

圖源：[worldskills](https://www.worldskills.sg/skills/skill-areas/it-software-solutions-for-business)

商務軟體設計

IT Software Solutions for Business

目錄

[壹、 前言 1](#_Toc174878254)

[貳、 SQL Server (SSMS) 1](#_Toc174878255)

[一、 學習網站 1](#_Toc174878256)

[二、 程式碼 1](#_Toc174878257)

[1. 增刪查改 (CRUD) 1](#_Toc174878258)

[2. 創建資料表 3](#_Toc174878259)

[3. 創建檢視表 3](#_Toc174878260)

[4. 匯入大量資料 3](#_Toc174878261)

[三、 經驗 4](#_Toc174878262)

[參、 C# (Visual Studio) 5](#_Toc174878263)

[一、 推薦網站 5](#_Toc174878264)

[二、 教學影片 5](#_Toc174878265)

[三、 程式碼 5](#_Toc174878266)

[1. 串接API GET Method 取得字串 5](#_Toc174878267)

[2. 串接API POST Method Post json 字串 6](#_Toc174878268)

[3. 串接API GET Method 取得圖片 7](#_Toc174878269)

[4. Json 序列化 (C#物件轉Json字串) 7](#_Toc174878270)

[5. Json 反序列化 (Json字串轉C#物件) 8](#_Toc174878271)

[6. 自定義類別 (Class) 8](#_Toc174878272)

[7. 實體資料模型 9](#_Toc174878273)

[8. LINQ 9](#_Toc174878274)

[9. LINQ to SQL 11](#_Toc174878275)

[10. 匿名觀念 13](#_Toc174878276)

[11. Lambda表達式 14](#_Toc174878277)

[12. Func<>、Action、Action<> 15](#_Toc174878278)

[13. 擴充方法 15](#_Toc174878279)

[14. 顯示表格 DataTable DataGridView 16](#_Toc174878280)

[15. 篩選表格 BindingSource、DataGridView 18](#_Toc174878281)

[16. 多線程 Task 19](#_Toc174878282)

[17. 委派 Invoke、Delegate 19](#_Toc174878283)

[四、 經驗 20](#_Toc174878284)

[肆、 Android Studio (Java) 24](#_Toc174878285)

[一、 推薦網站 24](#_Toc174878286)

[二、 教學影片 24](#_Toc174878287)

[三、 程式碼 24](#_Toc174878288)

[1. 串接API GET Method 取得字串 24](#_Toc174878289)

[2. 串接API POST Method Post json 字串 27](#_Toc174878290)

[3. 串接API GET Method 取得圖片 29](#_Toc174878291)

[4. Json序列化 JSONObject to JsonString 29](#_Toc174878292)

[5. Json反序列化 JsonString to JSONObject 30](#_Toc174878293)

[6. try-catch Exception 30](#_Toc174878294)

[7. 多線程 31](#_Toc174878295)

[8. runOnUiThread 31](#_Toc174878296)

[9. Context 32](#_Toc174878297)

[10. Activity、Fragment、Context 32](#_Toc174878298)

[四、 經驗 33](#_Toc174878299)

[伍、 Web API (全國賽) 35](#_Toc174878300)

[一、 前言 35](#_Toc174878301)

[二、 程式碼 35](#_Toc174878302)

[1. 新增控制器 35](#_Toc174878303)

[2. 實體資料模型 35](#_Toc174878304)

[3. 使用Json格式作為回傳值 36](#_Toc174878305)

[4. GET Method 回傳資料 36](#_Toc174878306)

[5. GET Method 回傳圖片 37](#_Toc174878307)

[6. POST Method 新增資料 38](#_Toc174878308)

[7. Put Method 修改資料 40](#_Toc174878309)

[8. Delete Method 刪除資料 43](#_Toc174878310)

[9. Body 48](#_Toc174878311)

[10. 路由 48](#_Toc174878312)

[11. HTTP方法 49](#_Toc174878313)

[12. 發佈 Web API 49](#_Toc174878314)

[13. 部署到IIS 49](#_Toc174878315)

[三、 經驗 50](#_Toc174878316)

[陸、 總結 50](#_Toc174878317)

