## **Creating ASP.NET Core Web API project using .NET Core CLI**

**Creating ASP.NET Core Web API Project**

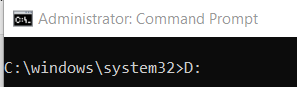
In .NET Core, there are two ways to create a project. They are as follows:

1. **Using .NET Core CLI (Command Line Interface)**
2. **Using Visual Studio 2019**

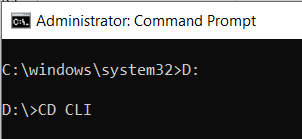
**Note**: In this article, I will show you how to create an ASP.NET Core Web API project using .NET Core CLI (Command Line Interface) and in the next article, I will show you how to create an ASP.NET Core Web API project using Visual Studio 2019.

**ASP.NET Core Web API project using .NET Core CLI**

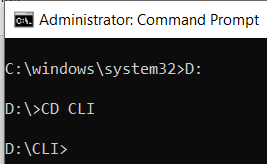
First, we need to select the folder location where we need to create the Web API Project. Suppose we want to create the ASP.NET Core Web API Project in the **D:\CLI** folder. Then open the command prompt in Administrator mode and change the directory location to D drive in the command prompt by typing **D:** and then press the enter key as shown below.



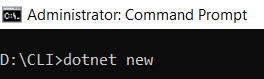
Once you type **D:** and press enter button, then it will change the directory path to the D drive. Then decide inside which folder you to create the project. I am going to create the project inside the CLI folder. So, change the directory location to the CLI folder by typing **CD CLI** and press the enter as shown below.



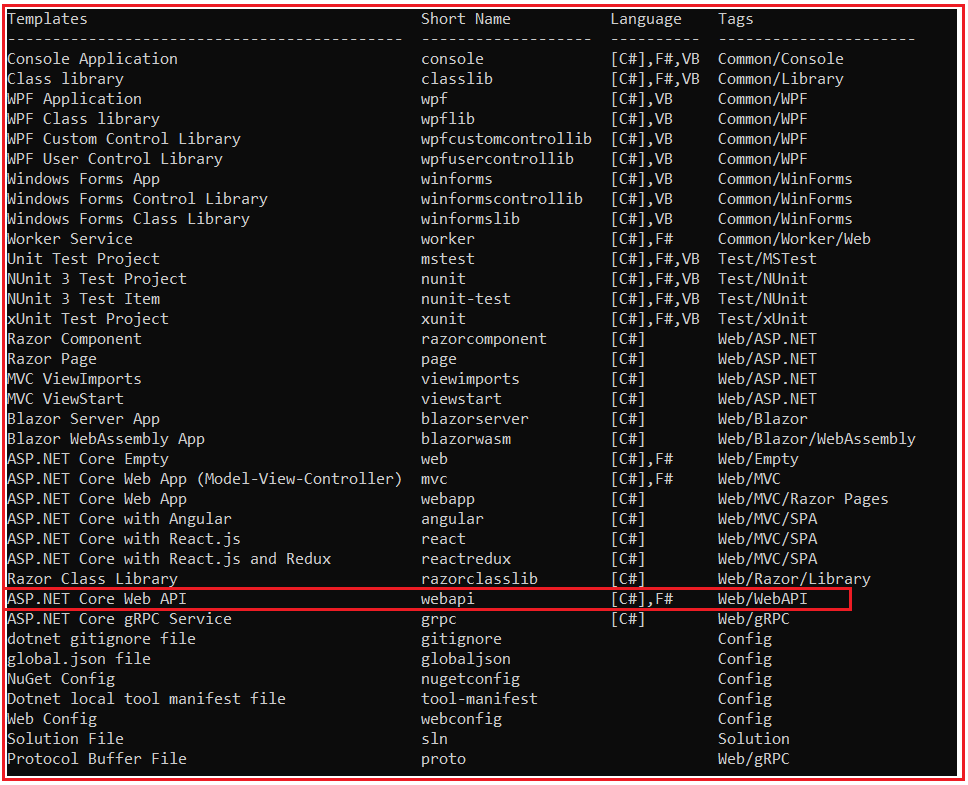
Once you type **CD CLI**and press the enter key, it will change the directory path to the CLI folder as shown in the below image.



