

ASP.NET Core Web API 建置教學

建置不含資料庫之 MathBmi_WebApi 雲端服務專案

by Min-Hsiung Hung

目錄

一、 建立一個空白的 ASP.NET Core 6 Web API 服務專案(Core6_MathBmiService)，並用 Swagger UI、瀏覽器及 Postman 分別測試這個 Web API 服務：	2
二、 建立[MathBmiServiceController.cs] Web API Controller，然後建立利用 GET 及路由參數呼叫的 Web API(ComputeMath 服務)	10
三、 建立利用 POST 及 JSON Request Body 呼叫的 Web API(ComputeBmi 服務)	18
四、 將此專案部署到私有雲虛擬機上，使它成為 REST 雲端服務.....	22

一、建立一個空白的 ASP.NET Core 6 Web API 服務專案 (**Core6_MathBmiService**)，並用 Swagger UI、瀏覽器及 Postman 分別測試這個 Web API 服務：

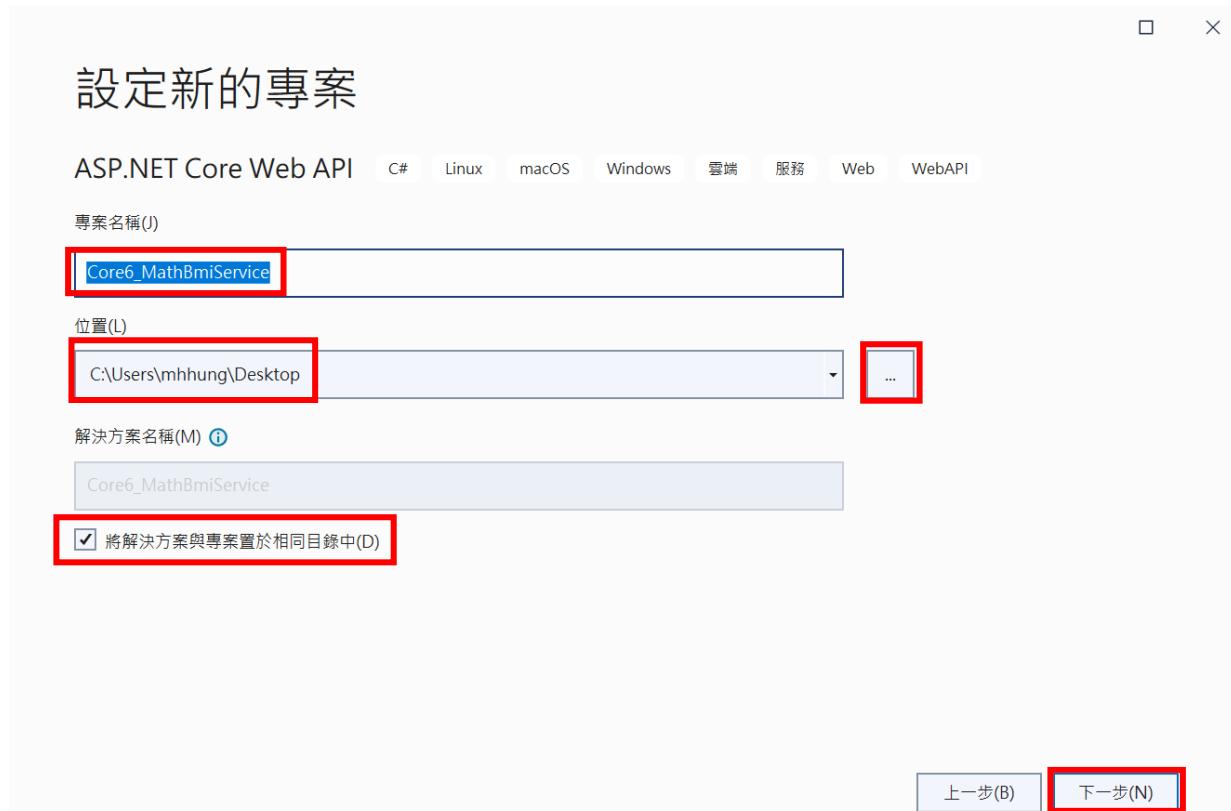
1. 開啟 Visual Studio 2022，點選[建立新的專案]：



2. 點選[ASP.NET Core Web API]，然後點選[下一步]：



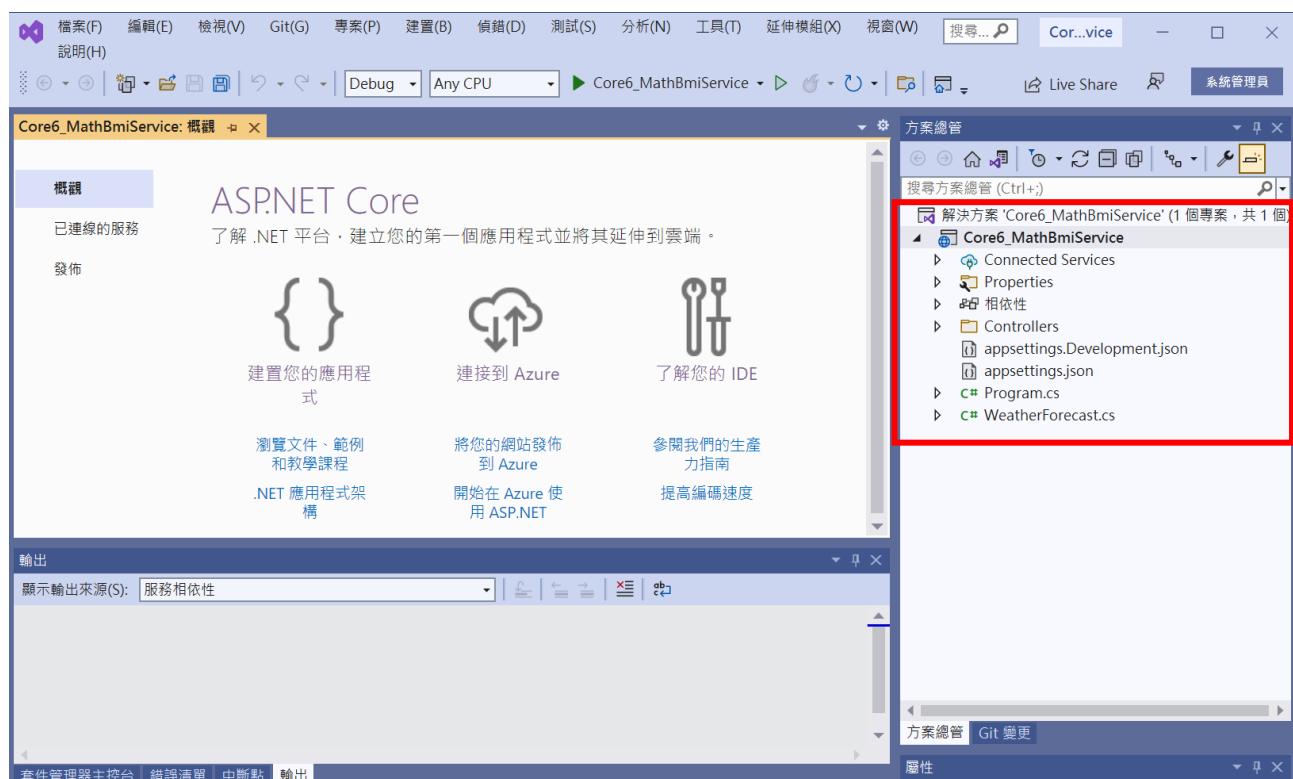
3. 輸入專案名稱：Core6_MathBmiService，位置：點選右邊[...]，然後選擇桌面/Desktop)。勾選[將解決方案與專案置於相同目錄中]，然後點選[下一步]：



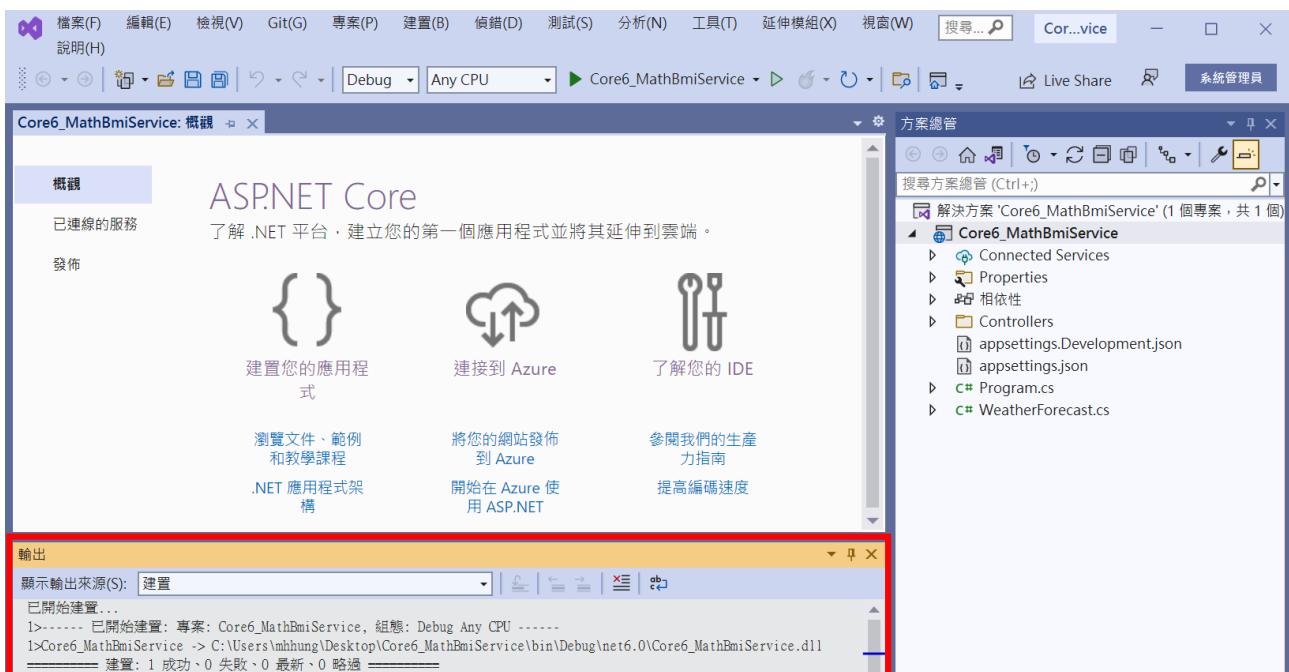
4. 取消勾選[針對 HTTPS 進行設定]，然後點選[建立]：



5. 建立專案後，在右側[方案總管]中可看到 Core6_MathBmiService 專案的結構：



6. 同時按下<Ctrl>+<Shift>+，以建置專案：



7. 點選[WeatherForecast.cs]，在編輯器可看到 WeatherForecast 類別內有 5 個屬性，詳細看程式中的中文註解：

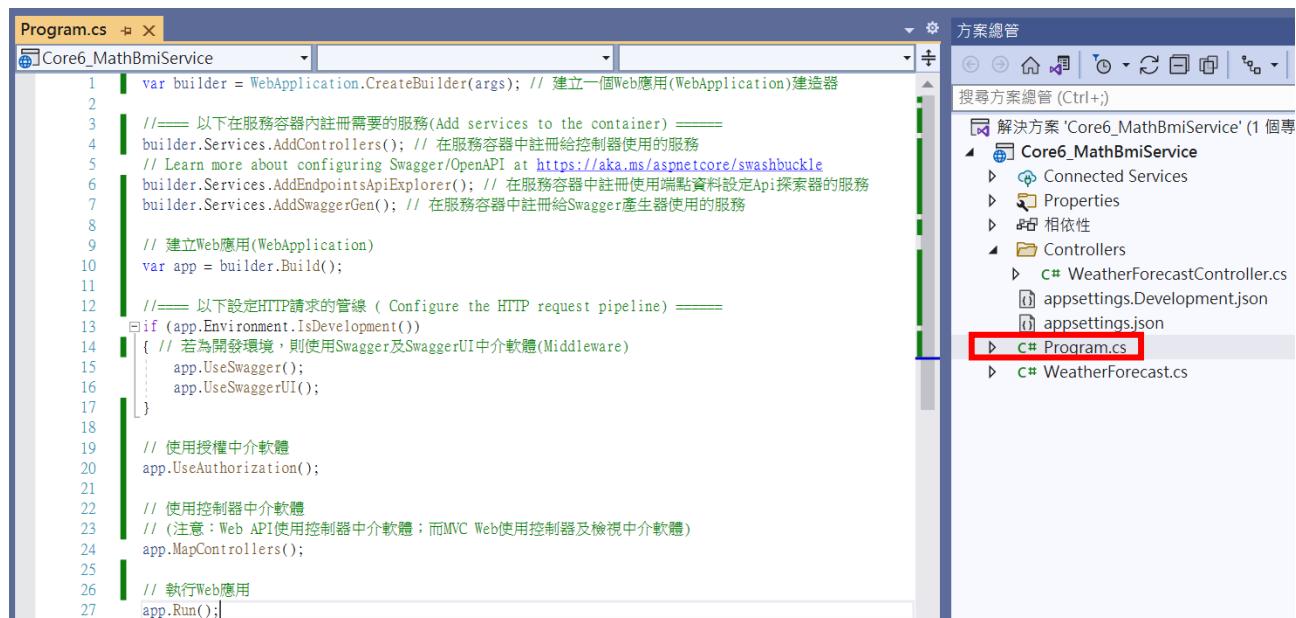


```

1  namespace Core6_MathBmiService
2  {
3      2 個參考
4      public class WeatherForecast
5      {
6          1 個參考
7          public DateTime Date { get; set; } // 用於儲存日期
8
9          2 個參考
10         public int TemperatureC { get; set; } // 用於儲存攝氏溫度
11
12         0 個參考
13         public int TemperatureF => 32 + (int)(TemperatureC / 0.5556); // 用於儲存華氏溫度
14
15         1 個參考
16         public string? Summary { get; set; } // 用於儲存溫度資訊摘要
17     }
18 }

