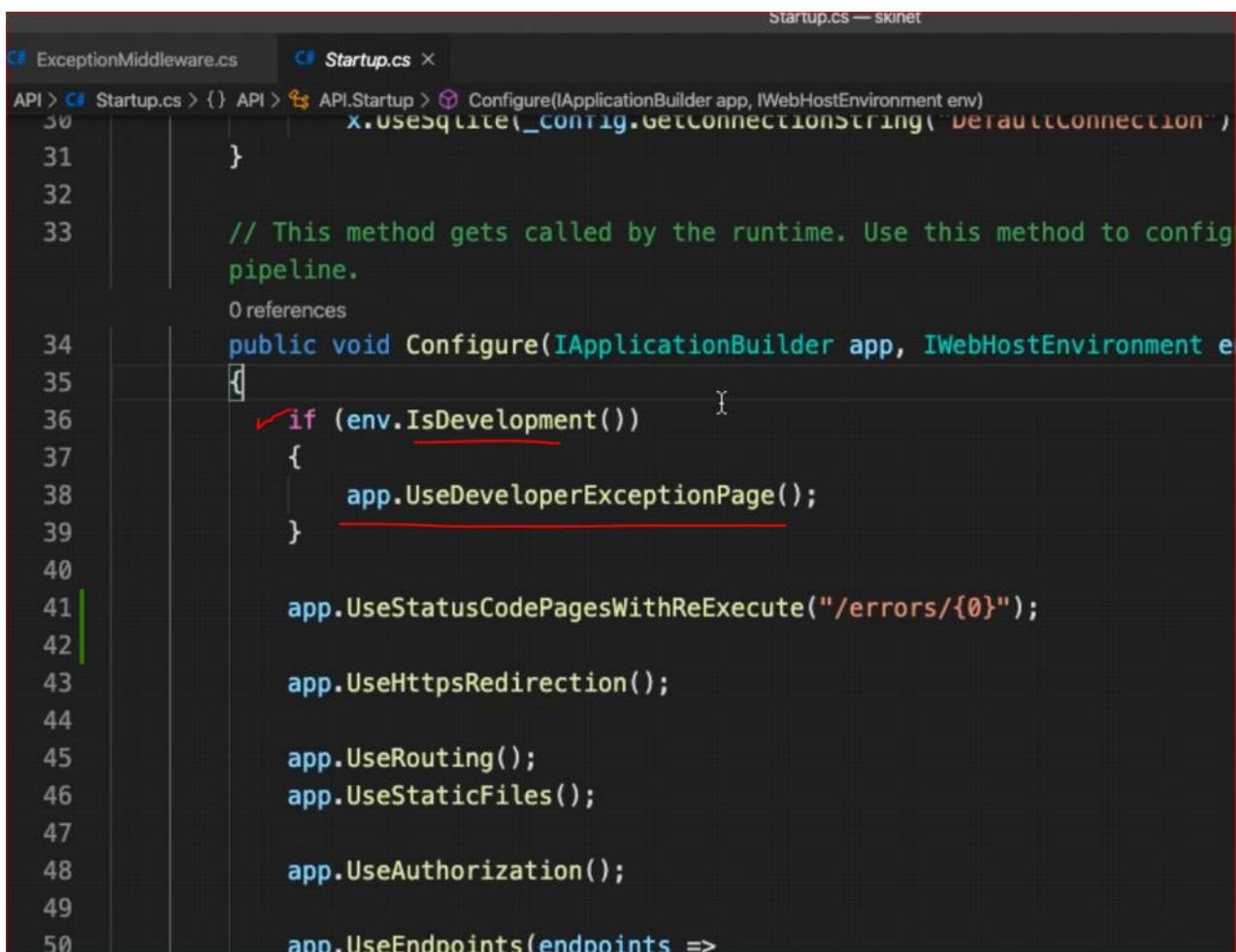
```
API > C# ExceptionMiddleware.cs > {} API.Middleware > C# API.Middleware.ExceptionMiddleware > ⚡ ExceptionMiddleware(RequestDelegate next, ILogger<ExceptionMiddleware> logger, IHostEnvironment env)
9     private readonly RequestDelegate _next;
10    1 reference
11    ✓ private readonly ILogger<ExceptionMiddleware> _logger;
12    1 reference
13    ✓ private readonly IHostEnvironment _env;
14    0 references
15    ✓ public ExceptionMiddleware(RequestDelegate next, ILogger<ExceptionMiddleware> logger, IHostEnvironment env)
16    {
17        ✓ _env = env;
18        ✓ _logger = logger;
19        ✓ _next = next;
20    }
21 }
```

Then middleware method

ExceptionMiddleware.cs

```
API > Middleware > C# ExceptionMiddleware.cs > {} API.Middleware > C# API.Middleware.ExceptionMiddleware > ⚡ InvokeAsync(HttpContext context)
0 references
14     public ExceptionMiddleware(RequestDelegate next, ILogger<ExceptionMiddleware> logger, IHostEnvironment env)
15     {
16         _env = env;
17         _logger = logger;
18         _next = next;
19     }
20
21     0 references
22     public async Task InvokeAsync(HttpContext context)
23     {
24         try
25         {
26         }
27         catch (Exception ex)
28         {
29         }
30     }
31 }
32 }
```

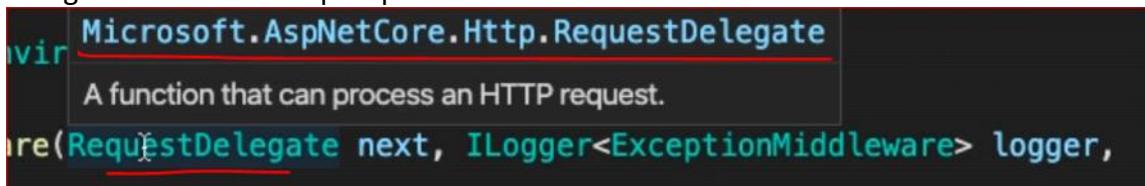
Declaration , as request come in to API ,and our startup file implement as middleware which in configuration ordering is important , So if we are in development mode and some exception raised in development it will catch from



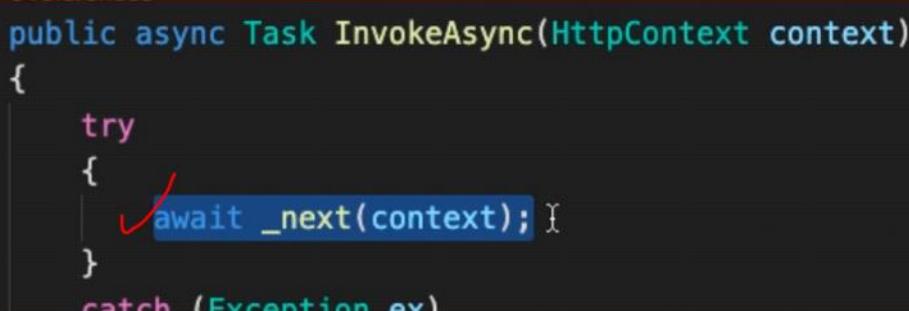
The screenshot shows the Visual Studio code editor with the file 'Startup.cs' open. The code is part of an ASP.NET Core application. It includes configuration for SQLite and the use of developer exception pages in development mode. The code is as follows:

```
API > C# Startup.cs > {} API > API.Startup > Configure(IApplicationBuilder app, IWebHostEnvironment env)
  30     .UseSqlite(_config.GetConnectionString("DefaultConnection"))
  31   }
  32
  33   // This method gets called by the runtime. Use this method to config
  34   // pipeline.
  35   public void Configure(IApplicationBuilder app, IWebHostEnvironment e
  36   {
  37     if (env.IsDevelopment())
  38     {
  39       app.UseDeveloperExceptionPage();
  40
  41       app.UseStatusCodePagesWithReExecute("/errors/{0}");
  42
  43       app.UseHttpsRedirection();
  44
  45       app.UseRouting();
  46       app.UseStaticFiles();
  47
  48       app.UseAuthorization();
  49
  50       app.UseEndpoints(endpoints =>
```

But in production mode we don't have middleware to handle exceptions , so in our class **RequestDelegate** as delegator receive an http request



And if there is no exception we want middleware to move to then next piece of middleware stage



The screenshot shows the 'InvokeAsync' method implementation. It uses a try-catch block to handle exceptions and calls the '_next' middleware stage.

```
public async Task InvokeAsync(HttpContext context)
{
    try
    {
        await _next(context);
    }
    catch (Exception ex)
```

And if there is an exception then we want to catch it

```
catch (Exception ex)
{
    _logger.LogError(ex, ex.Message);
    context.Response.ContentType = "application/json";
    context.Response.StatusCode = (int)HttpStatusCode.InternalServerError;

    var response = _env.IsDevelopment()
        ? new ApiException((int)HttpStatusCode.InternalServerError, ex.Message,
            ex.StackTrace.ToString())
        : new ApiException((int)HttpStatusCode.InternalServerError);

    var json = JsonSerializer.Serialize(response);

    await context.Response.WriteAsync(json);
}
```

```
# ExceptionMiddleware.cs •
API > Middleware > C# ExceptionMiddleware.cs > () API.Middleware > API.Middleware.ExceptionMiddleware > InvokeAsync(HttpContext context)
1  using System;
2  using System.Net;
3  using System.Text.Json;
4  using System.Threading.Tasks;
5  using API.Errors;
6  using Microsoft.AspNetCore.Http;
7  using Microsoft.Extensions.Hosting;
8  using Microsoft.Extensions.Logging;
9
10 namespace API.Middleware
11 {
12     2 references
13     public class ExceptionMiddleware
14     {
15         2 references
16         private readonly RequestDelegate _next;
17         2 references
18         private readonly ILogger<ExceptionMiddleware> _logger;
19         2 references
20         private readonly IHostEnvironment _env;
21         0 references
22         public ExceptionMiddleware(RequestDelegate next, ILogger<ExceptionMiddleware> logger, IHostEnvironment env)
23         {
24             _env = env;
25             _logger = logger;
26             _next = next;
27         }
28         0 references
29         public async Task InvokeAsync(HttpContext context)
30         {
31             try
32             {
33                 await _next(context);
34             }
35             catch (Exception ex)
36             {
37                 _logger.LogError(ex, ex.Message);
38                 context.Response.ContentType = "application/json";
39                 context.Response.StatusCode = (int)HttpStatusCode.InternalServerError;

40                 var response = _env.IsDevelopment()
41                     ? new ApiException((int)HttpStatusCode.InternalServerError, ex.Message,
42                         ex.StackTrace.ToString())
43                     : new ApiException((int)HttpStatusCode.InternalServerError);

44                 var options = new JsonSerializerOptions(PropertyNamingPolicy = JsonNamingPolicy.CamelCase);

45                 var json = JsonSerializer.Serialize(response, options);

46                 await context.Response.WriteAsync(json);
47             }
48         }
49     }
}
```

```

using System.Net;
using System.Text.Json;
using API.Errors;
namespace API.Middleware
{
    public class ExceptionMiddleware
    {
        private readonly RequestDelegate _next;
        private readonly ILogger<ExceptionMiddleware> _logger;
        private readonly IHostEnvironment _env;
        public ExceptionMiddleware(
            RequestDelegate next,
            ILogger<ExceptionMiddleware> logger,
            IHostEnvironment env
        )
        {
            _env = env;
            _logger = logger;
            _next = next;
        }
        public async Task InvokeAsync(HttpContext context)
        {
            try
            {
                await _next(context);
            }
            catch (Exception ex)
            {
                _logger.LogError(ex, ex.Message);
                context.Response.ContentType = "application/json";
                context.Response.StatusCode = (int) HttpStatusCode.InternalServerError;
                var response = _env.IsDevelopment()
                    ? new ApiException(
                        (int) HttpStatusCode.InternalServerError,
                        ex.Message,
                        ex.StackTrace.ToString()
                    )
                    : new ApiException((int) HttpStatusCode.InternalServerError);
                var options = new JsonSerializerOptions
                {
                    PropertyNamingPolicy = JsonNamingPolicy.CamelCase
                };
                var json = JsonSerializer.Serialize(response, options);
                await context.Response.WriteAsync(json);
            }
        }
    }
}

```

For the time being let work in development mode just replace old exception page

```

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
        app.UseSwagger();
        app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "API v1"));
    }
}

```

With this

```

app.UseMiddleware<ExceptionMiddleware>();
app.UseSwagger();
app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "API v1"));

```

No let we test this **dotnet watch run**

GET /api/Buggy/servererror

Parameters

No parameters

Execute **Clear**

Responses

Curl

```
curl -X 'GET' \
  'https://localhost:5001/api/Buggy/servererror' \
  -H 'accept: */*'
```