Now let see the command to create a new project using .NET Core CLI. First, type **dotnet new** and press the enter button as shown in the below image.



Once you type **dotnet new** and press the enter button, it will display the following. As you can see in the below image, we have the templates and the short name for the template as well as the default language for the template.

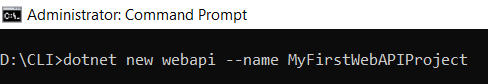


As you can see in the above image, it provides one template called ASP.NET Core Web API, and to create that project we can use the command webapi (Short Name of the template). The default programming language for this Web API Project is C# and apart from C# language, you can also use F# language.

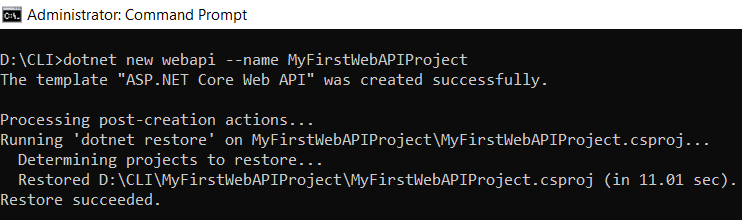
Let us create the web API project. You can create a web API project using two ways. They are as follows:

1. **dotnet new webapi:** If you only type **dotnet new webapi**, then a new project will be created inside the CLI folder with the default name.
2. **dotnet new webapi –name MyFirstWebAPIProject**: With this command, the .NET Core CLI will create a project inside the CLI folder with the name MyFirstWebAPIProject.

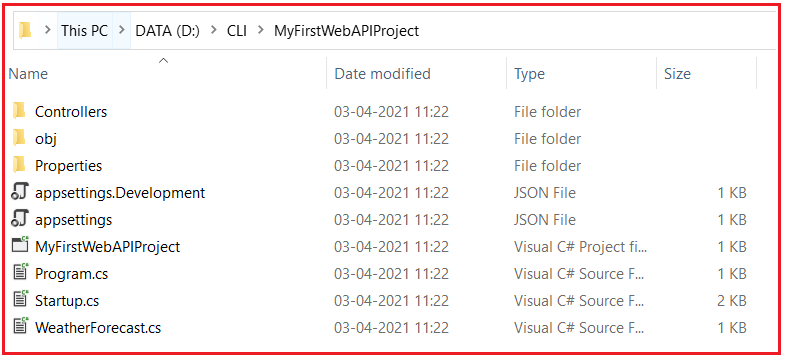
So, let us use the second option to create the ASP.NET Core Web API Project. Type **dotnet new webapi –name MyFirstWebAPIProject** command and press enter in the command prompt as shown in the below image.



Once you type dotnet new webapi –name MyFirstWebAPIProject and press enter, you will get the below output saying that the template ASP.NET Core Web API was created successfully.

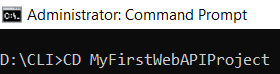


Now, inside the CLI Folder, you can see a folder with your project name (in my case the folder name is **MyFirstWebAPIProject**) and inside that folder, you can see the following default project files and folders provided by the ASP.NET Core Web API template as shown in the below image.

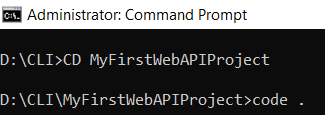


**Opening ASP.NET Core Web API project in Visual Studio Code:**

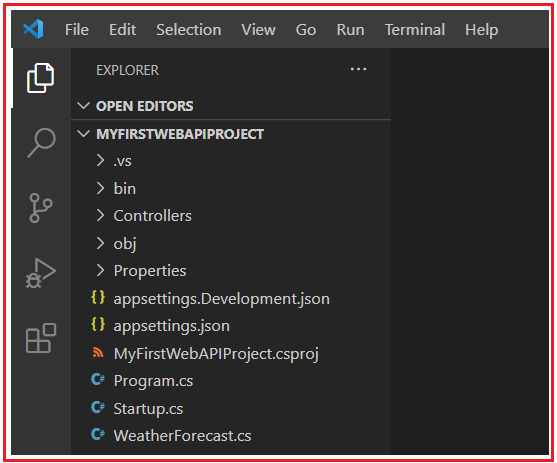
Now let us see how to open the above project in Visual Studio Code. To do so, first change the directory in the command prompt to the project directory (CD MyFirstWebAPIProject) as shown below.