```

8. 點選[Program.cs]，在編輯器可看到在服務容器註冊所需服務及為 HTTP 請求管線(HTTP Request Pipeline)增加了一些需要的中介軟體(Middleware)，詳細看程式中的中文註解：



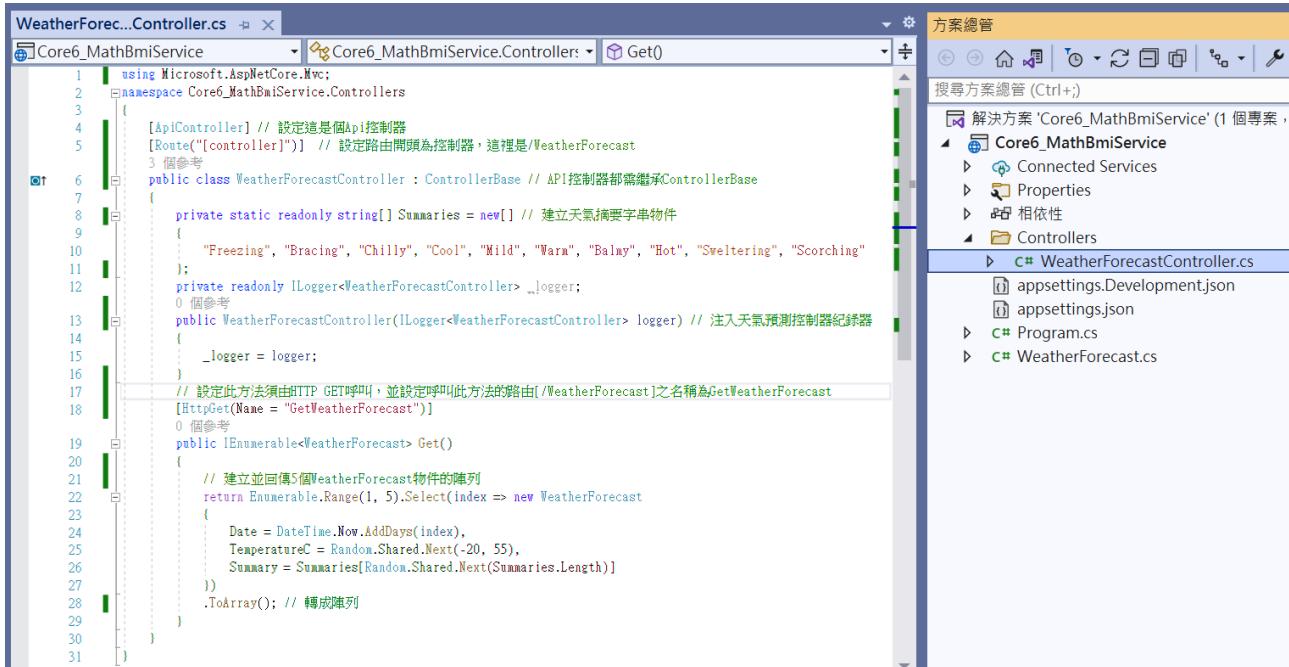
```

1  var builder = WebApplication.CreateBuilder(args); // 建立一個Web應用(WebApplication)建構器
2
3  //== 以下在服務容器內註冊需要的服務(Add services to the container) ==
4  builder.Services.AddControllers(); // 在服務容器中註冊給控制器使用的服務
5  // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
6  builder.Services.AddEndpointsApiExplorer(); // 在服務容器中註冊使用端點資料設定Api探索器的服務
7  builder.Services.AddSwaggerGen(); // 在服務容器中註冊給Swagger產生器使用的服務
8
9  // 建立Web應用(WebApplication)
10 var app = builder.Build();
11
12 //== 以下設定HTTP請求的管線 ( Configure the HTTP request pipeline ) ==
13 if (app.Environment.IsDevelopment())
14 { // 若為開發環境，則使用Swagger及SwaggerUI中介軟體(Middleware)
15     app.UseSwagger();
16     app.UseSwaggerUI();
17 }
18
19 // 使用授權中介軟體
20 app.UseAuthorization();
21
22 // 使用控制器中介軟體
23 // (注意：Web API使用控制器中介軟體；而MVC Web使用控制器及檢視中介軟體)
24 app.MapControllers();
25
26 // 執行Web應用
27 app.Run();

```

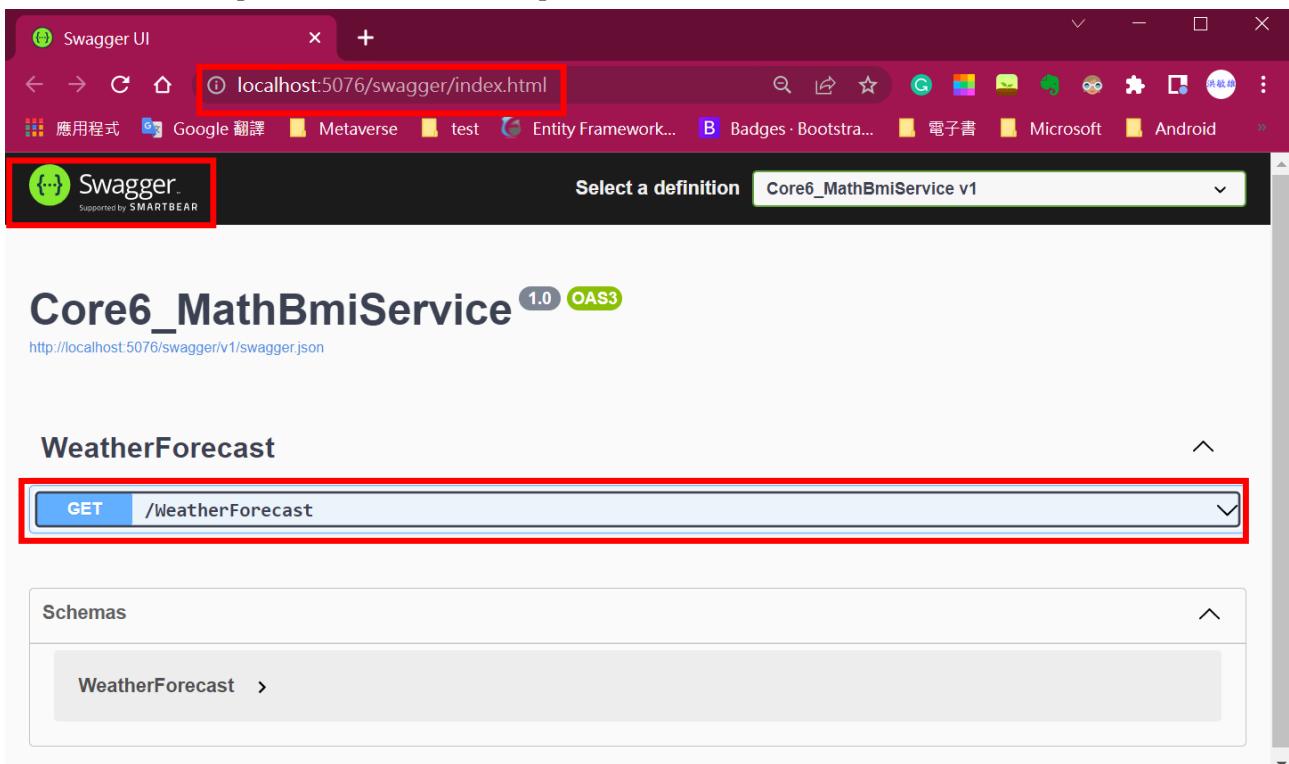
9. 點選[WeatherForecastController.cs]，在編輯器中可看到天氣預測服務的實作，詳細看程式中的中文註解：

● 這個利用 **Get /WeatherForecast** 呼叫這個天氣預測服務，即可傳回 5 筆天氣預測資料：



```
1  using Microsoft.AspNetCore.Mvc;
2  namespace Core6_MathBmiService.Controllers
3  {
4      [ApiController] // 設定這是個API控制器
5      [Route("[controller]")] // 設定路由由開頭為控制器，這裡是/WeatherForecast
6      3 個參考
7      public class WeatherForecastController : ControllerBase // API控制器都需繼承ControllerBase
8      {
9          private static readonly string[] Summaries = new[] // 建立天氣摘要字符串物件
10         {
11             "Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweating", "Scorching"
12         };
13         private readonly ILogger<WeatherForecastController> _logger;
14         0 個參考
15         public WeatherForecastController(ILogger<WeatherForecastController> logger) // 注入天氣預測控制器紀錄器
16         {
17             _logger = logger;
18         }
19         // 設定此方法須由HTTP GET呼叫，並設定呼叫此方法的路徑[/WeatherForecast]之名稱為GetWeatherForecast
20         [HttpGet(Name = "GetWeatherForecast")]
21         0 個參考
22         public IEnumerable<WeatherForecast> Get()
23         {
24             // 建立並回傳5個WeatherForecast物件的陣列
25             return Enumerable.Range(1, 5).Select(index => new WeatherForecast
26             {
27                 Date = DateTime.Now.AddDays(index),
28                 TemperatureC = Random.Shared.Next(-20, 55),
29                 Summary = Summaries[Random.Shared.Next(Summaries.Length)]
30             })
31             .ToArray(); // 轉成陣列
         }
    }
```

10. 同時按下<Ctrl>+<F5>，以不除錯模式執行專案，即可看到此 WebAPI 的 Swagger UI 介面。然後展開[GET /WeatherForecast]：



