

# Web API trong Asp.net Core

[Hướng dẫn tự học lập trình ASP.NET Core toàn tập](#) > [Web API trong Asp.net Core](#)

Bắt đầu từ bài học này chúng ta chuyển sang một dạng ứng dụng web hơi khác biệt, trong đó Asp.net Core chạy trên server giờ chỉ đóng vai trò cung cấp dữ liệu ở dạng JSON hoặc XML. Loại chương trình Asp.net Core như thế được gọi là Web API. Web API trong Asp.net Core chia sẻ phần lớn kỹ thuật với MVC framework mà bạn vừa học.

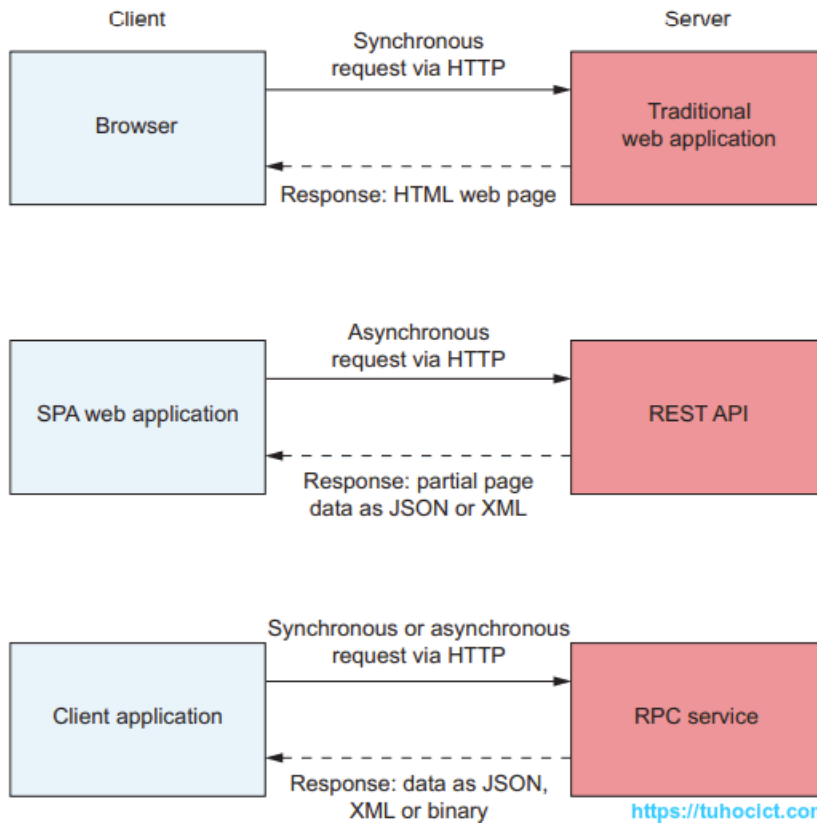
## NỘI DUNG CỦA BÀI [ Ấn ]

1. Ứng dụng client/server qua HTTP
2. Web API trong Asp.net Core
3. Tạo dự án Asp.net Core Web API
  - 3.1. Sử dụng Visual Studio
  - 3.2. Sử dụng dotnet CLI
4. Tạo API Controller đầu tiên
5. Khai thác dịch vụ Web API
6. Kết luận

## Ứng dụng client/server qua HTTP

Trong các phần trước của tập tài liệu bạn đã học cách xây dựng *ứng dụng web truyền thống* sử dụng hai framework MVC và Razor Pages. Điểm chung của loại ứng dụng web này là giao diện người dùng (HTML + JS + CSS) được server sinh ra và gửi cho trình duyệt qua HTTP. Asp.net Core sử dụng Razor template cho nhiệm vụ này.

Tuy nhiên hiện nay rất phổ biến các loại ứng dụng trong đó phần giao diện người dùng là một ứng dụng độc lập chạy trên một thiết bị khác và tương tác với ứng dụng server qua giao thức HTTP.



Phần giao diện người dùng được gọi là client. Client rất đa dạng, bao gồm ứng dụng đơn trang SPA (Single-Page Application, chạy trong trình duyệt), ứng dụng desktop truyền thống, ứng dụng mobile. Thậm chí một ứng dụng web khác cũng có thể trở thành client.