Once you type CD MyFirstWebAPIProject and press the enter button, it will change the directory. Then to open the project files and folders in Visual Studio Code, type “**code .**” (code space dot) and press the enter button as shown in the below image.



Once you type “**code .**” (code space dot) and press the enter button, it will open the project in Visual Studio code as shown in the below image.



In our upcoming articles, we will discuss each of the above files and folders in detail.

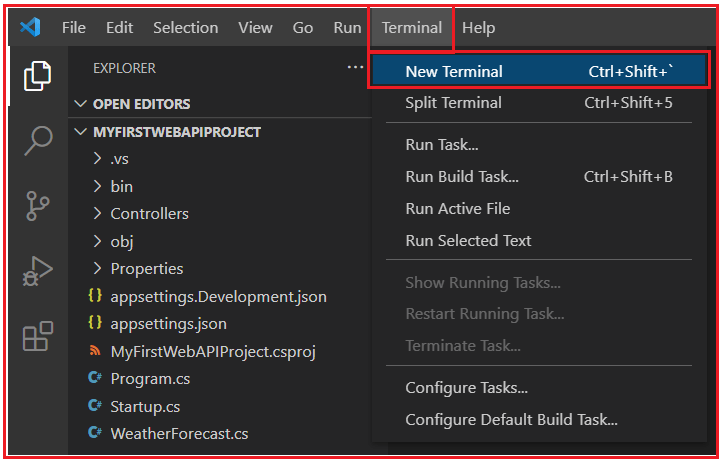
**How to build an ASP.NET Core Web API project using .NET Core CLI?**

Now let us see how to build an ASP.NET Core Web API project using .NET Core CLI. You can build the project using two ways.

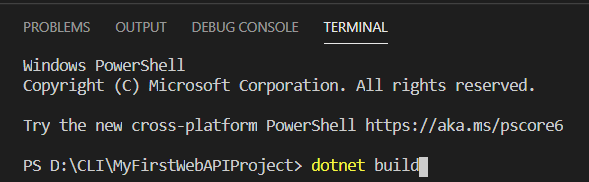
1. **Using Visual Studio Code Terminal**
2. **Using Command Prompt**

**Build ASP.NET Core Web API Project Using Visual Studio Code Terminal**

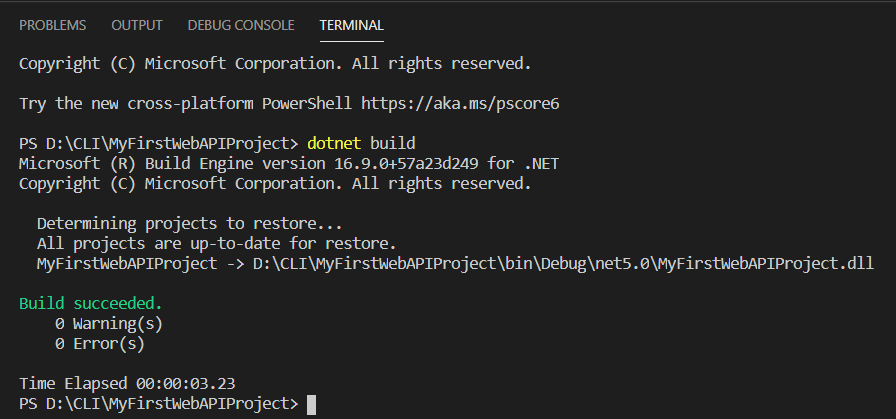
First, open a new terminal. To do so, click on the **Terminal** Menu and then select the **New Terminal** option as shown in the below image.



