**WEB API dùng MVC**

**1. Hướng dẫn thực hành và công cụ phát triển web API**

- Để hiểu bản chất các ví dụ thực hành các bạn cần có các kiến thức về web MVC ,Web Page hay các ứng dụng Client như winform , wpf , Console

- Đặc biệt các bạn có kiến thức về lập trình web MVC sẽ rất dê dàng hiểu được web API , do kỹ thuật lập trình trong web API khá giống web MVC ( không giống như web Page (Razor ) hay web form)

- Ngoài ra với các ứng dụng web kết nối tới web Api dùng javascript các bạn cần có các kiến thức về javascript , jQuery đặc biệt là lập trình hướng đối tượng trong javascript để xây dựng các ứng dụng tương tác tốt hơn với người dùng .

- Trong quá trình học các bạn có thể kết hợp xem thêm mã nguồn chúng tôi đính kèm.

- Mã nguồn đính kèm có 2 Project :

+) Mã nguồn web API , và các file html kết nối tới web api lấy dữ liệu

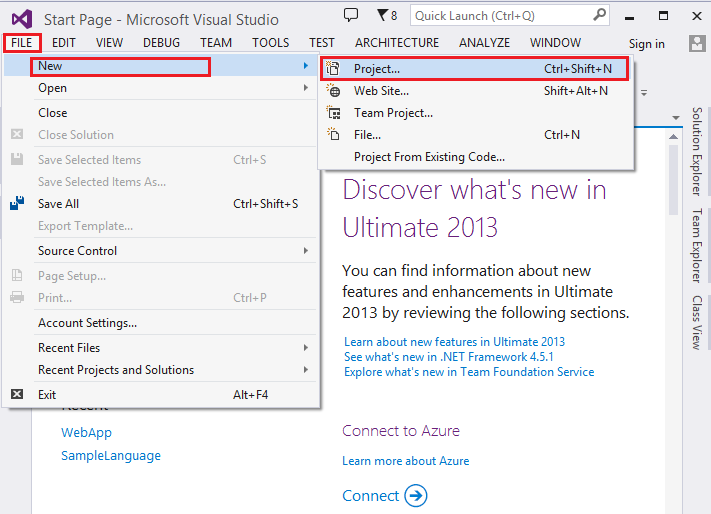
+) Mã nguồn window App là project kết nối tới web API từ các ứng dụng Console , wpf

- Máy tính các bạn cần cài đặt Visual Studio 2013 (có thể là Visual Studio 2012) để mở Project mẫu cũng như làm công cụ để phát triển ứng dụng web API

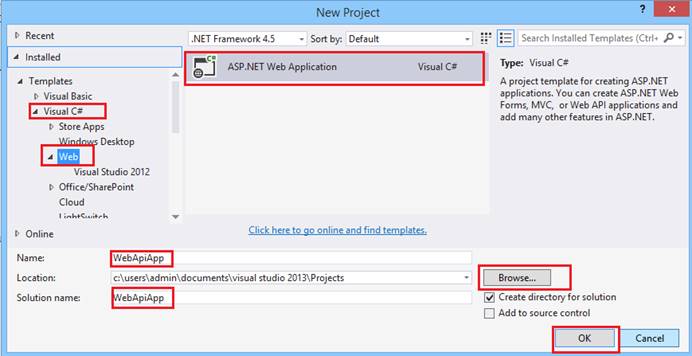
**2. Khởi tạo và cấu hình Web API**

**2.1 Tạo ứng dụng web API bằng Visual Studio**

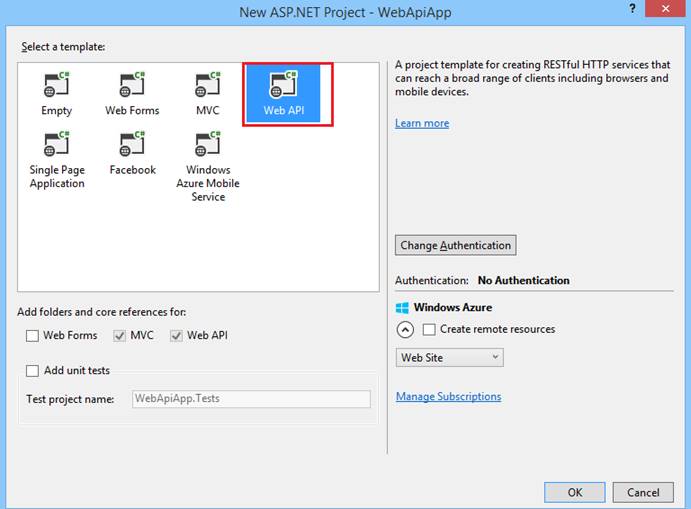
- Mở Visual Studio 2013 tạo mới Projetc theo hình sau :