**Ứng dụng đơn trang** (Single-Page Application, SPA) trong những năm qua trở nên rất phổ biến. Hàng loạt framework do các hãng lớn phát triển dành cho phát triển SPA client như React (Facebook), Angular (Google).

Trong SPA, thành phần client chạy trên trình duyệt và được xây dựng hoàn toàn bằng JavaScript. Thành phần server gửi cho trình duyệt code JavaScript một lần lúc trình duyệt truy xuất ứng dụng. Sau đó SPA và server chỉ còn trao đổi dữ liệu (qua HTTP) ở dạng JSON hoặc XML.

**Ứng dụng mobile** cũng vô cùng phổ biến trong nhiều năm qua. Do đặc thù của thiết bị di động, việc xử lý và lưu trữ dữ liệu của các loại ứng dụng này thường không thực hiện ngay trên thiết bị. Thay vào đó, chúng được thực hiện trên một server. Ứng dụng mobile cùng truy xuất server theo cách tương tự như ứng dụng SPA qua HTTP.

**Ứng dụng desktop** hiện nay, nhất là các ứng dụng business, cũng theo xu hướng sử dụng một server chung để lưu trữ và xử lý dữ liệu. Loại ứng dụng desktop triển khai theo mô hình này tiện lợi trong mạng cục bộ và thậm chí có thể hoạt động qua Internet.

Thành phần server xây dựng bằng Asp.net Core được gọi là Web API. Web API chịu trách nhiệm cung cấp dữ liệu, chủ yếu ở dạng JSON và XML, cho client. Web API cung cấp một loạt URL mà client có thể sử dụng để truy xuất hoặc thay đổi dữ liệu trên server qua HTTP.

Một điểm đặc biệt là cùng một Web API có thể đồng thời được sử dụng bởi SPA, mobile, desktop và các ứng dụng khác. Điều này dẫn đến Web API trở thành một phần vô cùng quan trọng trong các hệ thống ứng dụng hiện đại.

Web API có thể triển khai song song với một ứng dụng web truyền thống hoặc có thể hoạt động hoàn toàn độc lập phục vụ các ứng dụng client khác nhau.

## Web API trong Asp.net Core

---

Khi xây dựng Web API trong Asp.net Core, về cơ bản bạn tiếp tục sử dụng MVC framework với một số thay đổi về kỹ thuật.

Điều này có nghĩa là khi xây dựng Web API, bạn xây dựng hoặc sử dụng các thành phần Model như trong ứng dụng MVC. Controller trong Web API chính là Controller của MVC.

Trong Asp.net cổ điển, MVC và Web API là hai framework khác nhau. Trong Asp.net Core bạn sử dụng cùng một framework MVC để tạo ra cả ứng dụng web truyền thống và web API.

Do vậy bạn tiếp tục làm việc với những khái niệm quen thuộc từ MVC như action, routing, model binding, model validation, view model.

Sự khác biệt lớn nhất khi sử dụng MVC để tạo ra Web API nằm ở chỗ: thành phần View của MVC giờ được thay thế bằng JSON hoặc XML. Thay vì sử dụng các Razor template để sinh HTML, action trong Web API sẽ chuyển đổi dữ liệu thành chuỗi JSON hoặc XML. Quá trình chuyển đổi này được gọi là serialization.

Có thể hình dung rằng thành phần View của MVC trong Web API giờ không còn dành cho người dùng (user-friendly) nữa mà là dành cho chương trình khác sử dụng (machine-friendly).

Như vậy, Web API trong Asp.net Core không phải là một framework riêng rẽ.

Ứng dụng Web API có thể triển khai song song bên cạnh ứng dụng MVC hoặc có thể xây dựng hoàn toàn độc lập.

Tuy nhiên cần lưu ý, mỗi loại client có cách khác nhau để sử dụng dịch vụ do Web API cung cấp.

## Tạo dự án Asp.net Core Web API

---

### Sử dụng Visual Studio

Chúng ta bắt đầu bằng một bài thực hành: tạo dự án Asp.net Core Web Api đầu tiên trong **Visual Studio**.