Once you click on the New Terminal option, it will open the terminal window as shown in the below image. In the terminal, type **dotnet build and press the enter button** as shown in the below image. This command is used to build an existing project.



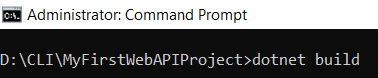
Once you type dotnet build and press the enter button, it will build the project and you get the following message.



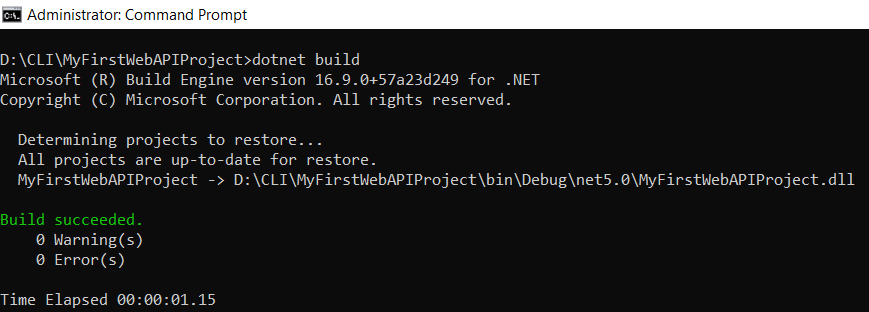
As you can see in the above image, the build is successful and there is no error and no warning.

**Build ASP.Net Core Web API Project Using Command Prompt:**

In the command prompt, first set the directory to your project folder and then type dotnet build and press the enter button as shown in the below image.

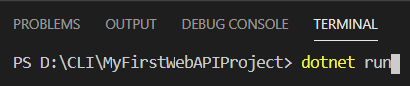


Once you type dotnet build and press the enter button, it will build your project and you will get the below message.

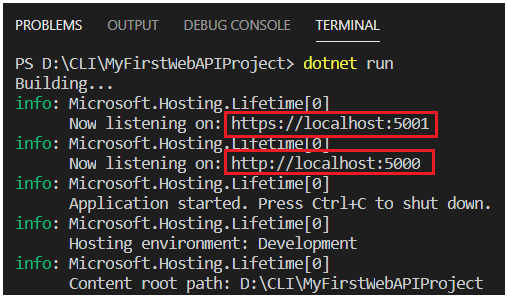


**How to run the ASP.NET Core Web API project using .NET Core CLI?**

Now let us see how to run the above ASP.NET Core Web API Project. The .NET Core CLI provides the run command to run the ASP.NET Core Web API Application. So, in the terminal type dotnet run and press the enter button as shown in the below image.

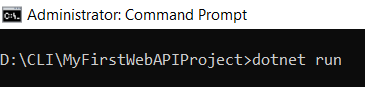


Once you type dotnet run and press the enter button, you will get the below message.

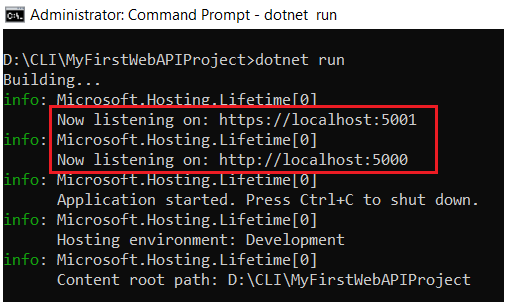


**How to run ASP.NET Core Web API Project using Command Prompt?**

In the command prompt, first set the directory to your project folder and then type **dotnet run** command and press the enter button as shown in the below image.



Once you type dotnet run and press the enter button, it will run your project and you will get the below message.