Request URL

<https://localhost:5001/api/Buggy/servererror>

Server response

Code **Details**

500 Undocumented **Error:**

Response body

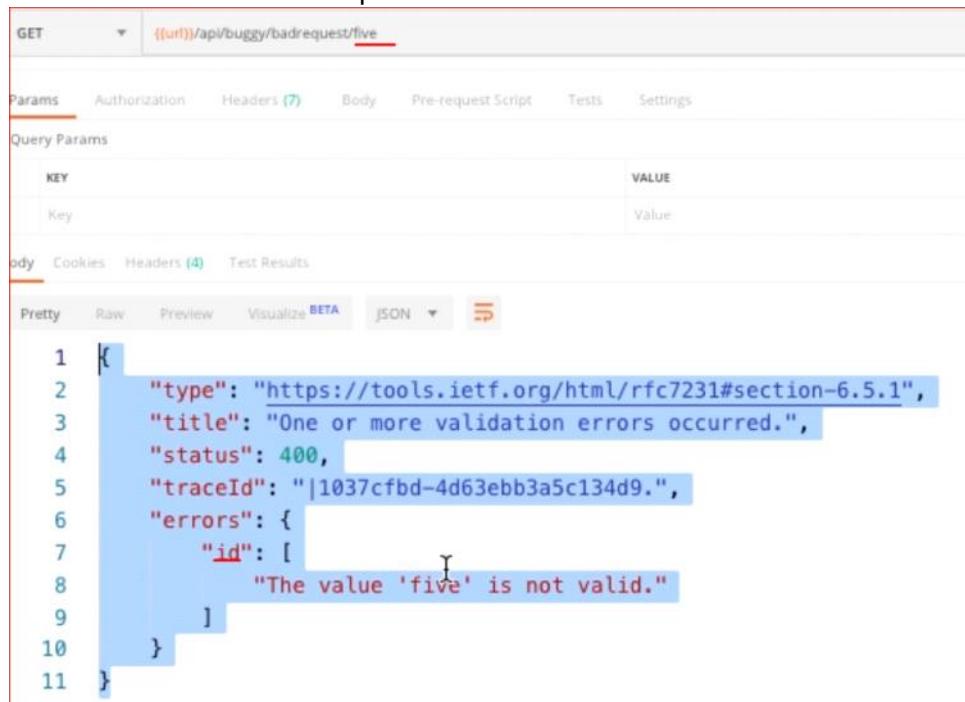
```
{ "details": " at API.Controllers.BuggyController.GetServerError() in C:\Dev_GIT\VisualCode\skinet\api\Controllers\BuggyController.cs:line 32\n      at lambda_method1(Closure , Object , Object[] )\n      at Microsoft.AspNetCore.Mvc.Infrastructure.ActionMethodExecutor.SynchronousActionExecutor.Execute(IActionResultTypeMapper mapper, ObjectMethodExecutor executor, Object controller, Object[] arguments)\n      at Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.InvokeActionMethodAsync()\n      --- End of stack trace from previous location ---\n      at Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.InvokeNextActionFilterAsync()\n      --- End of stack trace from previous location ---\n      at Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.Rethrow(ActionExecutedContextSealed context)\n      at Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.InvokeInnerFilterAsync()\n      --- End of stack trace from previous location ---\n      at Microsoft.AspNetCore.Mvc.Infrastructure.ResourceInvoker.<>InvokeFilterPipelineAsync()\n      --- Awaiting 19_0(ResourceInvoker invoker)\n      Task lastTask, State next, Scope scope, Object state, Boolean isCompleted)\n      at Microsoft.AspNetCore.Mvc.Infrastructure.ResourceInvoker.<InvokeAsynchronousFilter>A(17_0(ResourceInvoker invoker)\n      Task task, IDisposable scope)\n      at Microsoft.AspNetCore.Routing.EndpointMiddleware.<Invoke>A(AwaitRequestTask _6_0(Endpoint endpoint, Task requestTask, ILogger logger))\n      at Microsoft.AspNetCore.Diagnostics.StatusCodePagesMiddleware.<Invoke>A(HttpContext context)\n      at Swashbuckle.AspNetCore.SwaggerUI.Middleware.Invoke(HttpContext context)\n      at Swashbuckle.AspNetCore.Swagger.SwaggerMiddleware.<Invoke>A(HttpContext httpContext, ISwaggerProvider swaggerProvider)\n      at API.Middleware.ExceptionMiddleware.<InvokeAsynchronousFilter>A(HttpContext context) in C:\Dev_GIT\VisualCode\skinet\api\Middleware\ExceptionMiddleware.cs:line 28",\n      \"statusCode\": 500,\n      \"message\": \"Object reference not set to an instance of an object.\" }
```

Improving the validation error responses

23 June 2021 10:58

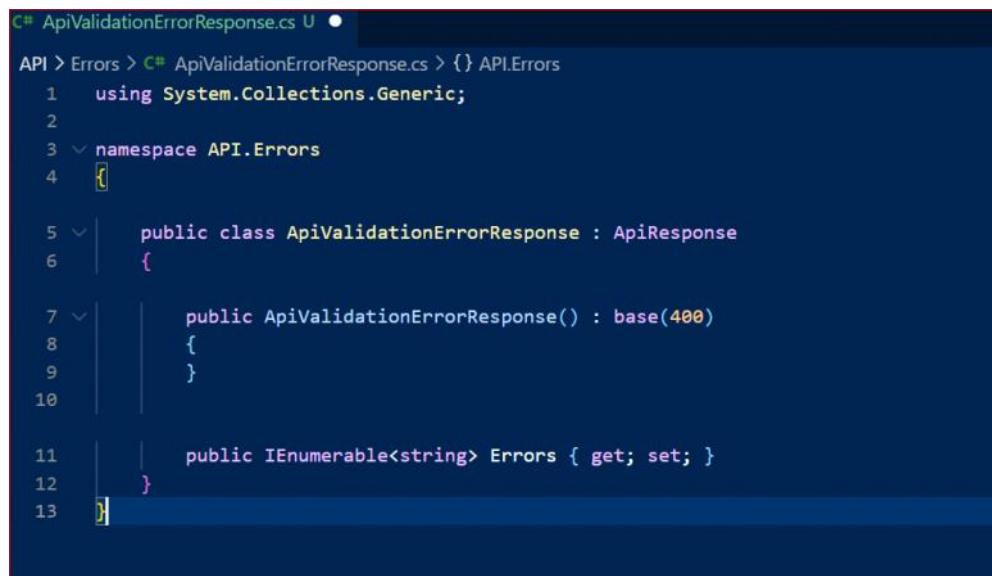
Here we need to handle response if task error below



The screenshot shows a Postman request for a GET endpoint at `https://{{url}}/api/buggy/badrequest/five`. The response body is a JSON object representing a validation error:

```
1  {
2      "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",
3      "title": "One or more validation errors occurred.",
4      "status": 400,
5      "traceId": "|1037cfbd-4d63ebb3a5c134d9.",
6      "errors": {
7          "id": [
8              "The value 'five' is not valid."
9          ]
10     }
11 }
```

This kind of error mostly related to array of error we mostly implement @ a forms for example above assume user send a string of ID such as Five instead 5 for id , so it is one value , but in some cases it could be array of 2 or more example user name or password empty / null then we have 2 rows errors as array one for username and one for password , So let we create a class naming it **ApiValidationErrorResponse** under **API/Errors** folder



```
C# ApiValidationErrorResponse.cs U ●
API > Errors > C# ApiValidationErrorResponse.cs > {} API.Errors
1  using System.Collections.Generic;
2
3  namespace API.Errors
4  {
5      public class ApiValidationErrorResponse : ApiResponse
6      {
7          public ApiValidationErrorResponse() : base(400)
8          {
9          }
10
11         public IEnumerable<string> Errors { get; set; }
12     }
13 }
```

Declaration

GET {{url}}/api/buggy/badrequest/five

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
Key	Value

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize BETA JSON

```

1  {
2      "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",
3      "title": "One or more validation errors occurred.", [
4          "status": 400,
5          "traceId": "|1037cfbd-4d63ebb3a5c134d9.",
6          "errors": {
7              "id": [
8                  "The value 'five' is not valid."
9              ]
10         }
11     ]

```

It come from

API > Controllers > BaseApiController.cs > API.Controllers > API.Controllers.BaseApiController

```

1  using Microsoft.AspNetCore.Mvc;
2
3  namespace API.Controllers
4  {
5      [ApiController]
6      [Route("api/[controller]")]
7      public class BaseApiController : ControllerBase
8      {
9      }
10 }
11

```

Let we comment

```

✓/ [ApiController]
[Route("api/[controller]")]
3 references
public class BaseApiController : ControllerBase
{
}

```

Nothing come , we don't see the response , because we remove tools checking to see the kind & Value of parameter

The screenshot shows a Postman request configuration for a GET method to the URL `{{url}}/api/buggy/badrequest/five`. The 'Params' tab is selected, showing a table with one row: 'Key' (Value) and 'Value' (Description). The 'Body' tab is also visible. The response preview shows the number '1'.

SO using

```
[ApiController]
[Route("api/[controller]")]
3 references
public class BaseApiController : ControllerBase
{}
```

It mean we used

actionContext.ModelState

```
"errors": {
  "id": [
    "The value 'five' is not valid."
  ]
}
```

So let we override the behavior of

```
[ApiController]
[Route("api/[controller]")]
3 references
public class BaseApiController : ControllerBase
{}
```

At startup we add

```

    References
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllers();
    services.AddSwaggerGen(c =>
    {
        c.SwaggerDoc("v1", new OpenApiInfo { Title = "API", Version = "v1" });
    });

    services.AddDbContext<StoreContext>(x =>
        x.UseSqlite(_config.GetConnectionString("DefaultConnection")));
    services.AddScoped<IProductRepository, ProductRepository>();
    services.AddScoped(typeof(IGenericRepository<>), typeof(GenericRepository<>));
    services.AddAutoMapper(typeof(MappingProfiles));

    ✓ services.Configure<ApiBehaviorOptions>(options =>
    {
        options.InvalidModelStateResponseFactory = actionContext =>
        {
            var errors = actionContext.ModelState
                .Where(e => e.Value.Errors.Count > 0)
                .SelectMany(x => x.Value.Errors)
                .Select(x => x.ErrorMessage).ToArray();

            var errorResponse = new ApiValidationErrorResponse
            {
                ✓ Errors = errors
            };

            return new BadRequestObjectResult(errorResponse);
        };
    });
}

```

```

builder.Services.Configure<ApiBehaviorOptions>(options =>
{
    options.InvalidModelStateResponseFactory = actionContext =>
    {
        var errors = actionContext.ModelState
            .Where(e => e.Value.Errors.Count > 0)
            .SelectMany(x => x.Value.Errors)
            .Select(x => x.ErrorMessage)
            .ToArray();
        var errorResponse = new ApiValidationErrorResponse { Errors = errors };
        return new BadRequestObjectResult(errorResponse);
    };
});

```

Note , normally @

```

public void ConfigureServices(IServiceCollection services)
{

```

Ordering is not important , but as exception we put our service.configuration after

```

public void ConfigureServices(IServiceCollection services)
{
    ✓ services.AddControllers();
}

```

Not let we test **dotnet watch run**

A screenshot of a web browser window. The address bar shows "localhost:5001/api/Buggy/badrequest/five". The page content is a JSON object with a red border:

```
✓errors: [
  "The value 'five' is not valid."
],
✓statusCode: 400,
✓message: "A bad request, you have made"
```

Adding Swagger for documenting our API

23 June 2021 12:04

Now in Dotnet 5 when you create webapi it add swagger to your project direct , but what assume if you don't have

The screenshot shows the NuGet Gallery interface. In the search bar, 'Swashbuckle' is typed. Below the search bar are 'Filter' and 'Prerelease' buttons. On the right, there's a 'nuget.org' link. The main list displays four items:

- Swashbuckle** by Richard Morris, v5.6.0. Description: Combines ApiExplorer and Swagger/swagger-ui to provide a rich discovery, documentation and playground experience to your API consumers.
- Swashbuckle.AspNetCore** by Swashbuckle.AspNetCore, v6.1.4. A red checkmark is placed next to this item.
- Swashbuckle.AspNetCore.Swagger** by Swashbuckle.AspNetCore.Swagger, v6.1.4. Description: Middleware to expose Swagger JSON endpoints from APIs built on ASP.NET Core.
- Swashbuckle.AspNetCore.SwaggerGen** by Swashbuckle.AspNetCore.SwaggerGen, v6.1.4. Description: Swagger Generator for APIs built on ASP.NET Core.