- Đặt tên project là WebApiApp như hình sau :

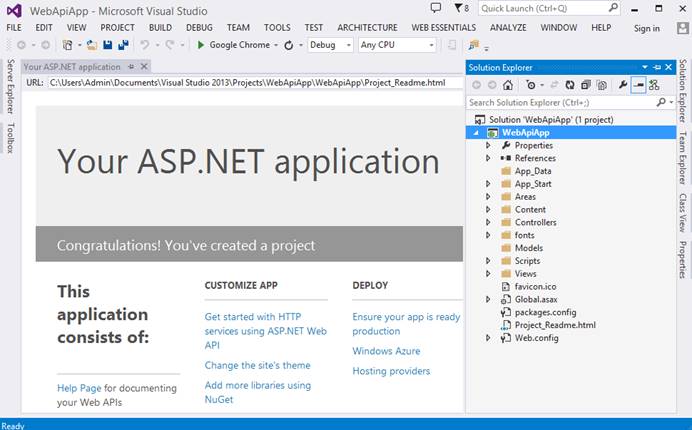


- Nhấn OK sau đó chọn template là web API như hình sau :



- Nhấp OK và chờ Visual Studio tạo các file mẫu cho Project

- Visual Studio tạo ra projetc có mẫu như hình sau :



- Visual Studio tạo sẵn khá nhiều thư mục và những thư viện hô trợ lập trình web API.Chúng ta sẽ lần lượt học cách sử dụng tất cả nhưng thư viện này trong các phần của tài liệu .

- Nếu bạn tạo 1 ứng dụng trống để dễ quản lý các thư viện thì bạn cũng lần lượt phải tạo các thư mục và add các thư viện cần thiết vào project.Ở đây chúng tôi muốn các bạn sử dụng sẵn các thư viện mà Visual Studio tạo ra để hiểu bản chất tất cả các thư viện và thư mục

- Trước tiên chúng ta đi tìm hiểu những thư mục cần thiết cho các ví dụ dưới, các thư viện khác sẽ được đề cập tới ở phần sau

- Thư mục App\_Data : nơi lưu trữ dữ liệu của projetc .Các file cơ sở dữ liệu như Sql Compact , Sql Local , xml thường được đặt ở đây. Hiện tại chúng ta chư dùng tới nó

- Thư mục App\_Start : Chức các file .cs khai báo các lớp cấu hình ứng dụng web MVC và web API :

+) File BundleConfig.cs : Khai báo tham chiếu các thư viện javascript , cs

+)

**2.2 Tạo ứng dụng Web Api đầu tiên**

- Mặc định Visual Studio khởi tạo cho ta 1 ApiController là : ValuesController .

- ValuesController chứa các phương thức trả về các đối tượng giá trị ( string , int …) và thực thi một số phương thức cơ bản như : post , put , delete ( tương ứng các tính năng thêm , sửa , xóa )

- Đây có thể là bản mẫu về 1 Controller API để các bạn tham khảo .Khi đã hiểu các bạn có thể xóa chúng đi

- Phần lớn các thao tác dịch vụ web đều thao tác trên các lớp đối tượng tham chiếu ( các lớp đối tượng do người dùng tự định nghĩa ) .Do vậy để hiểu bản chất vấn đề chúng ta đi ngay vào xây dựng 1 tương tác với 1 lớp đối tượng chúng ta tự định nghĩa

**(1) Tạo Model**

- Tạo lớp Product trong thưc mục Model có code như sau :

public class Product

    {

        public int ID { get; set; }

        public string Name { get; set; }

        public static IEnumerable<Product> GetData(int Count)

        {

            var Persons = new List<Product>();

            for (int i = 1; i <= Count; i++)

            {

                Persons.Add(new Product { ID = i, Name = "Person" + i.ToString() });

            }

            return Persons;

        }

    }

Giải thích code :

- Ở đây ta tạo lớp Product có 2 thuộc tính là ID (mã sản phẩm) và Name (Tên sản phẩm)