Swagger UI

localhost:5076/swagger/index.html

Core6_MathBmiService v1

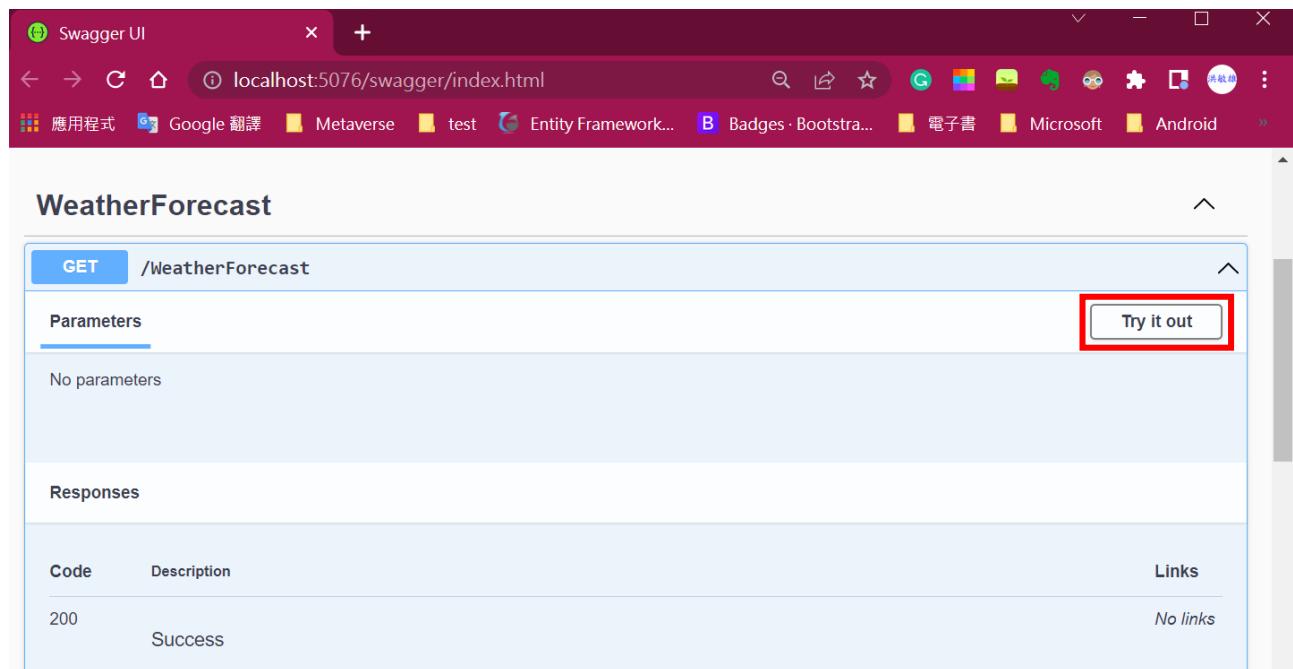
WeatherForecast

GET /WeatherForecast

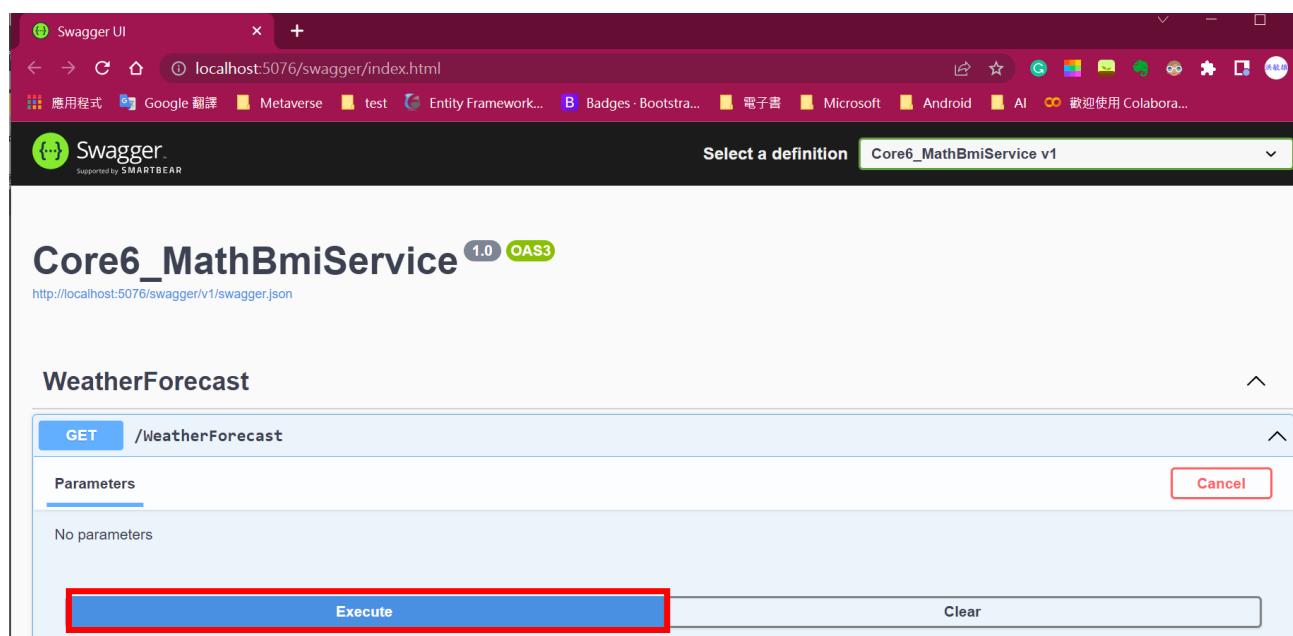
Schemas

WeatherForecast >

11. 點選[Try it out]，然後按下[Excecute]呼叫此 Web API 服務，即可看到回傳的結果(5 個天氣預測物件的陣列：



The screenshot shows the Swagger UI interface for the **WeatherForecast** API. At the top, there's a navigation bar with icons for application, Google Translate, Metaverse, test, Entity Framework, Badges, Microsoft, and Android. The main content area has a title **WeatherForecast**. Below it, a method section shows a **GET /WeatherForecast** entry. Under the **Parameters** section, it says "No parameters". To the right, there's a **Try it out** button, which is highlighted with a red box. The **Responses** section shows a single entry for **Code 200 Success**, with a note "No links".



The screenshot shows the Swagger UI interface for the **Core6_MathBmiService v1** API. At the top, there's a navigation bar with icons for application, Google Translate, Metaverse, test, Entity Framework, Badges, Microsoft, and Android. The main content area has a title **Core6_MathBmiService** with a version **1.0 OAS3**. Below it, it says **http://localhost:5076/swagger/v1/swagger.json**. Under the **WeatherForecast** section, there's a **GET /WeatherForecast** entry. The **Parameters** section says "No parameters". At the bottom, there are two buttons: **Execute** (highlighted with a red box) and **Cancel**.

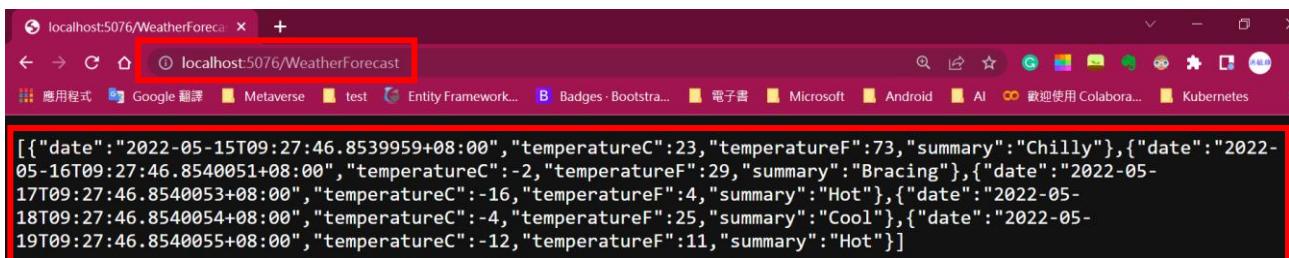
```

[{"date": "2022-05-15T09:20:49.8255619+08:00", "temperatureC": 6, "temperatureF": 42, "summary": "Sweltering"}, {"date": "2022-05-16T09:20:49.8260726+08:00", "temperatureC": -1, "temperatureF": 31, "summary": "Bracing"}, {"date": "2022-05-17T09:20:49.8260764+08:00", "temperatureC": -5, "temperatureF": 24, "summary": "Warm"}, {"date": "2022-05-18T09:20:49.8260767+08:00", "temperatureC": -3, "temperatureF": 27, "summary": "Freezing"}, {"date": "2022-05-19T09:20:49.8260769+08:00", "temperatureC": -1, "temperatureF": 31, "summary": "Hot"}]

```

[Copy](#) [Download](#)

12. 利用瀏覽器測試此服務：直接在瀏覽器輸入這個天氣預測服務的網址 (<http://localhost:5076/WeatherForecast>)，即可看到回傳的結果(5 個天氣預測物件的陣列)：



13. 利用 Postman 測試呼叫此服務：開啟 Postman，選擇[GET]，輸入此服務的網址：<http://localhost:5076/WeatherForecast>，然後按下[Send]，即可看到回傳的結果(5 個天氣預測物件的陣列)：

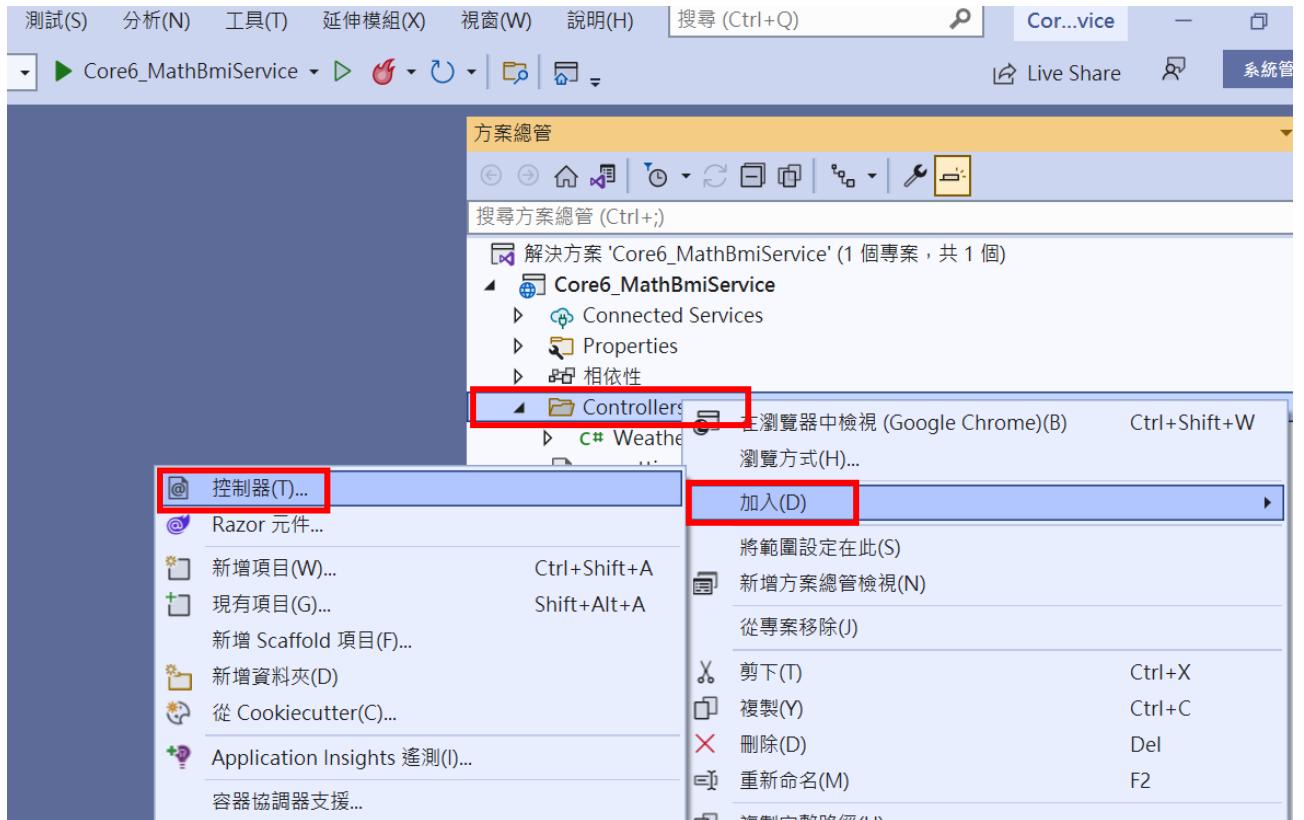
```

[{"date": "2022-05-15T09:51:56.1541305+08:00", "temperatureC": -15, "temperatureF": 6, "summary": "Bracing"}, {"date": "2022-05-16T09:51:56.1541414+08:00", "temperatureC": -16, "temperatureF": 4, "summary": "Bracing"}]

```

二、建立[MathBmiServiceController.cs] Web API Controller，然後建立利用 GET 及路由參數呼叫的 Web API(ComputeMath 服務)

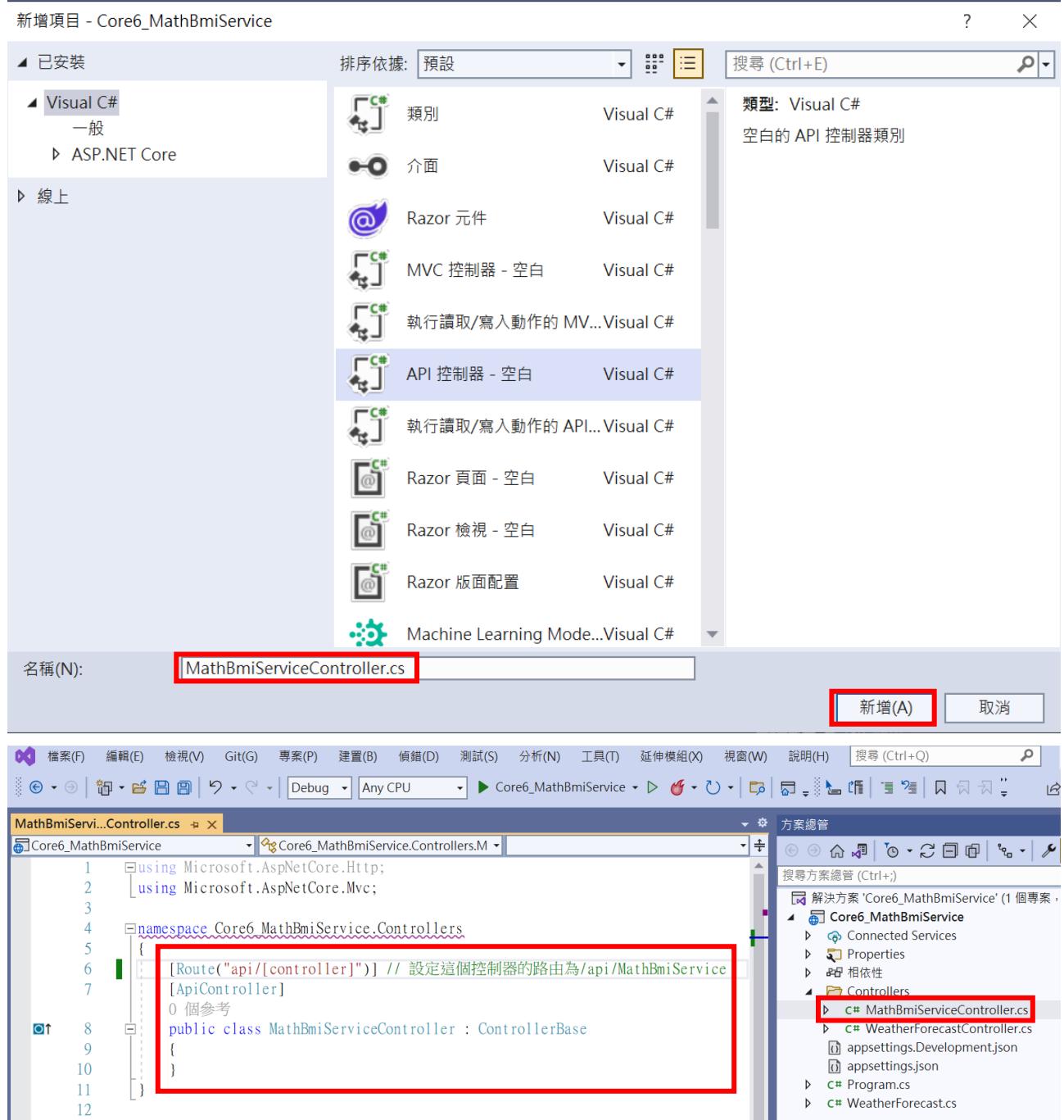
1. 用滑鼠右鍵點選 Controllers 資料夾，選擇[加入]，選擇[控制器]：