On the right side of the page, there's a sidebar for the selected item 'Swashbuckle.AspNetCore':

- Refresh button
- Swashbuckle.AspNetCore** by Swashbuckle.AspNetCore
- API.csproj (6.1.2) [Uninstall]
- Core.csproj [Install]
- Infrastructure.csproj [Install]
- 6.1.4 dropdown menu with 'Install' and 'Uninstall' buttons
- Description: Swagger tools for documenting APIs built on ASP.NET Core
- Open in: nuget.org
- Version: 6.1.4

The screenshot shows a code editor with C# syntax highlighting. The code is part of a startup configuration, likely in a 'ConfigureServices' method of a 'Startup.cs' file. It configures the 'ActionContext' to handle validation errors and adds a 'SwaggerGen' service to generate an OpenAPI document.

```
options.InvalidModelStateResponseFactory = actionContext =>
{
    var errors = actionContext.ModelState
        .Where(e => e.Value.Errors.Count > 0)
        .SelectMany(x => x.Value.Errors)
        .Select(x => x.ErrorMessage).ToArray();

    var errorResponse = new ApiValidationErrorResponse
    {
        Errors = errors
    };

    return new BadRequestObjectResult(errorResponse);
};

services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new OpenApiInfo {Title = "SkiNet API", Version = "v1"});
});
```

```

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    app.UseMiddleware<ExceptionMiddleware>();
    app.UseStatusCodePagesWithReExecute("/errors/{0}");

    app.UseHttpsRedirection();

    app.UseRouting();
    app.UseStaticFiles();

    app.UseAuthorization();

    ✓ app.UseSwagger();
    ✓ app.UseSwaggerUI(c => c
        ✓ .SwaggerEndpoint("/swagger/v1/swagger.json", "SkiNet API v1"));});

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllers();
    });
}

```

or keep as below

The screenshot shows the `Startup.cs` file in a code editor. Several lines of code are highlighted with red checkmarks, indicating they are part of the solution. The annotated lines are:

- `✓ app.UseSwagger();`
- `✓ app.UseSwaggerUI(c => c`
- `✓ .SwaggerEndpoint("/swagger/v1/swagger.json", "SkiNet API v1"));});`
- `✓ app.UseEndpoints(endpoints =>`
- `✓ {`
- `✓ endpoints.MapControllers();`
- `✓ };`
- `✓ }`
- `✓ // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.`
- `✓ public void Configure(IApplicationBuilder app, IWebHostEnvironment env)`
- `✓ {`
- `✓ if (env.IsDevelopment())`
- `✓ {`
- `✓ //app.UseDeveloperExceptionPage();`
- `✓ app.UseMiddleware<ExceptionMiddleware>();`
- `✓ app.UseSwagger();`
- `✓ ✓ app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "API v1"));`
- `✓ }`
- `✓ }`

Improving the swagger documentation

23 June 2021 13:08

Here we need to improve documentation of

The screenshot shows a Swagger UI interface with three main sections:

- Buggy**: Contains four GET endpoints:
 - /api/Buggy/notfound
 - /api/Buggy/servererror
 - /api/Buggy/badrequest
 - /api/Buggy/badrequest/{id}
- Products**: Contains five GET endpoints:
 - /api/Products
 - /api/Products/{id} (marked with a red checkmark)
 - /api/Products/brands
 - /api/Products/types
- WeatherForecast**: Contains one GET endpoint:
 - /WeatherForecast

In above method it could be return

- 200 (ID) exists
- 204
- 404

200

The screenshot shows the '200' details page for the `GET /api/Products/{id}` endpoint. It includes the following sections:

- Parameters**: A table with a single row for `id` (required, integer, path), with value `2`.
- Responses**:
 - Curl**: `curl -X 'GET' \ 'https://localhost:5001/api/Products/2' \ -H 'accept: text/plain'`
 - Request URL**: `https://localhost:5001/api/Products/2`
 - Server response**:
 - Code**: 200
 - Details**: Response body (marked with a red checkmark)

```
{
  "id": 2,
  "name": "Core Purple Boots",
  "description": "Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci.",
  "price": 399.99,
  "pictureUrl": "https://localhost:5001/images/products/boot-core1.png",
  "productType": "Boots",
  "productBrand": "NetCore"
}
```

204

GET /api/Products/{id}

Parameters

Name	Description
id <small>* required</small>	integer(\$int32) (path) 42

Execute Clear

Responses

Curl

```
curl -X 'GET' \
'https://localhost:5001/api/Products/42' \
-H 'accept: text/plain'
```

Request URL

```
https://localhost:5001/api/Products/42
```

Server response

Code	Details
204 <small>Undocumented</small>	Response headers <small>content-length: 0 date: Wed,23 Jun 2021 11:04:08 GMT server: Kestrel</small>

Responses

Code	Description	Links
200	Success	No links

404

Not secure | localhost:5001/api/Products/id=null

Apps دانلود رایگان نرم افزار... نظام الرعائية الاجتماعية HelpSmith Help Building Micros...

```
{
- errors: [
    "The value 'id=null' is not valid."
],
statusCode: 400,
message: "A bad request, you have made"
}
```

So let we add

```
[HttpGet("{id}")]
0 references
public async Task<ActionResult<ProductToReturnDto>> getProduct(int id)
{
    var spec = new ProductsWithTypesAndBrandsSpecification(id);
    var product = await _productsRepo.GetEntityWithSpec(spec);
    if (product == null) return NotFound(new ApiResponse(404));
    return _mapper.Map<ProductToReturnDto>(product);
}
```

The screenshot shows the Swagger UI interface for a REST API. At the top, there are navigation icons and a status bar indicating 'Not secure | localhost:5001/swagger/index.html'. Below the header, there are two main sections: 'GET /api/Products' and 'GET /api/Products/{id}'. The second section is expanded to show 'Parameters' for 'Id' (a required integer path parameter with value '42'). Below the parameters are 'Execute' and 'Clear' buttons. Under the 'Responses' section, there is a 'Curl' block with a command to execute a GET request to 'https://localhost:5001/api/Products/42'. The 'Request URL' is also shown as 'https://localhost:5001/api/Products/42'. The 'Server response' section shows a 404 error with the message 'Resource found, it was not'. A red arrow points to the 'Error:' part of the response body. There are 'Code' and 'Details' tabs, and a 'Download' button.

404

Undocumented Error:

We add also

```
[HttpGet("{id}")]
[ProducesResponseType(StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
0 references
public async Task<ActionResult<ProductToReturnDto>> getProducts(int id)
{
    var spec = new ProductsWithTypesAndBrandsSpecification(id);
    var product= await _productsRepo.GetEntityWithSpec(spec);

    if (product == null) return NotFound(new ApiResponse(404));

    return _mapper.Map<ProductToReturnDto>(product);
}
```

The screenshot shows a Swagger UI interface for a .NET API. At the top, a GET request is selected for the endpoint `/api/Products/{id}`. A parameter `id` is set to the value `42`. Below the parameters, there are sections for 'Responses' and 'Responses'. The 'Responses' section for the main request shows a 404 error response with a JSON body containing `{"statusCode": 404, "message": "Resource found, it was not"}`. The 'Responses' section also includes examples for 200 and 404 status codes, each with its own JSON schema example.

We could be more specific

```
[HttpGet("{id}")]  
  
[ProducesResponseType(StatusCodes.Status200OK)]  
[ProducesResponseType(typeof(ApiResponse), StatusCodes.Status404NotFound)]  
0 references  
public async Task<ActionResult<ProductToReturnDto>> getProducts(int id)  
{  
    var spec = new ProductsWithTypesAndBrandsSpecification(id);  
    var product= await _productsRepo.GetEntityWithSpec(spec);  
  
    if (product == null) return NotFound(new ApiResponse(404));  
  
    return _mapper.Map<ProductToReturnDto>(product);  
}
```

```
[HttpGet("{id}")]  
[ProducesResponseType(StatusCodes.Status200OK)]  
[ProducesResponseType(typeof(ApiResponse), StatusCodes.Status404NotFound)]  
  
public async Task<ActionResult<ProductToReturnDto>> getProduct(int id)  
{  
    var spec = new ProductsWithTypesAndBrandsSpecification(id);
```

```
var product = await _productsRepo.GetEntityWithSpec(spec);
if (product == null) return NotFound(new ApiResponse(404));
return _mapper.Map<ProductToReturnDto>(product);
}
```

The screenshot shows a Swagger UI interface for a .NET Core API. The endpoint is `/api/Products/{id}`. A parameter `id` is required and set to `42`.

Responses

- 404**: Error:
 - Response body:

```
{ "statusCode": 404, "message": "Resource found, it was not" }
```
 - Response headers:

```
Content-Type: application/json; charset=utf-8
date: Wed, 13 Jan 2021 11:27:51 GMT
server: Kestrel
```
- 200**: Success
 - Media type: `text/plain`
 - Example Value | Schema:

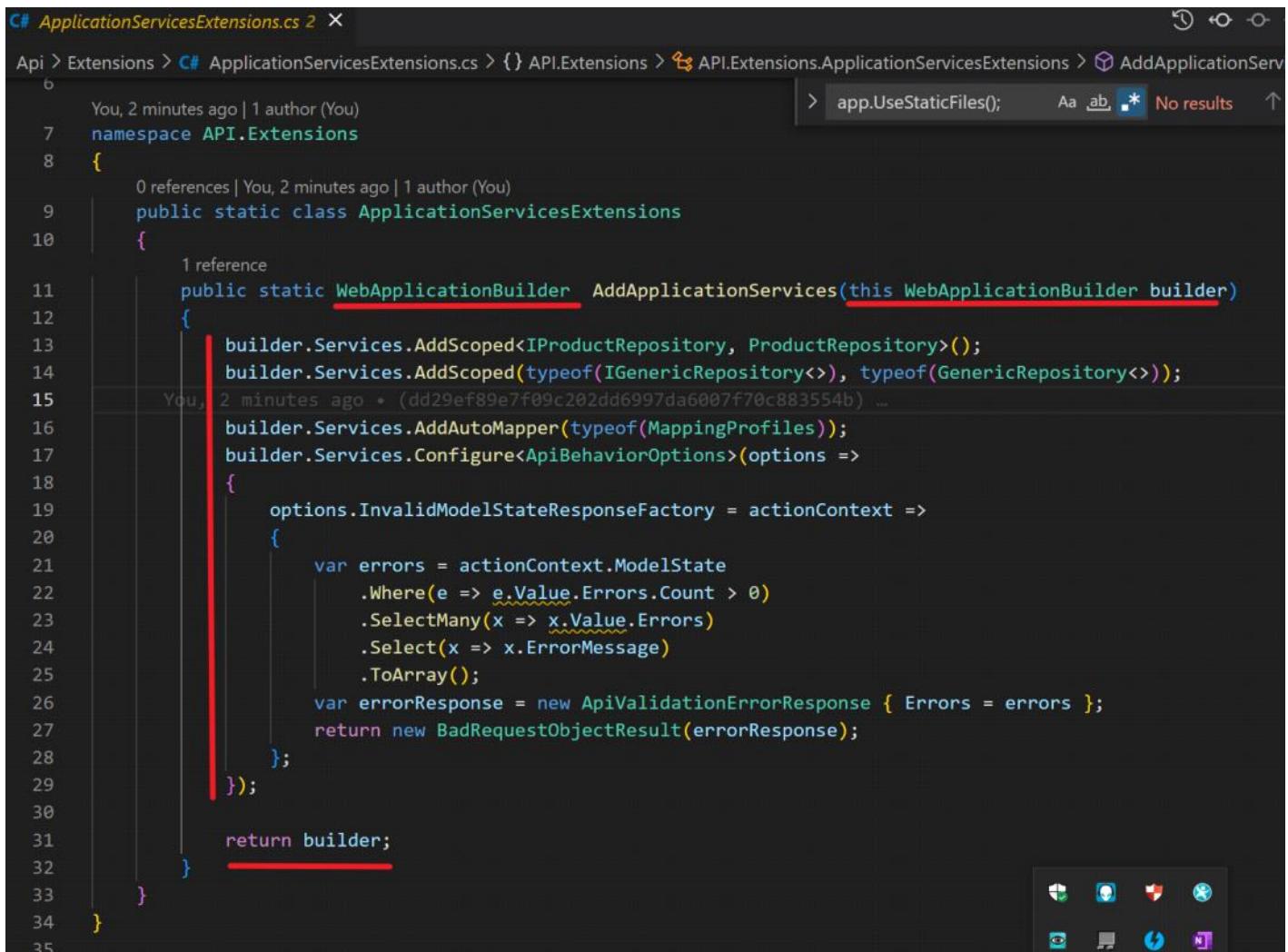
```
{ "Id": 0, "Name": "string", "Description": "string", "Price": 0, "PictureUrl": "string", "ProductType": "string", "ProductBrand": "string" }
```
- 404**: Not Found
 - Media type: `text/plain`
 - Example Value | Schema:

```
{ "statusCode": 404, "message": "string" }
```