- Do ta chưa tương tác với 1 cơ sở dữ liệu thực sự nên ta tạo sẵn 1 phương thức trả về danh sách dữ liệu là GetData()

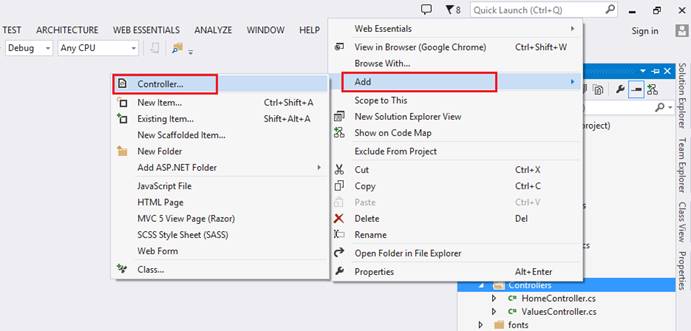
- Phương thức GetData trả về danh sách Product ( kiểu IEnumable<T>) nên việc tương tác với danh sách này sẽ tương tự với các đối tượng truy vấn cơ sở dữ liệu thực sự dùng Entity Framework.

- Chúng ta sẽ trở lại vấn đề này trong phần lập trình cơ sở dữ liệu web API với Entity Framework.

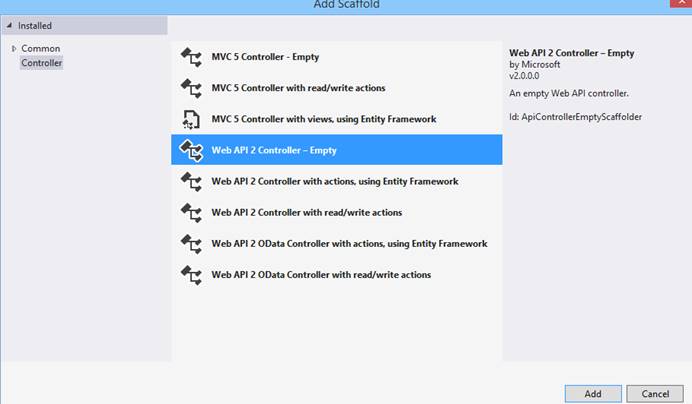
**(2) Tạo ProductApiController**

- Ta sẽ tạo 1 ApiController để tương tác với bảng Product (bảng Product là bảng dưới cơ sở dữ liệu ,ở đây ta tạo kiểu dữ liệu ảo nhưng có tính năng như 1 bảng cơ sở dữ liệu thực sự) bằng cách sau :

- Chuột phải vào thư mục Controller chọn theo hình sau :



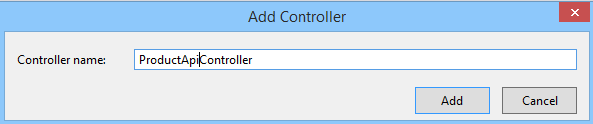
- Chọn web API 2 Controller Emtry như hình sau :



- Ở đây ta có 2 loại template là WebApiController và Controller thông thường .

- Mỗi loại template có các tính năng sinh mã tự động cho các Action tương ứng với 1 đối tượng ( đọc, thêm,sửa,xóa) .Ở đây để ví dụ cho các bạn dẽ hiểu chúng ta chọn Emtry tức là tạo ra 1 ApiController trống và tự code các method

- Một bảng thông báo đặt tên cho ApiController như sau :



Chúng ta đặt tên cho ApiController là ProductApi.Chúng ta có thể đặt tên bất kỳ , và không nhất thiết phải có hậu tố Api đằng sau.Ở đây chúng tôi đặt tên có hậu tố Api đằng sau để dễ phân biệt với các Controller thông thường

- Mã ProductApiController như sau :

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

namespace WebApiApp.Controllers

{

    public class ProductApiController : ApiController

    {

    }

}

- Ta thấy lớp ProductApiController kế thừa lớp ApiController .Khác với lớp Controller thông thường kế thừa lớp Controller

- Ta tạo phương thức Get có mã như sau :

public class ProductApiController : ApiController

    {

        public IEnumerable<Product> Get()

        {

            return Product.GetData(5);