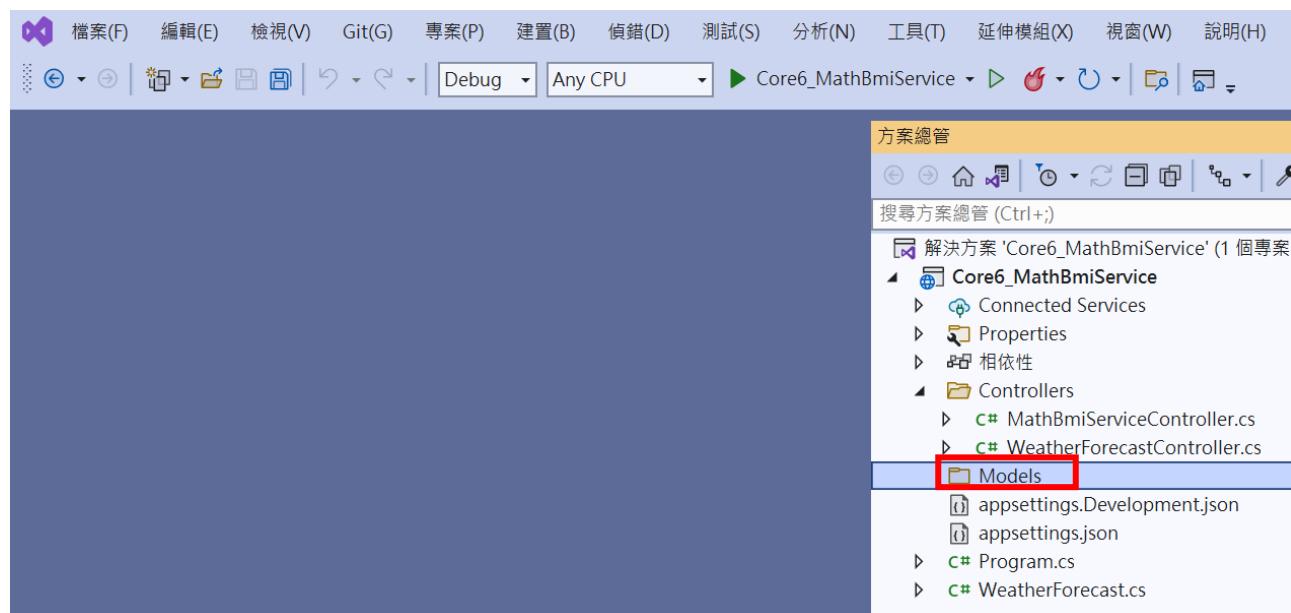
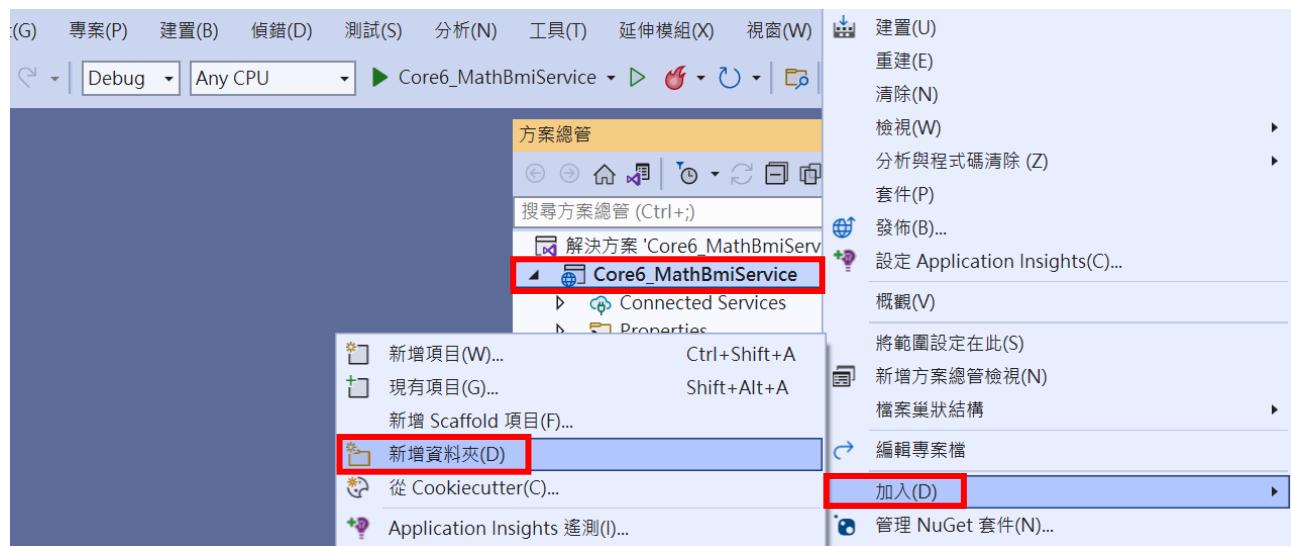
2. 選擇[API]=>[API 控制器-空白]，然後選擇[加入]：



3. 輸入[MathBmiServiceController.cs]，然後按下[新增]，即可看到新增的空白 Web API Controller：

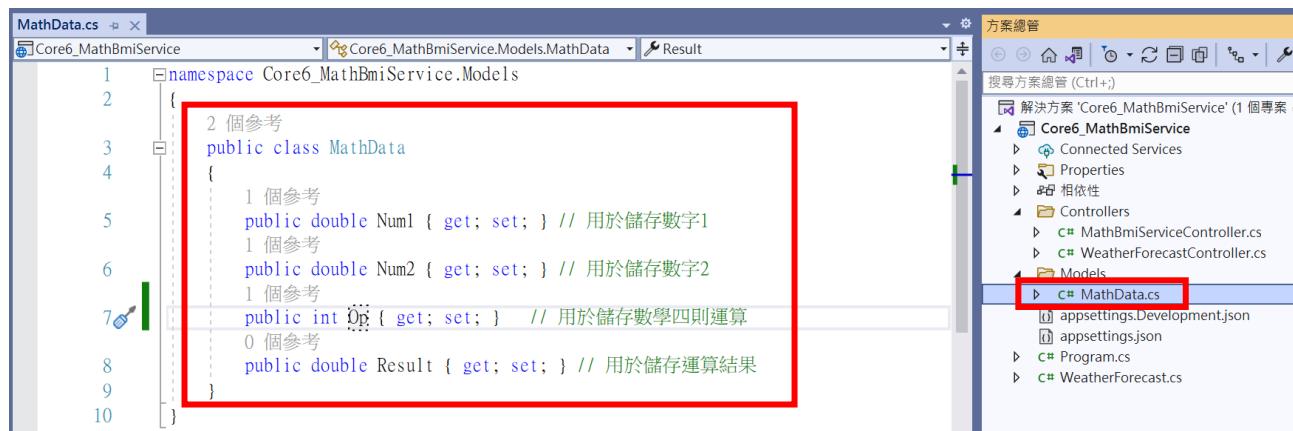


4. 用滑鼠右鍵點選專案名稱，選擇[加入]，選擇[新增資料夾]，將資料夾取名 Models：

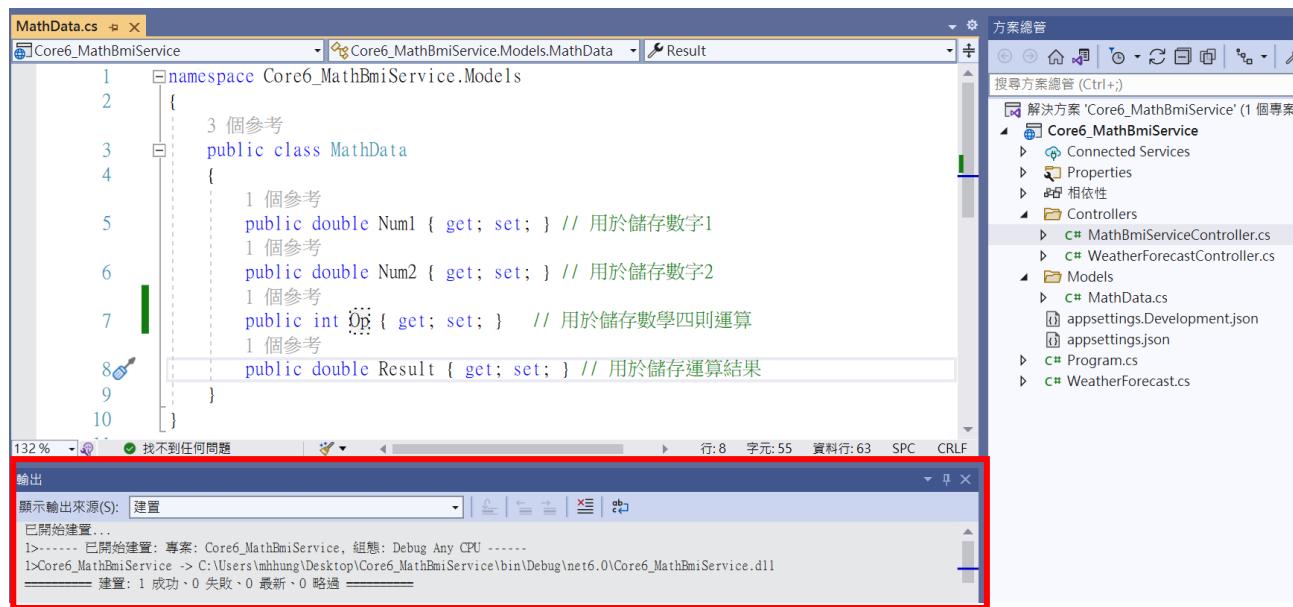


5. 在 Models 資料夾新增[MathData.cs]類別，包含 4 個屬性：

```
public class MathData
{
    public double Num1 { get; set; } // 用於儲存數字1
    public double Num2 { get; set; } // 用於儲存數字2
    public int Op { get; set; } // 用於儲存數學四則運算
    public double Result { get; set; } // 用於儲存運算結果
}
```