Cleaning up the Program class

23 June 2021 14:26

Let's create an extension and application services there. So let's create a folder naming **Extensions** under API folder then add class **ApplicationServicesExtensions**



```
using System;
using Microsoft.AspNetCore.Builder;
using Microsoft.Extensions.DependencyInjection;
using API.Helpers;
using API.Errors;
using Base.Repository;
using Core.Interfaces;
using Microsoft.AspNetCore.Mvc;
namespace API.Extensions
{
    public static class ApplicationServicesExtensions
    {
        public static WebApplicationBuilder AddApplicationServices(this WebApplicationBuilder builder)
        {
            builder.Services.AddScoped<IProductRepository, ProductRepository>();
            builder.Services.AddScoped(typeof(IGenericRepository<>), typeof(GenericRepository<>));
            builder.Services.AddAutoMapper(typeof(MappingProfiles));
            builder.Services.Configure<ApiBehaviorOptions>(options =>
            {
                options.InvalidModelStateResponseFactory = actionContext =>
                {
                    var errors = actionContext.ModelState
                        .Where(e => e.Value.Errors.Count > 0)
                        .SelectMany(x => x.Value.Errors)
                        .Select(x => x.ErrorMessage)
                        .ToArray();
                    var errorResponse = new ApiValidationErrorResponse { Errors = errors };
                    return new BadRequestObjectResult(errorResponse);
                };
            });
            return builder;
        }
    }
}
```

```
using Api.Helpers;
using API.Errors;
using Base.Repository;
using Core.Interfaces;
using Microsoft.AspNetCore.Mvc;
namespace API.Extensions
{
    public static class ApplicationServicesExtensions
    {
        public static WebApplicationBuilder AddApplicationServices(this WebApplicationBuilder builder)
        {
            builder.Services.AddScoped<IProductRepository, ProductRepository>();
            builder.Services.AddScoped(typeof(IGenericRepository<>), typeof(GenericRepository<>));
            builder.Services.AddAutoMapper(typeof(MappingProfiles));
            builder.Services.Configure<ApiBehaviorOptions>(options =>
            {
                options.InvalidModelStateResponseFactory = actionContext =>
                {
                    var errors = actionContext.ModelState
                        .Where(e => e.Value.Errors.Count > 0)
                        .SelectMany(x => x.Value.Errors)
                        .Select(x => x.ErrorMessage)
                        .ToArray();
                    var errorResponse = new ApiValidationErrorResponse { Errors = errors };
                    return new BadRequestObjectResult(errorResponse);
                };
            });
        }
    }
}
```

```
        return builder;
    }
}
```

And then we need to **ApplicationServiceExtensions** on Program

```
var builder = WebApplication.CreateBuilder(args)
    .AddApplicationServices();
```

```
var builder = WebApplication.CreateBuilder(args)
    .AddApplicationServices();
```

Finally remove unassay references

The screenshot shows a code editor window with the file 'Startup.cs' open. The code is a standard ASP.NET Core startup configuration. Numerous 'using' statements are highlighted with red squiggly underlines, indicating they are unused or unnecessary. The statements include System.Linq, API.Errors, API.Extensions, API.Helpers, API.Middleware, Core.Interfaces, Infrastructure.Data, Microsoft.AspNetCore.Builder, Microsoft.AspNetCore.Hosting, Microsoft.AspNetCore.Mvc, Microsoft.EntityFrameworkCore, Microsoft.Extensions.Configuration, Microsoft.Extensions.DependencyInjection, Microsoft.Extensions.Hosting, and Microsoft.OpenApi.Models.

```
# Startup.cs M X
API > C# Startup.cs > {} API > API.Startup > Configure(IApplicationBuilder)
1 ✓ using System.Linq;
2 ✓ using API.Errors;
3 using API.Extensions;
4 using API.Helpers;
5 using API.Middleware;
6 ✓ using Core.Interfaces;
7 using Infrastructure.Data;
8 using Microsoft.AspNetCore.Builder;
9 using Microsoft.AspNetCore.Hosting;
10 using Microsoft.AspNetCore.Mvc;
11 using Microsoft.EntityFrameworkCore;
12 using Microsoft.Extensions.Configuration;
13 using Microsoft.Extensions.DependencyInjection;
14 ✓ using Microsoft.Extensions.Hosting;
15 ✓ using Microsoft.OpenApi.Models;
16
```

Then **dotnet watch run**

Reference

Tuesday, December 31, 2024 1:12 PM

[Lesson 7: Add Logging to Minimal Web API Calls](#)



[The log4net Tutorial: Logging in C# \(hands-on from beginner to advanced\)](#)



[Request Logging with Minimal API Endpoint Filter in C# .NET](#)



<https://stackoverflow.com/questions/59311444/simplest-correct-way-to-configure-log4net-for-a-net-core-3-1-application>

<https://www.c-sharpcorner.com/article/how-to-configure-log4net-in-net-7-api/>

<https://hackajob.com/talent/blog/how-to-log-in-asp-net-core-web-apps-using-log4net>

MVC-Controller

11 February 2024 12:56

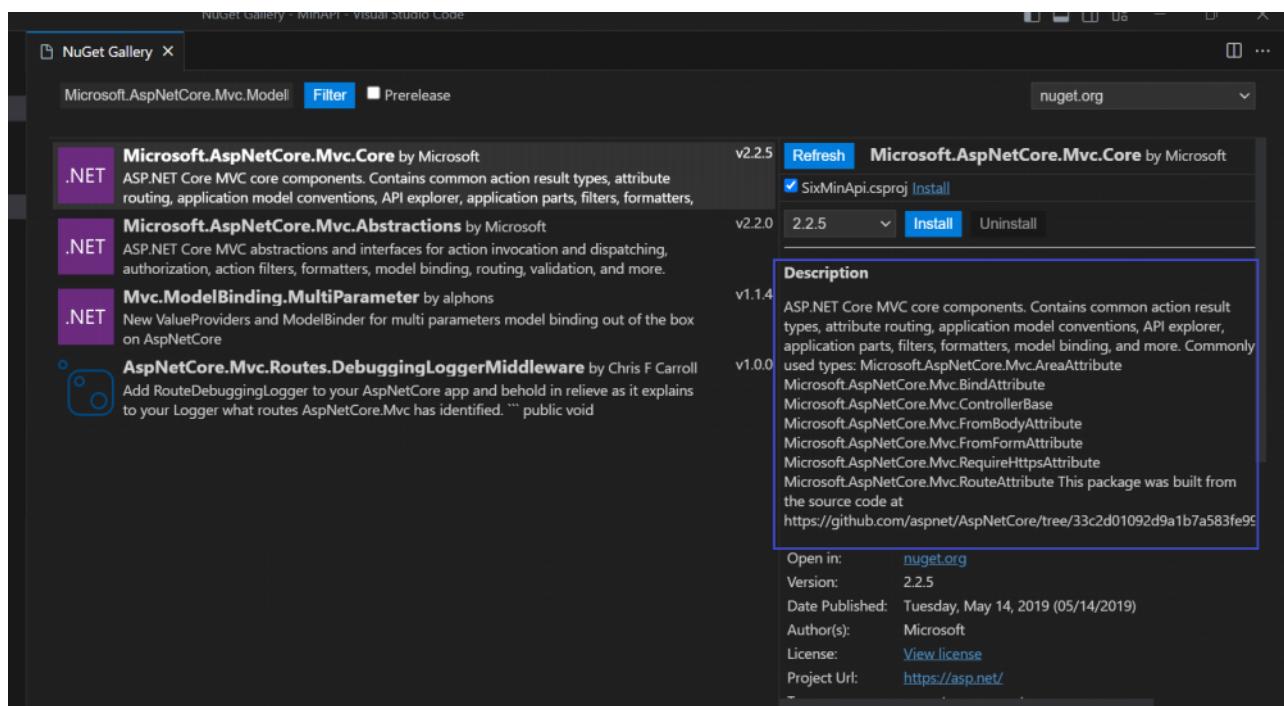
Using Controller API Instead Of Minimal API

By Missing some of feature in Minimal API such as

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`)

Let we Add to our Minimal API to controller base

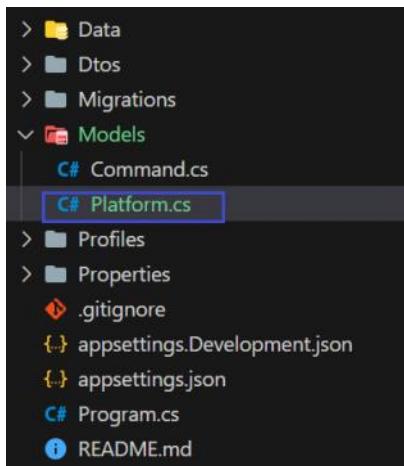


Register Controller

```
# Program.cs X
C# Program.cs > Program > Main
154     app.MapDelete("api/commands/{id}
155     {
156         var command = await context.
157         if (command != null)
158         {
159             context.Commands.Remove(
160                 await context.SaveChangesAsync());
161             return Results.NoContent();
162         }
163         return Results.NotFound();
164     });
165     app.MapControllers();
166
167     app.Run();
168 }
169 }
170 }
```

app.MapControllers();

Let we add class



```
using System.ComponentModel.DataAnnotations;
using Microsoft.AspNetCore.Mvc;

namespace MinLookUp.Models
{
    public class Platform
    {
        [Key]
        public int Id { get; set; }
        [Required]
        public string? PlatformName { get; set; }
    }
}
```

Creating DTO

```
using System.ComponentModel.DataAnnotations;
namespace SixAPI.Dtos
{
    public class PlatformCreateDto
    {
        [Required]
```

```

        public string? PlatformName { get; set; }
    }
}

```

Creating Profile

```

using System.ComponentModel.DataAnnotations;
namespace MinAPI.Dtos
{
    public class PlatformCreateDto
    {
        [Required]
        public string? PlatformName { get; set; }
    }
}

```

Adding to AppDbContext

The screenshot shows the Visual Studio interface with the project structure on the left and the code editor on the right. The code editor displays the `AppDbContext.cs` file. The line `public DbSet<Platform> Platforms => Set<Platform>();` is highlighted with a red rectangle.

```

14 references | You, 1 second ago | 2 authors (Barefoot@Dell8 and others)
public class AppDbContext : DbContext
{
    0 references
    public DbSet<Command> Commands { get; set; }
    9 references
    public DbSet<Command> Commands => Set<Command>();
    1 reference
    public DbSet<Platform> Platforms => Set<Platform>();
}

```

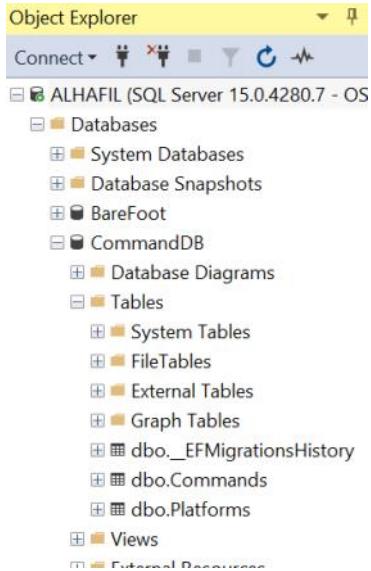
```
public DbSet<Platform> Platforms => Set<Platform>();
```

Migration

`dotnet ef migrations add AddPlatform`

Creating Database

`dotnet ef database update`



	Id	Platform...
1		.Net
2		Angular
3		React
*	NULL	NULL

Adding custom Binding

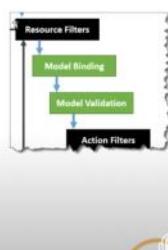
<https://learn.microsoft.com/en-us/aspnet/core/mvc/advanced/custom-model-binding?view=aspnetcore-7.0>

<https://www.c-sharpcorner.com/article/custom-model-binding-in-asp-net-core-mvc/>

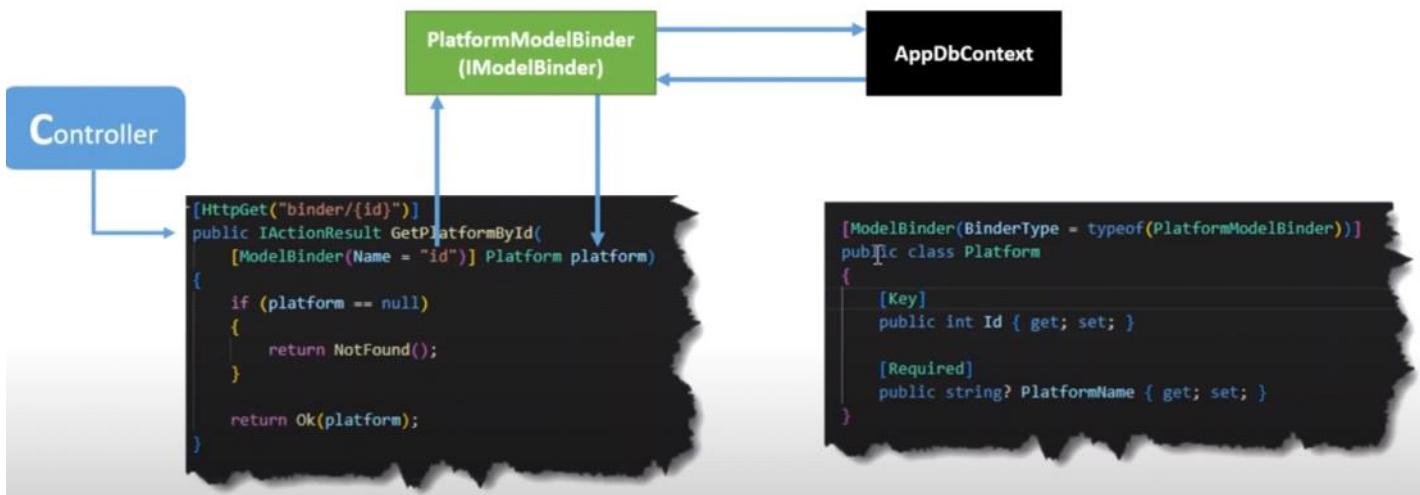
Model binding allows controller actions to work directly with model types (passed in as method arguments), rather than HTTP requests. Mapping between incoming request data and application models is handled by model binders. Developers can extend the built-in model binding functionality by implementing custom model binders (though typically, you don't need to write your own provider).

