# Creating Basic WebAPI

web api

Clear all

All languages | All platforms | All project types

**ASP.NET Core Web API**

A project template for creating a RESTful Web API using ASP.NET Core controllers or minimal APIs, with optional support for OpenAPI and authentication.

C# | Linux | macOS | Windows | API | Cloud | Service | Web | Web API

**ASP.NET Core Web API (native AOT)**

A project template for creating a RESTful Web API using ASP.NET Core minimal APIs published as native AOT.

C# | Linux | macOS | Windows | API | Cloud | Service | Web | Web API

**ASP.NET Core Web API**

A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

F# | Linux | macOS | Windows | Cloud | Service | Web | Web API

# Configure your new project

## ASP.NET Core Web API

C# | Linux | macOS | Windows | API | Cloud | Service | Web | Web API

Project name

WebAPIDemo

Location

D:\Projects

...

Solution name

WebAPIDemo

☐ Place solution and project in the same directory

Project will be created in "D:\Projects\WebAPIDemo\WebAPIDemo\"

# ASP.NET Core Web API

C# | Linux | macOS | Windows | API | Cloud | Service | Web | Web API

Framework

.NET 8.0 (Long Term Support)

Authentication type

None

- [x] Configure for HTTPS
- [ ] Enable container support

Container OS

Linux

Container build type

Dockerfile

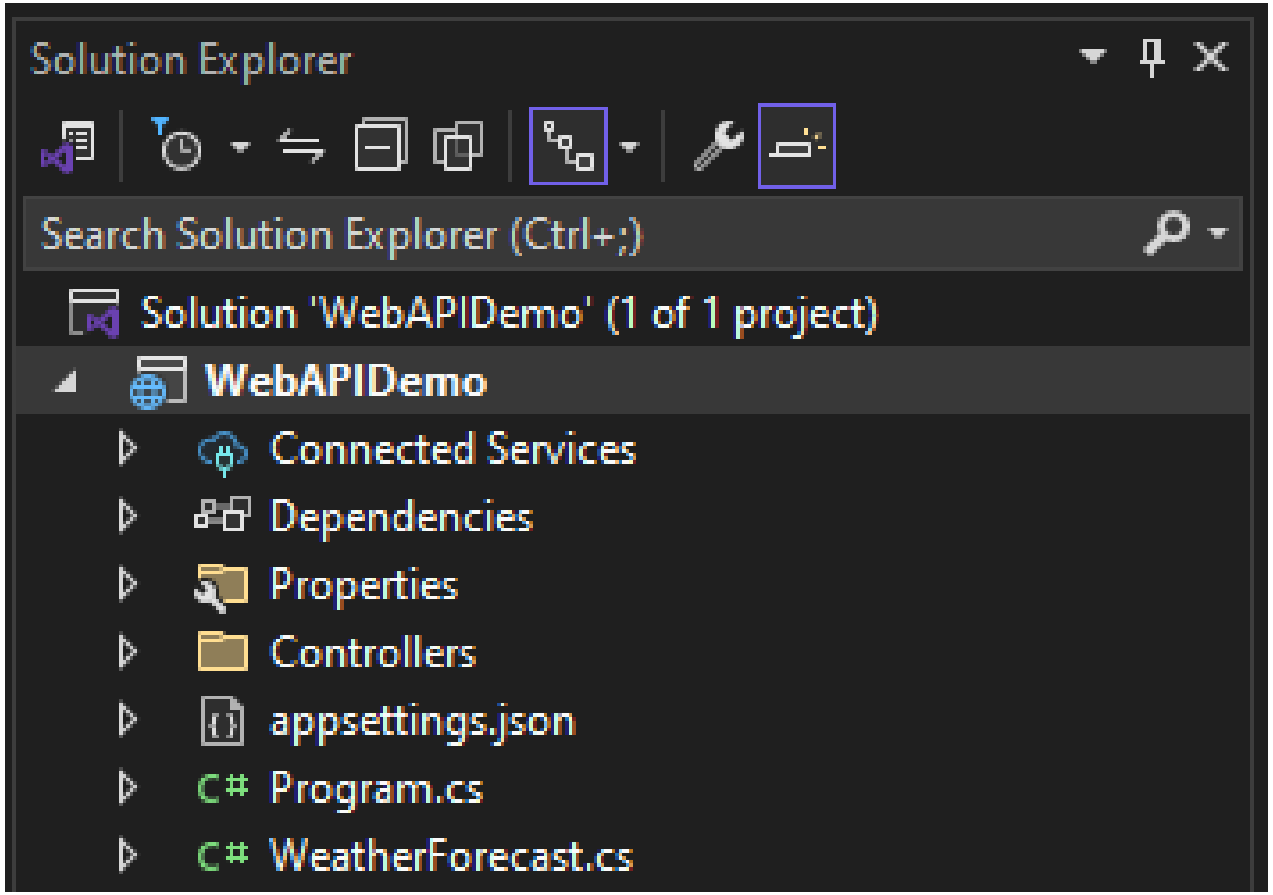
- [x] Enable OpenAPI support
- [x] Do not use top-level statements
- [x] Use controllers

