# BÀI 1 GIỚI THIỆU VỀ ASP.NET WEB API

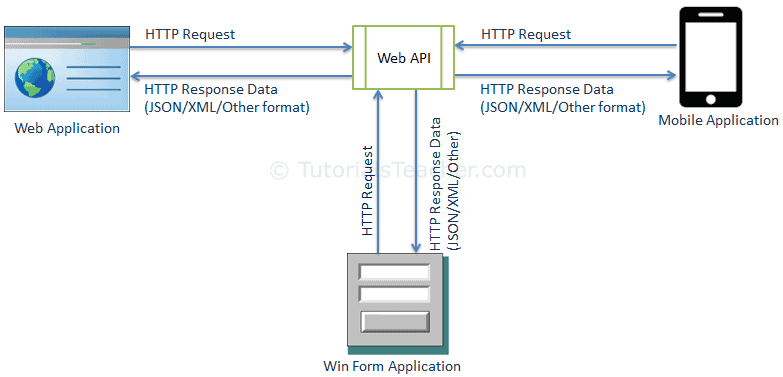
## API là gì?

**API** là từ viết tắt của “Application Programming Interface” với nghĩa là một giao diện lập trình ứng dụng. Đây là phần mềm trung gian cho phép kết nối 2 ứng dụng với nhau.

## Web API là gì?

**Web** **API** **là** một **API** trên **web** có thể được truy cập bằng giao thức HTTP. Nó **là** một khái niệm và không phải **là** một công nghệ.

Các dịch vụ Web API có thể được sử dụng bởi nhiều loại clients khác nhau thông qua các request. Ví dụ: Browsers, Mobile applications, Desktop applications, IOTs, …

[](https://www.tutorialsteacher.com/Content/images/webapi/webapi-overview.png)

Hình 1.1 Web API

**Cấu trúc của 1 request**

|  |
| --- |
| URL |
| Method |
| Header |
| Body |

Hình 1.2 Cấu trúc request

Trong đó:

URL là 1 cái địa chỉ duy nhất

Method: là cái hành động client muốn tác động lên “resources”, và nó thường là động  
từ. Có 4 loại Method (Create – POST, Read – GET, Update – PUT, Delete – DELETE) được dùng:  
– GET: Yêu cầu server đưa lại data   
– POST: Yêu cầu server cho tạo ra 1 resource mới.– PUT: Yêu cầu server cho sửa / thêm vào resource đã có trên hệ thống.   
– DELETE: Yêu cầu server cho xóa 1 resourse.

Header: nơi chứa các thông tin cần thiết của 1 request nhưng end-users không biết có sự tồn tại của nó. Ví dụ: độ dài của request body, thời gian gửi request, loại thiết bị đang sử  
dụng, loại định dạng cái response mà client có đọc được…

Body: nơi chứa thông tin mà client sẽ điền. Giả sử bạn đặt 1 cái bánh pizza, thì thông tin  
ở phần body sẽ là: Loại bánh pizza, kích cỡ, số lượng đặt.

*Response:*Sau khi nhận được request từ phía client, server sẽ xử lý cái request đó và gửi ngược lại cho client 1 cái response.

**Cấu trúc của 1** *Response*

|  |
| --- |
| Status Code |
| Header |
| Body |

Hình 1.3 Cấu trúc Response

Trong đó

Status code là những con số có 3 chữ số và có duy nhất 1 ý nghĩa. Chắc các bạn cũng  
không còn lạ lẫm với những Error “404 Not Found” hoặc “503 Service Unavailable”. Full  
list có ở đây.

Header và Body tương đối giống với request.

## REST và RESTful API

REST ( **RE** trình bày **S** tate **T** ransfer) về cơ bản là một kiểu kiến ​​trúc để phát triển ứng dụng có một số nguyên tắc như sau:

* Nó nên không quốc tịch
* Nó sẽ truy cập tất cả các tài nguyên từ máy chủ chỉ bằng URI
* Nó không có mã hóa sẵn có
* Nó không có phiên
* Nó sử dụng một và chỉ một giao thức - HTTP
* Để thực hiện thao tác CRUD, cần sử dụng HTTP động từ như get, post, putvàdelete
* Nó sẽ chỉ trả về kết quả dưới dạng JSON hoặc XML, nguyên tử, OData, v.v. (dữ liệu nhẹ)

Trong đó

* REST based services Thực hiện theo một số nguyên tắc trên và không phải tất cả
* RESTFUL services có nghĩa là nó tuân theo tất cả các nguyên tắc trên.

## APS.NET Web API

### APS.NET Web API là gì

The **ASP.NET Web API** là một framework có khả năng mở rộng để xây dựng các dịch vụ dựa trên HTTP, nó có thể thể truy cập trong các ứng dụng khác nhau trên các nền tảng khác nhau như web, windows, mobile, v.v.

Cách thức hoạt động của **ASP.NET Web API** hầu như giống như cách thức ứng dụng web ASP.NET MVC, **ASP.NET Web API** hoạt động khác ASP.NET MVC ở chỗ nó gửi dữ liệu dưới dạng các respond (phản hồi) thay vì trả lại các html view.

