<https://docs.microsoft.com/en-us/learn/modules/build-web-api-aspnet-core/3-exercise-create-web-api>

In Visual Studio Code, select **File** > **Open Folder**.

Create a new folder named **ContosoPizza** in the location of your choice, and then click **Select Folder**.

Open the integrated terminal from Visual Studio Code by selecting **View** > **Terminal** from the main menu.

In the terminal window, copy and paste the following command:

dotnet new webapi -f net6.0

This command creates the files for a basic web API project that uses controllers, along with a C# project file named ContosoPizza.csproj that will return a list of weather forecasts. If you get an error, ensure that you have the [.NET 6 SDK](https://dotnet.microsoft.com/download) installed.

**Important**

Web API projects are secured with https by default. If you have problems, [**configure the ASP.NET Core HTTPS development certificate**](https://docs.microsoft.com/en-us/aspnet/core/security/enforcing-ssl#trust-the-aspnet-core-https-development-certificate-on-windows-and-macos).

You might receive a prompt from Visual Studio Code to add assets to debug the project. Select **Yes** in the dialog.

The command uses an ASP.NET Core project template, aliased as *webapi*, to scaffold a C#-based web API project. A *ContosoPizza* directory is created. This directory contains an ASP.NET Core project running on .NET. The project name matches the *ContosoPizza* directory name.

You should now have access to these files:

-| Controllers

-| obj

-| Properties

-| appsettings.Development.json

-| appsettings.json

-| ContosoPizza.csproj

-| Program.cs

-| WeatherForecast.cs

Examine the following files and directories:

| **TABLE 1** | |
| --- | --- |
| **Name** | **Description** |
| *Controllers/* | Contains classes with public methods exposed as HTTP endpoints |
| *Program.cs* | Configures services and the app's HTTP request pipeline, and contains the app's managed entry point |
| *ContosoPizza.csproj* | Contains configuration metadata for the project |

## Build and test the web API

1. Run the following .NET Core CLI command in the command shell:

dotnet run

* + Locates the project file at the current directory.
  + Retrieves and installs any required project dependencies for this project.
  + Compiles the project code.
  + Hosts the web API with the ASP.NET Core Kestrel web server at both an HTTP and HTTPS endpoint.

A port from 5000 to 5300 will be selected for HTTP, and from 7000 to 7300 for HTTPS, when the project is created. The ports that you use during development can be easily changed by editing the project's launchSettings.json file. This module uses the secure localhost URL that begins with https.

A variation of the following output appears to indicate that your app is running:

Building...

info: Microsoft.Hosting.Lifetime[14]

Now listening on: https://localhost:7294

info: Microsoft.Hosting.Lifetime[14]

Now listening on: http://localhost:5118

info: Microsoft.Hosting.Lifetime[0]

Application started. Press Ctrl+C to shut down.

info: Microsoft.Hosting.Lifetime[0]

Hosting environment: Development

**Important**

Check terminal output if you encounter any unexpected behavior. If the build fails or other errors occur, the log file's information helps you troubleshoot. As you make changes to the code, you'll need to stop the web API by selecting CTRL+C on the keyboard and rerunning the dotnet run command.

Open a web browser and go to:

https://localhost:{PORT}/weatherforecast

The following output represents an excerpt of the JSON that's returned:

[

{

"date": "2021-11-09T20:36:01.4678814+00:00",

"temperatureC": 33,

"temperatureF": 91,

"summary": "Scorching"

},

{

"date": "2021-11-09T20:36:01.4682337+00:00",

"temperatureC": -8,

"temperatureF": 18,

"summary": "Cool"

},

// ...

]

Open a new integrated terminal from Visual Studio Code by selecting **Terminal** > **New Terminal** from the main menu. Then run the following command:

dotnet tool install -g Microsoft.dotnet-httprepl

The preceding command installs the .NET HTTP REPL command-line tool that you'll use to make HTTP requests to the web API.

Connect to the web API by running the following command:

httprepl https://localhost:{PORT}

Alternatively, run the following command at any time while HttpRepl is running:

(Disconnected)> connect https://localhost:{PORT}

Explore available endpoints by running the following command:

ls

The preceding command detects all APIs available on the connected endpoint. It should display the following:

https://localhost:{PORT}/> ls

. []

WeatherForecast [GET]

Go to the WeatherForecast endpoint by running the following command:

cd WeatherForecast

The preceding command shows an output of available APIs for the WeatherForecast endpoint:

https://localhost:{PORT}/> cd WeatherForecast

/WeatherForecast [GET]

Make a GET request in HttpRepl by using the following command:

get

The preceding command makes a GET request similar to going to the endpoint in the browser:

HTTP/1.1 200 OK

Content-Type: application/json; charset=utf-8

Date: Fri, 02 Apr 2021 17:31:43 GMT

Server: Kestrel

Transfer-Encoding: chunked

[

{

"date": 4/3/2021 10:31:44 AM,

"temperatureC": 13,

"temperatureF": 55,

"summary": "Sweltering"

},

{

"date": 4/4/2021 10:31:44 AM,

"temperatureC": -13,

"temperatureF": 9,

"summary": "Warm"

},

// ..

]

End the current HttpRepl session by using the following command:

exit

Return to the dotnet terminal in the drop-down list in Visual Studio Code. Shut down the web API by selecting CTRL+C on your keyboard.

Now that you've created the web API, you'll modify it to meet the needs of the pizza web API.