# 前言

成為選手後，最重要的是培養自己解決問題的能力，不論是察看這份文件、上網找答案、使用Ai工具…等等。不論是何種方法，都是為了學習拆分問題、尋找解答。

這份文件將會上傳到[GitHub儲存庫](https://github.com/wnilnay/IT-Software-Solutions-for-Business-Experience.git)，未來有更新文件也會在GitHub上進行更新。如果有問題嘗試了各種方法仍然無法解決，可到GitHub上進行討論(Discussions)。若文件有錯誤，也請在[GitHub儲存庫](https://github.com/wnilnay/IT-Software-Solutions-for-Business-Experience.git)提出，我會盡快改正。

比賽過程中無法使用任何第三方程式庫!比賽前會有測試環境時間，請好好利用

以下所有程式碼僅供參考，請勿直接複製貼上，親手撰寫過才會有印象。

以下內容如有侵權，請告知刪除。

# SQL Server (SSMS)

## 學習網站

1. [SQL語法教學-1Keydata](https://www.1keydata.com/tw/sql/sql.html)
2. [W3Schools-SQL](https://www.w3schools.com/sql/default.asp)
3. [Learn SQL | Codecademy](https://www.codecademy.com/learn/learn-sql)

## 程式碼

### 增刪查改 (CRUD)

**增加 (Create) 增加一筆資料**

INSERT INTO TableName (Column1, Column2, Column3, ...)

VALUES (Value1, Value2, Value3, ...);

**查詢 (Read) 查詢符合條件的資料**

SELECT Column1, Column2, ...

FROM TableName

WHERE Condition;

SELECT \*

FROM TableName

WHERE Condition;

**改正 (Update) 修改符合條件的資料**

UPDATE TableName

SET Column1 = Value1, Column2 = Value2, ...

WHERE Condition;

**刪除 (Delete) 刪除符合條件的資料**

DELETE FROM TableName

WHERE Condition;

Condition 指的是篩選條件，例如，查詢Name是John的資料

SELECT \*

FROM TableName

WHERE Name = 'John';

Condition中可以使用LIKE、萬用字元、AND、OR，組合出不同篩選條件。

具體可參考[SQL LIKE](https://www.1keydata.com/tw/sql/sqllike.html)、[SQL 萬用字元](https://www.1keydata.com/tw/sql/sql-wildcard.html)、[SQL AND OR](https://www.1keydata.com/tw/sql/sqlandor.html)。

### 創建資料表

資料庫中儲存資料的基本架構，實際儲存資料。

SSMS中常用工具創建資料表，不需撰寫SQL語法，不過在Android Studio中要創建資料表就必須使用SQL語法。

CREATE TABLE TableName (

Column1 DataType CONSTRAINTS,

Column2 DataType CONSTRAINTS,

Column3 DataType CONSTRAINTS,

...

);

CONSTRAINTS：為表格的某一個欄位增加限制，可有可無，常見的限制有NOT NULL、UNIQUE、CHECK、主鍵 (Primary Key)、外來鍵 (Foreign Key)。具體可參考[SQL CONSTRAINTS](https://www.1keydata.com/tw/sql/sql-constraint.html)。

### 創建檢視表

檢視表只儲存語法，它本身並不實際儲存資料。顯示的資料僅為語法執行結果

CREATE VIEW ViewName AS

SELECT Column1, Column2, ...

FROM TableName

WHERE Condition;

### 匯入大量資料

可直接從Excel中複製貼上，不過面對大量資料，效率極差。

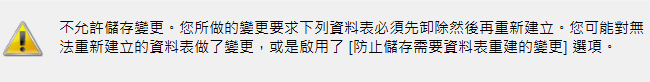
使用匯入匯出精靈，選擇Microsoft Excel 作為資料來源匯入，不過疑似需要額外安裝軟體，比賽中部一定會安裝。或是將資料複製至記事本，資料來源選擇Flat File Source。上述匯入方法皆有可能失敗，目前無較好的匯入資料方法。

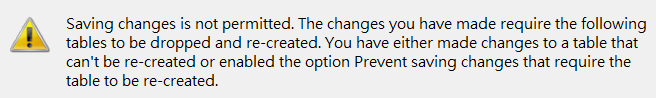
## 經驗

1. SQL 語法的**關鍵字(保留字)不區分大小寫**，但通常會全部大寫。

SELECT、FROM、WHERE、VIEW、AS…等等，都是保留字。SSMS中保留字會以藍色顯示，具體可查詢[保留字列表](https://en.wikipedia.org/wiki/List_of_SQL_reserved_words)。