**Bước 1.** Tạo một ứng dụng Asp.net Core Web Application trong Visual Studio theo các thông tin cấu hình sau:

**1**

C# All platforms Web

C# Windows Web Test

**ASP.NET Core Web Application**  
Project templates for creating ASP.NET Core web apps and web APIs for Windows, Linux and macOS using .NET Core or .NET Framework. Create web apps with Razor Pages, MVC, or Single Page Apps (SPA) using Angular, React, or React + Redux.

C# Linux macOS Windows Cloud Service Web

**ASP.NET Core Web Application** C# Linux macOS Windows Cloud Service Web

**2**

Project name  
HelloWebApi

Location  
Z:\OneDrive\Learn ASP NET Core\

Solution name **1**  
WebApi

☐ Place solution and project in the same directory

.NET Core ASP.NET Core 3.1

**3**

**C# Empty**  
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.

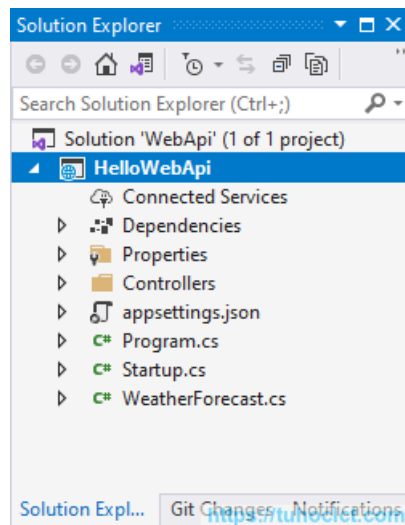
**C# API**  
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

**C# Web Application**  
A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.

**C# Web Application (Model-View-Controller)**  
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.

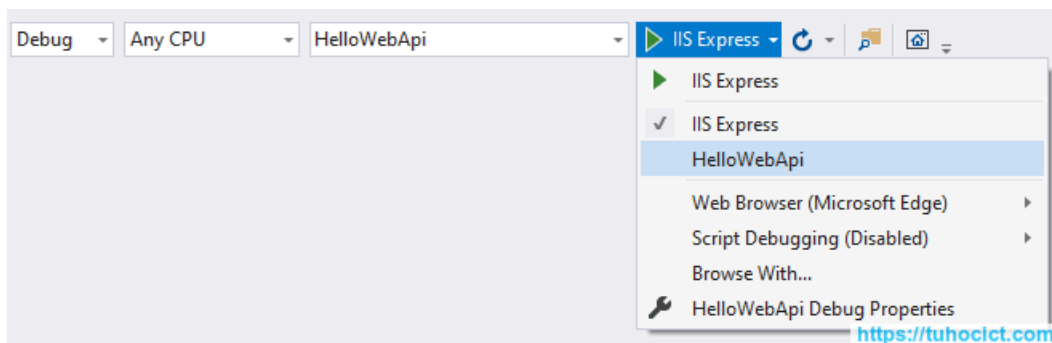
<https://tuhocict.com>

Bạn thu được một project với cấu trúc như sau:



**Bước 2.** Chạy thử ứng dụng mới.

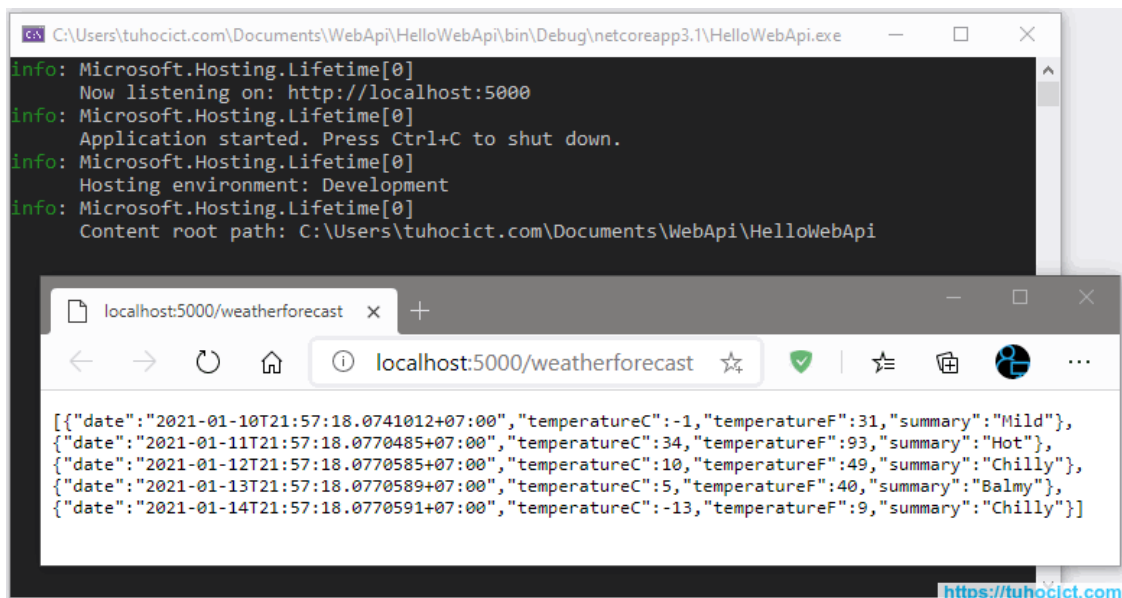
Bạn có thể lựa chọn chạy ứng dụng tích hợp với IIS Express hoặc trong ứng dụng Console.



**Bước 3.** Mở trình duyệt và nhập vào URL `localhost:5000/weatherforecast`

Lưu ý: nếu chạy ứng dụng console, cổng mặc định là 5000 hoặc 5001 (https); nếu chạy cùng IIS Express, cổng mặc định là 50040. Các giá trị này có thể thay đổi trong file Properties/launchSettings.json.

Bạn thu được kết quả như sau:



Hãy để ý, kết quả bạn thu được giờ là một chuỗi JSON chứ không phải là trang web bình thường nữa.

## Sử dụng dotnet CLI

Tương tự như với dự án Razor Pages và MVC, bạn cũng có thể tạo dự án Web API từ **dotnet CLI**. Hãy cùng thực hiện.

**Bước 1.** Mở command prompt hoặc power shell và chuyển thư mục (lệnh CD) đến thư mục làm việc mong muốn.

**Bước 2.** Gõ lệnh sau: `dotnet new webapi -f netcoreapp3.1 -o HelloWebApi2`

Kết quả chạy lệnh trên như sau:

```
The template "ASP.NET Core Web API" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on HelloWebApi2\HelloWebApi2.csproj...
  Determining projects to restore...
  Restored C:\Users\tuhocict.com\Documents\WebApi\HelloWebApi2\HelloWebApi2.csproj
  (in 93 ms).
Restore succeeded.
```

Ở đây bạn đã yêu cầu dotnet CLI tạo dự án Web API mới trong thư mục HelloWebApi2 và sử dụng framework .NET Core 3.1. Nếu sử dụng framework .NET 5 mới, bạn cần thay

`-f netcoreapp3.1` bằng tham số `-f net5.0`.

Khi này bạn có thể chạy luôn dự án từ giao diện dòng lệnh (`dotnet run -p .\HelloWebApi2\`), hoặc mở trong Visual Studio và chạy debug như trường hợp 1. Bạn sẽ thu được cùng kết quả như trường hợp 1.