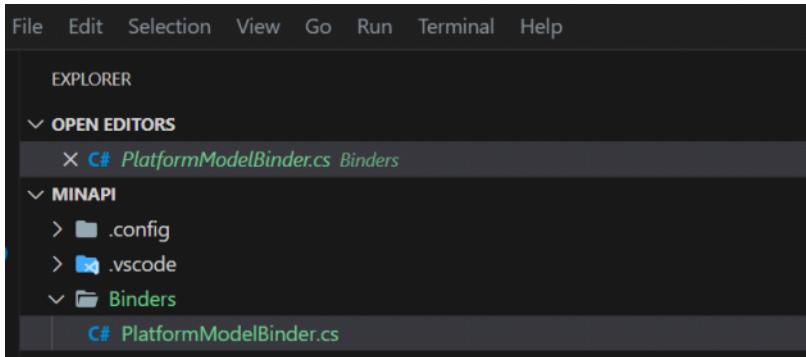
Custom Model Binding (IModelBinder)

- 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*





```

using Microsoft.AspNetCore.Mvc.ModelBinding;
using MinAPI.Data;
namespace MinAPI.Binders
{
    public class PlatformModelBinder : IModelBinder
    {
        private readonly AppDbContext _context;
        public PlatformModelBinder(AppDbContext context)
        {
            _context = context;
        }
        public Task BindModelAsync(ModelBindingContext bindingContext)
        {
            if (bindingContext == null)
            {
                Console.WriteLine("--> Binding Context is Null");
                throw new ArgumentNullException(nameof(bindingContext));
            }

            var modelName = bindingContext.ModelName;
            Console.WriteLine($"--> modelName: {modelName}");
            // Try to fetch the value of the argument by name
            var valueProviderResult =
                bindingContext.ValueProvider.GetValue(modelName);
            if (valueProviderResult == ValueProviderResult.None)
            {
                Console.WriteLine("--> ValueProviderResult is none");
                return Task.CompletedTask;
            }
            bindingContext.ModelState.SetModelValue(modelName,
                valueProviderResult);
            var value = valueProviderResult.FirstValue;
            // Check if the argument value is null or empty
            if (string.IsNullOrEmpty(value))
            {
                return Task.CompletedTask;
            }
            if (!int.TryParse(value, out var id))
            {
                // Non-integer arguments result in model state errors
                bindingContext.ModelState.TryAddModelError(
                    modelName, "Platform Id must be an integer.");
                return Task.CompletedTask;
            }
            // Model will be null if not found, including for
            // out of range id values (0, -3, etc.)
            var model = _context.Platforms.Find(id);
            bindingContext.Result = ModelBindingResult.Success(model);
            return Task.CompletedTask;
        }
    }
}

```

Applying ModelBinder Attribute on Model

From <<https://www.c-sharpcorner.com/article/custom-model-binding-in-asp-net-core-mvc/>>

```

using System.ComponentModel.DataAnnotations;
using Microsoft.AspNetCore.Mvc;
using MinAPI.Binders;

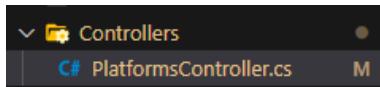
```

```

namespace MinAPI.Models
{
    [ModelBinder(BinderType = typeof(PlatformModelBinder))]
    public class Platform
    {
        [Key]
        public int Id { get; set; }
        [Required]
        public string? PlatformName { get; set; }
    }
}

```

Add Controller by Applying ModelBinding Attribute on Action method



```

using Microsoft.AspNetCore.Mvc;
using MinAPI.Models;
namespace MinAPI.Controllers
{
    [Route("api/v1/[controller]")]
    [ApiController]
    public class PlatformsController : ControllerBase
    {

        [HttpGet("binder/{id}")]
        public IActionResult GetPlatformById([ModelBinder(Name = "id" )]
Platform platform,int? id )
        {
            if (platform == null)
            {
                return NotFound();
            }
            return Ok(platform);
        }
    }
}

```

Test Controller API

The screenshot shows the Swagger UI interface for testing the API. The 'Platforms' section has a red box around the 'GET /api/v1/Platforms/binder/{id}' endpoint. The 'SixMinApi' section lists several other endpoints with different status colors: green for POST, blue for GET, orange for PUT, red for DELETE, and light blue for others.

The screenshot shows a Swagger UI interface for a REST API. At the top, a blue bar indicates a **GET** request to the endpoint `/api/v1/Platforms/binder/{id}`. Below this, a section titled "Parameters" lists a single parameter named "id". The "id" parameter is described as an integer (int32) required path parameter. A sample value "1" is entered into the input field. At the bottom of the main request area is a blue "Execute" button. Below the main request area, under the heading "Responses", there is a "Curl" section containing the command:

```
curl -X 'GET' \
'http://localhost:5196/api/v1/Platforms/binder/1?id=0&platformName=string' \
-H 'accept: */*'
```

Under "Request URL", the URL `http://localhost:5196/api/v1/Platforms/binder/1?id=0&platformName=string` is shown. The "Server response" section shows a 200 status code. The "Details" table has two rows: one for the "Response body" containing the JSON object `{"id": 1, "platformName": ".Net"}`, and one for "Response headers" showing the values `content-length: 30`, `content-type: application/json; charset=utf-8`, `date: Tue, 05 Sep 2023 10:57:11 GMT`, and `server: Kestrel`.

Adding Filters

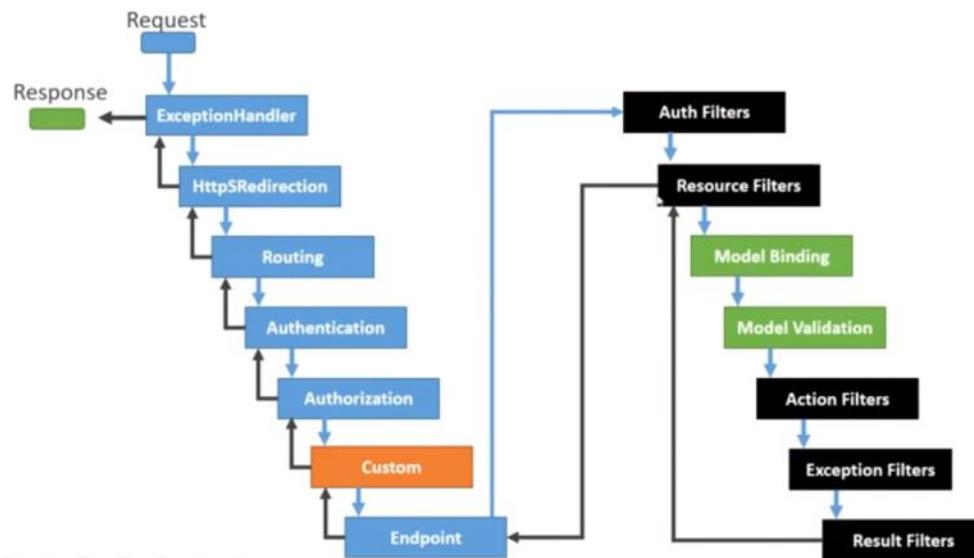
Web API includes filters to add extra logic before or after action method executes. Filters can be used to provide cross-cutting features such as logging, exception handling, performance measurement, authentication and authorization.

Filters

- Allow you to run code before or after stages in the *Filter Pipeline*
- Each Filter Type is executed at a different stage in the pipeline:
 - Authorization
 - Resource
 - Action
 - Exception
 - Result

- Custom filters can be scoped:
 - Globally (all controllers, actions and Razor Pages)
 - To a Controller
 - To an Action
- We have Synchronous and Asynchronous versions

Request Vs Filter Pipelines



Our implementation here in controller

So let we start to add Filter before and after controller

The screenshot shows the Visual Studio code editor with the 'TestAsyncActionFilter.cs' file open. The file is located in the 'Controllers' folder of a project named 'MinAPI'. The code defines a class 'TestAsyncActionFilter' that implements the 'IAutoFilter' interface. The 'OnActionExecutionAsync' method is overridden to log the action name and query string before and after the action execution.

```

EXPLORER
OPEN EDITORS
TestAsyncActionFilter.cs Controllers
MINAPI
Data
Dtos
Migrations
Models
Profiles
Properties
.gitignore
appsettings.Development.json
appsettings.json
Program.cs
README.md

... TestAsyncActionFilter.cs 1.0 x
Controllers > TestAsyncActionFilter.cs > TestAsyncActionFilter > OnActionExecutionAsync
1. using Microsoft.AspNetCore.Mvc;
2. using Microsoft.AspNetCore.Mvc.Filters;
3.
4. namespace MinAPI.Controllers;
5. {
6.     [Route("[controller]")]
7.     public class TestAsyncActionFilter : Controller
8.     {
9.         public async Task OnActionExecutionAsync(ActionExecutingContext context, ActionExecutionDelegate next)
10.        {
11.            var qs = context.HttpContext.Request.QueryString;
12.
13.            //var requestBody = context.HttpContext.Request.Body;
14.            Console.ForegroundColor = ConsoleColor.Red;
15.            Console.WriteLine("--> Action Filter doing something BEFORE the Action...");
16.            Console.WriteLine($"--> Query string: {qs}");
17.
18.            var result = await next();
19.            Console.ForegroundColor = ConsoleColor.Green;
20.            Console.WriteLine($"--> Action Filter doing something AFTER the Action...");
21.            Console.ResetColor();
22.        }
23.    }
24. }

```

```

using Microsoft.AspNetCore.Mvc.Filters;
namespace SixAPI.Controllers
{
    public class TestAsyncActionFilter : IAutoFilter
    {
        public async Task OnActionExecutionAsync(ActionExecutingContext context, ActionExecutionDelegate next)
        {
            var qs = context.HttpContext.Request.QueryString;
            //var requestBody = context.HttpContext.Request.Body.
        }
    }
}

```

```

        Console.ForegroundColor = ConsoleColor.Red;
        Console.WriteLine("---> Action Filter doing something BEFORE the
Action...");
        Console.WriteLine($"---> Query string: {qs}");

        var result = await next();
        Console.ForegroundColor = ConsoleColor.Green;
        Console.WriteLine("---> Action Filter doing something AFTER the
Action...");
        Console.ResetColor();
    }
}
}

```

Register it

The screenshot shows the `Program.cs` file in a code editor. The cursor is positioned at line 33, which contains the registration of the `TestAsyncActionFilter`. The line is highlighted with a red rectangle. The code snippet is as follows:

```

22
23     builder.Services.AddEndpointsApiExplorer();
24     builder.Services.AddSwaggerGen();

25
26     var sqlConBuilder = new SqlConnectionStringBuilder();
27     sqlConBuilder.ConnectionString = builder.Configuration.GetConnectionString("MsSqlDbConnection");
28     sqlConBuilder.UserID = builder.Configuration["UserId"];
29     sqlConBuilder.Password = builder.Configuration["Password"];

30
31     builder.Services.AddDbContext<AppDbContext>(opt => opt.UseSqlServer(sqlConBuilder.ConnectionString));
32     builder.Services.AddScoped< ICommandRepo, CommandRepo>();
33     builder.Services.AddScoped<TestAsyncActionFilter>();
34     builder.Services.AddAutoMapper(AppDomain.CurrentDomain.GetAssemblies());

35
36
37     var app = builder.Build();

38
39     // Configure the HTTP request pipeline.
40     if (app.Environment.IsDevelopment())
41     {
42         You, yesterday * Custom Binding ...
43         app.UseSwagger();
44         app.UseSwaggerUI();
45     }

46     app.UseHttpsRedirection();

