要使用WebAPI，首先需要知道MVC和WebAPI的区别。

Mvc相当于一个成熟的网站架构，主要作用于网站建设，而webapi则是构建http服务。

应用层面：  
先有的mvc后有的webapi，mvc 主要是用来建站；webapi 主要是用来构建 http 数据服务；微软对后者经过精心设计改良，尤其适合 面向移动设备(节省带宽)提供数据服务。  
技术实现层面：webapi 被砍去了 view 层 ，整体轻快。  
1: webapi 有自己独特的消息处理管道；   
2: webapi 有自己独特的路由系统；   
3: webapi 的模型绑定也有些特别，用的是自己独特 httpParameterBinding   
4: webapi 的托管层也不一样，不单只web托管(例如: IIS) , 还能自托管 (寄宿在windows 进程)，还能托管在 OWIN的实现着（例如: katana）；MVC 只能托管在 IIS 上，当然不包括 借助 mono 托管在linux

先决条件：.net framework 4.5

<https://docs.microsoft.com/zh-cn/aspnet/web-api/overview/getting-started-with-aspnet-web-api/tutorial-your-first-web-api>

Webapi又分两种：

1.基于http的ApiController;

2.基于mvc的Controller;

而创建webapi的方式又有四种：

1.创建空模板的api

2.创建webapi模板同时带mvc

3.创建mvc同时带webapi

4.创建web窗体同时带webapi

现在主要用空模板的webapi以及带mvc视图的webapi这两种！

而它两者的区别在于mvc的多了一层视图且会多一种空模板webapi所没有的Controller。

3、MVC和WebApi路由区别汇总

WebApi的默认路由机制通过http请求的类型匹配Action，MVC的默认路由机制通过url匹配Action

WebApi的路由配置文件是WebApiConfig.cs，MVC的路由配置文件是RouteConfig.cs

WebApi的Controller继承自Web.Http.ApiController，MVC的Controller继承自Web.Mvc.Controller

针对空模板实例:WebApiEmptyTemplate

针对mvc模板实例:WebApiByMVC

Webapi在调用时可以提供前缀一般为api可更改

<http://localhost:53139/api/values>

<http://localhost:53139/api/values?id=1>

注意：?代表QueryString 查询字符串

<http://localhost:53139/api/values?id=1&name=Mark>

注意：&代表’,’多个入参之间用&隔开

Mvcapi在调用时没有前缀直接在url拼接就行

<http://localhost:53139/home>

Webapi四大内置对象：

1.Get():用于查询

2.Post():用于插入

3.Put(Int id):用于更新，必须提供条件

4.delete(int id):用于删除，必须提供条件

详情请参考实例中解释！

Asp.Net WebApi+JQuery Ajax的Post请求整理

一、总结

1.WebApi 默认支持Post提交处理，返回的结果为json对象，前台不需要手动反序列化处理。  
2.WebApi 接收Post提交参数需要指定([FromBody] string name)  
3.WebApi 中如果只接收一个基础类型参数，不能指定key的名称  
4.WebApi Post请求Action只能接收一个参数，也就是说一个Action中[FromBody]仅可指定一次  
5.WebApi Post请求处理多个参数可以使用类对象方式接收参数例如：Student  
6.在接收Post参数时，如果不想定义类，可以使用Newtonsoft.Json.Linq的JObject json对象容器接收参数  
7.(不推荐使用)此接收参数可以使用dynamic本质是使用的JObject，但是提交参数需要指定字符串类型，contentType: 'application/json',类似WebServer中的指定方式  
8.在WebApi的Post请求处理中，后台的Action名称不能使用“GetXXX”方式命名

WebAPI使用流程：

方式1：

创建带mvc的webapi

配置WebApiConfig文件，在其中配置一个特性路由。

//特性路由，

//1.在ValuesController中给方法前添加ActionName指定名称后

//2.在请求url时在默认controller后在添加ActionName指定名称

config.Routes.MapHttpRoute(

name: "ActionApi",

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

defaults: new { id = RouteParameter.Optional }

);

在xxxController中get、post等方法前加上[HttpGet]、[HttpPost]