**ASP.NET Web API** giống như một webservice hoặc WCF service nhưng nó khác là nó chỉ hỗ trợ giao thức HTTP.

Các dịch vụ Web API có thể được sử dụng bởi nhiều loại clients khác nhau thông qua các request. Ví dụ: Browsers, Mobile applications, Desktop applications, IOTs, …

### Đặc điểm của ASP.NET Web API

* Là platform lý tưởng cho việc xây dựng RESTful services.
* Kiến trúc lý tưởng cho các thiết bị có băng thông giới hạn như các thiết bị di động.
* Giúp cho việc xây dựng các HTTP service rất đơn giản, nhanh chóng, maps các HTTPGet/Post/Put/Delete với tên của phương thức
* Mã nguồn mở (Open Source) và có thể được sử dụng bởi bất kì client nào hỗ trợ XML, JSON.
* Hỗ trợ đầy đủ các thành phần HTTP: URI, request/response headers, caching, versioning, content formats.
* Có thể host trong ứng dụng hoặc trên IIS, Self-hosted hoặc web server (yêu cầu .NET 4.0 trở lên)
* Hỗ trợ nhiều định dạng dữ liệu khác nhau như có thể là JSON, XML, BSON hoặc một kiểu dữ liệu bất kỳ.

### Các phiên bản của ASP.NET Web API

| Web API Version | Supported .NET Framework | Coincides with | Supported in |
| --- | --- | --- | --- |
| Web API 1.0 | .NET Framework 4.0 | ASP.NET MVC 4 | VS 2010 |
| Web API 2 - Current | .NET Framework 4.5 | ASP.NET MVC 5 | VS 2012, 2013 |

### So sánh ASP.NET Web API và WCF

| Web API | WCF |
| --- | --- |
| Open source and ships with .NET framework. | Ships with .NET framework |
| Supports only HTTP protocol. | Supports HTTP, TCP, UDP and custom transport protocol. |
| Maps http verbs to methods | Uses attributes based programming model. |
| Uses routing and controller concept similar to ASP.NET MVC. | Uses Service, Operation and Data contracts. |
| Does not support Reliable Messaging and transaction. | Supports Reliable Messaging and Transactions. |
| Web API can be configured using HttpConfiguration class but not in web.config. | Uses web.config and attributes to configure a service. |
| Ideal for building RESTful services. | Supports RESTful services but with limitations. |

### 1.4.5. Khi nào nên chọn ASP.NET Web API?

* Sử dụng .NET framework 4.0 trở lên.
* Xây dựng webservice chỉ hỗ trợ giao thức HTTP protocol.
* Xây dựng webservice RESTful HTTP based services.
* Biết về ASP.NET MVC

## Tạo ứng dụng Hello Web API

### Tạo dự án ASP.NET Web API sử dụng Visual Studio

Các bước xây dựng một dự án ASP.NET Web API như sau

**Bước 1:**Đầu tiên mở ứng dụng Visual Studio

Khởi động Visual Studio → tạo một project ASP.NET Web Application và chọn template Web API.

**Bước 2:**

Tạo Data Model sử dụng Entity Framework để Web API service có thể tương tác CRUD (Create, Read, Update, Delete) dữ liệu được (không tạo cũng được). Mỗi bảng là 1 Controller

**Bước 3:**Tạo ra các Web API

Nhấp chuột phải vào thư mục Controllers và chọn thêm controller.