```

Create Command Controller

```

using AutoMapper;
using Microsoft.AspNetCore.JsonPatch;
using Microsoft.AspNetCore.Mvc;
using MinAPI.Data;
using MinAPI.Dtos;
using SixMinApi.Models;
namespace MinAPI.Controllers
{
    [ServiceFilter(typeof(TestAsyncActionFilter))]
    [Route("api/v2/[controller]")]
    [ApiController]
    public class CommandsController : Controller
    {
        private readonly ICommandRepo _repo;
        private readonly IMapper _mapper;
        public CommandsController(ICommandRepo repo, IMapper mapper)
        {
            _repo = repo;
            _mapper = mapper;
        }
        [HttpGet]
        public async Task<ActionResult<IEnumerable<CommandReadDto>>>
GetAllCommands(

```

```

        [FromHeader] bool flipSwitch
    }

    var commands = await _repo.GetAllCommands();
    Console.ForegroundColor = ConsoleColor.Blue;
    Console.WriteLine($"--> The flip switch is: {flipSwitch}");
    Console.ResetColor();
    return Ok(_mapper.Map<IEnumerable<CommandReadDto>>(commands));
}

[HttpGet("{id}", Name = "GetCommandById")]
public async Task<ActionResult<CommandReadDto>> GetCommandById(int id)
{
    var commandModel = await _repo.GetCommandById(id);
    if (commandModel != null)
    {
        return Ok(_mapper.Map<CommandReadDto>(commandModel));
    }
    return NotFound();
}

[HttpPost]
public async Task<ActionResult<CommandReadDto>>
CreateCommand(CommandCreateDto cmdCreateDto)
{
    var commandModel = _mapper.Map<Command>(cmdCreateDto);
    await _repo.CreateCommand(commandModel);
    await _repo.SaveChanges();
    var cmdReadDto = _mapper.Map<CommandReadDto>(commandModel);
    Console.WriteLine($"Model State is: {ModelState.IsValid}");
    return CreatedAtRoute(nameof(GetCommandById), new { Id =
cmdReadDto.Id }, cmdReadDto);
}

//PUT api/commands/{id}
[HttpPut("{id}")]
public async Task<ActionResult> UpdateCommand(int id, CommandUpdateDto
commandUpdateDto)
{
    var commandModelFromRepo = await _repo.GetCommandById(id);
    if (commandModelFromRepo == null)
    {
        return NotFound();
    }
    _mapper.Map(commandUpdateDto, commandModelFromRepo);
    await _repo.SaveChanges();
    return NoContent();
}

//DELETE api/commands/{id}
[HttpDelete("{id}")]
public async Task<ActionResult> DeleteCommand(int id)
{
    var commandModelFromRepo = await _repo.GetCommandById(id);
    if (commandModelFromRepo == null)
    {
        return NotFound();
    }
    _repo.DeleteCommand(commandModelFromRepo);
    await _repo.SaveChanges();
    return NoContent();
}
}
}

```

Let we test our console Filter

```

var qs = context.HttpContext.Request.QueryString;

//var requestBody = context.HttpContext.Request.Body;
Console.ForegroundColor = ConsoleColor.Red;
Console.WriteLine("--> Action Filter doing something BEFORE the Action...");
Console.WriteLine($"--> Query string: {qs}");

var result = await next();
Console.ForegroundColor = ConsoleColor.Green;
Console.WriteLine("--> Action Filter doing something AFTER the Action...");
Console.ResetColor();

```

Dotnet run

Use any API

SixMinApi 1.0 OAS3

http://localhost:5196/swagger/v1/swagger.json

Commands

GET /api/v2/Commands

Parameters

Name	Description
flipSwitch	boolean (header)

Execute

Responses

Curl

```
curl -X 'GET' \
'http://localhost:5196/api/v2/Commands' \
-H 'accept: text/plain'
```

Request URL

```
http://localhost:5196/api/v2/Commands
```

Server response

Code	Details
200	<p>Response body</p> <pre>[{ "id": 1, "howTo": "run dotnet", "platform": ".Net", "commandLine": "dotnet run" }, { "id": 2, "howTo": "run dotnet hot reload", "platform": ".Net", "commandLine": "dotnet watch" }]</pre> <p>Response headers</p> <pre>content-length: 164 content-type: application/json; charset=utf-8 date: Wed, 06 Sep 2023 06:55:03 GMT server: Kestrel</pre>

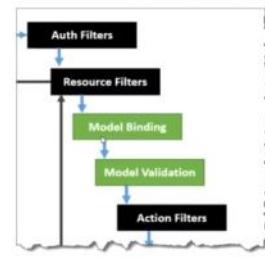
```

info: Microsoft.Hosting.Lifetime[0]
Content root path: D:\VC\MinAPI
warn: Microsoft.AspNetCore.HttpsPolicy.HttpsRedirectionMiddleware[3]
Failed to determine the https port for redirect.
--> Action Filter doing something BEFORE the Action...
--> Query string:
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
Executed DbCommand (27ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
SELECT [c].[Id], [c].[CommandLine], [c].[HowTo], [c].[Platform]
FROM [Commands] AS [c]
--> The flip switch is: False
--> Action Filter doing something AFTER the Action...
[]
```

Model Validation

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



```
C# Command.cs ×

Models > C# Command.cs > ...
Barefoor@Dell8, 6 days ago | 2 authors (AlhafiSurface and others)
1 using System.ComponentModel.DataAnnotations;
2
3 namespace SixMinApi.Models
4 {
5     17 references | Barefoor@Dell8, 6 days ago | 2 authors (AlhafiSurface and others)
6     public class Command
7     {
8         [Key]
9             5 references
10        public int Id { get; set; }
11        [Required]
12            2 references
13        public string? HowTo { get; set; }
14        [Required]
15            2 references
16        [MaxLength(5)]
17        2 references
18        public string? Platform { get; set; }
19        [Required]
20            2 references
21        public string? CommandLine { get; set; }
22    }
23 }
```

POST https://localhost:7014/api/v1/commands

Send 400 Bad Request 1.91 ms 297 B

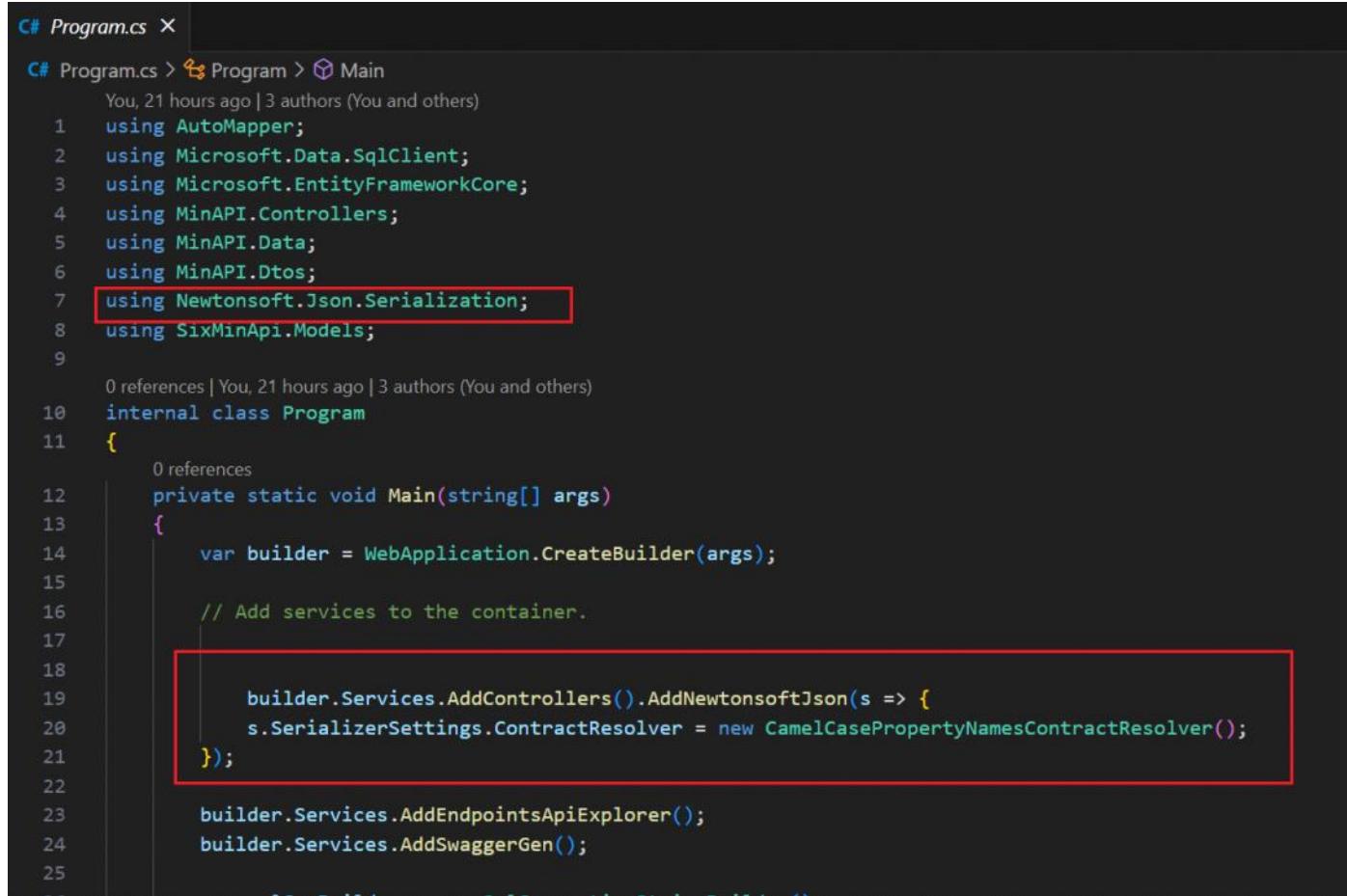
SON	Auth	Query	Header	Docs	Preview	Header	Cookie	Timeline
1 + { 2 "howTo": "Build a net app", 3 "platform": ".NET core version 4", 4 "commandline": "dotnet build" 5 }	Base64 => Encode Base64 => Decode Hash = SHA256 Timestamp = ISO-8601				1 + { 2 "errors": [3 "platform": [4 "The field Platform must be a string or array type with 5 length of '7'." 6] 7], 8 "type": "https://tools.ietf.org/html/rfc2231#section-6.5.1", 9 "title": "One or more validation errors occurred.", 10 "status": 400, 11 "traceId": "00-3b3e185346c23cb4bfe5bb083292e0ef-8f1a6732dd92"			

Batching

When a client needs to replace an existing Resource entirely, they can use PUT.
When they're doing a partial update, they can use HTTP PATCH.

For instance, when updating a single field of the Resource, sending the complete Resource representation can be cumbersome and uses a lot of unnecessary bandwidth. In such cases, the semantics of PATCH make a lot

more sense.



```
C# Program.cs X
C# Program.cs > Program > Main
You, 21 hours ago | 3 authors (You and others)
1  using AutoMapper;
2  using Microsoft.Data.SqlClient;
3  using Microsoft.EntityFrameworkCore;
4  using MinAPI.Controllers;
5  using MinAPI.Data;
6  using MinAPI.Dtos;
7  using Newtonsoft.Json.Serialization;
8  using SixMinApi.Models;
9
10 internal class Program
11 {
12     private static void Main(string[] args)
13     {
14         var builder = WebApplication.CreateBuilder(args);
15
16         // Add services to the container.
17
18
19         builder.Services.AddControllers().AddNewtonsoftJson(s => {
20             s.SerializerSettings.ContractResolver = new CamelCasePropertyNamesContractResolver();
21         });
22
23         builder.Services.AddEndpointsApiExplorer();
24         builder.Services.AddSwaggerGen();
25     }
}
```

```
builder.Services.AddControllers().AddNewtonsoftJson(s => {
    s.SerializerSettings.ContractResolver = new
CamelCasePropertyNamesContractResolver();
});

//PATCH api/v1/commands/{id}
[HttpPatch("{id}")]
public async Task<ActionResult> PartialCommandUpdate(
    int id,
    JsonPatchDocument<CommandUpdateDto> patchDoc
)
{
    var commandModelFromRepo = await _repo.GetCommandById(id);
    if (commandModelFromRepo == null)
    {
        return NotFound();
    }
    var commandToPatch = _mapper.Map<CommandUpdateDto>
(commandModelFromRepo);
    patchDoc.ApplyTo(commandToPatch, ModelState);
    if (!TryValidateModel(commandToPatch))
    {
        return ValidationProblem(ModelState);
    }
    _mapper.Map(commandToPatch, commandModelFromRepo);
    //await _repo.UpdateCommand(commandModelFromRepo);
    await _repo.SaveChangesAsync();
    return NoContent();
}
```

