114年智慧行動商務系統開發班第1期

伺服端程式設計與開發學習評量

|  |  |  |  |
| --- | --- | --- | --- |
| **評量編號** | 2-5-2 | | |
| **評量主題** | Restful API實作 | **實施時間** | 2025/8/15 |
| **評量形式** | 作業紀錄 | **評量日期** | 2025/9/2 |
| **評量目的** | 評定學員是否具備下列能力：   1. 能介接第三方Open Data。 2. 能使用ASP.net Core製作Restful API。 3. 能實作不同的查詢Get方法。 4. 能實作Service與DI。 5. 能提升程式碼之內聚力並降低耦合力。 | | |
| **評量說明** | **請使用題目所指定之方式實作Restful AP，並使用Swagger測試成功。**  **任務一：**第三方Open Data API資料介接。  **任務二：**分離商業邏輯與控制邏輯。  **任務三：**程式架構必須為高內聚力及低耦合力。 | | |
| **提交要求** | 繳交時請以FTP上傳**相關程式檔案**與**作業文件檔**各一份，**相關程式檔案**請另放置在「完成結果檔」資料夾內，最外層資料夾名稱請用**兩碼學號與姓名**，例如1號王小明，資料夾請命名為「01王小明」。 | | |

座號： 05 姓名： 鄭冠怡

**任務一：第三方Open Data API資料介接。**

1. 請自選一個來第三方的Open Data資料，資料種類不限。
2. 將第三方資料介接至自己的API。
3. 至少實作三種不同的資料查詢Get方法。
4. 在Swagger測試所有Get方法且必須可成功執行。

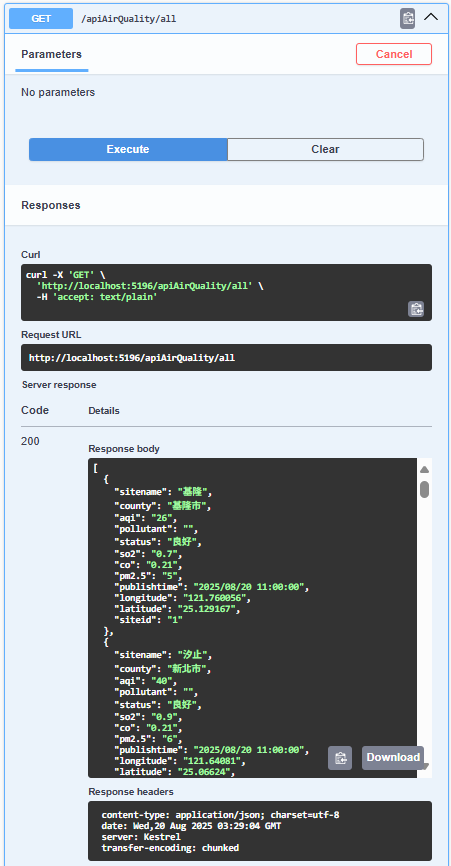
**任務二：分離商業邏輯與控制邏輯。**

1. 將商業邏輯實作成服務(Service)。
2. 將控制邏輯置於Controller裡。
3. 實作依賴注入(DI)將商業邏輯Service注入Controller中供使用。