**Web API 2 Controller Empty:**tự viết các phương thức từ đầu.

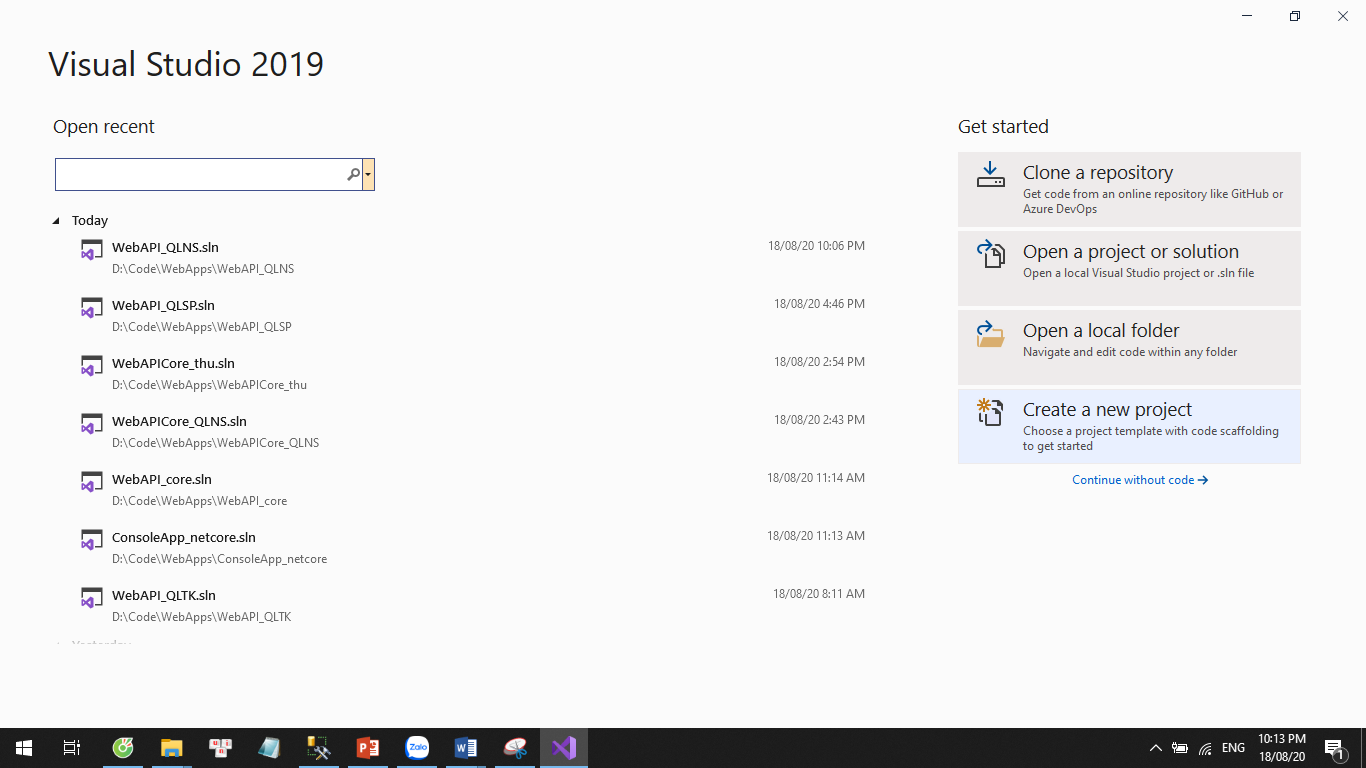
**Web API 2 Controller with read/write actions**: phát sinh các phương thức ví dụ để bạn có thể biết cách viết các service này.