1. 必須要知道什麼是[主鍵](https://www.1keydata.com/tw/sql/sql-primary-key.html)、[外來鍵](https://www.1keydata.com/tw/sql/sql-foreign-key.html)、[GROUP BY](https://www.1keydata.com/tw/sql/sqlgroupby.html)、[JOIN](https://www.1keydata.com/tw/sql/sqljoins.html)、[別名](https://www.1keydata.com/tw/sql/sqlalias.html)。要建立檢視表前先熟知這些知識，才不會在撰寫SQL語法時感到混亂。
2. 在使用設計工具修改資料表結構時，可能會出現





可依照 工具🡪選項🡪設計師🡪資料表和資料庫設計工具🡪防止儲存需要資 料表重建的變更🡪取消勾選

Tools🡪Options🡪Designers🡪Table and Database Designers🡪Prevent saving changes that require table re-creation🡪取消勾選

不過實務上不建議這樣做，只是為了比賽中方便快速更改資料表結構。

實務上需要使用[ALTER TABLE](https://www.1keydata.com/tw/sql/sql-alter-table.html) 變更資料表結構。

1. 在SSMS中，通常會使用工具創建資料表、建立資料庫關聯圖(創建外來鍵關係)，還可以創建檢視表，不過不常用，大多使用SQL語法[創建檢視表](#_創建檢視表)。

先前提及[創建資料表](#_創建資料表)語法與增加限制語法常用在Android Studio，目前並未提供SQL工具，故仍須了解其語法。

1. 測試時間可先將選項取消勾選，接著做其他測試。(三個科目共用測試時間)。

# C# (Visual Studio)

## 推薦網站

1. [.Net文件](https://learn.microsoft.com/zh-tw/dotnet/fundamentals/)
2. [C# 文件](https://learn.microsoft.com/zh-tw/dotnet/csharp/tour-of-csharp/)
3. [C# 語言參考](https://learn.microsoft.com/zh-tw/dotnet/csharp/language-reference/)
4. [C# 教程](https://www.runoob.com/csharp/csharp-tutorial.html)

## 教學影片

請參考原始文件

## 程式碼

由於種類繁多，只將重要的程式碼片段收錄其中。若有問題無法解決，可至[GitHub儲存庫](https://github.com/wnilnay/IT-Software-Solutions-for-Business-Experience.git)提出討論。

使用 .NET Framework架構，專案類型為Windows Forms App(.NET Framework)

### 串接API GET Method 取得字串

Ex:Uri = http://localhost:5000/api/UserVoice/GetRequestToken

HttpClient client = new HttpClient();

client.BaseAddress = new Uri("YourUri");

//client.BaseAddress = new Uri("http://localhost:5000");

HttpResponseMessage responseMessage = client.GetAsync("YourUrl").Result;

//HttpResponseMessage responseMessage = client.GetAsync("/api/UserVoice/GetRequestToken").Result;

if (responseMessage.IsSuccessStatusCode == true)

{

string result = responseMessage.Content.

ReadAsStringAsync().Result;

}

### 串接API POST Method Post json 字串

Ex:Uri = http://localhost:5000/api/UserVoice/ApplyUserVoice

HttpClient client = new HttpClient();

client.BaseAddress = new Uri("YourUri");

// client.BaseAddress = new Uri("http://localhost:5000");

StringContent content = new StringContent(post\_string, Encoding.UTF8, "application/json");

HttpResponseMessage responseMessage = client.PostAsync("YourUri", content).Result;

//HttpResponseMessage responseMessage = client.PostAsync("/api/UserVoice/ApplyUserVoice", content).Result;

if(responseMessage.IsSuccessStatusCode)

{

string result = responseMessage.Content.ReadAsStringAsync().Result;//如果有回傳值

}

### 串接API GET Method 取得圖片

與[取得字串](#_串接API_GET_Method)相似

//接續先前內容

if (httpResponseMessage.IsSuccessStatusCode == true)

{

// 讀取回應內容作為 byte array

byte[] imageBytes = httpResponseMessage.Content.ReadAsByteArrayAsync().Result;

// 使用 MemoryStream 將 byte array 轉換為 Bitmap

using (MemoryStream ms = new MemoryStream(imageBytes))

{

Bitmap bitmap = new Bitmap(ms);