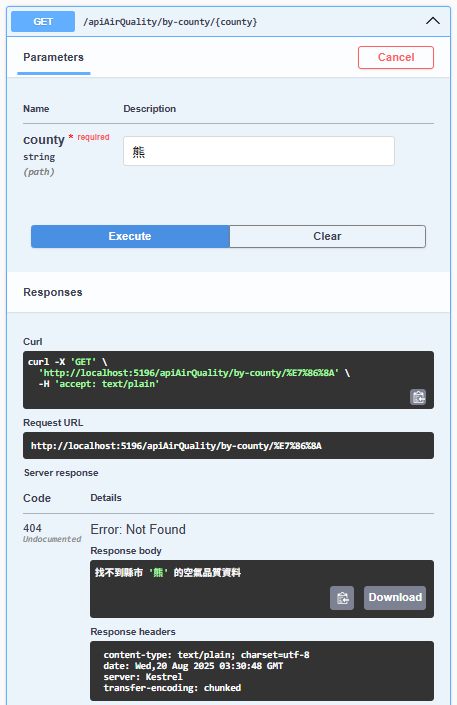
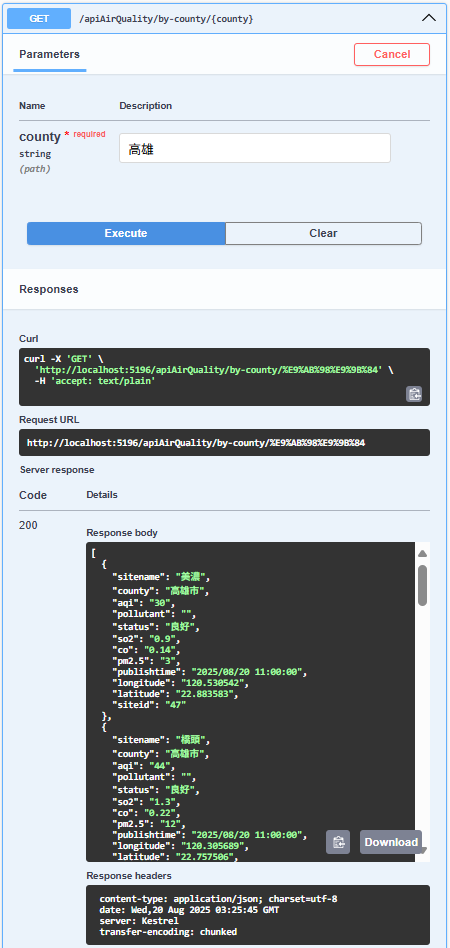
**任務三：程式架構必須為高內聚力及低耦合力。**

1. Swagger執行結果畫面(每個Get方法均要有至少一張圖片)：

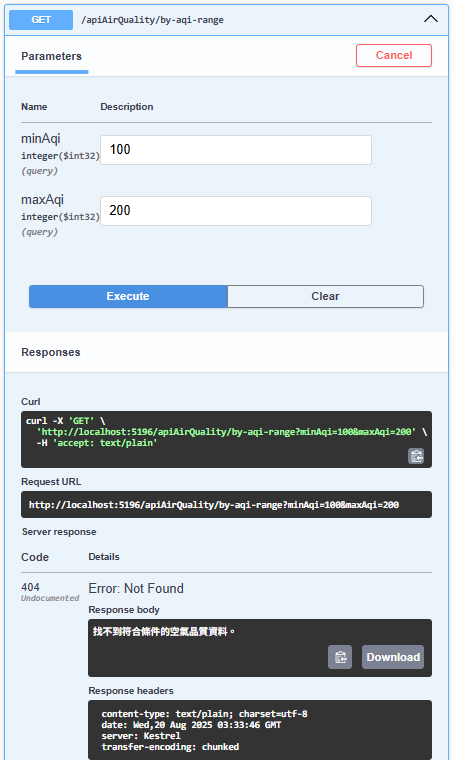
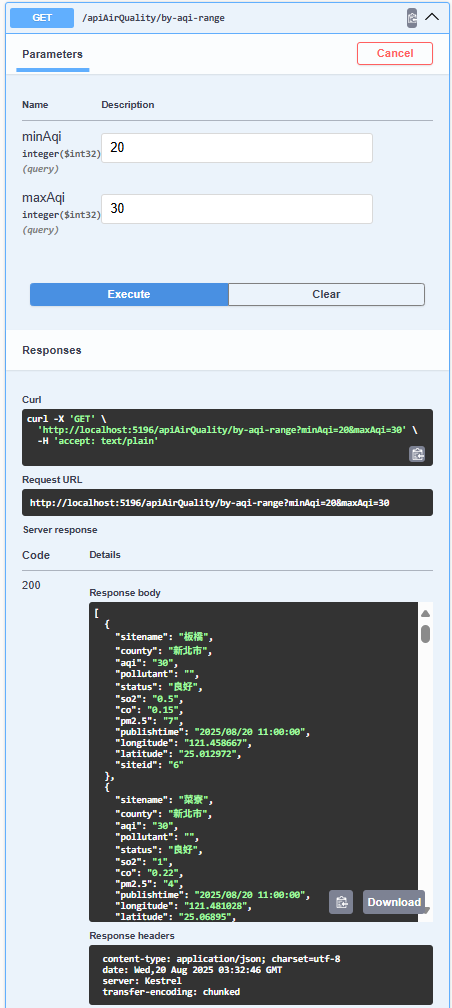
GetAllAirQualityData