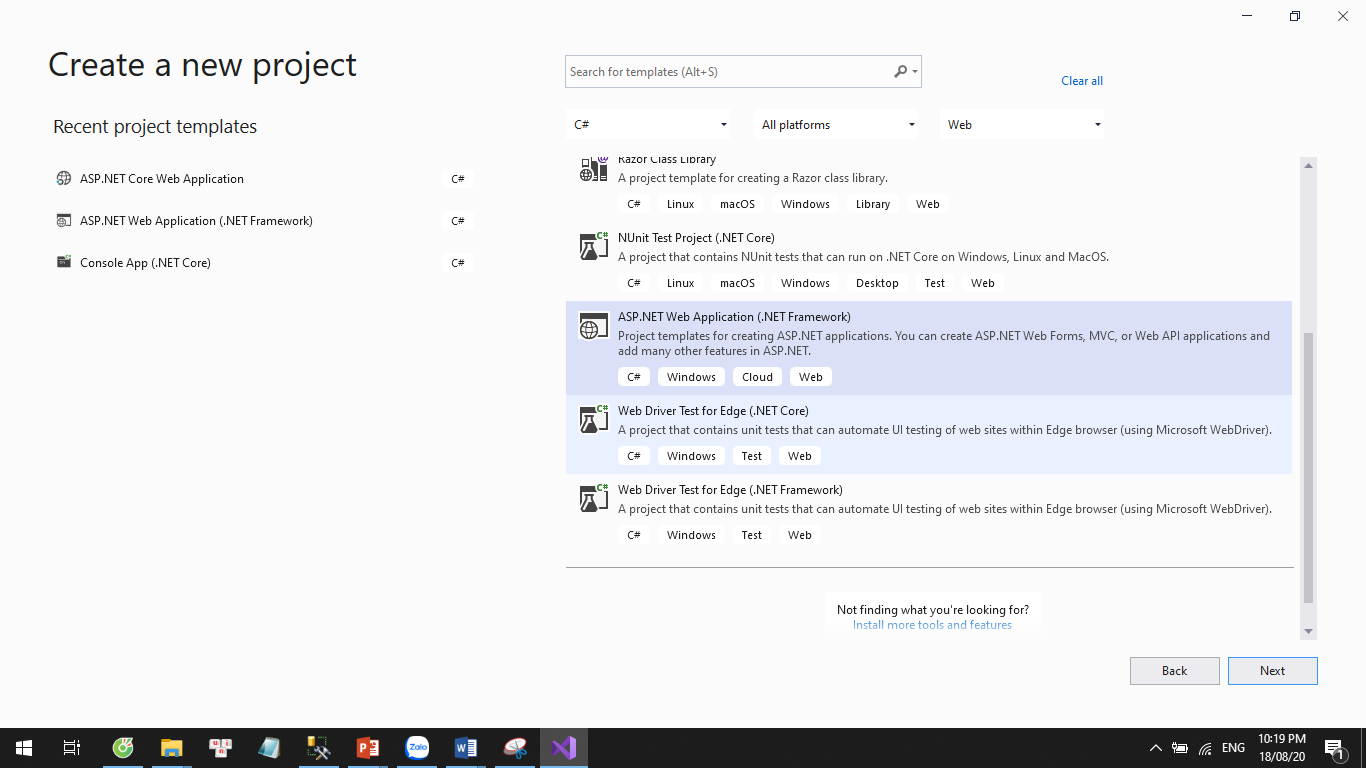
**Bước 4:**Chạy thử và kiểm tra

Ta thực hiện xây dựng ứng dụng Hello Web API như sau

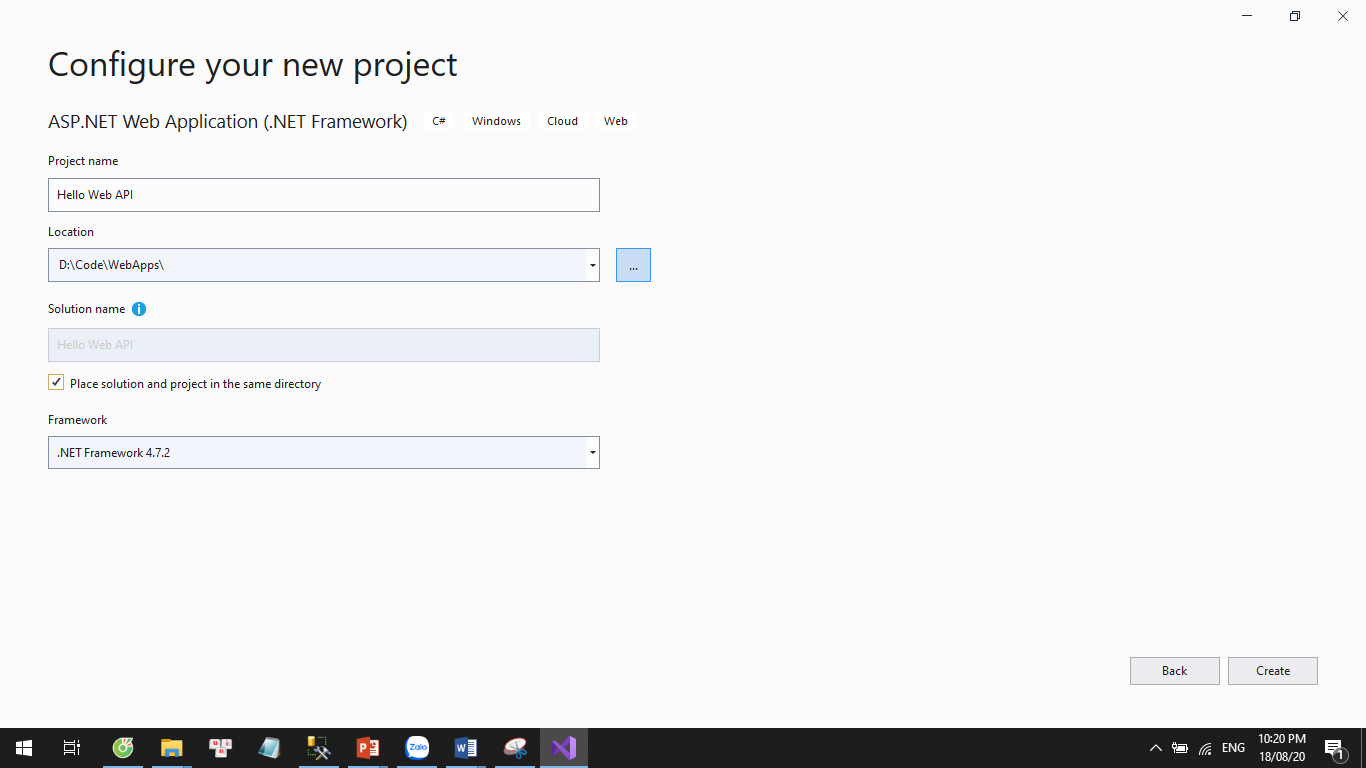
#### **Bước 1:** Khởi động Visual Studio