        }

    }

- Chú ý cần khai báo namespace tham chiếu tới lớp Product

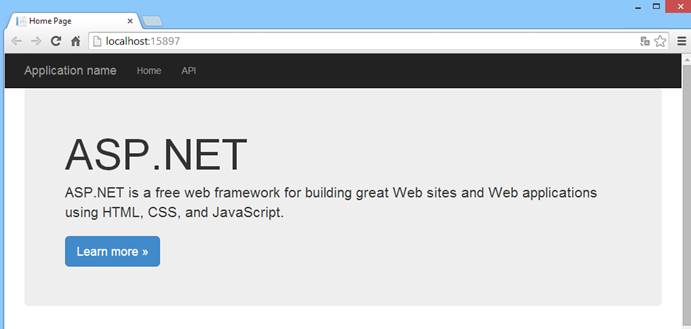
- Phương thức Get() trả về danh sách 5 Product , được gọi theo địa chỉ :

/api/productApi

- Mặc định khi gọi địa chỉ này thì phương thức get được gọi, việc qui định này được chỉ định ở tiền tố get và tham số đầu vào , cũng như được khai báo trong lớp WebApiConfig.cs ở thư mục App\_Start

- Chúng ta sẽ quay lại vấn đề về route trong web API ở phần sau

- Chạy ứng dụng thì HomeController của ứng dụng web được gọi (khai báo trong file RouteConfig.cs )như hình sau

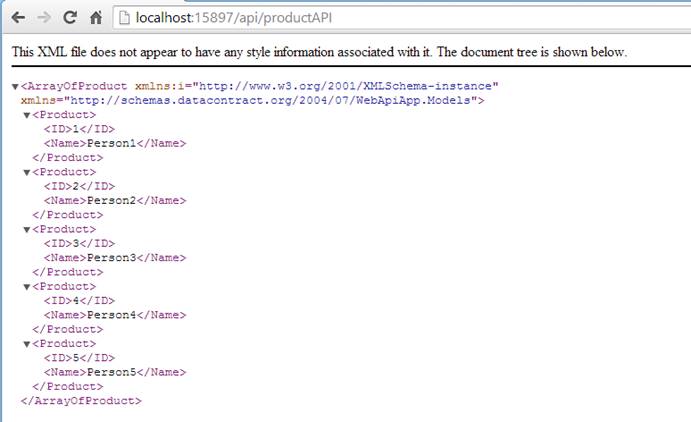


- Trên thành địa chỉ bạn gõ đường dẫn sau để truy vấn tới ProductApiController

/api/productAPI

( link toàn bộ theo công máy tôi sẽ là : <http://localhost:15897/api/productAPI> )

- Ta được kết quả như sau :



- OK tới đây là Service web API của các bạn đã hoạt động thành công .

- ProductApiController trả về danh sách 5 Product , kiểu dữ liệu trả về dưới dạng JSON

- JSON (JavaScript Object Notation) là 1 kiểu dữ liệu rất hay được dùng để trao đổi trong các ứng dụng web .Các bạn có thể tìm hiểu cấu trúc loại dữ liệu này , các kỹ thuật lập trình Ajax áp dụng json rất nhiều nhằm tăng tính tính tương tác cung như tốc độ web

- Do web Api trả về kiểu dư liệu json , nên tại các ứng dụng web Client bất kỳ chúng ta có thể dùng javascript gọi tới Web API lấy dữ liệu và xử lý tùy thích tại client

- Các ứng dụng web, destop cũng có thể gọi tới web API để lấy dữ liệu trả về .

- Tới đây các bạn cũng đã hiểu bản chất của web API , và ứng dụng tương tác web API gồm 2 phần chính đó là:

+) web API cung cấp các dịch vụ thông tin , trả về thông tin dạng JSON .

+) Các ứng dụng Client như là ứng dụng web ( trang web độc lập bên ngoài web API , trang web của hang thứ 3 ) hay các ứng dụng desktop truy vấn tới web API để lấy dữ liệu đồng thời tương tác dữ liệu với web API

- Từ bản chất này chúng ta hiểu ứng dụng WebAPI có thể được gọi ngay cả từ các ứng dụng web PHP , JSP hay bất kỳ loại ứng dụng nào có sử dụng phương thức truyền tải dữ liệu HTTP.

**(3) Dùng Javascript kết nối lấy dữ liệu từ web API**

- Ta sẽ tạo 1 ứng dụng web Client kết nối tới web API để lấy dữ liệu và hiển thị lên trang html

- Dĩ nhiên khi đã thực hiện được kết nối lấy dữ liệu thì ta hoàn toàn có thể thực hiện các thao tác khác xử lý dữ liệu như là : lọc , thêm , sửa , xóa , tìm kiếm ….