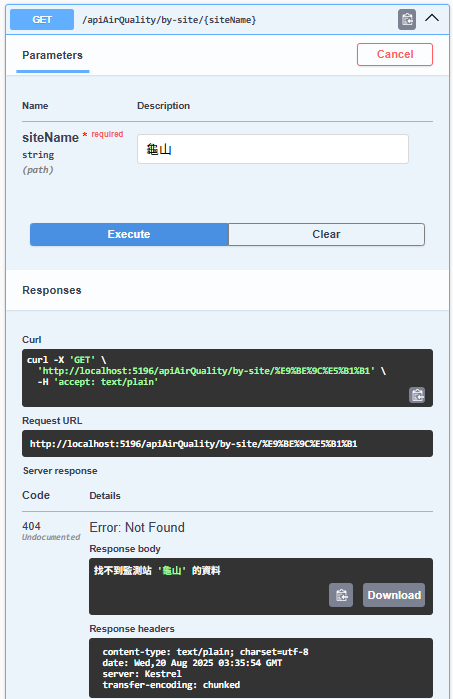
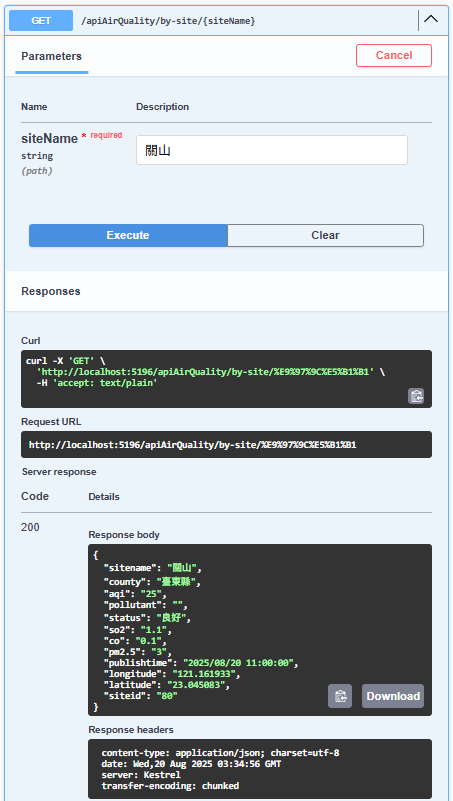
GetAirQualityByCounty