Chọn Create a new project để tạo một project ASP.NET Web Application

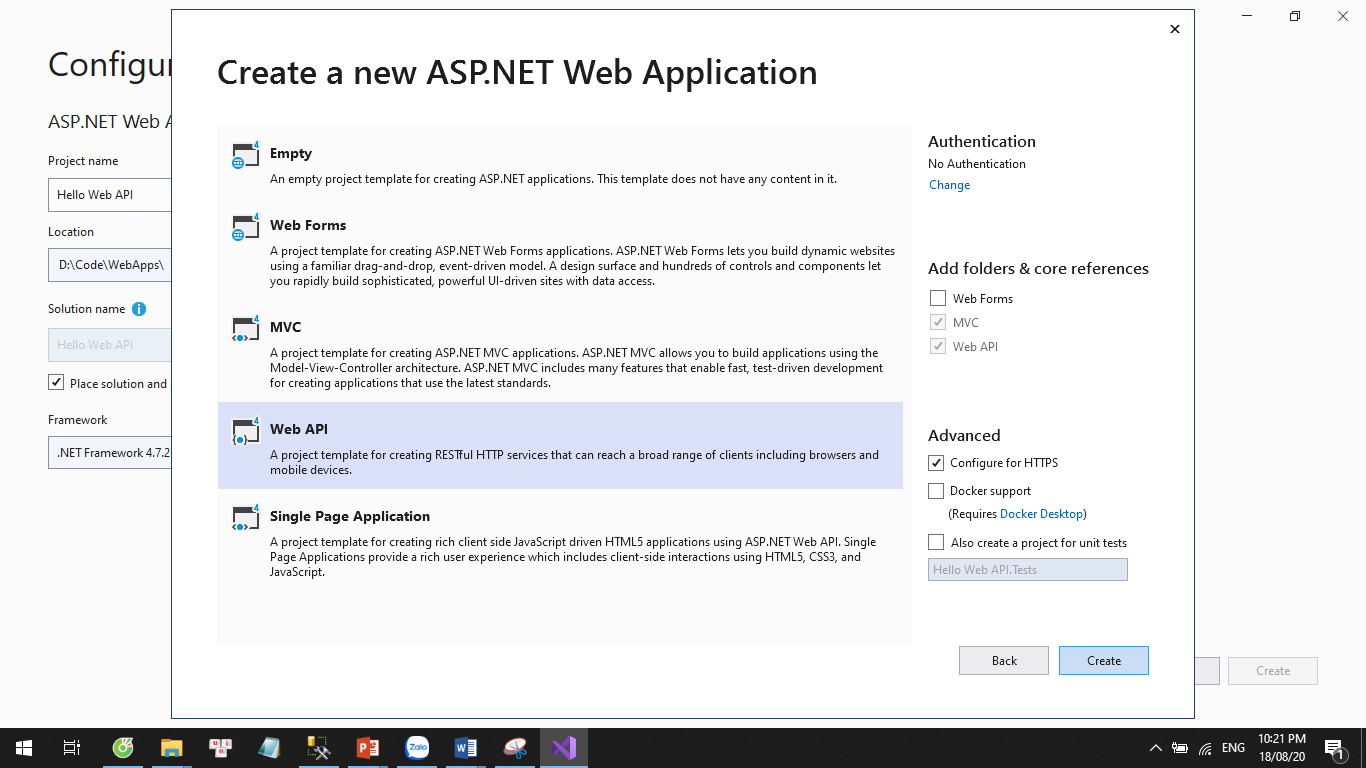




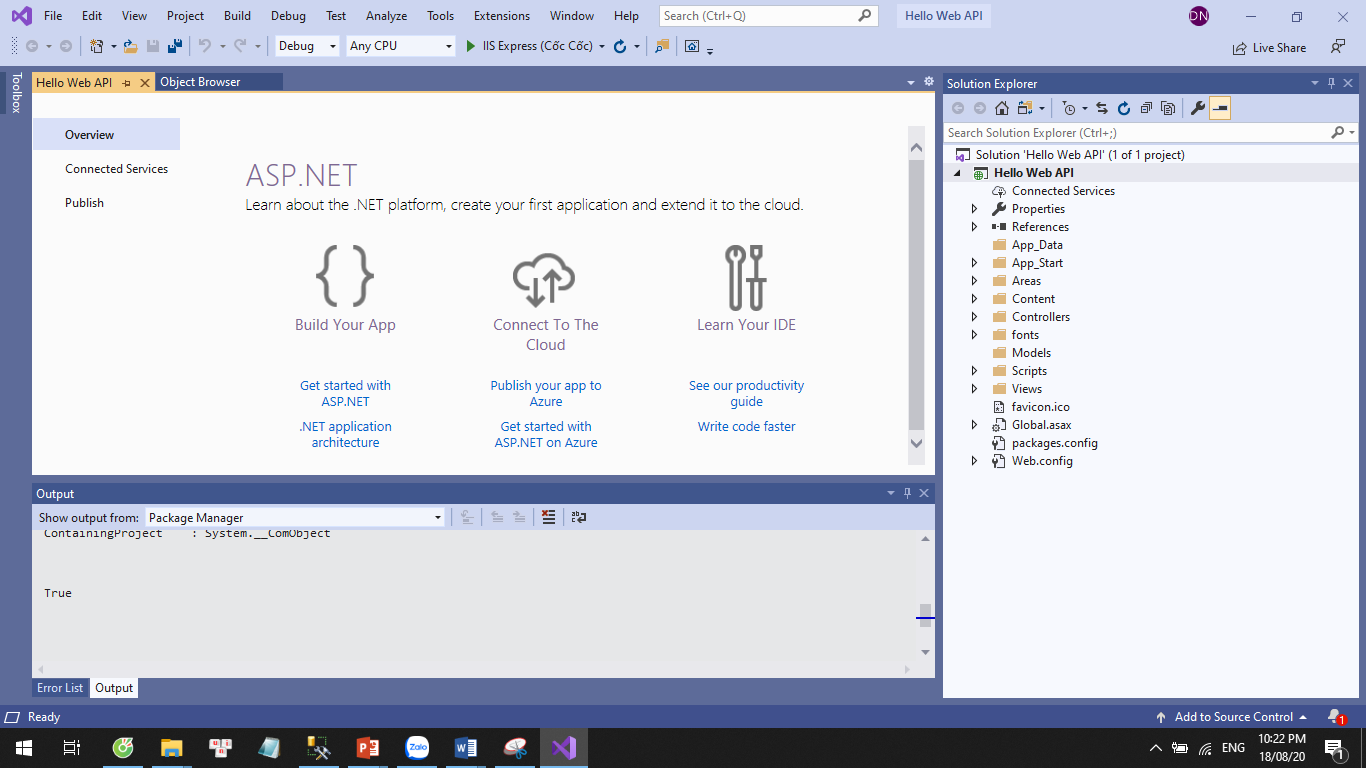
Đặt tên cho ứng dụng, chọn nơi lưu trữ và FrameWork rồi chọn Create



Chọn template Web API.