如果存在多个post方法需要用到特性路由，在方法前加上 [ActionName("NewPostAdd")]，其中NewPostAdd为别名。

使用ajax调用时需要在url拼接上别名，如下：

$.ajax({

url: '/api/Values/NewPoststring',

type: "post",

data: {

"":"Mark"

}

}).done(function (s) {

console.log(s);

console.log(this.url);

}).fail(function (err) {

console.log(err.responseText);

});

如果前台是通过json格式，使用data对象传值那么我们需要：

每个方法只能有一个参数，若有多个参数则封装为对象在入参。

给方法中入参前边加关键字[FromBody]。

方式2：

1.创建带mvc的webapi

2.配置WebApiConfig文件，在其中默认配置下的defaults下加条件：

Default：new { controller = “Home”,action = “Index”,id = RouteParameter.Optionan }

3.在xxxController中get、post等方法前加上

[ HttpGet,, Route(“NewGetAll”) ]，

[ HttpPost, Route(“NewPostAdd”) ]，其中NewPostAdd为别名。

4.给xxxController类前加上[RoutePrefix(“api/Values”)]

5.使用ajax调用时需要在url拼接上别名，如下：

$.ajax({

url: '/api/Values/NewPostAdd,

type: "post",

data: {

ID: 6,

Name: "刘七",

Sex: "女",

Age: 30

}

}).done(function (s) {

console.log(s);

console.log(this.url);

}).fail(function (err) {

console.log(err.responseText);

});

如果前台是通过json格式，使用data对象传值那么我们需要：

每个方法只能有一个参数，若有多个参数则封装为对象在入参。

给方法中入参前边加关键字[FromBody]。

其中有[FromUri] 和 [FromBody]

Uel：是通过拼接url来添加操作post

Body：是通过正文htmljs，例如ajax中data或者vue中data来添加

都是基于HttpRequestMessage来操作，bidy为实体正文，url为链接，简要说明：一个通过浏览器链接url来操作，一个通过html网页来操作。

注意：以上例举方式1、2的区别主要在于方法前是加 [ActionName("NewPutUpd")]来区分还是直接在 [HttpPut,Route("PutUpData")]中指定路由，通过第二种能简单一点，不用配置路由就可以直接使用！

在vuecli3中调用webapi时一定要配置vue.config.js中的devServer

// 配置跨域请求 webpack-dev-server 相关配置

devServer: {

//默认端口

port: 8080,

//设置为0.0.0.0则所有地址都能访问

host: 'localhost',

// 自动打开浏览器

open: true,

https: false,

//使用代理解决跨域问题(跨域的常用方式：1.在前端解决跨域问题 2.后端直接设置访问控制允许源Access-Control-Allow-Origin:\* 代表允许全部域名跨域，也可单独设置指定域名跨域 3.后端配置Nginx反向代理)

proxy: {

//详解

//1.proxy里边的'/api'什么意思？

//答：作用是告诉node，我的接口要是以'/api'开头的才用代理，例如:App.vue中请求的接口地址为"/api/user"符合以/api开口的条件，所以会被代理，最后代理的路径由http://localhost:8080/api/user ==> 变成http://10.10.38.94:3000/api/user

// 虽然浏览器的NetWork的Headers/General URL还是http://localhost:8080/api/user,但实际上在请求后，就被node代理服务器悄悄代理成为了http://10.10.38.943000/api/user再去请求真实后台接口地址

//2.pathRewrite里面的'^/api':''什么意思？

//答：由上面可知，代理成了http://10.10.38.94:3000/api/user,但是我们实际的真是后台接口地址是http://10.10.38.94:3000/user，所以在请求前一刻，需要将/api去除(把/api给重写成空字符串了)。

//'^/api是一个正则表达式，表示匹配请求的url中，全部http://localhost:8081/api/user转接为http://10.10.38.94:3000/user'

'/api': {

//目标代理服务器地址

target: 'http://localhost:53139/api',

//如果要代理 websockets，配置这个参数

//ws: true,

// 如果是https接口，需要配置这个参数

//secure: false,

//允许跨域

changeOrigin: true,

pathRewrite: {

'^/api': ''

}

}

}

}