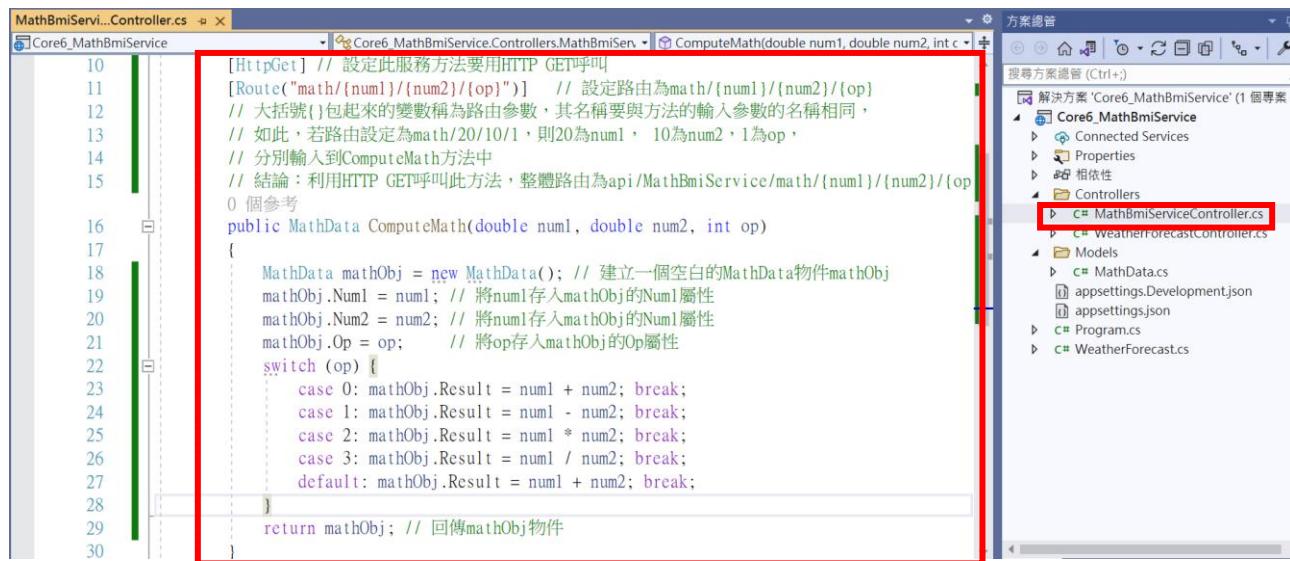
6. 同時按下<Ctrl>+<Shift>+建置專案：



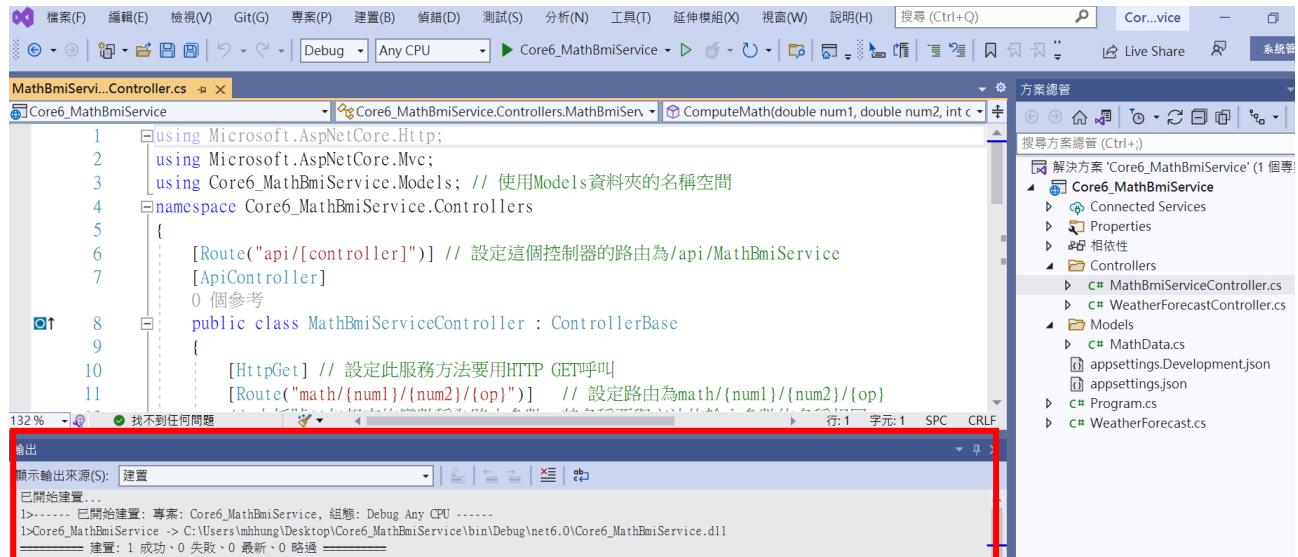
7. 建立 ComputeMath Web API 服務：

- 利用 HTTP GET 呼叫。
- 設定路由為 math/{num1}/{num2}/{op}

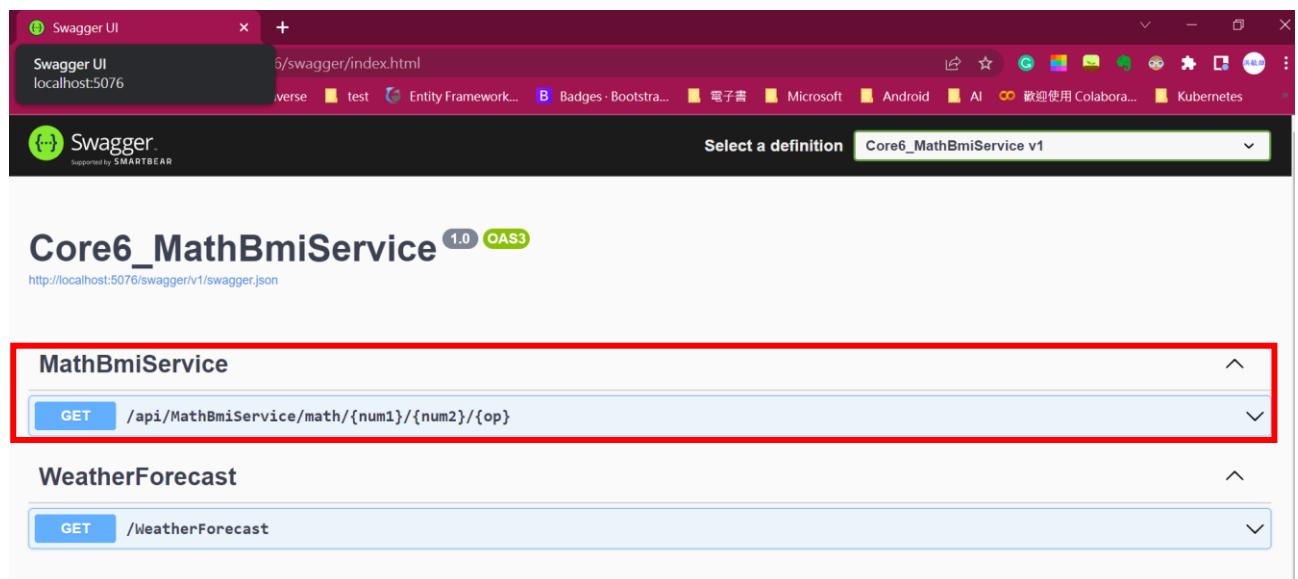
```
using Core6_MathBmiService.Models; // 使用Models資料夾的名稱空間
namespace Core6_MathBmiService.Controllers
{
    [Route("api/[controller]")] // 設定這個控制器的路由為/api/MathBmiService
    [ApiController]
    public class MathBmiServiceController : ControllerBase
    {
        [HttpGet] // 設定此服務方法要用HTTP GET呼叫
        [Route("math/{num1}/{num2}/{op}")] // 設定路由為math/{num1}/{num2}/{op}
        // 大括號{}包起來的變數稱為路由參數，其名稱要與方法的輸入參數的名稱相同，
        // 如此，若路由設定為math/20/10/1，則20為num1， 10為num2，1為op，
        // 分別輸入到ComputeMath方法中
        // 結論：利用HTTP GET呼叫此方法，整體路由為api/MathBmiService/math/{num1}/{num2}/{op}
        public MathData ComputeMath(double num1, double num2, int op)
        {
            MathData mathObj = new MathData(); // 建立一個空白的MathData物件mathObj
            mathObj.Num1 = num1; // 將num1存入mathObj的Num1屬性
            mathObj.Num2 = num2; // 將num2存入mathObj的Num2屬性
            mathObj.Op = op; // 將op存入mathObj的Op屬性
            switch (op)
            {
                case 0: mathObj.Result = num1 + num2; break;
                case 1: mathObj.Result = num1 - num2; break;
                case 2: mathObj.Result = num1 * num2; break;
                case 3: mathObj.Result = num1 / num2; break;
                default: mathObj.Result = num1 + num2; break;
            }
            return mathObj; // 回傳mathObj物件
        }
    }
}
```



8. 同時按下<Ctrl>+<Shift>+建置專案：



9. 同時按下<Ctrl>+<F5>，以不除錯的模式執行專案，即可看到 Swagger UI 介面，已經有了上面建置的 MathBmiService 的 ComputeMath Web API 的操作指引：



10. 利用 Swagger UI 測試此服務：

- 下拉[GET /api/MathBmiService/math/{num1}/{num2}/{op}]，然後點選[Try it out]：

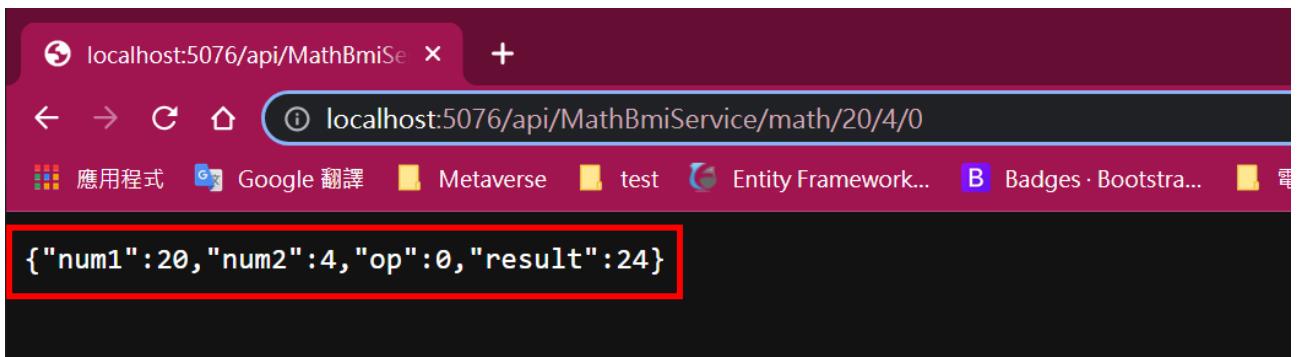
The screenshot shows the 'MathBmiService' API documentation. Under the 'Parameters' section for the GET /api/MathBmiService/math/{num1}/{num2}/{op} endpoint, there are three input fields: 'num1' (value: num1), 'num2' (value: num2), and 'op' (value: op). The 'Try it out' button is highlighted with a red box.

- 輸入 num1(例如：20)，num2(例如：4)及 op(例如：0)的值，然後按下[Execute]，即可看到回傳物件的 JSON 格式結果：

The screenshot shows the 'MathBmiService' API documentation. The 'Parameters' section has three input fields: 'num1' (value: 20), 'num2' (value: 4), and 'op' (value: 0). The 'Execute' button is highlighted with a red box. Below the form, the 'Request URL' is displayed as <http://localhost:5076/api/MathBmiService/math/20/4/0>. The 'Server response' section shows a 200 status code. The 'Response body' is a JSON object: { "num1": 20, "num2": 4, "op": 0, "result": 24 }. The 'Response headers' section includes: content-type: application/json; charset=utf-8, date: Sat, 14 May 2022 03:29:57 GMT, server: Kestrel, transfer-encoding: chunked. A 'Download' button is also present.

11. 利用瀏覽器測試此服務：

- 在瀏覽器輸入此服務的路由網址：<http://localhost:5076/api/MathBmiService/math/20/4/0>



12. 利用 Postman 測試呼叫此服務：

- 開啟 Postman，選擇[GET]，輸入此服務的網址：

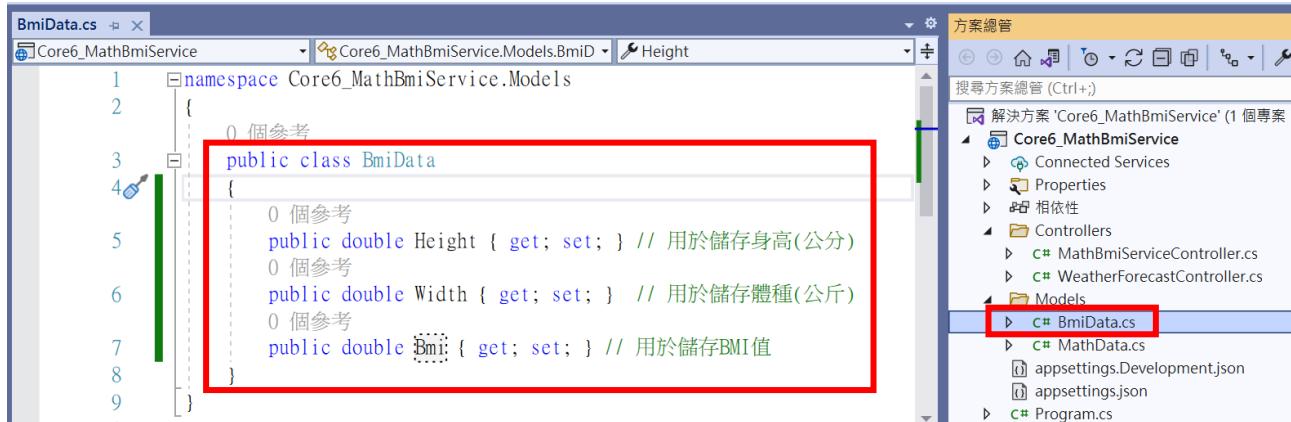
<http://localhost:5076/api/MathBmiService/math/20/4/0>，然後按下[Send]，即可看到回傳的結果：

A screenshot of the Postman application. The left sidebar shows 'My Workspace' with collections, APIs, environments, mock servers, monitors, flows, and history. The main area shows a 'My Collection' entry for 'http://localhost:5076/api/MathBmiService/math/20/4/0'. A GET request is selected with the URL <http://localhost:5076/api/MathBmiService/math/20/4/0> in the 'Send' button. The 'Body' tab shows a JSON response: {"num1": 20, "num2": 4, "op": 0, "result": 24}, which is highlighted with a red rectangular box. The status bar at the bottom indicates a 200 OK response with 10 ms latency and 187 B size.