- Ở đây chúng ta thực hiện kết nối ngay trong Project web của chúng ta, với các ứng dụng web bên ngoài sau này chúng ta kết nối tương tự chỉ thay URL gồm cả domain (tên miền) tới webAPI .

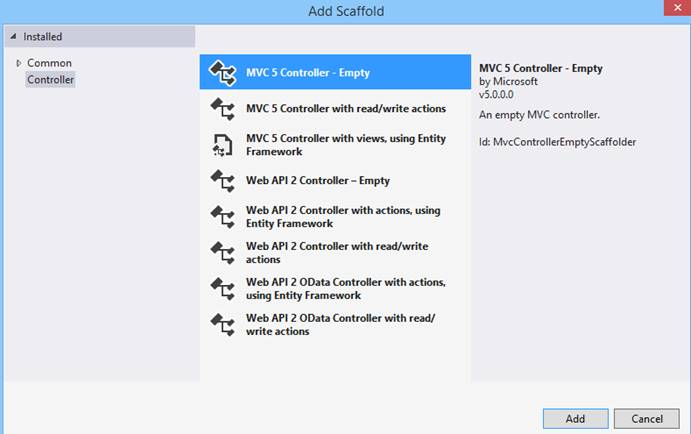
- Giải sử muốn kết nối tới web API từ 1 ứng dụng bên ngoài ứng dụng web API như project web khác , desktop App thì chúng ta cần khai báo đầy đủ url là :

<http://localhost:15897/api/productAPI>

Và thêm cả điều kiện nữa là ứng dụng web API phải đang chạy ( Service phải đang hoạt động)

- Còn nếu kết nối từ ngay ứng dụng web API thì chỉ cần lời gọi sau : api/productAPI

**Tạo 1 Controller thông thường**



- Đặt tên là Sample1Controller và có code như sau :

- Code Sample1Controller như sau :

public class Sample1Controller : Controller

    {

        // GET: Sample1

        public ActionResult Index()

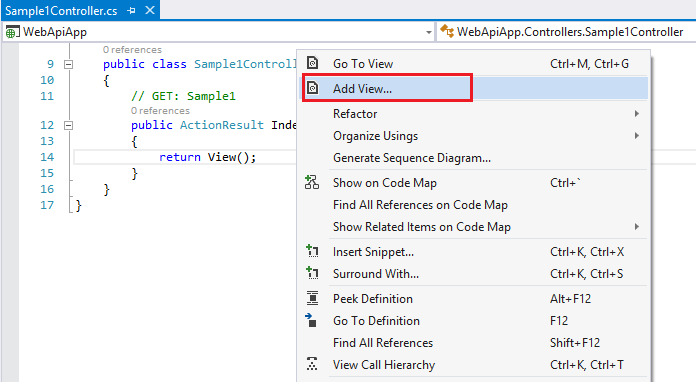
        {

            return View();