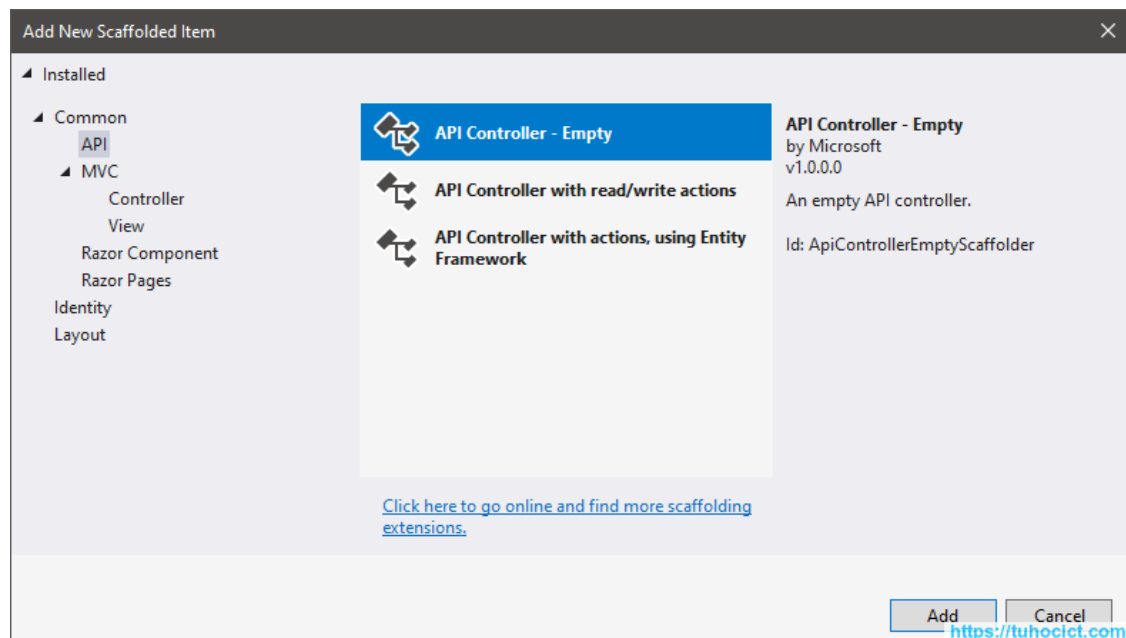
# Tạo API Controller đầu tiên

Ở phần trước chúng ta đã tạo được dự án Web API đầu tiên. Tuy nhiên chúng ta chưa thay đổi bất cứ nội dung gì. Tiếp theo đây chúng ta sẽ tạo API Controller riêng đầu tiên.

Giả sử bạn sử dụng Visual Studio.

**Bước 1.** Click phải vào thư mục Controller => Chọn Add => Controller ...

**Bước 2.** Chọn node Common / API (bên trái) và chọn mục API Controller – Empty bên khung phải.



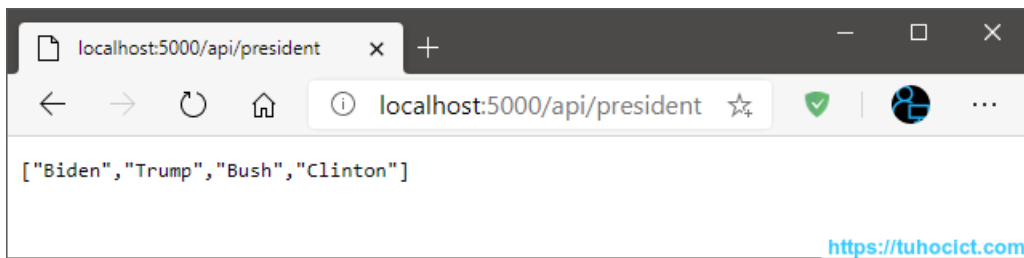
Đặt tên cho file code mới là PresidentController.cs

**Bước 3.** Viết code cho PresidentController.cs như sau:

```
1. using Microsoft.AspNetCore.Mvc;
2. using System.Collections.Generic;
3.
4. namespace HelloWebApi.Controllers {
5.     [Route("api/[controller]")]
6.     [ApiController]
7.     public class PresidentController : ControllerBase {
8.         IEnumerable<string> _presidents = new List<string> {
9.             "Biden", "Trump", "Bush", "Clinton"
10.        };
11.
12.        public IEnumerable<string> Index() {
13.            return _presidents;
14.        }
15.    }
16. }
```

**Bước 4.** Chạy debug và mở Url sau trong trình duyệt `http://localhost:5000/api/president`

Bạn thu được kết quả như sau:



Hãy để ý mấy vấn đề sau:

- Bạn tiếp tục sử dụng các class trong namespace `Microsoft.AspNetCore.Mvc` như trong dự án MVC.
- Chúng ta đang sử dụng **attribute routing** (`[Route("api/[controller]")]`) với `PresidentController` ;
- Lớp `PresidentController` được đánh dấu với attribute `[ApiController]` ;
- Lớp `PresidentController` kế thừa từ `ControllerBase` ;
- Action Index có kiểu trả về là `IEnumerable<T>` chứ không phải `IActionResult` như thường lệ trong MVC.