三、建立利用 POST 及 JSON Request Body 呼叫的 Web API(ComputeBmi 服務)

- 在 Models 資料夾新增[BmiData.cs]類別，包含 3 個屬性，然後重建專案：

```
public class BmiData
{
    public double Height { get; set; } // 用於儲存身高(公分)
    public double Width { get; set; } // 用於儲存體重(公斤)
    public double Bmi { get; set; } // 用於儲存BMI值
}
```



- 建立 ComputeBmi Web API 服務：

- 利用 HTTP POST 呼叫。
- 設定路由為 bmi 且利用 Request Body 傳送 JSON 格式物件給方法的輸入參數：

```
[HttpPost] // 設定此服務方法要用HTTP POST呼叫
[Route("bmi")] // 設定路由為bmi
// 結論：利用HTTP POST呼叫此方法，整體路由為api/MathBmiService/bmi
// 利用Request Body傳送JSON物件給方法的輸入參數
// 例如：POST /api/MathBmiService/bmi
// {"Height":170, "Weight":67}
public BmiData ComputeMath([FromBody] BmiData bmiObj)
{
    // 計算BMI值：BMI=體重(公斤)除以身高(公尺)的平方
    bmiObj.Bmi = bmiObj.Weight / ((bmiObj.Height / 100.0) * (bmiObj.Height / 100.0));
    return bmiObj; // 回傳bmiObj物件
}
```



14. 同時按下<Ctrl>+<Shift>+建置專案：

The screenshot shows the Visual Studio IDE interface during the build process of the 'Core6_MathBmiService' project. The code editor displays the 'MathBmiServiceController.cs' file, which contains a POST method for calculating BMI. The output window at the bottom shows the build logs:

```
1>----- 已開始建置：專案: Core6_MathBmiService, 組態: Debug Any CPU -----  
1>Core6_MathBmiService -> C:\Users\mhung\Desktop\Core6_MathBmiService\bin\Debug\net6.0\Core6_MathBmiService.dll  
----- 建置: 1 成功、0 失敗、0 最新、0 略過 -----
```

15. 同時按下<Ctrl>+<F5>，以不除錯的模式執行專案，即可看到 Swagger UI 介面，已經有了上面建置的 MathBmiService 的 ComputeBmi Web API 的操作指引：

The screenshot shows the Swagger UI interface for the 'Core6_MathBmiService' project. The top navigation bar shows the URL as 'localhost:5076'. The main content area displays the API documentation for the 'MathBmiService' controller. It includes a 'Select a definition' dropdown set to 'Core6_MathBmiService v1'. Below it, there are two API endpoints listed:

- A GET endpoint: /api/MathBmiService/math/{num1}/{num2}/{op}
- A POST endpoint: /api/MathBmiService/bmi

16. 利用 Swagger UI 測試此服務：

- 下拉[POST /api/MathBmiService/bmi]，然後點選[Try it out]：

The screenshot shows the 'Try it out' interface for the POST /api/MathBmiService/bmi endpoint. At the top, there is a green button labeled 'POST' and the URL '/api/MathBmiService/bmi'. To the right of the URL is a red-bordered button labeled 'Try it out'. Below the URL, there is a section titled 'Parameters' with a green underline, followed by the message 'No parameters'. Under 'Request body', there is a dropdown menu set to 'application/json'. In the 'Example Value' section, there is a code block containing the following JSON:

```
{  
    "height": 0,  
    "weight": 0,  
    "bmi": 0  
}
```

- 在 Request body 中輸入 height(例如：170)，weight(例如：67)，然後按下[Execute]，即可看到回傳物件的 JSON 格式結果：

The screenshot shows the 'Execute' interface for the same endpoint. The 'Request body' field contains the following JSON, which is highlighted with a red rectangle:

```
{  
    "height": 170,  
    "weight": 67,  
    "bmi": 0  
}
```

Below the request body is a large blue rectangular area representing the 'Execute' button, which is also highlighted with a red rectangle. At the bottom of the interface, there is a 'Server response' section. It shows a table with two columns: 'Code' and 'Details'. Under 'Code', there is a row for '200'. Under 'Details', there is a 'Response body' section containing the following JSON, which is highlighted with a red rectangle:

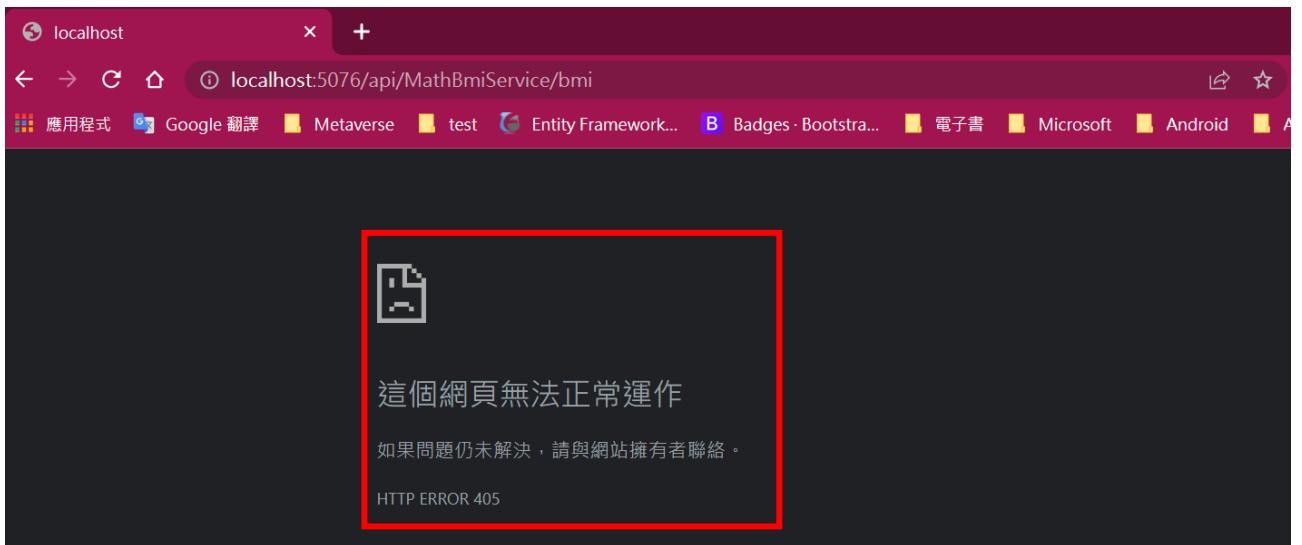
```
{  
    "height": 170,  
    "weight": 67,  
    "bmi": 23.18339100346021  
}
```

Next to the JSON is a 'Download' button. Below the response body, there is a 'Response headers' section containing the following text, which is highlighted with a red rectangle:

```
content-type: application/json; charset=utf-8  
date: Sat,14 May 2022 04:15:43 GMT  
server: Kestrel  
transfer-encoding: chunked
```

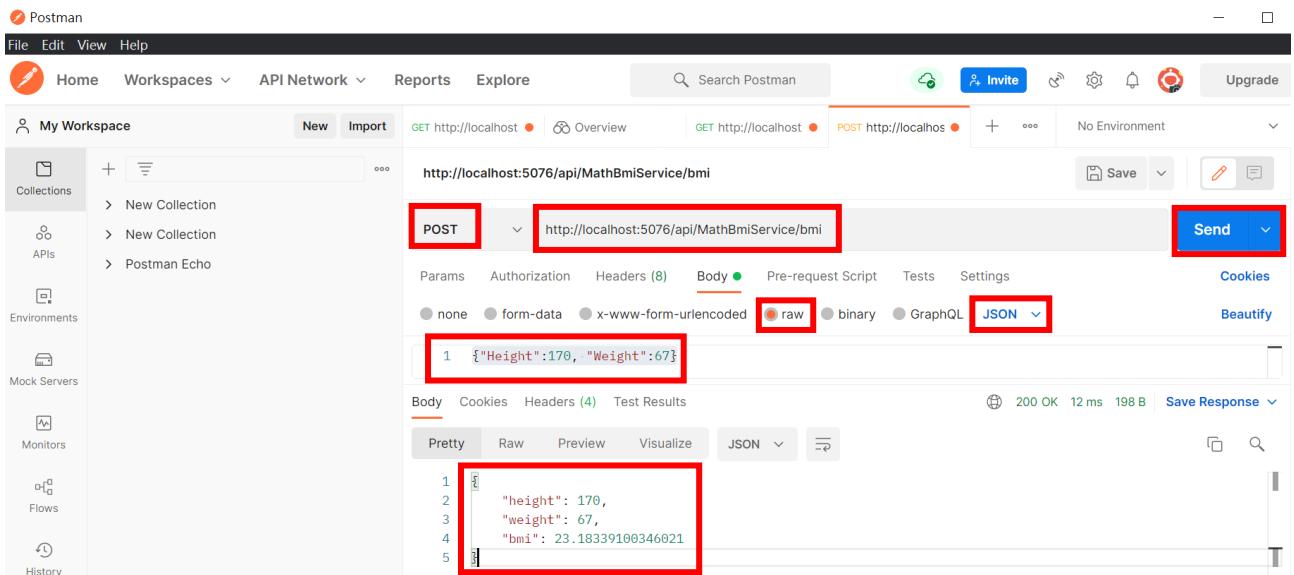
17. 無法利用瀏覽器測試此服務(因為瀏覽器的請求預設使用 GET)

- 在瀏覽器輸入此服務的路由網址：<http://localhost:5076/api/MathBmiService/bmi>



18. 利用 Postman 測試呼叫此服務：

- 開啟 Postman，選擇[POST]
- 輸入此服務的網址：<http://localhost:5076/api/MathBmiService/bmi>，
- 選擇[Body]，選擇[raw]，[選擇 JSON]，然後輸入 `{"Height":170, "Weight":67}`
- 然後按下[Send]，即可看到回傳的結果：



四、將此專案部署到私有雲虛擬機上，使它成為 REST 雲端服務

- 開啟 NuGet 套件管理頁面，新增 **NSwag.AspNetCore** 套件，改用 NSwag 產生 Web API 文件。
- 修改[Program.cs]，改用 Swagger 2.0 及 Open API 3.0 文件：

● 修改 1：

原來：

```
// 在服務容器中註冊給Swagger產生器使用的服務  
builder.Services.AddSwaggerGen();
```

改為：

```
// 在服務容器中註冊給Swagger 2.0產生器使用的服務(產生符合Open API 3.0的文件)  
builder.Services.AddSwaggerDocument();
```

● 修改 2：

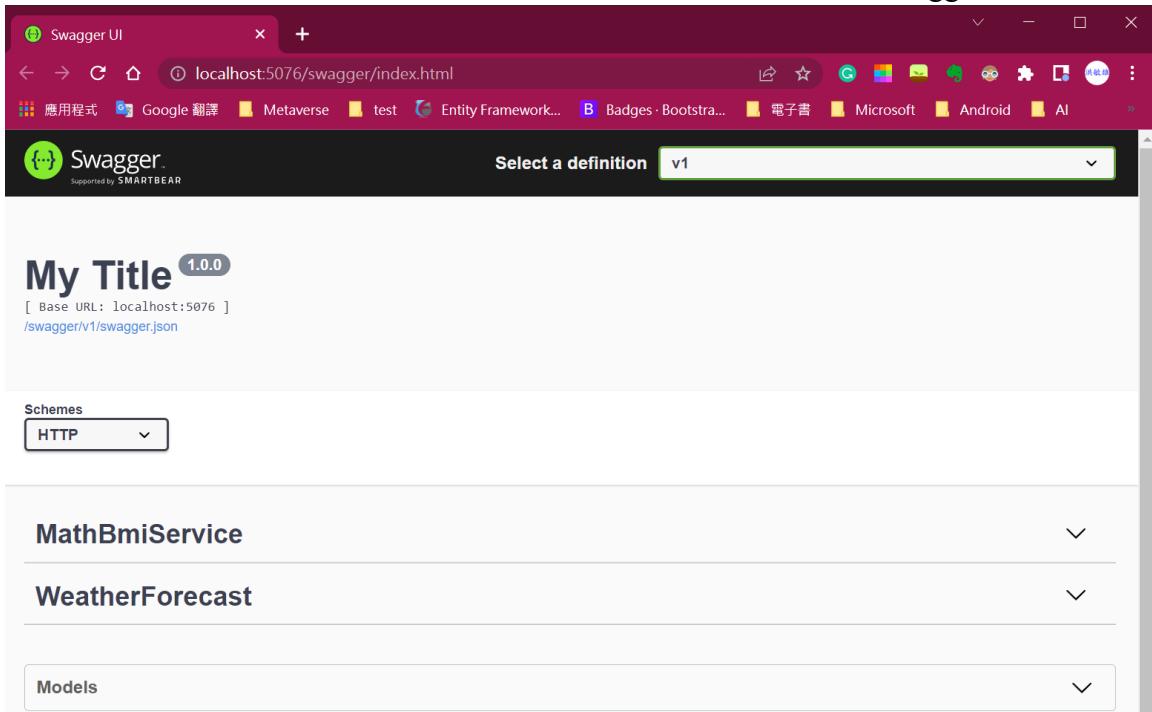
原來：

```
// 假如是開發環境，才使用Swagger及SwaggerUI中介軟體(Middleware)  
if (app.Environment.IsDevelopment())  
{  
    app.UseSwagger();  
    app.UseSwaggerUI();  
}
```

改為：

```
// 不管開發環境或部署的環境都使用Swagger產生器及Swagger UI  
// 增加Swagger產生器及Swagger UI中介軟體到Request Pipeline  
app.UseOpenApi(); // 使用Swagger 2.0 (OpenApi)產生器中介軟體  
app.UseSwaggerUi3(); // 使用 Swagger UI 3.0 中介軟體
```

- 重新建置專案。
- 同時按下<Ctrl>+<F5>，以不除錯的模式執行專案，可看到新的 Swagger UI：



5. 展開 MathBmiService，進行本機端的測試，驗證測試成功：

- 測試 ComputeMath Web API 服務：

MathBmiService

GET /api/MathBmiService/math/{num1}/{num2}/{op}

Parameters

Cancel

Name	Description
num1 * required	number(\$double) (path)
num2 * required	number(\$double) (path)
op * required	integer(\$int32) (path)

Execute Clear

Request URL

http://localhost:5076/api/MathBmiService/math/20/4/2

Server response

Code Details

200 Response body

```
{  
    "num1": 20,  
    "num2": 4,  
    "op": 2,  
    "result": 80  
}
```

Download

● 測試 ComputeBmi Web API 服務：

The screenshot shows the Swagger UI interface for testing a Web API. The URL is `localhost:5076/swagger/index.html#/`. The main area displays the `POST /api/MathBmiService/bmi` endpoint. The **Parameters** section includes a required parameter `bmiObj` with a JSON schema:

```
{  
    "height": 170,  
    "weight": 67,  
    "bmi": 0  
}
```

Below the parameters are **Execute** and **Clear** buttons. The **Responses** section shows a **Curl** command:

```
curl -X 'POST' \  
'http://localhost:5076/api/MathBmiService/bmi' \  
-H 'accept: text/plain' \  
-H 'Content-Type: application/json' \  
-d '{  
    "height": 170,  
    "weight": 67,  
    "bmi": 0  
'
```

The **Request URL** is `http://localhost:5076/api/MathBmiService/bmi`. The **Server response** section shows a 200 status with a JSON body:

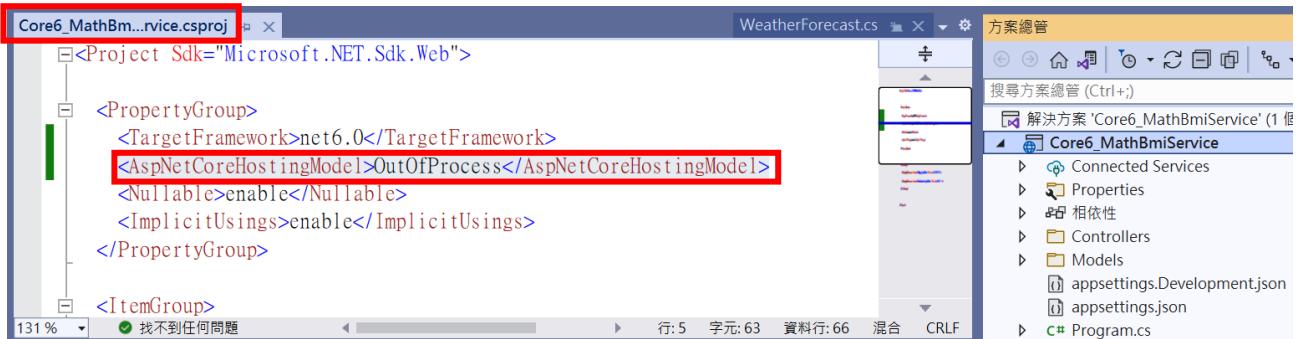
```
{  
    "height": 170,  
    "weight": 67,  
    "bmi": 23.18339100346021  
}
```

With **Download** and **Edit** buttons.