}

}

### Json 序列化 (C#物件轉Json字串)

必須先將System.Text.Json 與 System.Memory加入參考，加入的參考並不是第三方程式庫，故比賽中也可使用。

加入完參考後，必須在程式碼最前面using區塊，增加 using System.Text.Json;

//將 C# 物件序列化為 JSON 字串

string serializedJson = JsonSerializer.Serialize(YourObject);

或是

string serializedJson = JsonSerializer.Serialize(YourObject, new JsonSerializerOptions { WriteIndented = true }); //將字串增加縮排與換行，增加可讀性

更多資訊請參考[JSON序列化與反序列化](https://learn.microsoft.com/zh-tw/dotnet/standard/serialization/system-text-json/overview)

### Json 反序列化 (Json字串轉C#物件)

//反序列化 JSON 字串為 C# 物件

YourClass obj = JsonSerializer.Deserialize<YourClass>(jsonString);

同[Json序列化](#_Json_序列化_(C#物件轉Json字串))，必須先加入參考。更多資訊請參考[JSON序列化與反序列化](https://learn.microsoft.com/zh-tw/dotnet/standard/serialization/system-text-json/overview)

### 自定義類別 (Class)

在Json序列與反序列化中，會需要自定義類別，對應Json的格式。例如現在有個Json字串

{

"Name" : "wnilnay",

"Gender" : "Man",

"Age" : 18

}

在C#中就要自定義一個類別

public class Person

{

public string Name { get; set; }

public string Gender { get; set; }

public int Age { get; set; }

}

屬性名稱必須一樣，才能與之相互對應。若需要C#屬性名與Json屬性名不同時，例如 "Body Mass Index" : 17.5，因為屬性中包含空格，無法與C#作對應，則自定義類別中可寫成：

[JsonPropertyName("Body Mass Index")]

public float Body\_Mass\_Index { get; set; }

### 實體資料模型

依照資料庫生成相應實體資料模型，可使用資料模型快速為資料表新增與移除資料。

根據資料庫關聯圖，可生成包括外部表格之資料表。類似使用[表格連接](https://www.1keydata.com/tw/sql/sqljoins.html)，連接兩個資料表。

### LINQ

請參考[LINQ概觀](https://learn.microsoft.com/zh-tw/dotnet/csharp/linq/)，裡面有更詳細的介紹。LINQ內容非常多，無法全部詳細介紹，若有疑問，請參考官方文件。

1. 如果資料屬於實作 IEnumerable<T> 的型別或介面，則可以使用 LINQ to Objects 來查詢資料。包括但不限於Array、List、DbSet…等。LINQ相比於傳統資料查詢有著更高的可讀性與便利性，並且支援IntelliSense。
2. 具體操作程式碼可參考[LINQ概觀](https://learn.microsoft.com/zh-tw/dotnet/csharp/linq/)。.
3. LINQ查詢可分為「查詢運算式」及「方法查詢」兩種，兩種查詢彼此等價且可互相轉換。具體可參考[LINQ 查詢](https://learn.microsoft.com/zh-tw/dotnet/csharp/linq/get-started/write-linq-queries)。
4. 下列範例示範簡單「查詢運算式」以及撰寫為「方法查詢」的語意對等查詢。範例來源：[LINQ 查詢](https://learn.microsoft.com/zh-tw/dotnet/csharp/linq/get-started/write-linq-queries)。
5. 方法查詢，方法內傳入 [Lambda](#_Lambda表達式) 運算式。

int[] numbers = [ 5, 10, 8, 3, 6, 12 ];

//Query syntax:

//可使用var numQuery1 =

IEnumerable<int> numQuery1 =

from num in numbers

where num % 2 == 0

orderby num

select num;

//Method syntax:

//可使用 var numQuery2 =

IEnumerable<int> numQuery2 = numbers.Where(num => num % 2 == 0).OrderBy(n => n);

foreach (int i in numQuery1)

{

Console.Write(i + " ");

}

Console.WriteLine(System.Environment.NewLine);

foreach (int i in numQuery2)

{

Console.Write(i + " ");

}

### LINQ to SQL

1. 創建實體資料模型 YourEntities，YourEntities內包含許多資料表，資料模型內的資料表屬於DbSet<T>類別，故可用兩種LINQ語法，也支援其[擴充方法](#_擴充方法)。
2. 程式碼

using (YourEntities db = new YourEntities())