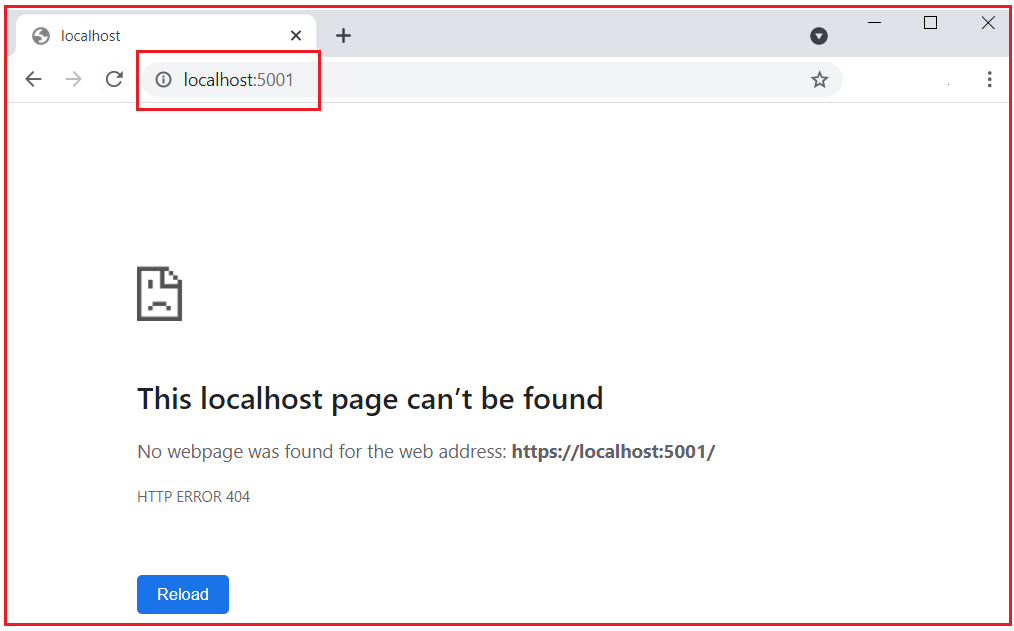


As you can see in the above image, our ASP.NET Core Web API Application is running on two different ports. They are as follows:

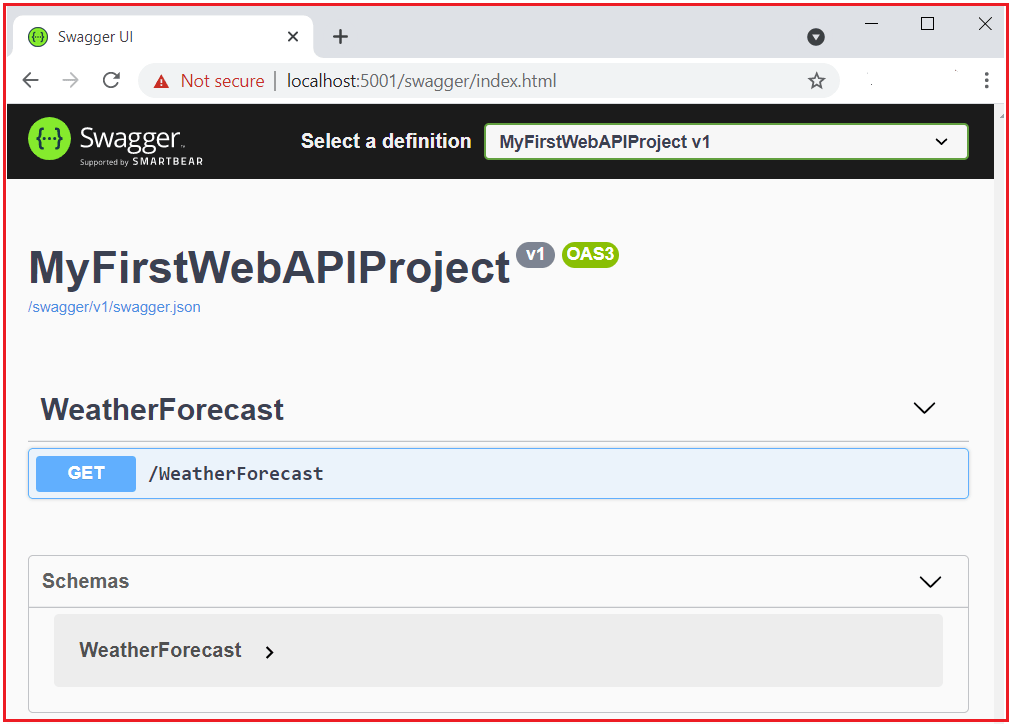
**https://localhost:5001**

**http://localhost:5000**

Now, open the above URL in any of your browsers and you will get a 404 error.