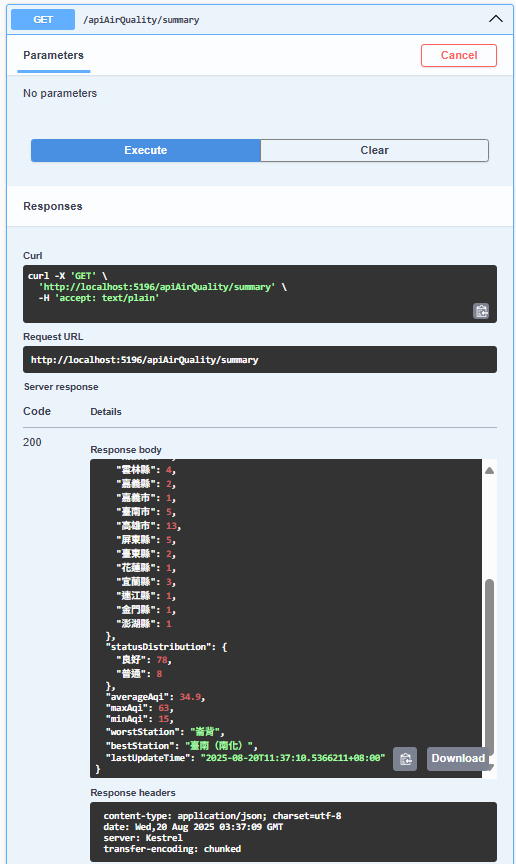
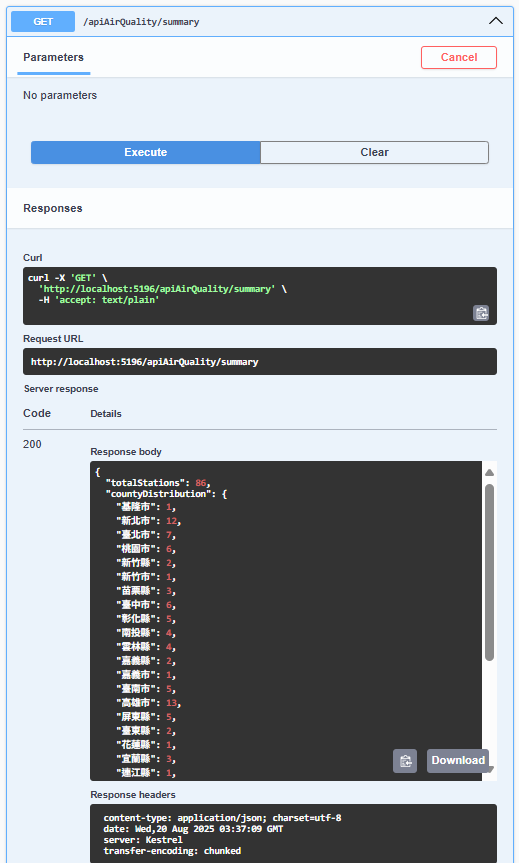
GetAirQualityByAqiRange



GetAirQualityBySite



GetAirQualitySummary



1. 程式碼(所有的程式碼均須貼上)：
   1. apiController (請標示Controller名稱)

➡️AirQualityController  
using HW\_2\_Practice.Models;

using HW\_2\_Practice.Services;

using Microsoft.AspNetCore.Mvc;

namespace AirQualityAPI.Controllers

{

[ApiController]

[Route("api[controller]")]

public class AirQualityController : ControllerBase

{

private readonly IAirQualityService \_airQualityService;

public AirQualityController(IAirQualityService airQualityService)

{

\_airQualityService = airQualityService;

}

[HttpGet("all")]

public async Task<ActionResult<IEnumerable<AirQualityData>>> GetAllAirQualityData()

{

try

{

var data = await \_airQualityService.GetAllAirQualityDataAsync();

return Ok(data);

}

catch (Exception ex)

{

return StatusCode(500, $"內部伺服器錯誤: {ex.Message}");

}

}

[HttpGet("by-county/{county}")]

public async Task<ActionResult<IEnumerable<AirQualityData>>> GetAirQualityByCounty(string county)

{

try

{

var data = await \_airQualityService.GetAirQualityByCountyAsync(county);

if (!data.Any())

{

return NotFound($"找不到縣市 '{county}' 的空氣品質資料");

}

return Ok(data);

}

catch (Exception ex)

{

return StatusCode(500, $"內部伺服器錯誤: {ex.Message}");

}

}

[HttpGet("by-aqi-range")]

public async Task<ActionResult<IEnumerable<AirQualityData>>> GetAirQualityByAqiRange(

int minAqi = 0,

int maxAqi = 300)

{

try

{

var data = await \_airQualityService.GetAirQualityByAqiRangeAsync(minAqi, maxAqi);

if (!data.Any())

{

return NotFound("找不到符合條件的空氣品質資料。");