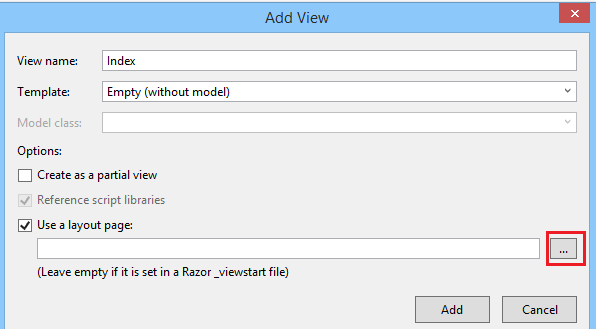
        }

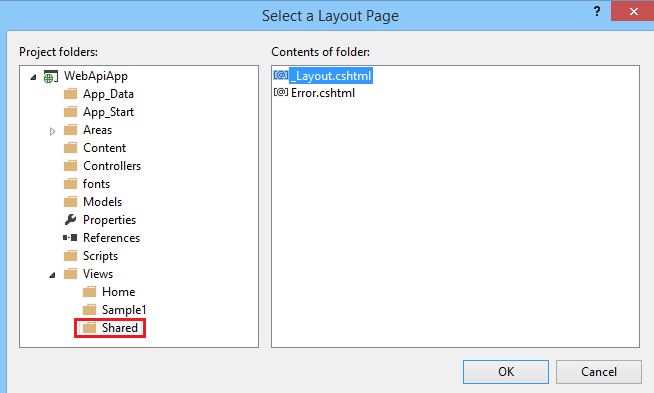
    }

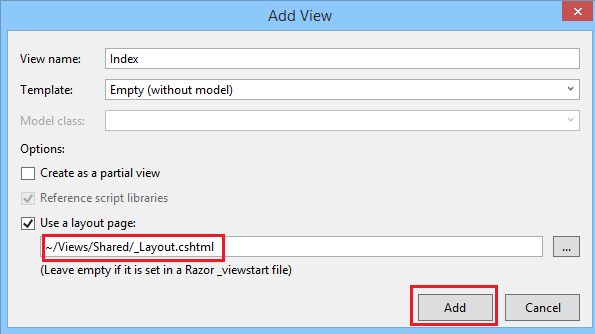
- Action Index của Sample1Controller trả về 1 View , chúng ta cần tạo View Index như sau :



- Chọn tiếp các thông số layout cho View như hình sau :







- Cách tạo View trong MVC5 có đôi chút khác so với MVC4 .Các bạn có thể tìm hiểu các nội dung này trong tài liệu MVC5 của chúng tôi

- Visual Studio sẽ tạo View Index.cshtml trong thưc mục View/Sample1

- Mã View Index.cshtml như sau :

@{

    ViewBag.Title = "Index";

    Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h2>Index</h2>

- Để đơn giản trong thao tác kết nối tới web API bằng javascript chúng ta dùng thư viện jQuery để thực hiện lời gọi Ajax tới web API

- Ta thay đổi mã View Index như sau :

@{

    ViewBag.Title = "Index";

    Layout = "~/Views/Shared/\_Layout.cshtml";

}

@Scripts.Render("~/bundles/jquery")

<h1>Trang Sample1 Index</h1>

<h2>Danh sách sản phẩm load từ server</h2>

<div id="div1">

</div>

<script>

$(function () {

        $.get('/api/productApi', function (data) {

            data.forEach(function (item) {

                $('#div1').append("<p>" + item.ID + "-" + item.Name + "</p>");

            });

        });

    });

</script>

**Giải thích mã :**

**Khai báo thư viện jQuery** :

@Scripts.Render("~/bundles/jquery")

- Đây là tính năng bundled các file jquery , css (từ phiên bản MVC4) .Được định nghĩa trong file BundleConfig.cs

- Các bạn cũng có thể khai báo trực tiếp thư viện jQuery như cách thông thường như sau :

<script src="~/Scripts/jquery-1.10.2.js"></script>

**Mã javascript :**

- Các bạn đã quá quen thuộc với đoạn mã dùng ajax của jquery để load dữ liệu từ server

- Ở đây ta chú ý tới đối tượng success như sau:

function (data) {

                data.forEach(function (item) {

                    $('#div1').append("<p>" + item.ID + "-" + item.Name + "</p>");

                });

            }

- Sau khi lời gọi ajax thành công dữ liệu được trả về trong biến data ( có thể đặt tên biến bất kỳ ) .Và ta thực hiện 1 hàm lặp qua các đối tượng trong biến data để hiển thị dữ liệu lên thẻ div

- Ở đây ta dùng thẻ div để hiển thị dữ liệu , thường dữ liêu dạng collection hiển thị dùng thẻ ul(li) hoặc thẻ table .Tuy nhiên mã thẻ này rối gây khó dễ cho việc quan sát bản chất ví dụ .

- Chạy ví dụ và gọi tới Action Index của Sample1Controller  ( /sample1) được kết quả như hình sau:



**(4) Một chút kiến thức về jQuery Ajax**

- jQuery cung cấp các phương thức lấy dữ liệu kiểu ajax khá mạnh mẽ như : get , post , load,ajax .