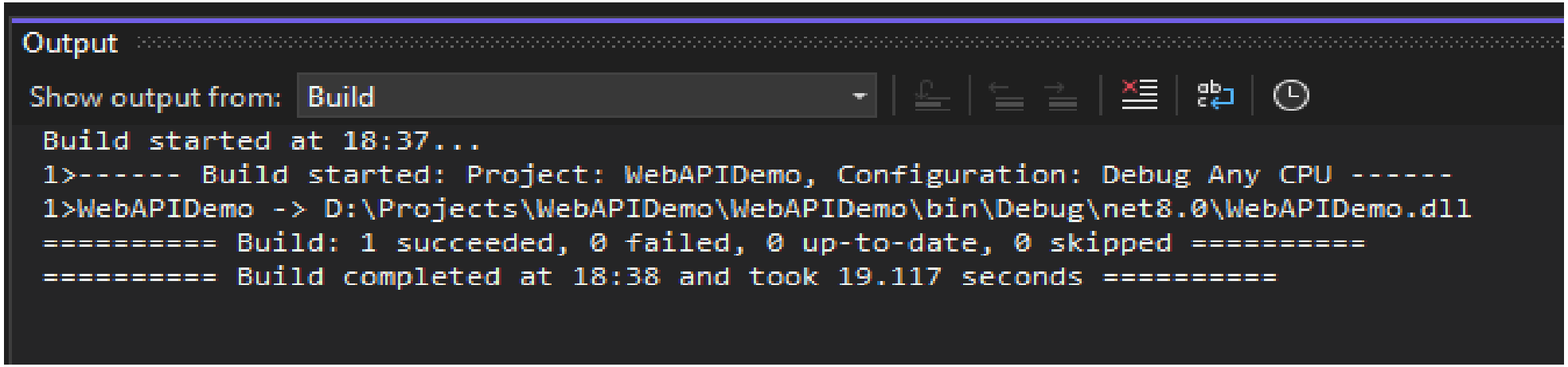
# Options

- **Configure for HTTPS:** This option configures the ASP.NET Core application to use HTTPS (Hypertext Transfer Protocol Secure) by default.
- HTTPS ensures that the communication between the client and server is encrypted, enhancing security.
- When this option is selected, Visual Studio sets up the necessary SSL (Secure Sockets Layer) certificates and configures the application to listen on an HTTPS endpoint.
- **Enable Docker Support:** Selecting this option adds a Dockerfile to the project and configures it for Docker containerization.
- Docker is a platform used for developing, shipping, and running applications inside containers.

# Options

- **Use Controllers:** This option structures the Web API project to use MVC (Model-View-Controller) controllers.
- Controllers are classes that handle incoming HTTP requests and return responses.
- **Enable OpenAPI Support:** OpenAPI, also known as Swagger, is a specification for building APIs.
- Enabling OpenAPI support automatically generates documentation for the API, provides a UI for testing API methods, and helps with client generation.


Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'WebAPIDemo' (1 of 1 project)
WebAPIDemo
Connected Services
Dependencies
Properties
Controllers
appsettings.json
Program.cs
WeatherForecast.cs


Output
Show output from: Build
Build started at 18:37...
1>------ Build started: Project: WebAPIDemo, Configuration: Debug Any CPU ------
1>WebAPIDemo -> D:\Projects\WebAPIDemo\WebAPIDemo\bin\Debug\net8.0\WebAPIDemo.dll
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
========== Build completed at 18:38 and took 19.117 seconds ==========

Swagger UI

localhost:7087/swagger/index.html

Swagger
Supported by SMARTBEAR

# WebAPIDemo 1.0 OAS 3.0

https://localhost:7087/swagger/v1/swagger.json

## WeatherForecast

GET /WeatherForecast

### Schemas

WeatherForecast >

localhost:7087/weatherforecast

Pretty-print


```
[
  {
    "date": "2024-12-13",
    "temperatureC": 46,
    "temperatureF": 114,
    "summary": "Mild"
  },
  {
    "date": "2024-12-14",
    "temperatureC": -1,
    "temperatureF": 31,
    "summary": "Scorching"
  },
  {
    "date": "2024-12-15",
    "temperatureC": -8,
    "temperatureF": 18,
    "summary": "Warm"
  },
  {
    "date": "2024-12-16",
    "temperatureC": -8,
    "temperatureF": 18,
    "summary": "Freezing"
  },
  {
    "date": "2024-12-17",
    "temperatureC": 49,
    "temperatureF": 120,
    "summary": "Balmy"
  }
]
```

# Features of Swagger

- **API Documentation:** Swagger automatically generates interactive API documentation, known as Swagger UI, based on the ASP.NET Core Web API Project.
- This documentation includes details about endpoints, parameters, request and response schemas, and even allows users to try out API calls directly from the documentation interface.
- **Standardization:** Swagger uses the OpenAPI Specification (OAS), which is a widely adopted industry standard for documenting RESTful APIs.
- **Testing and Debugging:** Through Swagger UI, developers can send requests to the API, view the responses, and effectively test API functionality in a user-friendly web interface.

# WeatherForecast

GET /WeatherForecast

**Parameters**

Try it out

No parameters

**Responses**

| Code | Description | Links |
| --- | --- | --- |
| 200 | OK<br>Media type<br>text/plain<br>Controls Accept header. | *No links* |

Execute | Clear

## Responses

Curl

```
curl -X 'GET' \
  'https://localhost:7235/WeatherForecast' \
  -H 'accept: text/plain'
```

Request URL

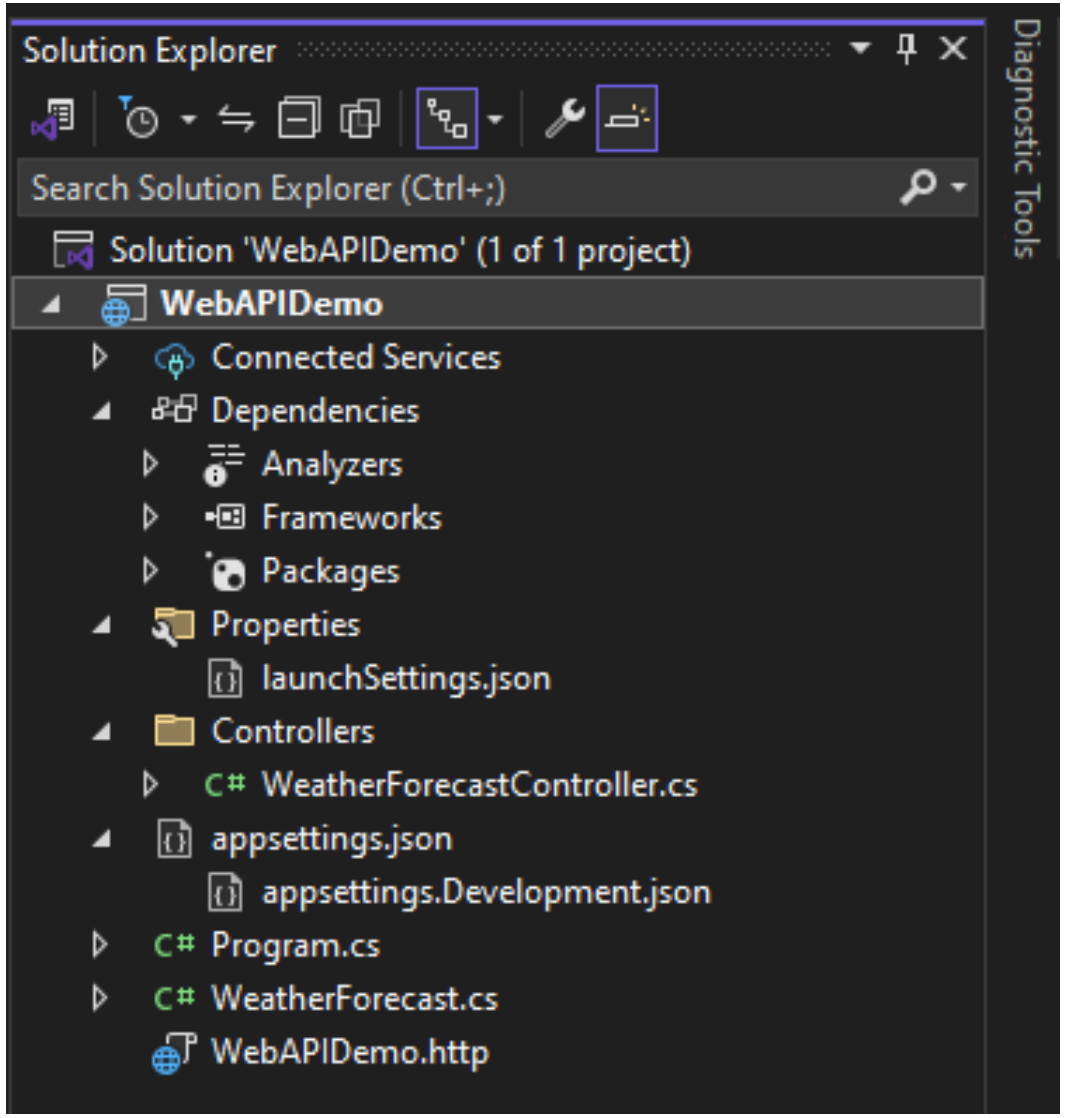
```
https://localhost:7235/WeatherForecast
```