{

var table1 = (from a in db.YourTable

select a);

foreach(var a in table1)

{

//Do something…

}

}

LINQ搭配資料庫關聯圖所生成的實體資料模型，可大幅增加篩選的便利性、增加篩選的速率。

例如現在有個實體資料模型 MyEntities，包括以下資料表，關聯圖也如下圖所示



若我現在想篩選某個帳號所屬的所有旅行社名稱，可使用

using (MyEntities db = new MyEntities ())

{

var travelAgencyUserData = (from a in db.AccountData

where a.FirstName == "Maurice"

select a.TravelAgencyUserData).FirstOrDefault();

foreach (var userData in travelAgencyUserData)

{

Console.WriteLine(userData.TravelAgencyData.Name);

}

}

即可篩選出FirstNmae為 Maurice 的第一筆資料，其所屬旅行社名稱。

若要查看篩選結果的資料結構，可使用LINQPad查看。

### 匿名觀念

C#中有幾種類型的資料結構或方法可以匿名使用，使用它們而無需明確地為它們命名或定義類型。以下是幾個主要的可以匿名的項目

1. 匿名方法 (Anonymous Methods)，多用Lambda表達式

Func<int, int, int> add = delegate(int x, int y)

{

return x + y;

};

1. Lambda表達式 (Lambda Expressions)，Lambda 表達式提供更精簡且更具表達性的方式來建立匿名函式。 使用 [**=> 運算子**](https://learn.microsoft.com/zh-tw/dotnet/csharp/language-reference/operators/lambda-operator)來建構 Lambda 運算式：

Func<int, int, int> add = (x, y) => x + y;//Lambda運算式

Func<int, int, int> add = (x, y) =>

{

return x + y;

};//Lambda陳述式

1. 匿名物件 (Anonymous Types)，匿名物件允許你在不定義類型的情況下臨時創建物件。

var person = new { Name = "John", Age = 30 };

1. 匿名陣列 (Anonymous Arrays)，匿名陣列允許你創建一個無名稱的陣列，陣列中的元素可以是不同類型。

var array = new object[] { "text", 42, true };

1. 動態物件 (Dynamic Objects)，使用 dynamic 關鍵字，你可以創建一個類型在編譯時未定義的物件。(不常用)
2. 隱含類型區域變數 (Implicitly Typed Local Variables)

使用 var 關鍵字，C# 允許你在編譯時由編譯器自動推斷變量的類型，而不是顯式聲明類型。

var number = 42; // 編譯器自動推斷為 int

var text = "Hello"; // 編譯器自動推斷為 string

1. 匿名委派 (Anonymous Delegates)

你可以在使用匿名方法或 Lambda 表達式時創建匿名委派。這些委派可以在不定義具名委派的情況下直接使用。

Action greet = delegate { Console.WriteLine("Hello, World!"); };

Action greet = () => { Console.WriteLine("Hello, World!"); };

greet();

//greet.Invoke();

### Lambda表達式

分為Lambda運算式、Lambda陳述式

使用 「 [**=> 運算子**](https://learn.microsoft.com/zh-tw/dotnet/csharp/language-reference/operators/lambda-operator)」，建立Lambda表達式，可使用[Func、Action](#_Func<>、Action、Action<>)封裝

(input-parameters) => expression //Lambda運算式

() => ();//Lambda運算式

() => {};//Lambda陳述式

如果 Lambda 運算式只有一個輸入參數，括弧是選擇性的：

### Func<>、Action、Action<>

皆是用於封裝方法，常用於[Lambda表達式](#_Lambda表達式)封裝，當然也可封裝具名函式與具名方法。

Func<>支援最多16個參數，一個回傳值。

Action支援0個參數0個回傳值。Action<>支援最多16個參數，沒有回傳值。

### 擴充方法

擴充方法可讓您在現有類型中「加入」方法，而不需要建立新的衍生類型、重新編譯，或是修改原始類型。

通常要為類別增加方法，通常會使用繼承類別的方式擴充方法。但若不想繼承，或是類別本身無法被繼承，例如string，就可以使用擴充方法。在靜態類別內建立靜態方法，參數內在要擴充類別前增加「this」。

最常見的擴充方法是 LINQ 標準查詢運算子。

使用第54屆分區賽歷屆試題，密碼雜湊功能作為範例程式

public static class ProcessSHA256

{

public static string ExtensionProcess(this string value)//擴充方法

{

return \_Process(value);