- Với jQuery ajax ta có thể kết nối tới web API dễ dàng để lấy dữ liệu thay cho việc phải viết các đối tượng XML resquest javascript thuần rất tốn thời gian và rối mã nguồn

- Tuy nhiên các công việc như CRUD ( đọc , thêm , sửa , xóa ) dữ liệu dùng jQuery Ajax kết nối tới web api rất nhiều , do đó các bạn có thể xây dựng những hàm jQuery Ajax sẵn xử lý các thao tác cơ bản này hơn là dùng jQuery thuần ( chúng tôi sẽ hướng dẫn chi tiết trong phần tương tác cơ sở dữ liệu với web api dùng jquery)

- Giả sử với việc lấy dữ liệu danh sách Product ở trên , để tường mình và dễ xử lý tương tác trong ứng dụng Client , ta có thể lưu dữ liệu lấy về dưới 1 mảng đối tượng JavaScript .Và ta có thể dễ dàng tương tác hiển thị , thêm ,sửa , xóa tại client theo đối tượng này

Ví dụ như sau :

function Product(\_id,\_name) {

        this.ID = \_id;

        this.Name = \_name;

    };

    $(function () {

        var Products = [];

           $.get('/api/productApi', function (data) {

            data.forEach(function (item) {

                Products.push(new Product(item.ID, item.Name));

            });

            Products.forEach(function (item) {

                $('#div1').append("<p>" + item.ID + "-" + item.Name + "</p>");

            })

        });

    });

- Ta tạo 1 lớp Product và đối tượng Products là mảng các Product .Việc tương tác (thêm, sửa , xóa) mảng Product bằng javascript rất đơn giản và tường mình và theo phong cách hướng đối tượng nên mã rất dễ đọc .

- Đối tượng Products sẽ được gán giá trị trong ajax request , sau đó lặp qua đối tượng này để hiển thị lên view .

- Chạy mã này ta cũng được kết quả như trên

- Chú ý là lời gọi ajax là bất đồng bộ,khi gọi ajax thì các công việc khác trên website vẫn được tiến hành bình thường .Do đó lời gọi ajax không thể thay đổi 1 biến toàn cục .Giả sử ở đây ta khai báo hiển thị danh sách mảng Products ngoài phương thức ajax sẽ không có kết quả :

function Product(\_id,\_name) {

        this.ID = \_id;

        this.Name = \_name;

    };

    $(function () {

        var Products = [];

           $.get('/api/productApi', function (data) {

            data.forEach(function (item) {

                Products.push(new Product(item.ID, item.Name));

            });

           });

           Products.forEach(function (item) {

               $('#div1').append("<p>" + item.ID + "-" + item.Name + "</p>");

           })

    });

- Lý do là đối tượng Products đã được khởi tạo và hiển thị trước lời gọi ajax ( lời gọi ajax sẽ là sau cùng và âm thầm gọi )

- Để giải quyết vấn đề này ta cần khai báo đối tượng jquery ajax async là false .Tức là giao diện sẽ hiển thị sau khi lời gọi ajax kết thúc .Tức là trong khi load dữ liệu ajax thì ta không thể tương tác với website được nữa .

- Mã trường hợp này như sau :

function Product(\_id,\_name) {

        this.ID = \_id;

        this.Name = \_name;

    };

    $(function () {

        var Products = [];

           $.ajax({

               url: '/api/productApi',

               cache: false,

               async: false,

               type: 'GET',

               contentType: 'application/json; charset=utf-8',

               data: {},

               success: function (data) {

                   data.forEach(function (item) {

                       Products.push(new Product(item.ID, item.Name));

                   });

               }

           });

           Products.forEach(function (item) {

               $('#div1').append("<p>" + item.ID + "-" + item.Name + "</p>");

           })

    });

- Chạy ví dụ ta được kết quả y nguyên ví dụ ban đầu

- Phương thức $.ajax thực ra là phương thức tổng quát của các phương thức như $.get() hay $.post() .Chỉ thêm nhiều hơn các tham số qui định trong lời gọi ajax .

- Các phương thức $.get() hay $.post() giản lược đi các tham số mặc định để dễ sử dụng và không phải khai báo dài dòng trong những lời gọi đơn giản

- Qua các ví dụ thì các bạn thấy việc xử dụng javascript để tương tác với ứng dụng web API là khá quan trọng .Và các bạn cần nắm chắc kiến thức về Javascript để có thể xây dựng lên các ứng dụng web tương tác với web api .

- Dĩ nhiên các ứng dụng web như : MVC , Web Page , web form hay các ứng dụng desktop như wpf , winform chúng ta có sẵn các phương thức .net (C# , vb.net) để kết nối tới web API.