}

return Ok(data);

}

catch (Exception ex)

{

return StatusCode(500, $"內部伺服器錯誤: {ex.Message}");

}

}

[HttpGet("by-site/{siteName}")]

public async Task<ActionResult<AirQualityData>> GetAirQualityBySite(string siteName)

{

try

{

var data = await \_airQualityService.GetAirQualityBySiteAsync(siteName);

if (data == null)

{

return NotFound($"找不到監測站 '{siteName}' 的資料");

}

return Ok(data);

}

catch (Exception ex)

{

return StatusCode(500, $"內部伺服器錯誤: {ex.Message}");

}

}

[HttpGet("summary")]

public async Task<ActionResult<AirQualitySummary>> GetAirQualitySummary()

{

try

{

var summary = await \_airQualityService.GetAirQualitySummaryAsync();

return Ok(summary);

}

catch (Exception ex)

{

return StatusCode(500, $"內部伺服器錯誤: {ex.Message}");

}

}

}

}

* 1. Data Model (請標示檔名)

➡️AirQualityData

using System.Text.Json.Serialization;

namespace HW\_2\_Practice.Models

{

public class AirQualityData

{

[JsonPropertyName("sitename")]

public string? SiteName { get; set; }

[JsonPropertyName("county")]

public string? County { get; set; }

[JsonPropertyName("aqi")]

public string? AqiString { get; set; }

[JsonIgnore]

public int Aqi => int.TryParse(AqiString, out var result) ? result : 0;

[JsonPropertyName("pollutant")]

public string? Pollutant { get; set; }

[JsonPropertyName("status")]

public string? Status { get; set; }

[JsonPropertyName("so2")]

public string? SO2 { get; set; }

[JsonPropertyName("co")]

public string? CO { get; set; }

[JsonPropertyName("pm2.5")]

public string? PM25 { get; set; }

[JsonPropertyName("publishtime")]

public string? PublishTime { get; set; }

[JsonPropertyName("longitude")]

public string? Longitude { get; set; }

[JsonPropertyName("latitude")]

public string? Latitude { get; set; }

[JsonPropertyName("siteid")]

public string? SiteId { get; set; }

}

}

➡️AirQualitySummary

namespace HW\_2\_Practice.Models

{

public class AirQualitySummary

{

public int TotalStations { get; set; }

public Dictionary<string, int> CountyDistribution { get; set; } = new();

public Dictionary<string, int> StatusDistribution { get; set; } = new();

public double AverageAqi { get; set; }

public int MaxAqi { get; set; }

public int MinAqi { get; set; }

public string? WorstStation { get; set; }

public string? BestStation { get; set; }

public DateTime LastUpdateTime { get; set; }

}

}

* 1. Service類別(請標示檔名)

➡️AirQualityService

using HW\_2\_Practice.Models;

using System.Text.Json;

using System.Text.Json.Serialization;

namespace HW\_2\_Practice.Services

{

public class AirQualityService : IAirQualityService

{

private readonly HttpClient \_httpClient;

private readonly ILogger<AirQualityService> \_logger;

private const string API\_URL = "https://data.moenv.gov.tw/api/v2/aqx\_p\_432?api\_key=9e565f9a-84dd-4e79-9097-d403cae1ea75&limit=1000&sort=ImportDate%20desc&format=JSON";

public AirQualityService(HttpClient httpClient, ILogger<AirQualityService> logger)

{

\_httpClient = httpClient;

\_logger = logger;

}

private async Task<List<AirQualityData>> FetchAirQualityDataAsync()

{

try

{

\_logger.LogInformation("正在從政府開放資料平台取得空氣品質資料...");

var response = await \_httpClient.GetAsync(API\_URL);

response.EnsureSuccessStatusCode();

var jsonContent = await response.Content.ReadAsStringAsync();

var apiResponse = JsonSerializer.Deserialize<ApiResponse>(jsonContent);

if (apiResponse?.Records == null)