Trong các bài học tiếp theo chúng ta sẽ lần lượt phân tích các vấn đề trên.

## Khai thác dịch vụ Web API

Ở phần trên, khi chạy debug ứng dụng web API bạn đã mở xem thông tin trên trình duyệt. Hẳn bạn đã thấy, dữ liệu bạn nhận được chỉ là một chuỗi JSON.

Đối với người dùng đầu cuối, chuỗi JSON này không có nhiều tác dụng.

Trên thực tế, dữ liệu trả về của web API không hướng tới người dùng cuối mà dành cho chương trình khác sử dụng. Điều này có nghĩa là (1) bạn không thể đơn thuần khai thác Web API trực tiếp qua trình duyệt như đối với ứng dụng web truyền thống, và (2) bạn phải xây dựng chương trình client riêng để nhận và diễn giải dữ liệu.

Tương tự như vậy, nếu bạn có các yêu cầu tương tác khác với dữ liệu thông qua Web API (như thêm / sửa / xóa), bạn không thể trực tiếp thực hiện trên trình duyệt. Bạn phải xây dựng chương trình riêng để giúp người dùng đầu cuối làm việc này.

Như đã trình bày ở phần đầu tiên của bài, chương trình client của Web API rất đa dạng, bao gồm cả ứng dụng desktop, SPA, mobile.

Để minh họa việc khai thác dữ liệu từ Web API, chúng ta sẽ xây dựng một ứng dụng console rất đơn giản.

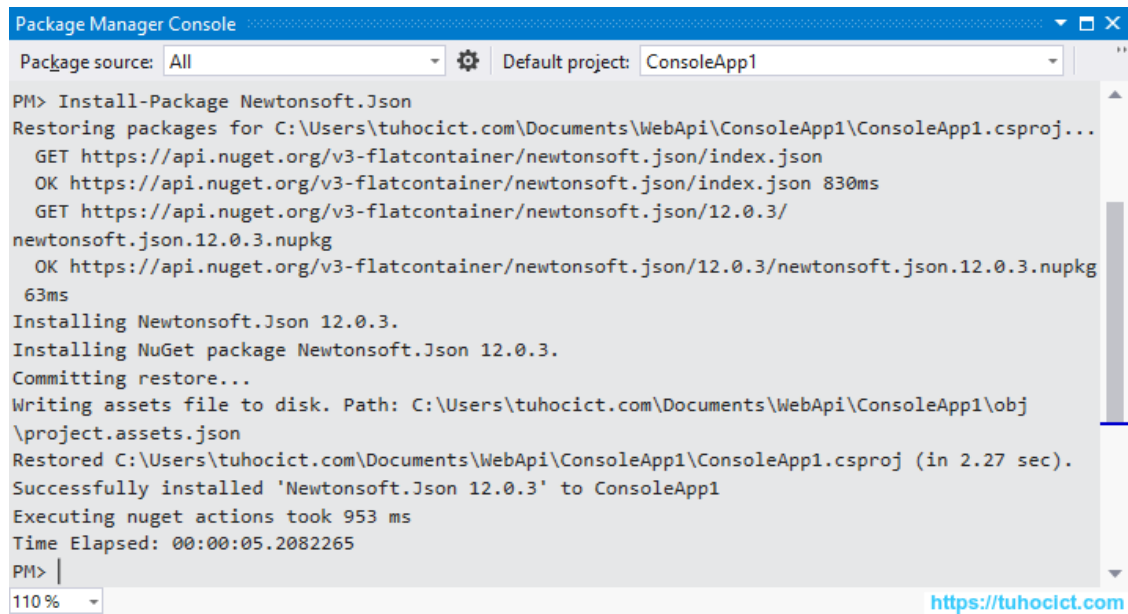
**Bước 1.** Tạo dự án Console trong cùng solution với `HelloWebApi`

**Bước 2.** Cài đặt gói thư viện `NewtonSoft.Json` cho dự án Console.



Bạn có thể sử dụng giao diện đồ họa Nuget Package Manager của Visual Studio hoặc qua giao diện dòng lệnh Nuget Package Manager Console.