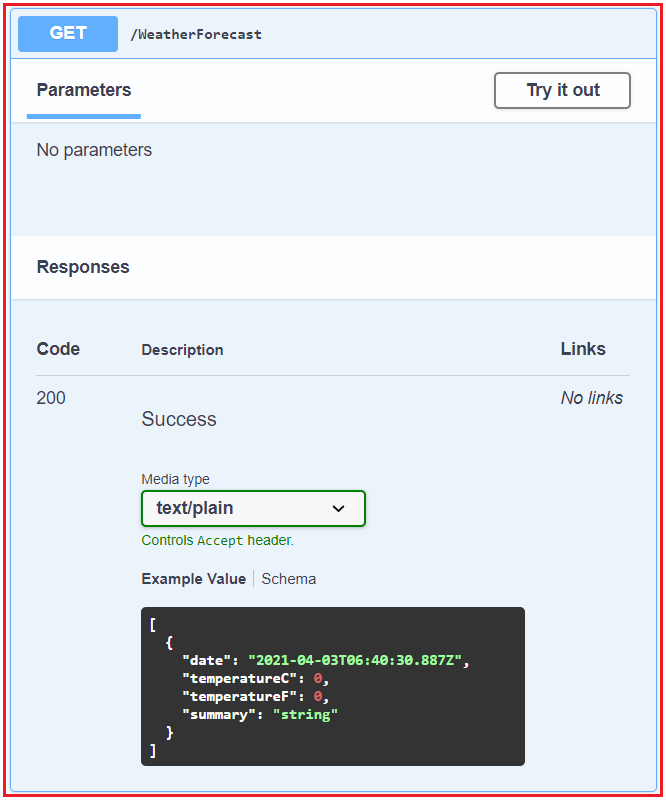
Don’t worry. Just type **swagger** at the end of the URL and press enter and you will get the following webpage.



The swagger will display the details of all the Web APIs available in your project. As you can see in the above image, it showing one API i.e. /WeatherForecast and the type is Get. Now click on the /WeatherForecast API to see details as shown in the below image.



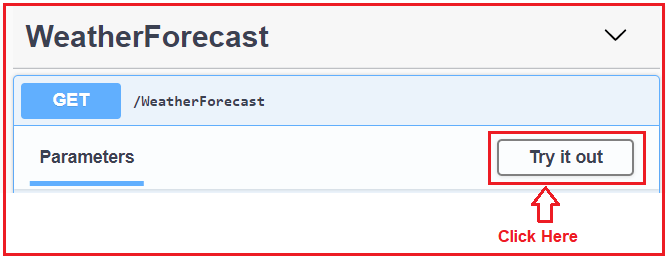
Once you click on the /WeatherForecast API, then it will show you the details of this API as shown in the below image.



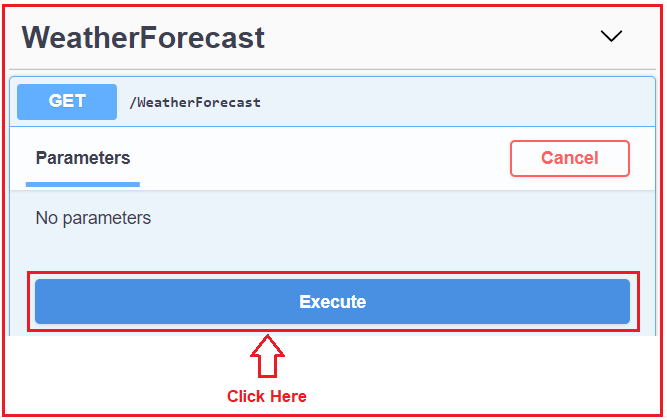
Note: Swagger is also a Client API Tool and using swagger we can also test the Web APIs. If you are using the default ASP.NET Core Web API project, then by default swagger is installed into the project.