{

\_logger.LogWarning("API 回應中沒有 records 資料");

return new List<AirQualityData>();

}

\_logger.LogInformation($"成功取得 {apiResponse.Records.Count} 筆空氣品質資料");

return apiResponse.Records;

}

catch (HttpRequestException ex)

{

\_logger.LogError(ex, "呼叫第三方 API 時發生網路錯誤");

throw new ApplicationException("無法連接到資料來源", ex);

}

catch (JsonException ex)

{

\_logger.LogError(ex, "解析 JSON 資料時發生錯誤");

throw new ApplicationException("資料格式錯誤", ex);

}

}

public async Task<IEnumerable<AirQualityData>> GetAllAirQualityDataAsync()

{

var data = await FetchAirQualityDataAsync();

return data.Where(x => !string.IsNullOrEmpty(x.SiteName));

}

public async Task<IEnumerable<AirQualityData>> GetAirQualityByCountyAsync(string county)

{

var data = await FetchAirQualityDataAsync();

return data.Where(x => !string.IsNullOrEmpty(x.County) &&

x.County.Contains(county, StringComparison.OrdinalIgnoreCase));

}

public async Task<IEnumerable<AirQualityData>> GetAirQualityByAqiRangeAsync(int minAqi, int maxAqi)

{

var data = await FetchAirQualityDataAsync();

return data.Where(x => x.Aqi >= minAqi && x.Aqi <= maxAqi && x.Aqi > 0)

.OrderByDescending(x => x.Aqi);

}

public async Task<AirQualityData?> GetAirQualityBySiteAsync(string siteName)

{

var data = await FetchAirQualityDataAsync();

return data.FirstOrDefault(x => !string.IsNullOrEmpty(x.SiteName) &&

x.SiteName.Contains(siteName, StringComparison.OrdinalIgnoreCase));

}

public async Task<AirQualitySummary> GetAirQualitySummaryAsync()

{

var data = await FetchAirQualityDataAsync();

var validData = data.Where(x => x.Aqi > 0).ToList();

var now = DateTime.Now;

if (!validData.Any())

{

return new AirQualitySummary { LastUpdateTime = now };

}

var maxAqi = validData.Max(x => x.Aqi);

var minAqi = validData.Min(x => x.Aqi);

var summary = new AirQualitySummary

{

TotalStations = validData.Count,

CountyDistribution = validData

.Where(x => !string.IsNullOrEmpty(x.County))

.GroupBy(x => x.County!)

.ToDictionary(g => g.Key, g => g.Count()),

StatusDistribution = validData

.Where(x => !string.IsNullOrEmpty(x.Status))

.GroupBy(x => x.Status!)

.ToDictionary(g => g.Key, g => g.Count()),

AverageAqi = Math.Round(validData.Average(x => x.Aqi), 2),

MaxAqi = maxAqi,

MinAqi = minAqi,

WorstStation = string.Join(", ", validData.Where(x => x.Aqi == maxAqi).Select(x => x.SiteName)),

BestStation = string.Join(", ", validData.Where(x => x.Aqi == minAqi).Select(x => x.SiteName)),

LastUpdateTime = now

};

return summary;

}

private class ApiResponse

{

[JsonPropertyName("records")]

public List<AirQualityData> Records { get; set; } = new();

}

}

}

* 1. 其他補充(若有才需貼，並請標示檔名)

※利用介面定義商業邏輯，方便擴充與測試

➡️IAirQualityService

using HW\_2\_Practice.Models;

namespace HW\_2\_Practice.Services

{

public interface IAirQualityService

{

Task<IEnumerable<AirQualityData>> GetAllAirQualityDataAsync();

Task<IEnumerable<AirQualityData>> GetAirQualityByCountyAsync(string county);

Task<IEnumerable<AirQualityData>> GetAirQualityByAqiRangeAsync(int minAqi, int maxAqi);

Task<AirQualityData?> GetAirQualityBySiteAsync(string siteName);

Task<AirQualitySummary> GetAirQualitySummaryAsync();

}

}