}

public static string StaticProcess(string value)//靜態方法

{

return \_Process(value);

}

}

在調用方法上，擴充方法有較高的可讀性。

str = str.ExtensionProcess();//擴充方法

str = ProcessSHA256.StaticProcess(str);//靜態方法

### 顯示表格 DataTable DataGridView

DataGridView，常用來顯示表格資料，我的做法是建立DataTable，操作DataTable，將DataGridView資料來源設為DataTable。

我將以54屆分區賽歷屆試題，帳號篩選功能需顯示的表格作為示範。

1. 在Form應用程式中建立DataGridView物件，為他增加所需的標題，修改控制項的屬性。我的作法：

加入、編輯、刪除皆不啟用，資料行唯獨，類型依需求改變

AllowDrop、AllowUser...、AllowUser…，全部False

AutoSizeColumnsMode = Fill， AutoSizeRowsMode = AllCells

ColumnHeadersDefultCellStyle、DefultCellStyle、RowHeadersDefultCellStyle

全部置中(MiddleCenter)，RowHeadersVisible = False

1. 關鍵程式碼

DataTable dataTable = new DataTable();

dataTable.Columns.Clear();dataTable.Rows.Clear();

//增加DataTable標題欄，為DataTable標題欄與DataGridView標題欄作連接

for(int i = 0; i < dataGridView1.Columns.Count; i++)

{

dataTable.Columns.Add(dataGridView1.Columns[i].HeaderText);

dataGridView1.Columns[i].DataPropertyName = dataTable.Columns[i].ColumnName;

}

//隨機增加10筆測資

for (int i = 0; i < 10; i++)

{

//new object[]為[匿名](#_匿名觀念(進階))陣列，可方便增加資料行

dataTable.Rows.Add(new object[]

{

name,

accountStatus,

updateTime,

orderCount,

ticketCount,

effiectedCount,

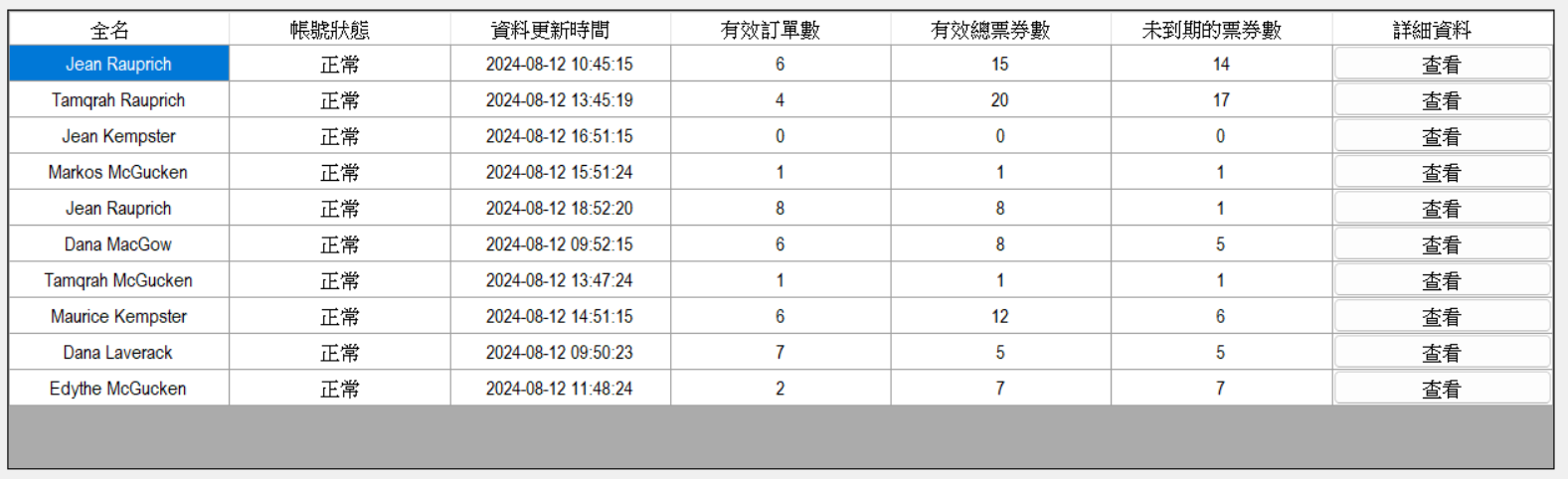
"查看"