Nếu sử dụng Package manager console, hãy sử dụng lệnh `Install-Package Newtonsoft.Json`.



```
Package Manager Console
Package source: All Default project: ConsoleApp1
PM> Install-Package Newtonsoft.Json
Restoring packages for C:\Users\tuhocict.com\Documents\WebApi\ConsoleApp1\ConsoleApp1.csproj...
GET https://api.nuget.org/v3-flatcontainer/newtonsoft.json/index.json
OK https://api.nuget.org/v3-flatcontainer/newtonsoft.json/index.json 830ms
GET https://api.nuget.org/v3-flatcontainer/newtonsoft.json/12.0.3/
newtonsoft.json.12.0.3.nupkg
OK https://api.nuget.org/v3-flatcontainer/newtonsoft.json/12.0.3/newtonsoft.json.12.0.3.nupkg
63ms
Installing Newtonsoft.Json 12.0.3.
Installing NuGet package Newtonsoft.Json 12.0.3.
Committing restore...
Writing assets file to disk. Path: C:\Users\tuhocict.com\Documents\WebApi\ConsoleApp1\obj\project.assets.json
Restored C:\Users\tuhocict.com\Documents\WebApi\ConsoleApp1\ConsoleApp1.csproj (in 2.27 sec).
Successfully installed 'Newtonsoft.Json 12.0.3' to ConsoleApp1
Executing nuget actions took 953 ms
Time Elapsed: 00:00:05.2082265
PM> |
```

Chú ý chọn Default project là console app bạn đang cần cài.

Tương tự, cài đặt gói thư viện sau cho console:

```
Install-Package System.Net.Http.Formatting.Extension
```

### Bước 3. Viết code như sau cho Program.cs

```
1. using System;
2. using System.Net.Http;
3. using System.Net.Http.Headers;
4. using System.Threading.Tasks;
5.
6. namespace ConsoleApp1 {
7.     class Program {
8.         static void Main(string[] args) {
9.
10.             RunAsync().Wait();
11.             Console.ReadKey();
12.         }
13.
14.         static async Task RunAsync() {
15.             using var client = new HttpClient {
16.                 BaseAddress = new Uri("http://localhost:5000")
17.             };
18.             client.DefaultRequestHeaders.Accept.Clear();
19.             client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue(
20.
21.                 HttpResponseMessage response = await client.GetAsync("/weatherforecast");
22.                 if (response.IsSuccessStatusCode) {
23.                     var data = await response.Content.ReadAsAsync<WeatherForecast[]>();
24.                     Console.WriteLine($"Records retrieved: {data.Length}");
25.                     foreach (var d in data) {
26.                         Console.WriteLine($"{d.Date}\t{d.TemperatureC}\t{d.Summary}");
27.                     }
28.                 }
29.             }
30.         }
31.
32.         public class WeatherForecast {
33.             public DateTime Date { get; set; }
34.             public int TemperatureC { get; set; }
```