Nếu chọn template WebAPI thì giao diện sẽ hỗ trợ theo form mô hình MVC và webAPi như sau:



#### Bước 2: Tạo Models

Để code tạo các model, nhấn chuột phải vào Models và chọn Add/New item

Nếu tự sinh code cho các entity thì search ADO

Tự code class thì chọn class và nhập tên class, rồi chọn Add

#### **Bước 3:**Tạo ra các Controller

Nhấp chuột phải vào thư mục Controllers và chọn thêm controller.

**Web API 2 Controller Empty:**tự viết các phương thức từ đầu.

**Web API 2 Controller with read/write actions**: phát sinh các phương thức ví dụ để bạn có thể biết cách viết các service này.

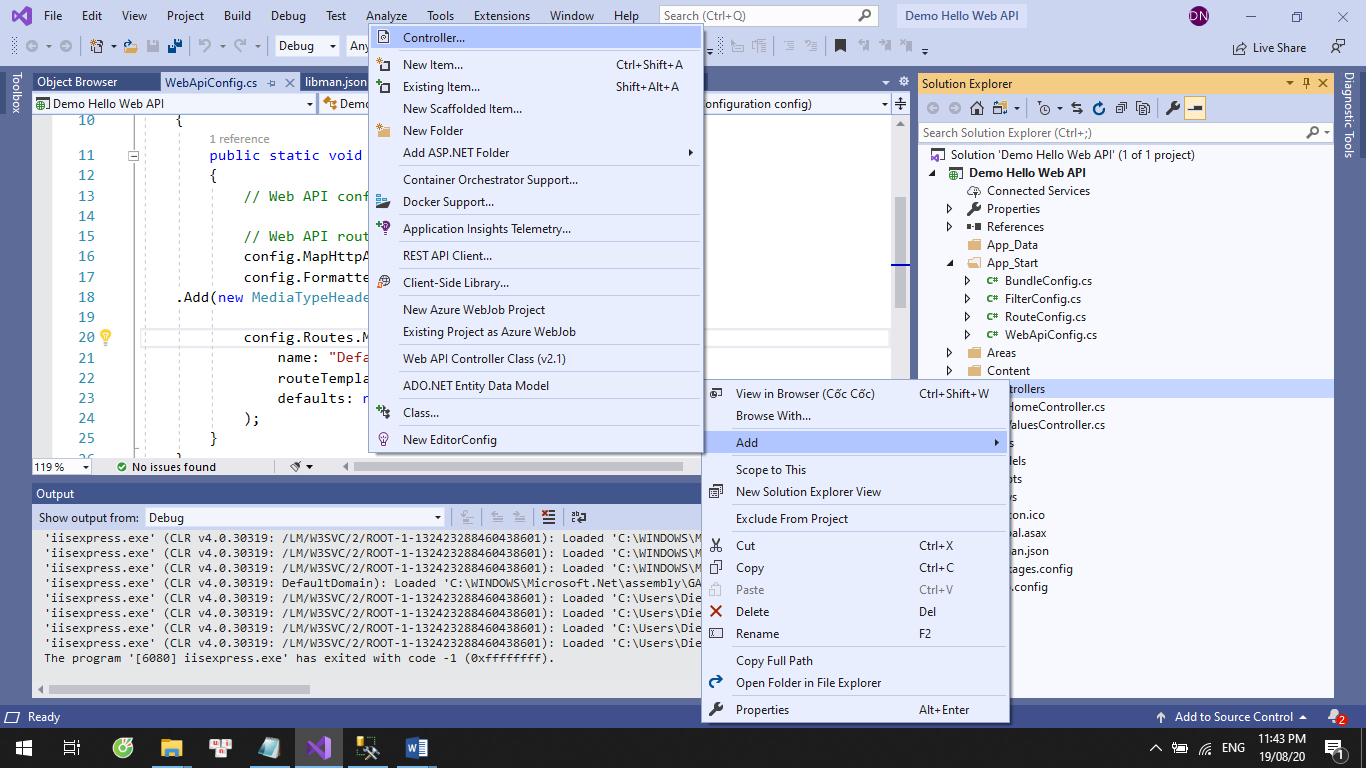
ValuesController.cs mặc định được sinh khi chọn webAPI template theo version VisualStudio 2010 và VisualStudio 2019 như sau:

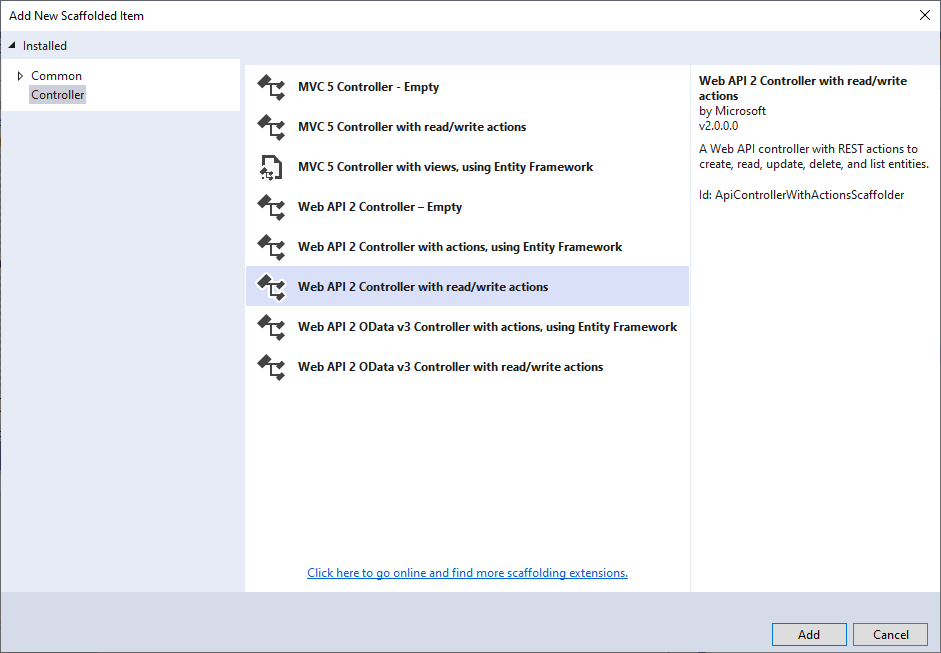
|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Net;  using System.Net.Http;  namespace MyWebAPI.Controllers  {  public class ValuesController : ApiController  {  [HttpGet]  public IEnumerable<string> Values()  {  return new string[] { "value1", "value2" };  }  [HttpGet]  public string Value(int id)  {  return "value";  }  [HttpPost]  public void SaveNewValue([FromBody]string value)  {  }  [HttpPut]  public void UpdateValue(int id, [FromBody]string value)  {  }  [HttpDelete]  public void RemoveValue(int id)  {  }  }  } |

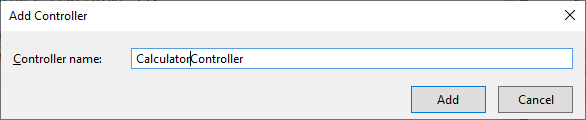
Hoặc

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Net;  using System.Net.Http;  using System.Web.Http;  namespace Hello\_Web\_API.Controllers  {  public class ValuesController : ApiController  {  // GET api/values  public IEnumerable<string> Get()  {  return new string[] { "value1", "value2" };  }  // GET api/values/5  //public string Get(int id)  //{  // return "value";  //}  // GET api/values  public string Get()  {  return "Chào mừng đến với WebAPI";  }  // POST api/values  public void Post([FromBody] string value)  {  }  // PUT api/values/5  public void Put(int id, [FromBody] string value)  {  }  // DELETE api/values/5  public void Delete(int id)  {  }  }  } |