6. 本機測試成功後，開始將專案部署到私有雲虛擬機之網頁伺服器上：

- 用滑鼠右鍵點選專案名稱，選擇[編輯專案檔]，設定使用 IIS out-of-process 模式運行 AspNetCore 應用：

<AspNetCoreHostingModel>OutOfProcess</AspNetCoreHostingModel>

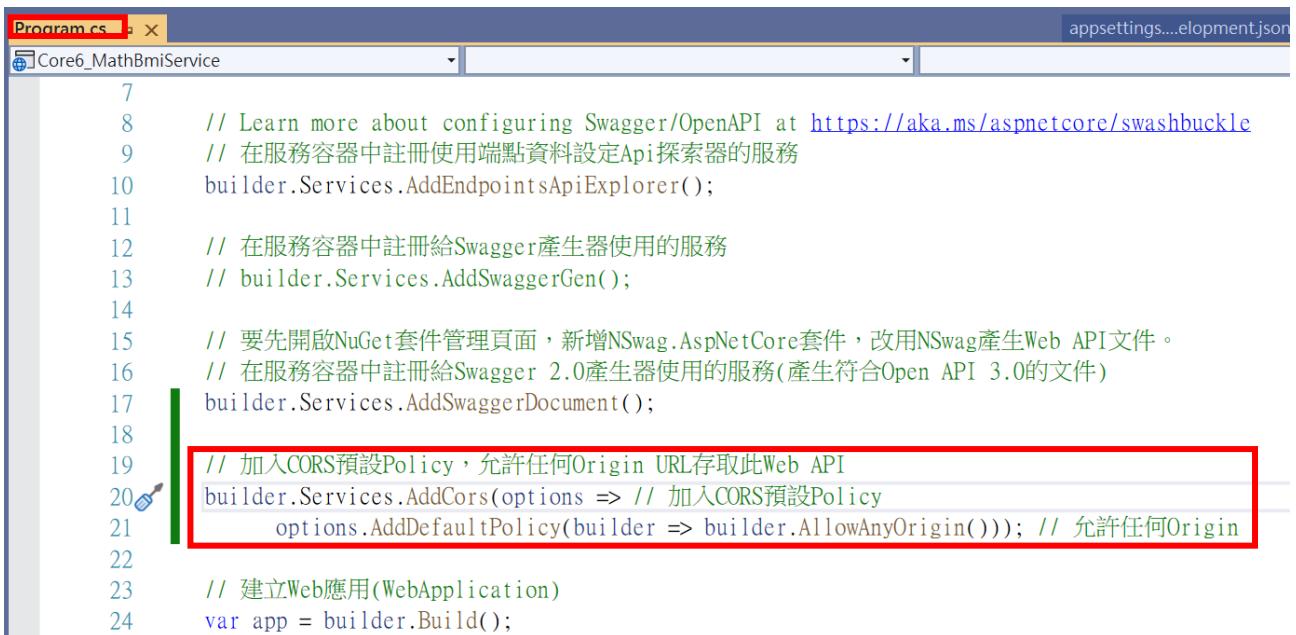


- 在 Program.cs 設定允許非同源 URL 存取此 Web API 雲端服務：

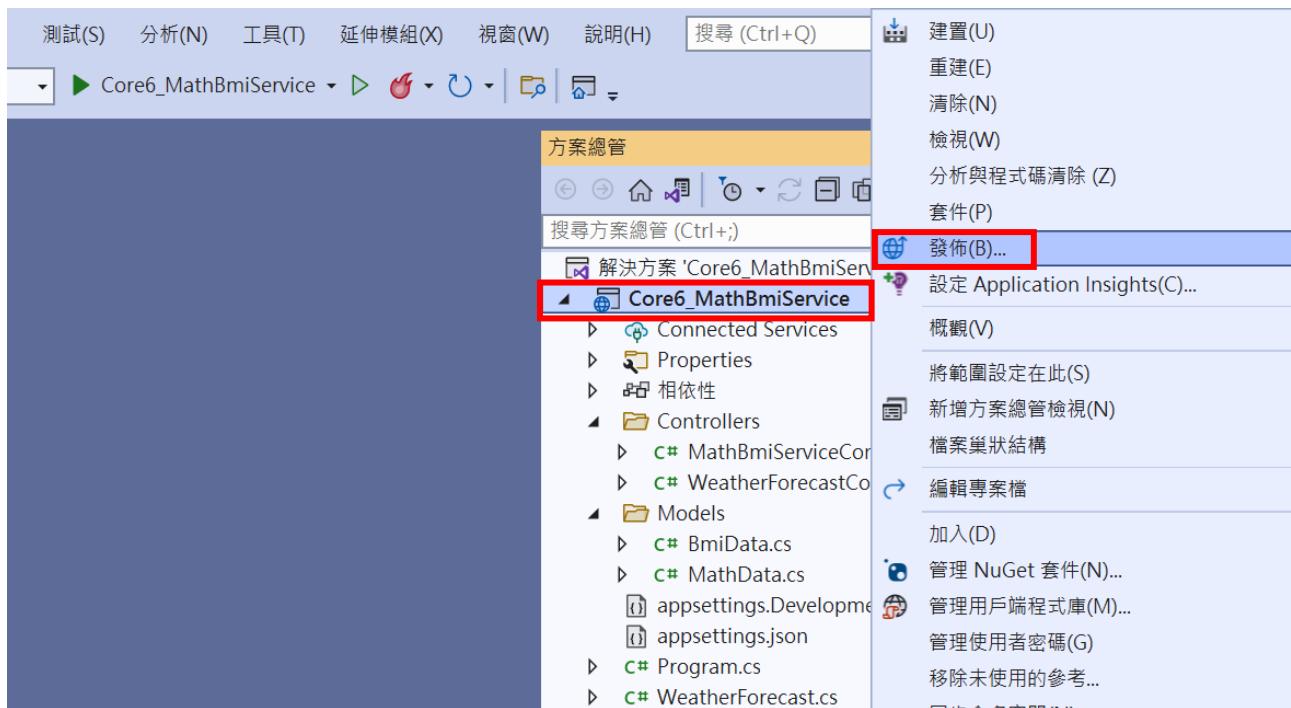
// 加入CORS預設Policy，允許任何Origin URL存取此Web API

builder.Services.AddCors(options => // 加入CORS預設Policy

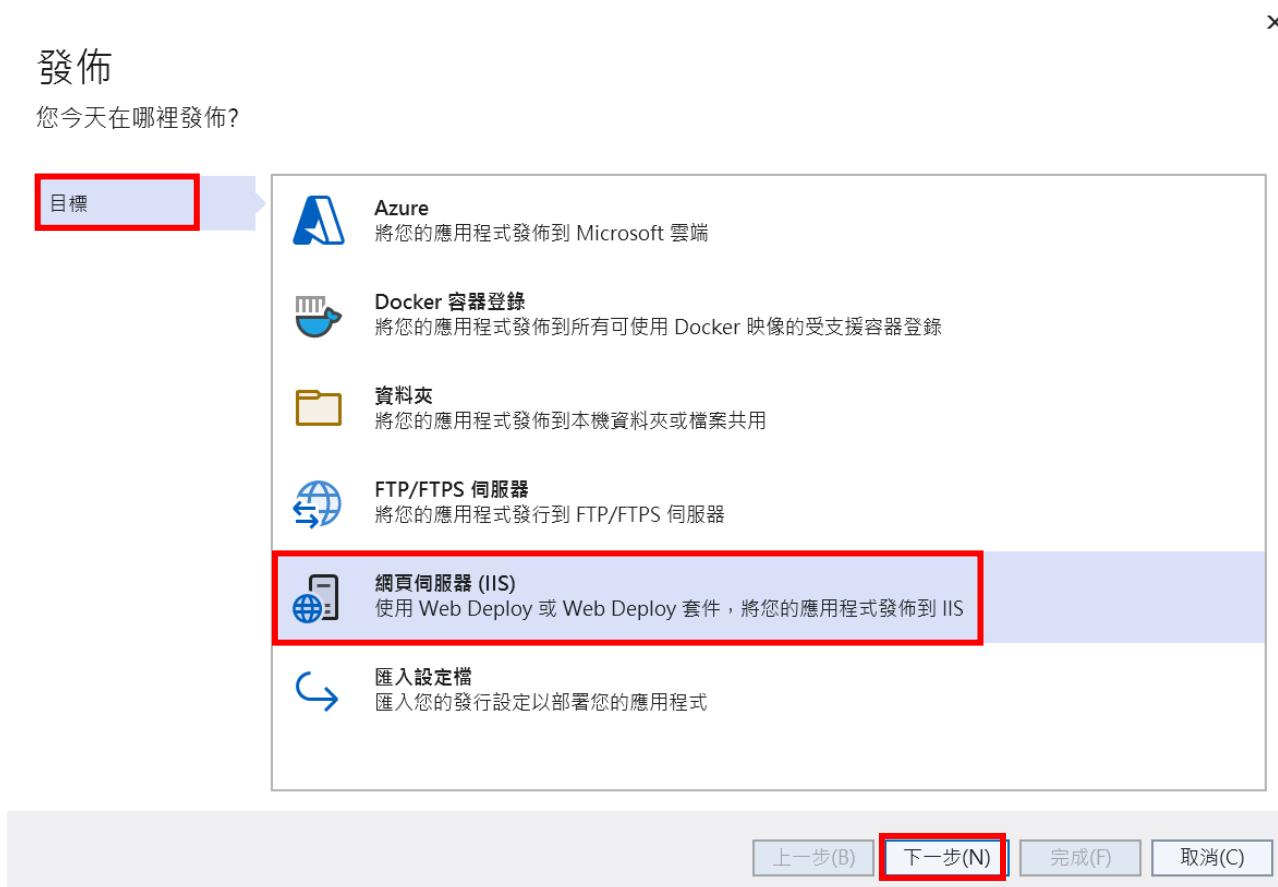
options.AddDefaultPolicy(builder => builder.AllowAnyOrigin()); // 允許任何Origin