Let we test by using batch to change only HotTo from item 2

Response body

```
[
  {
    "id": 1,
    "howTo": "run dotnet",
    "platform": ".Net",
    "commandline": "dotnet run"
  },
  {
    "id": 2,
    "howTo": "run dotnet hot loaded",
    "platform": ".Net",
    "commandline": "dotnet watch"
  }
]
```

PATCH /api/v2/Commands/{id}

Parameters

Name	Description
id <small>required</small>	integer(\$int32) 2 (path)

Request body

application/json-patch+json

```
[
  {
    "operationType": 0,
    "path": "string",
    "op": "string",
    "from": "string",
    "value": "string"
  }
]
```

Replace

PATCH /api/v2/Commands/{id}

Parameters

Name	Description
id <small>required</small>	integer(\$int32) 2 (path)

Request body

application/json-patch+json

```
[
  {
    "path": "/howto",
    "op": "replace",
    "value": "Testing Changing"
  }
]
```

Execute

Commands

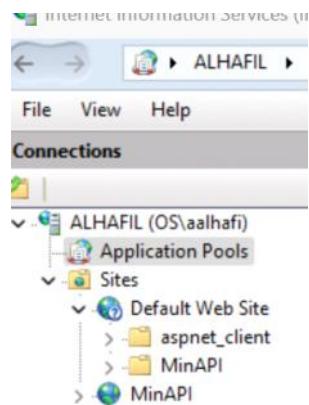
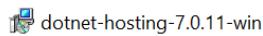
GET	/api/controller/v1/Commands
POST	/api/controller/v1/Commands
GET	/api/controller/v1/Commands/{id}
PATCH	/api/controller/v1/Commands/{id}
PUT	/api/controller/v1/Commands/{id}
DELETE	/api/controller/v1/Commands/{id}

Platforms

GET	/api/controller/binding/v1/Platforms/{id}
-----	---

Finally you can Publish to your IIS using Visual code

[Download ASP.NET Core 7.0 Runtime \(v7.0.11\) - Windows Hosting Bundle Installer \(microsoft.com\)](#)



Edit Site Binding

Type:	IP address:	Port:
http	All Unassigned	8032
Host name:		
Example: www.contoso.com or marketing.contoso.com		

OK **Cancel**

Add Site Binding

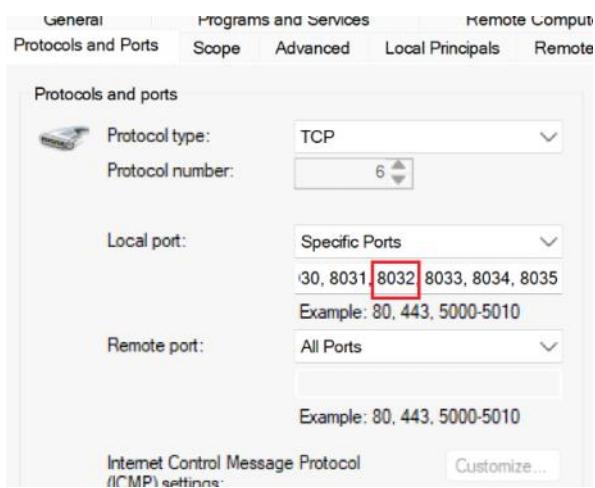
Type:	IP address:	Port:
https	All Unassigned	8033
Host name:		
<input type="checkbox"/> Require Server Name Indication		
<input type="checkbox"/> Disable TLS 1.3 over TCP	<input type="checkbox"/> Disable QUIC	
<input type="checkbox"/> Disable Legacy TLS	<input type="checkbox"/> Disable HTTP/2	
<input type="checkbox"/> Disable OCSP Stapling	<input type="checkbox"/> Negotiate Client Certificate	
SSL certificate:		
IIS Express Development Certificate		Select... View...

OK **Cancel**

Site Bindings

Type	Host Name	Port	IP Address	Binding Informa...	
http		8032	*		Add...
https		8033	*		Edit...

Remove **Browse**



Application Pools

This page lets you view and manage the list of application pools on the server. Application pools are associated with applications, and provide isolation among different applications.

Name	Status	.NET CLR V...
.NET v2.0	Started	v2.0
.NET v2.0 Classic	Started	v2.0
.NET v4.5	Started	v4.0
.NET v4.5 Classic	Started	v4.0
Classic .NET Ap...	Started	v2.0
DefaultAppPool	Started	v4.0
MinAPI	Started	No Manag...

```

appsettings.json •
appsettings.json > {} ConnectionStrings
...
1  {
2    "Logging": {
3      "LogLevel": {
4        "Default": "Information",
5        "Microsoft.AspNetCore": "Warning"
6      }
7    },
8    "AllowedHosts": "*",
9    "ConnectionStrings": {
10      "MsSqlDbConnection": "Server=localhost;initial catalog=CommandDB;TrustServerCertificate=True;MultipleActiveResultSets=true"
11    }
12  }
13}
14

```

C# Program.cs 2 ●

C# Program.cs > Program > Main

```

22
23     builder.Services.AddEndpointsApiExplorer();
24     builder.Services.AddSwaggerGen();

25
26     var sqlConBuilder = new SqlConnectionStringBuilder();
27     sqlConBuilder.ConnectionString = builder.Configuration.GetConnectionString("MsSqlDbConne");
28     //sqlConBuilder.UserID = builder.Configuration["UserId"];
29     //sqlConBuilder.Password = builder.Configuration["Password"];
30
31     sqlConBuilder.UserID = "cat";
32     sqlConBuilder.Password = "wd0rogP@ssd";

33
34     builder.Services.AddDbContext<AppDbContext>(opt => opt.UseSqlServer(sqlConBuilder.Connec);
35     builder.Services.AddScoped< ICommandRepo, CommandRepo>();
36     builder.Services.AddScoped< TestAsyncActionFilter>();
37     builder.Services.AddAutoMapper(AppDomain.CurrentDomain.GetAssemblies());
38
39
40     var app = builder.Build();

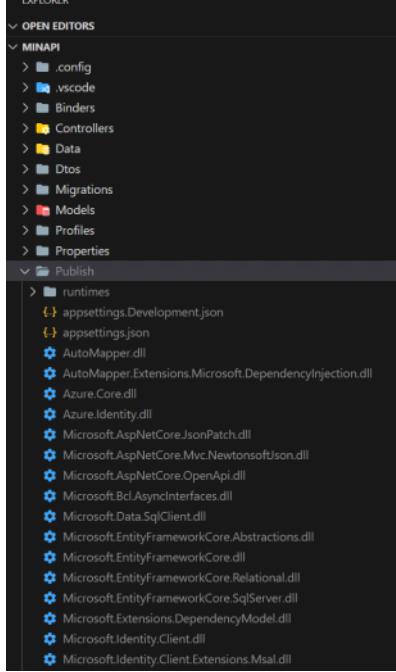
41
42     // Configure the HTTP request pipeline.
43     // if (app.Environment.IsDevelopment())
44     // {
45     //     app.UseSwagger();
46     //     app.UseSwaggerUI();
47     // }

48
49     app.UseSwagger();
50     app.UseSwaggerUI();

51

```

dotnet publish -c Release -o ./Publish



Copy to

↑	↻	This PC	OS (C:)	inetpub	Hosting	MinAPI	>
X	U	F	A	E	Delete	Sort	View
Name		Date modified	Type				
Publish		13-Sep-23 8:08 AM	File folder				
runtimes		13-Sep-23 8:08 AM	File folder				
appsettings.Development.json		04-Sep-23 8:28 AM	JSON File				

<https://localhost:8033/swagger/index.html>

The screenshot shows the Swagger UI homepage. At the top, there's a header with a logo, a search bar containing 'localhost:8033/swagger/index.html', and several navigation links. Below the header is a dark banner with the 'Swagger' logo and the text 'Supported by SMARTBEAR'. The main content area displays the title 'SixMinApi' with a '1.0 OAS3' badge, followed by the URL 'https://localhost:8033/swagger/v1/swagger.json'.

SixMinApi 1.0 OAS3

<https://localhost:8033/swagger/v1/swagger.json>

Commands

GET /api/controller/v1/Commands

POST /api/controller/v1/Commands

GET /api/controller/v1/Commands/{id}

Minimal API Implement Filtration For Model Validation

```
C# Program.cs 2 •
C# Program.cs > Program
4 references
279     public string? Platform { get; set; }
280
281     4 references
282     public string? CommandLine { get; set; }
283   }
284
285   0 references | You, 1 second ago | 1 author (You)
286   public class Person
287   {
288     0 references
289     public int Id { get; set; }
290
291     0 references
292     public string? FullName { get; set; }
293
294     0 references
295     public string? Phone { get; set; }
296
297     0 references
298     public string? Email { get; set; }
299
300   }
301
302   You, 1 second ago • Uncommitted changes
303 }
```

```
public class Person
{
    public int Id { get; set; }
    public string? FullName { get; set; }
    public string? Phone { get; set; }
    public string? Email { get; set; }

}
```

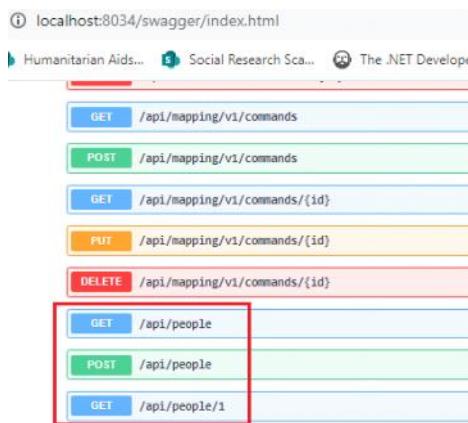
Creating var of list

```
C# Program.cs 2, M X
C# Program.cs > Program > Main
76     Platform = ".NET",
77     CommandLine = "dotnet Watch"
78   }
79 };
80
81 var people = new List<Person>
82 {
83   new Person
84   {
85     Id = 1,
86     FullName = "Shawn Wildermuth",
87     Phone = "",
88     Email = "shawn@aol.com"
89   }
90 };
91
92 // Simplest Static API///////////
93
94 app.MapGet("api/helloworld", () => "Hello World");
```

```
var people = new List<Person>
{
  new Person
  {
    Id = 1,
    FullName = "Shawn Wildermuth",
    Phone = "",
    Email = "shawn@aol.com"
  }
};
```

Let we add endpoint fake action

```
app.MapGet(
    "/api/people",
    () =>
{
    return Results.Ok(people);
}
);
app.MapGet(
    "/api/people/1",
    () =>
{
    return Results.Ok(people[0]);
}
);
app.MapPost(
    "/api/people",
    (Person person) =>
{
    // Fake some saving
    return Results.Created("/api/person/1", person);
}
);
```



Let post any data

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8034/api/people' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 1,
    "fullName": "xxx",
    "phone": "",
    "email": "xx"
}'
```

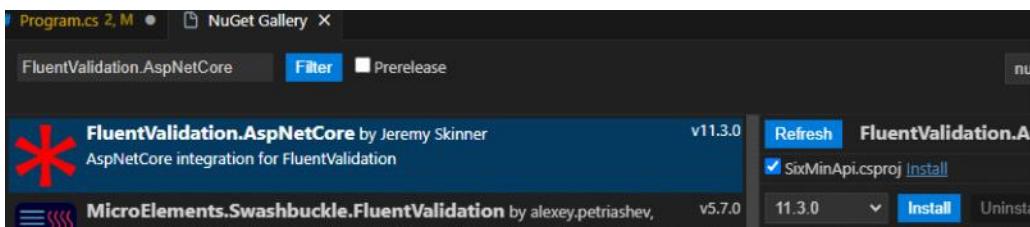
Request URL

<http://localhost:8034/api/people>

Server response

Code	Details
201 undocumented	Response body <pre>{ "id": 3, "fullName": "xxx", "phone": "", "email": "xx" }</pre>
Response headers	
<pre>content-type: application/json; charset=utf-8 date: Thu,14 Sep 2023 04:44:48 GMT location: /api/person/1 server: Kestrel transfer-encoding: chunked</pre>	
Responses	
Code	Description
200	Success

Let we start filtration



Let we add validator

```
C# Program.cs 2, M •
C# Program.cs > ...
354 |
355 |     4 references | You, 8 seconds ago | 1 author (You)
356 |     public class Person
357 |     {
358 |         2 references
359 |         public int Id { get; set; }
360 |         2 references
361 |         public string? FullName { get; set; }
362 |         2 references
363 |         public string? Phone { get; set; }
364 |         2 references
365 |         public string? Email { get; set; }
366 |
367 |         1 reference | You, 1 second ago | 1 author (You)
368 |         public class PersonValidator : AbstractValidator<Person>
369 |         {
370 |             0 references
371 |             public PersonValidator()
372 |             {
373 |                 RuleFor(p => p.Id).GreaterThan(0);
374 |                 RuleFor(p => p.FullName).MaximumLength(255).NotEmpty();
375 |                 RuleFor(p => p.Email).EmailAddress();
376 |                 RuleFor(p => p.Phone)
377 |                     .NotEmpty()
|                     .Must(phone => Regex.IsMatch(phone!, @"^[\+]?([0-9]\ ?){6,14}[0-9]$"))
|                     .WithMessage("Must be a valid phone number");
|             }
|         }
|     }
| }
```