**Test ASP.NET Core Web API using Swagger:**

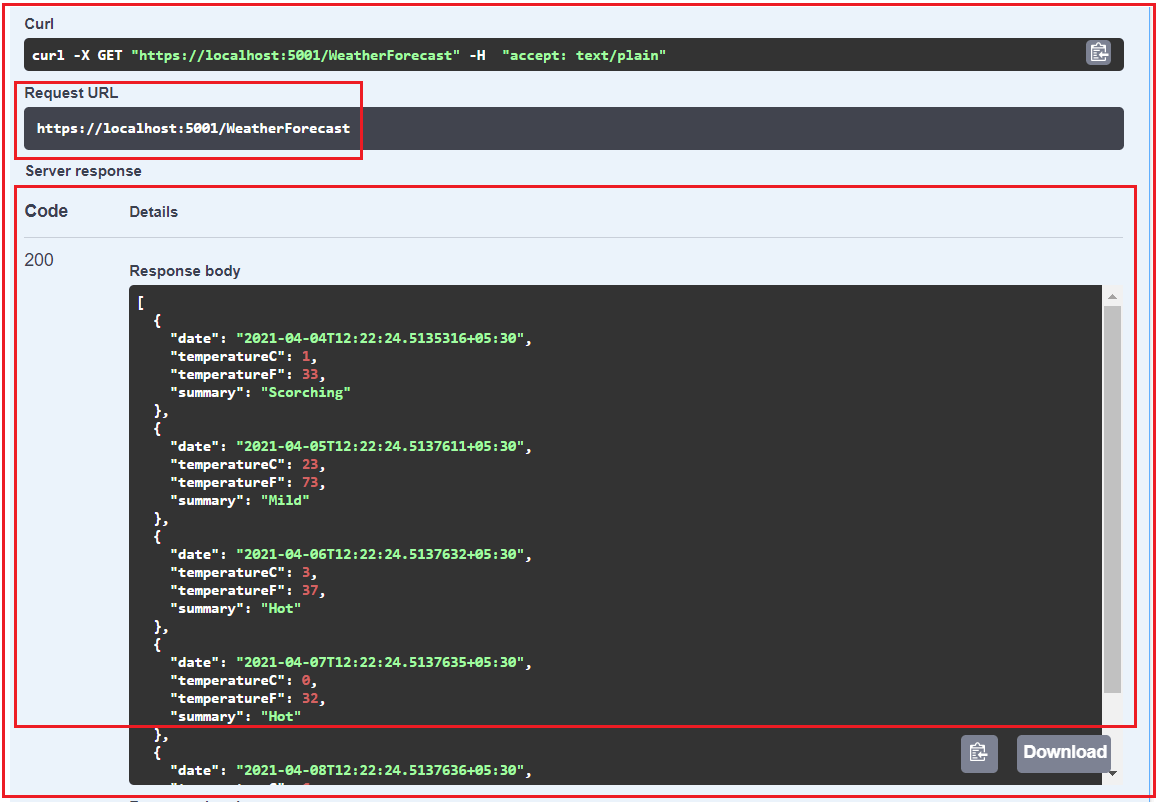
Now let us see how to test the API i.e. WeatherForecast API using swagger. To test the API using swagger, first, click on the **try it out** button as shown in the below image.



Once you click on the Try it Out button, it will open the below, and again here click on the Execute button as shown in the below image.



Once you click on the Execute button, it will give you the response as shown in the below image. Here, you can find the request URL, the response body, the response status code, and response headers.



In our upcoming article, we will discuss where this response is coming from. Even if you want you directly call the Request URL in any of your browsers and you will get the response as expected as shown in the below image.