- 用滑鼠點選專案名稱，選擇[發佈]：



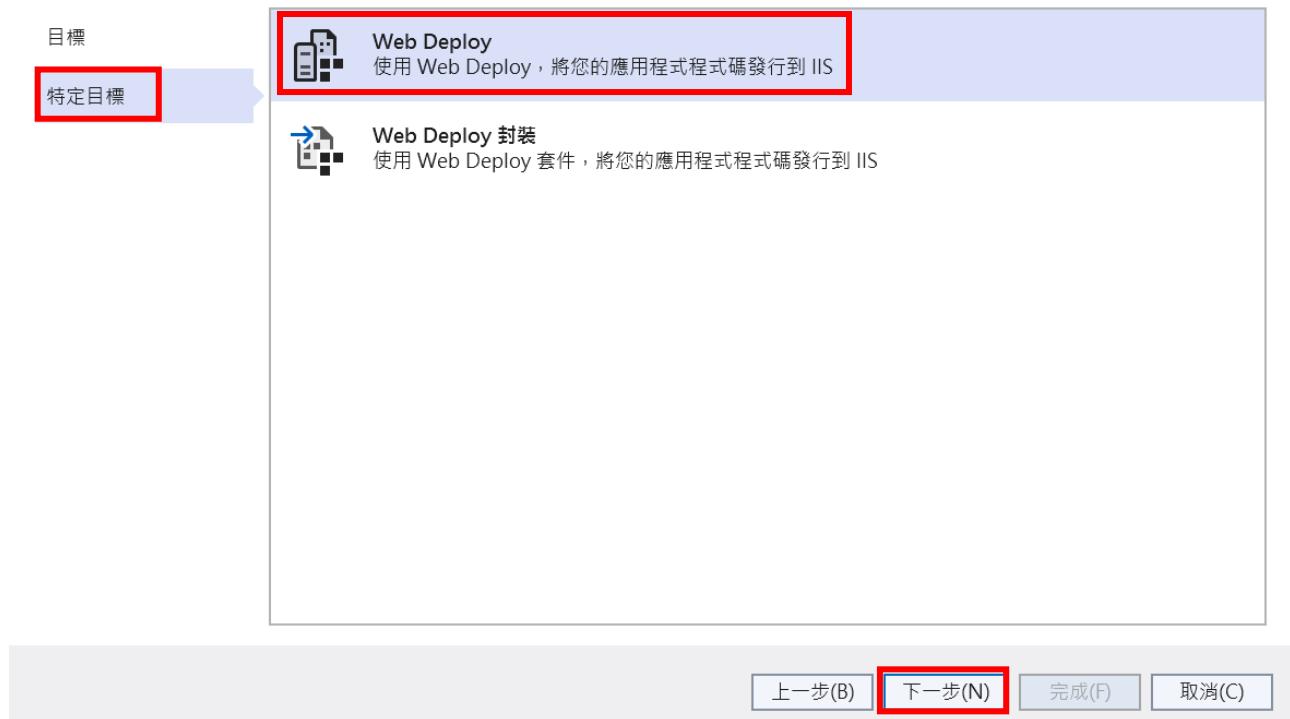
- 在[目標]頁面，選擇[網頁伺服器(IIS)]，然後選擇[下一步]：



- 在[特定目標]頁面，選擇[Web Deploy]，然後選擇[下一步]：

發佈

使用 Web Deploy 或 Web Deploy 套件，將您的應用程式發佈到 IIS



- 在[IIS 連線]頁面，輸入以下私有雲虛擬機之部署資訊，然後點選[完成]：

伺服器名稱：**140.137.41.136:1782** (每位同學都使用這個伺服器名稱)

網站名稱：**default web site/學號/專案名稱**

例如學號為A1234567，專案名稱為Core6_MathBmiService，則網站名稱為：

default web site/A1234567/Core6_MathBmiService

(請務必將專案部署在你的學號名稱目錄之下)

目的地URL：**http://140.137.41.136:1780/A1234567/專案名稱**

例如：專案名稱為Core6_MathBmiService，則URL為：

http://140.137.41.136:1780/A1234567/Core6_MathBmiService/swagger/index.html

使用者名稱：**ccpcsiepccu** (每位同學都使用這個帳號進行部署)

密碼：**\$A123a123!** (每位同學都使用這個密碼進行部署)

發佈

設定 Web 伺服器 (IIS) 連線

目標

伺服器
140.137.41.136:1782

特定目標

網站名稱
default web site/A1234567/Core6_MathBmiService

IIS 連線

目的地 URL(D)
http://140.137.41.136:1780/A1234567/Core6_MathBmiService/swagger/index.html

使用者名稱
ccpcsiepccu

密碼
●●●●●●●●●●●●

儲存密碼

驗證連線

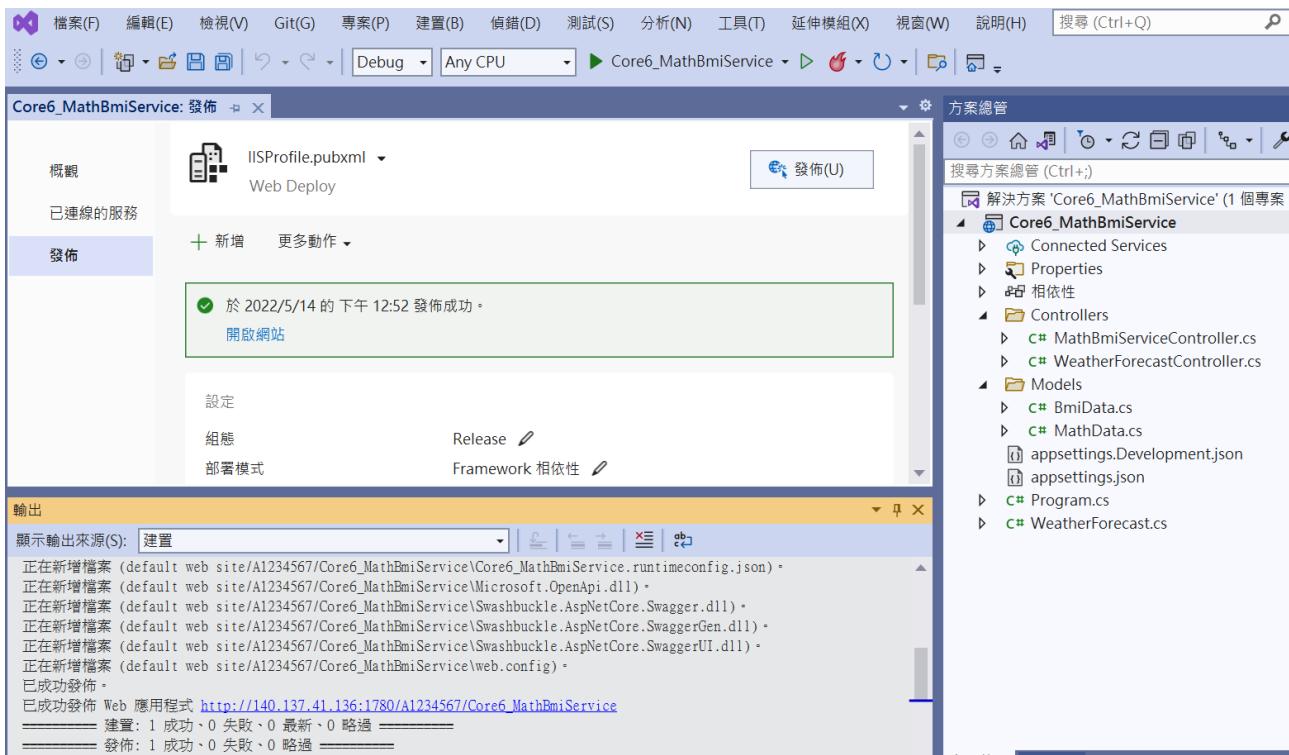
上一步(B)

下一步(N)

完成(F)

取消(C)

● 在[發佈]頁面點選[發佈]：



7. 瀏覽器自動顯示 Swagger UI 介面，網址顯示為私有雲虛擬機的公開 IP(14.137.41.136)，請比照本機的測試，進行這個雲端服務的測試：

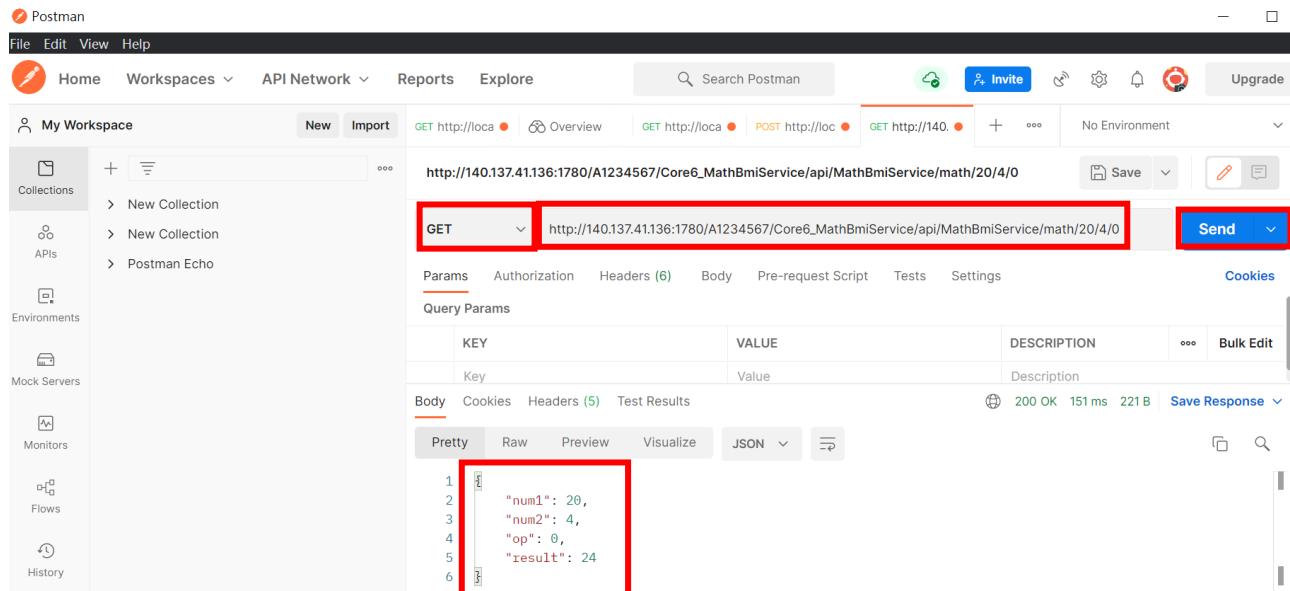
The screenshot shows a browser window displaying the Swagger UI interface for the 'MathBmiService'. The URL in the address bar is http://140.137.41.136:1780/A1234567/Core6_MathBmiService/swagger/index.html. The interface includes sections for 'MathBmiService' and 'WeatherForecast'.

8. 利用 Postman 測試呼叫 ComputeMath Web API 雲端服務：

- 開啟 Postman，選擇[GET]，輸入此服務的網址：

http://140.137.41.136:1780/A1234567/Core6_MathBmiService/api/MathBmiService/math/20/4/0

然後按下[Send]，即可看到回傳的結果：



The screenshot shows the Postman interface with a red box highlighting the URL field containing the API endpoint. The 'Send' button is also highlighted. Below the URL, the 'Body' tab is selected, showing a JSON response with the following content:

```
1
2
3
4
5
6
    "num1": 20,
    "num2": 4,
    "op": 0,
    "result": 24
```

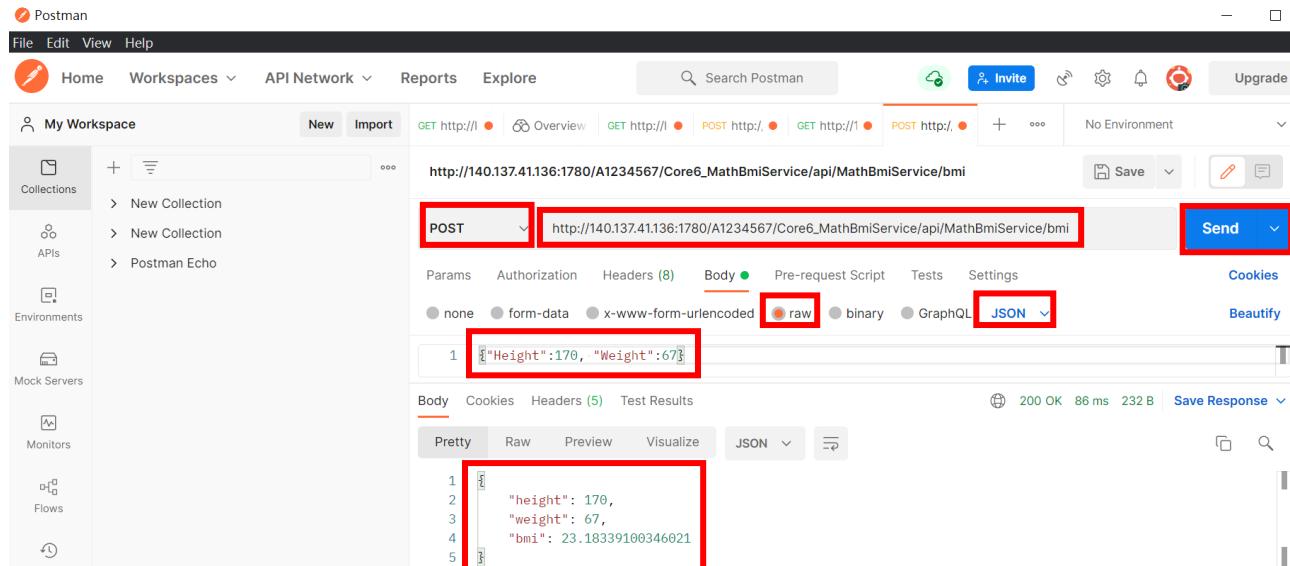
9. 利用 Postman 測試呼叫 ComputeBmi Web API 雲端服務：

- 開啟 Postman，選擇[POST]

- 輸入此服務的網址：

http://140.137.41.136:1780/A1234567/Core6_MathBmiService/api/MathBmiService/bmi

- 選擇[Body]，選擇[raw]，[選擇 JSON]，然後輸入 `{"Height":170, "Weight":67}`
- 然後按下[Send]，即可看到回傳的結果：



The screenshot shows the Postman interface with a red box highlighting the 'POST' method and the URL. The 'Send' button is also highlighted. Below the URL, the 'Body' tab is selected, showing the 'raw' option and a JSON payload:

```
1
2
3
4
5
{"Height":170, "Weight":67}
```

The response body is displayed below, with a red box highlighting the JSON output:

```
1
2
3
4
5
    "height": 170,
    "weight": 67,
    "bmi": 23.18339100346021
```