Tạo CalculatorController.cs như sau



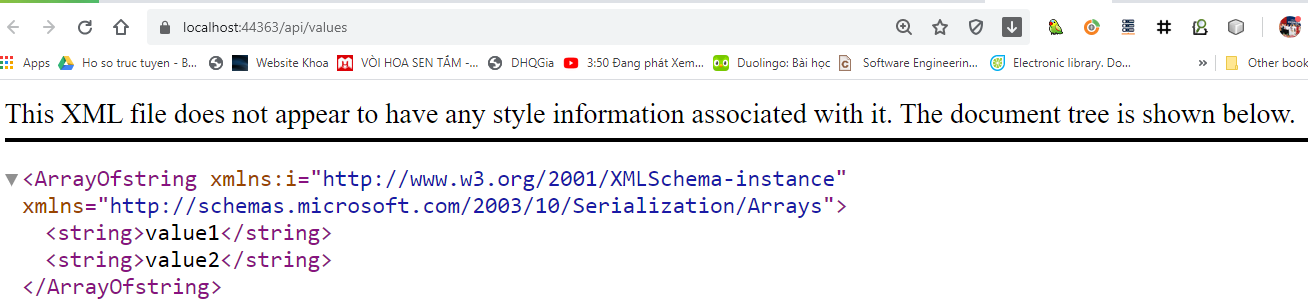




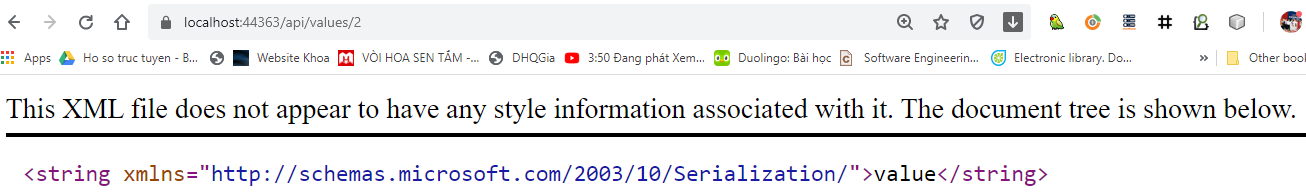
|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Net;  using System.Net.Http;  using System.Web.Http;  namespace Demo\_Hello\_Web\_API.Controllers  {  public class CalculatorController : ApiController  {    // GET: api/Calculator/5  public string Get(int id)  {  return "Welcome to Calculator Demo";  }  [HttpPut]  public float Sum(float a, float b)  {  return a + b;  }  [HttpPut]  public float Subtract(float a, float b)  {  return a - b;  }  [HttpPut]  public float Multi(float a, float b)  {  return a \* b;  }  [HttpPut]  public float Divide(float a, float b)  {  float kq = 0;  try  {  kq = a / b;  }  catch { }  return kq;  }  [HttpPut]  public float BasicCalculatator(float a, float b, char c)  {  float kq = 0;  switch (c)  {  case '+': kq = a + b; break;  case '-': kq = a - b; break;  case '\*': kq = a \* b; break;  case '/':  if (b != 0)  kq = a / b; break;  }  return kq;  }    }  } |

#### **Bước 4:**Chạy thử và kiểm tra

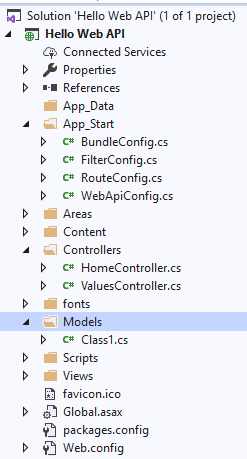
<https://localhost:44363/api/values>



<https://localhost:44363/api/values/2>



### Cấu trúc ứng dụng ASP.NET Web API



App\_Start\WebApiConfig

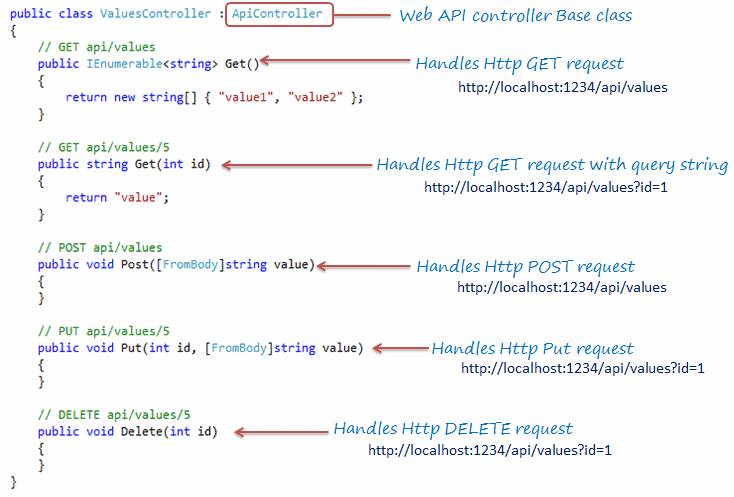
Controller

Models

Web.config

Global.asax

### Các phương thức trong Web API

[](https://www.tutorialsteacher.com/Content/images/webapi/webapi-controller.png)

| HTTP Method | Possible Web API Action Method Name | Usage |
| --- | --- | --- |
| GET | Get() get() GET() GetAllStudent() \*any name starting with Get \* | Retrieves data. |
| POST | Post() post() POST() PostNewStudent() \*any name starting with Post\* | Inserts new record. |
| PUT | Put() put() PUT() PutStudent() \*any name starting with Put\* | Updates existing record. |
| PATCH | Patch() patch() PATCH() PatchStudent() \*any name starting with Patch\* | Updates record partially. |
| DELETE | Delete() delete() DELETE() DeleteStudent() \*any name starting with Delete\* | Deletes record |

Contents

[BÀI 1 GIỚI THIỆU VỀ ASP.NET WEB API 1](#_Toc48683671)

[1.1. API là gì? 1](#_Toc48683672)

[1.2. Web API là gì? 1](#_Toc48683673)

[1.3. REST và RESTful API 1](#_Toc48683674)

[1.4. APS.NET Web API 1](#_Toc48683675)

[1.4.1. APS.NET Web API là gì 1](#_Toc48683676)

[1.4.2. Đặc điểm của ASP.NET Web API 2](#_Toc48683677)

[1.4.3. Các phiên bản của ASP.NET Web API 2](#_Toc48683678)

[1.4.4. So sánh ASP.NET Web API và WCF 3](#_Toc48683679)

[1.4.5. Khi nào nên chọn ASP.NET Web API? 3](#_Toc48683680)

[1.5. Tạo ứng dụng Hello Web API 3](#_Toc48683681)

[1.5.1. Tạo dự án ASP.NET Web API sử dụng Visual Studio 3](#_Toc48683682)

[1.5.2. Cấu trúc ứng dụng ASP.NET Web API 9](#_Toc48683683)

[1.5.3. Các phương thức trong Web API 9](#_Toc48683684)