```
public class Person
{
    public int Id { get; set; }
    public string? FullName { get; set; }
    public string? Phone { get; set; }
    public string? Email { get; set; }
    public class PersonValidator : AbstractValidator<Person>
    {
        public PersonValidator()
        {
            RuleFor(p => p.Id).GreaterThan(0);
            RuleFor(p => p.FullName).MaximumLength(255).NotEmpty();
            RuleFor(p => p.Email).EmailAddress();
            RuleFor(p => p.Phone)
                .NotEmpty()
                .Must(phone => Regex.IsMatch(phone!, @"^[\+]?([0-9]\ ?){6,14}[0-9]$"))
                .WithMessage("Must be a valid phone number");
        }
    }
}
```

Let we now register it

```

C# Program.cs 2, M X
C# Program.cs > Program > Main
15
16     var builder = WebApplication.CreateBuilder(args);
17
18     // Add services to the container.
19
20
21     builder.Services
22         .AddControllers()
23         .AddNewtonsoftJson(s =>
24     {
25         s.SerializerSettings.ContractResolver =
26             new CamelCasePropertyNamesContractResolver();
27     });
28     // builder.Services.AddValidatorsFromAssemblyContaining(typeof(Person.PersonValidator));
29     //builder.Services.AddValidatorsFromAssemblyContaining<Program>(ServiceLifetime.Singleton);
30     builder.Services.AddScoped < IValidator < Person > , Person.PersonValidator> ();

```

```
builder.Services.AddScoped < IValidator < Person > , Person.PersonValidator> ();
();
```

Then validation Filter

```

app.MapPost(
    "/api/people",
    async (IValidator<Person> validator, Person person) =>
{
    var validation = await validator.ValidateAsync(person);
    if (validation.IsValid)
    {
        return Results.Created("/api/person/1", person);
    }
    // Fake some saving
    return Results.ValidationProblem(validation.ToDictionary());
}
);

```

Let we test

The screenshot shows the Thunder Client interface. A request is being made to `POST http://localhost:8034/api/people`. The response status is `400 Bad Request`, with a size of `294 Bytes` and a time of `7 ms`. The response body is a JSON object representing validation errors:

```

1 { "type": "https://tools.ietf.org/html/rfc7231#section-6.3.1", "title": "One or more validation errors occurred.", "status": 400, "errors": { "id": [ "id" must be greater than '0' ], "Email": [ "Email" is not a valid email address. ], "Phone": [ "Phone" must not be empty., "Must be a valid phone number" ] } }

```

Validation ok ..let we create generic filtration

```

File Edit Selection View GO Run ... / MINAPI
EXPLORER ... C# Program.cs 2, M C# ValidationFilter.cs U C# CommandRepo.cs
OPEN EDITORS 1 unsaved Data > C# ValidationFilter.cs > ...
C# ValidationFilter.cs Data 2, M
C# ValidationFilter.cs Data U
C# CommandRepo.cs Data
MINAPI
> .config
> vscode
> Binders
> Controllers
> Data
C# ApplicationDbContext.cs
C# CommandRepo.cs
C# ICommandRepo.cs
C# ValidationFilter.cs U
C# Dtos
C# Migrations
C# Models
C# Profiles
C# Properties
people
1 using FluentValidation;
2
3 namespace MinAPI.Data
4 [
5     0 references
6     public class ValidationFilter<T> : IEndpointFilter
7     {
8         0 references
9             public async ValueTask<object?> InvokeAsync(
10                 EndpointFilterInvocationContext ctx,
11                 EndpointFilterDelegate next
12             )
13             {
14                 var validator =
15                     ctx.HttpContext.RequestServices.GetService<IValidator<T>>();
16                 if (validator is not null)
17                 {
18                     var entity =
19                         ctx.Arguments
20                             .OfType<T>()
21                             .FirstOrDefault(a => a?.GetType() == typeof(T));
22                     if (entity is not null)
23                     {
24                         var validation = await validator.ValidateAsync(entity);
25                         if (validation.IsValid)
26                         {
27                             return await next(ctx);
28                         }
29                     }
30                     return Results.ValidationProblem(validation.ToDictionary());
31                 }
32                 else
33                 {
34                     return Results.Problem("Could not find type to validate");
35                 }
36             }
37             return await next(ctx);
38         }
39     }
40 }

```

```

using FluentValidation;
namespace MinAPI.Data
{
    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());
            }
            else
            {
                return Results.Problem("Could not find type to validate");
            }
        }
        return await next(ctx);
    }
}

```

Let we implement extend filtration

From this

```

app.MapPost(
    "/api/people",
    (Person person) =>
{
    // Fake some saving
    return Results.Created("/api/person/1", person);
});

```

To this

```
app.MapPost(
    "/api/people",
    async (Person person) =>
{
    // Fake some saving
    return Results.Created("/api/person/1", person);
}
).AddEndpointFilter<ValidationFilter<Person>>();
```

```
app.MapPost(
    "/api/people",
    (Person person) =>
{
    // Fake some saving
    return Results.Created("/api/person/1", person);
}
).AddEndpointFilter<ValidationFilter<Person>>();
```

Let we test

The screenshot shows the Thunder Client interface. On the left, the activity log lists a POST request to `localhost:8034/api/people` just now. On the right, a detailed view of the POST request to `http://localhost:8034/api/people` shows the following details:

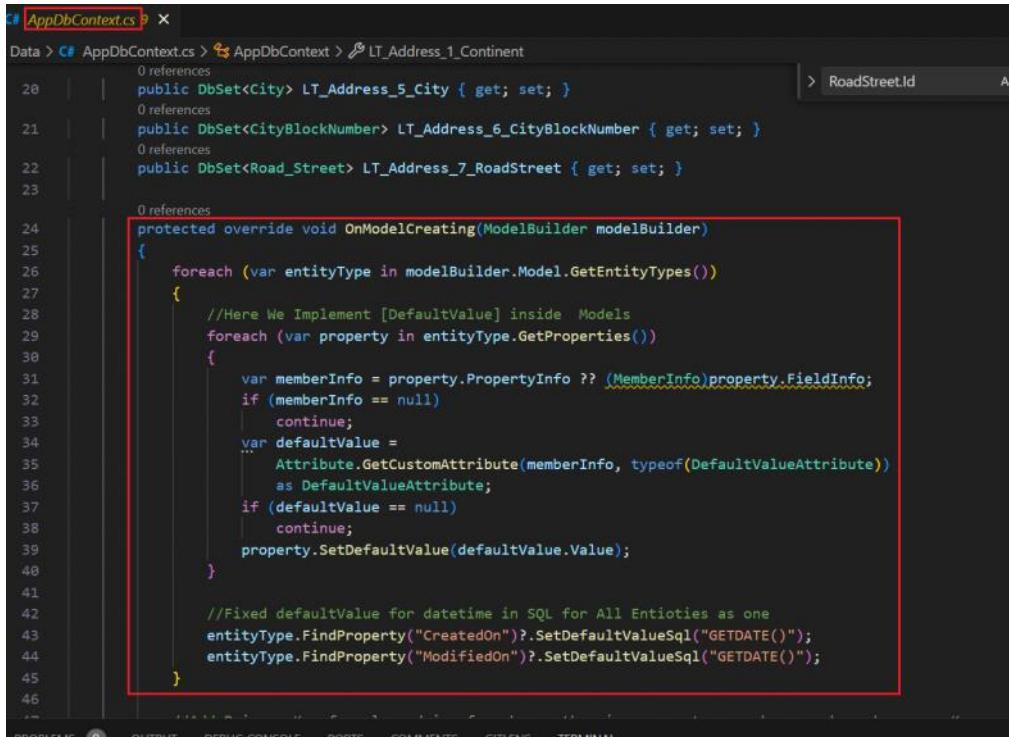
- Method:** POST
- Headers:** Content-Type: application/json
- Body:** JSON content:

```
{ "Id": 1, "fullName": "xxxxxxxxxxxxxx", "phone": "", "email": "a@a.com" }
```
- Response:** Status: 400 Bad Request, Size: 204 Bytes, Time: 11 ms
- Content:** Response body is a JSON object with validation errors:

```
{
  "type": "https://tools.ietf.org/html/rfc7231#err400",
  "title": "One or more validation errors occurred.",
  "status": 400,
  "errors": [
    {
      "Phone": [
        "'Phone' must not be empty."
      ],
      "Must be a valid phone number"
    }
  ]
}
```

We can create One SQL contain all database migration
dotnet ef migrations script -o Data/Seeding_All.sql

We can implement annotation model in migration such as default value



```
AppDbContext.cs
Data > AppDbContext.cs > AppDbContext > LT_Address_1_Continent
20     public DbSet<City> LT_Address_5_City { get; set; }
21     public DbSet<CityBlockNumber> LT_Address_6_CityBlockNumber { get; set; }
22     public DbSet<Road_Street> LT_Address_7_RoadStreet { get; set; }
23
24     protected override void OnModelCreating(ModelBuilder modelBuilder)
25     {
26         foreach (var entityType in modelBuilder.Model.GetEntityTypes())
27         {
28             //Here We Implement [DefaultValue] inside Models
29             foreach (var property in entityType.GetProperties())
30             {
31                 var memberInfo = property.PropertyInfo ?? (MemberInfo)property.FieldInfo;
32                 if (memberInfo == null)
33                     continue;
34                 var defaultValue =
35                     Attribute.GetCustomAttribute(memberInfo, typeof(DefaultValueAttribute))
36                     as DefaultValueAttribute;
37                 if (defaultValue == null)
38                     continue;
39                 property.SetDefaultValue(defaultValue.Value);
40             }
41
42             //Fixed defaultValue for datetime in SQL for All Entities as one
43             entityType.FindProperty("CreatedOn")?.SetDefaultValueSql("GETDATE()");
44             entityType.FindProperty("ModifiedOn")?.SetDefaultValueSql("GETDATE()");
45         }
46     }

```

```
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("GETD
ATE()");
    entityType.FindProperty("ModifiedOn")?.SetDefaultValueSql("GET
DATE()");
}

```

We can even use Fluent in migration

```
# AppDbContext.cs 9 X
Data > C# AppDbContext.cs > AppDbContext > OnModelCreating
29     foreach (var property in entityType.GetProperties())
30     {
31         var memberInfo = property.PropertyInfo ?? (MemberInfo)property.FieldInfo;
32         if (memberInfo == null)
33             continue;
34         var defaultValue =
35             Attribute.GetCustomAttribute(memberInfo, typeof(DefaultValueAttribute))
36             as DefaultValueAttribute;
37         if (defaultValue == null)
38             continue;
39         property.SetDefaultValue(defaultValue.Value);
40     }
41
42     //Fixed defaultValue for datetime in SQL for All Entioties as one
43     entityType.FindProperty("CreatedOn")?.SetDefaultValueSql("GETDATE()");
44     entityType.FindProperty("ModifiedOn")?.SetDefaultValueSql("GETDATE()");
45 }
46
47 //Add Primary Key for class drive from base otherwise you get error because base have noy Key
48 modelBuilder.Entity<BaseLookupTableReference>().HasKey(p => p.Lookup_Code);
49 modelBuilder.Entity<Continent>().HasKey(p => p.Continent_Code);
50 modelBuilder.Entity<Zone>().HasKey(p => p.Zone_Code);
51 modelBuilder.Entity<Country>().HasKey(p => p.Country_Code);
52 modelBuilder.Entity<Governorate>().HasKey(p => p.Governorate_Code);
53 modelBuilder.Entity<City>().HasKey(p => p.City_Code);
54 modelBuilder.Entity<CityBlockNumber>().HasKey(p => p.CityBlock_Code);
55 modelBuilder.Entity<Road_Street>().HasKey(p => p.Road_Code);
56 }
57 }
58 }
```

```
modelBuilder.Entity<BaseLookupTableReference>().HasKey(p =>
p.Lookup_Code);
modelBuilder.Entity<Continent>().HasKey(p => p.Continent_Code);
modelBuilder.Entity<Zone>().HasKey(p => p.Zone_Code);
modelBuilder.Entity<Country>().HasKey(p => p.Country_Code);
modelBuilder.Entity<Governorate>().HasKey(p =>
p.Governorate_Code);
modelBuilder.Entity<City>().HasKey(p => p.City_Code);
modelBuilder.Entity<CityBlockNumber>().HasKey(p =>
p.CityBlock_Code);
modelBuilder.Entity<Road_Street>().HasKey(p => p.Road_Code);
```