Server response

| Code | Details |
|---|---|
| 200 | Response body |


```
[
  {
    "date": "2024-12-14",
    "temperatureC": -10,
    "temperatureF": 15,
    "summary": "Chilly"
  },
  {
    "date": "2024-12-15",
    "temperatureC": 54,
    "temperatureF": 129,
    "summary": "Warm"
  },
```

# Folders and Files in WebAPI



# Folders and Files in WebAPI

- **Connected Services:**
- This section within the Solution Explorer in Visual Studio allows user to easily connect to external services such as Azure, Office 365, or third-party REST API services.
- **Dependencies:**
- The Dependencies folder contains all the packages and SDKs installed into the project.
- It contains three files (Analyzers, Frameworks, and Packages).

# Folders and Files in WebAPI

- **Analyzers:**
- Analyzers are tools that inspect the code for issues, style violations, or other problems.
- They help maintain code quality by providing suggestions or warnings directly within the IDE.
- **Frameworks:**
- Frameworks within the Dependencies folder refer to the core libraries and components on which the ASP.NET Core Web API project depends.
- This includes the ASP.NET Core framework itself, along with any additional frameworks or runtime libraries required by the project.

# Folders and Files in WebAPI

- **Packages:**
- Packages in the Dependencies folder contain all external libraries and tools that the project utilizes.
- Packages are managed via NuGet, allows user to easily add, update, or remove dependencies as the project evolves.
- **Properties:**
- The Properties Folder in the ASP.NET Core Web API Application, by default, contains one JSON file called launchsettings.json file.
- This launchsettings.json file contains configuration settings for launching the application, such as environment variables and application URLs that will be used by .NET Core Framework.

# Folders and Files in WebAPI

- **Controllers Folder:**
- The ASP.NET Core Web API is a Controller-Based Approach.
- All the controllers of ASP.NET Core Web API Application should and must reside inside the Controllers folder.
- These classes handle incoming HTTP requests, execute the appropriate logic, and return HTTP responses.
- Each controller typically corresponds to a resource or a set of related endpoints. By default, a sample WeatherForecastController is included, demonstrating how to build API endpoints.
- **appsettings.json file:**
- This file is used for configuration settings such as connection strings, API keys, logging settings, and custom configurations.

# Folders and Files in WebAPI

- **Program.cs Class File:**
- This is the entry point of the application.
- It contains the Main method where the application is configured and started.
- Here, it sets up the ASP.NET Core host, configures services, and the middleware pipeline settings for your application.
- **WeatherForecast.cs class file:**
- This is the model class. The model represents the structure and shape of data.