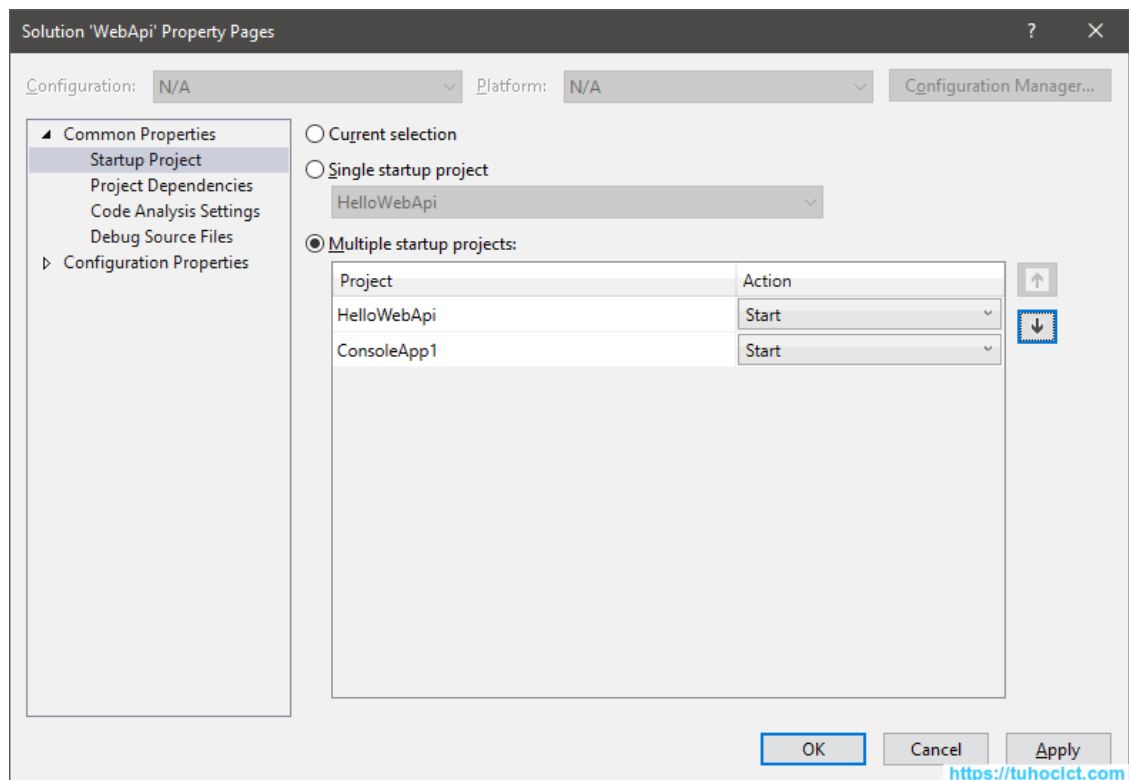
```

35.         public int TemperatureF => 32 + (int) (TemperatureC / 0.5556);
36.         public string Summary { get; set; }
37.     }
38. }

```

**Bước 4.** Thiết lập để có thể debug đồng thời cả HelloWorldApi và Console

Click phải vào tên solution chọn Properties. Điều chỉnh trong hộp thoại như dưới đây:



**Bước 5.** Chạy debug và thu được kết quả như sau:

The screenshot shows a console window with the following output:

```

Records retrieved: 5
1/13/2021 2:12:23 PM 43 Hot
1/14/2021 2:12:23 PM 45 Bracing
1/15/2021 2:12:23 PM 54 Scorching
1/16/2021 2:12:23 PM 1 Sweltering
1/17/2021 2:12:23 PM 28 Sweltering

```

A watermark 'https://tuhocict.com' is visible at the bottom right.

Như bạn thấy, trong .NET, để khai thác dịch vụ của Web API bạn phải sử dụng HttpClient – thư viện chuyên cho làm việc với giao thức HTTP.

Tùy thuộc vào từng công nghệ khi xây dựng client, bạn phải sử dụng các loại kỹ thuật khác nhau.

Ở phần sau của loạt bài học này chúng ta sẽ điểm qua một số công nghệ phổ biến.

- + Nếu bạn thấy site hữu ích, trước khi rời đi hãy **giúp đỡ** site bằng một hành động nhỏ để site có thể phát triển và phục vụ bạn tốt hơn.
- + Nếu bạn thấy bài viết hữu ích, hãy giúp **chia sẻ** tới mọi người.

+ Nếu có thắc mắc hoặc cần trao đổi thêm, mời bạn viết trong phần **thảo luận** cuối trang.  
Cảm ơn bạn!

## Kết luận

---

Trong bài học này chúng ta bắt đầu làm quen với Web API:

- Web API được sử dụng làm thành phần server cho nhiều loại ứng dụng khác nhau (SPA, mobile, desktop);
- Web API trao đổi dữ liệu ở dạng JSON hoặc XML với client qua HTTP;
- Trong Asp.net Core, Web API được xây dựng trên MVC framework chứ không phải là một framework riêng;
- Dự án Asp.net Core Web API cũng chính là dự án MVC nhưng có một số thay đổi về kỹ thuật.
- Để khai thác Web API bạn phải xây dựng client riêng.