- Nhưng chúng tôi muốn nhấn mạnh tới kỹ thuật lập trình web ajax kết nối tới web api là khá mạnh và hiện đại để xây dựng các website tương tác tốt với người dùng.Chúng ta sẽ tìm hiểu kỹ thuật này trong tài liệu về Single Page Application ( ứng dụng trang web đơn) .Tức là 1 website tất cả các tương tác với người dùng chỉ trên 1 trang .

**2.3 Cấu hình Route ứng dụng web API**

**(1) route mặc định**

- Cấu hình route là cấu hình url truy cập tới các Controller của ứng dụng .

- Với web API route được cấu hình trong file WebApiConfig.cs (thư mục App\_Start)

public static class WebApiConfig

    {

        public static void Register(HttpConfiguration config)

        {

            config.MapHttpAttributeRoutes();

            config.Routes.MapHttpRoute(

                name: "DefaultApi",

                routeTemplate: "api/{controller}/{id}",

                defaults: new { id = RouteParameter.Optional }

            );

        }

    }

- Routed qui định 2 giá trị : tên controller và tham số

- Với cấu hình trên có thể gọi web Api thông qua các tiền tố của Action .Trong trường hợp này trong cùng 1 Controller không thể có 2 phương thức cùng tham số và cùng hành động ( get , post) .Ví dụ :

|  |  |
| --- | --- |
| /api/product/ | gọi tới get method không tham số |
| /api/product/1 | gọi tới get method có chứa tham số |

-Với Route mặc định như trên ta xây dựng các phương thức get,post dữ liệu trong ApiController với các thao tác đọc , thêm, sửa , xóa dữ liệu .Giả sử ta có ProductController như sau :

public class ProductController : ApiController

{

// url: /api/product  :  lấy về danh sách Product ( ko cần tham số )

public IEnumerable<Product> Get() {

 }

// url:/api/product/?name=name  :lấy về danh sách product , với tham số là chuối tên Product

public IEnumerable<Product> Get(string name) {

 }

// url: /api/product/id  :lấy về 1 product theo id ( kiểu int)

public Product Get(int id) {

 }

// url : /api/product  : thêm mới product

public void Post(Product model) {

 }

// url : /api/product/id  : cập nhật product

public void Put(int id, Product model) {

 }

// url : /api/product/id  : xóa product

public void Delete(int id) {

}

}

- Ở đây ta chú ý quan tâm tới các đầu ngữ qui định việc Get , Post , Put hay Delete

-Giả sử phương thức get có thể đổi tên thành GetProduct như sau :

// url: /api/product/id  :lấy về 1 product theo id ( kiểu int)

public Product GetProduct(int id) {

 }

- Rote sẽ dựa vào tham số đầu vào , kiểu dữ liệu gửi lên , loại gửi (get , post) và kết hợp với các đầu ngữ qui ước để chọn Action tương ứng để xử lý

- Bạn có thể tạo các Action có tên bất kỳ rồi gọi chúng từ Client nếu thấy có thể gây sự nhậm lẫn giữa các tham số ?

**Chú ý**

Với Routed mặc đinh thì các lời gọi tới api có url giống nhau .Tùy vào dữ liệu gửi lên mà Routed sẽ gọi tới Action tương ứng .

Ví dụ :

Client gọi url là : /api/product

- nếu không kèm theo dữ liệu gì thì method get được gọi

- nếu kèm theo dữ liệu ( đối tượng Product ) thì method Post được gọi

- Để có lời gọi chính xác , có thể thiết lập routed để gọi theo kiểu :

/ControllerName/Action/Parameter

(như trong routed của MVC thì chỉnh sủa Routed lại cho web api )

**(2) Tùy chỉnh route**

- Nếu cần xây dựng API với những yêu cầu về tên , tham số nhiều hơn chúng ta có tự cấu hình route cho phù hợp với ứng dụng dịch vụ

- Giả sử ta cấu hình route 3 tham số như route của ứng dụng web MVC như sau :

public static void Register(HttpConfiguration config)

        {

            config.Routes.MapHttpRoute(

                name: "DefaultApi",

                routeTemplate: "api/{controller}/{action}/{id}",

                defaults: new { id = RouteParameter.Optional }

            );

        }

- Tương tự như ứng dụng MVC chúng ta có thể đăng ký nhiều route .Và khi nhận 1 yêu cầu thì ứng dụng sẽ load các route từ trên xuống theo nguyên tắc khi nào tìm được route đúng tên , đúng tham số , và đúng kiểu (get,post) thì dừng lại