});

}

//將DataGridView資料來源設定成DataTable

dataGridView1.DataSource = dataTable;

1. 最後效果

### 篩選表格 BindingSource、DataGridView

此節延續[顯示表格](#_顯示表格_DataTable_DataGridView)，為表格進行篩選。

BindingSource bindingSource = new BindingSource();

dataGridView1.DataSource = bindingSource.DataSource = dataTable;

//篩選與SQL篩選極為相似

//使用 like 進行模糊篩選，通常搭配 「%」或「\_」萬用字元

bindingSource.Filter = "全名 like '%" + name + "%'";

//也可使用「=」

bindingSource.Filter = "有效總票券數 = '" + textBox2.Text + "'";

bindingSource.Filter = "有效總票券數 like '%" + textBox2.Text + "%'"

//使用 AND 或是 OR 連接篩選條件

bindingSource.Filter = "全名 like '%" + name + "%' AND 有效總票券數 = '" + textBox2.Text + "'";

### 多線程 Task

載入表格資料大多需要進行許多操作與計算，大多為耗時任務，故運行在主線程會導致應用程式當機，因此需要多線程讓耗時任務運行在其他線程中。

要達成多線程有非常多種方法，這裡介紹我比較常用的方法「Task」。

Task.Run(() =>

{

//Do something on other thread

});

//若需要對任務做些操作，可使用  
Task task = Task.Run(() =>

{

//Do something on other thread

});

Task.Run()方法內傳入一個[Action](#_Func<>、Action、Action<>)或是[Func](#_Func<>、Action、Action<>)，也可用[Lambda表達式](#_Lambda表達式)，具體可參考[Task.Run方法](https://learn.microsoft.com/zh-tw/dotnet/api/system.threading.tasks.task.run?view=net-8.0#system-threading-tasks-task-run(system-action))。

多線程也常與Async、Await一起使用，不過我不太熟悉，若有需要，可參考

[使用 async 和 await 進行非同步程式設計](https://learn.microsoft.com/zh-tw/dotnet/csharp/asynchronous-programming/)

### 委派 Invoke、Delegate

我對Delegate不是很熟悉，本節只針對多線程中使用Invoke回到UI線程。

在多線程中變更UI線程，會觸發System.InvalidOperationException: 跨執行緒作業無效。這時會需要使用Invoke回到UI線程操作。

控制項.Invoke(new Action(() =>

{

//Do something on UI thread

}));

Invoke()需要一個Delegate類別作為參數傳入，我這裡使用[Action](#_Func<>、Action、Action<>)()搭配[Lambda](#_Lambda表達式) 創建一個匿名函式，當作參數傳入。除了使用[Action](#_Func<>、Action、Action<>)，也可使用[Func](#_Func<>、Action、Action<>)或是delegate等函式。

具體程式碼可能略有不同，請依照情況進行修改。

Func<> func = 具名函式;

Func<> func = () => {};

Func<> func = () => ();

Func<> func = delegate {};

Action func = 具名函式;

Action func = () => {};

Action func = () => ();

Action func = delegate {};

控制項.Invoke(func);

## 經驗

以上程式碼，關於委派、Delegate、匿名委派、Func、Action相關部分不是很熟悉，若有不清楚或錯誤的部分，請見諒。

若未來有更深的理解，將會在[GitHub儲存庫](https://github.com/wnilnay/IT-Software-Solutions-for-Business-Experience.git)上更新，歡迎指教與討論。

上述內容有些部分較偏重理論，建議以實作為主，先試著自己動手做做看，若有餘力，再將理論部分補足即可。先以歷屆試題開始做起，有不理解的部分，再找資料解決。

關於Language Integrated Query，縮寫：LINQ 音"link"，LINQ to SQL可使用[LINQPad](https://www.linqpad.net/)查看結構與轉換後的SQL語句，這在學習過程中非常有用。

上述內容對於類別與物件提及較少，我認為這是最基本要會的，且在教學影片中，對於 C# 物件導向有著清楚的介紹，故在此不再贅述。

補充關於Task.Run()、Invoke、表格操作

在DataTable設定成DataGridView的DataSource後，為DataTable增加資料，通常也會自動在DataGridView上進行更新。故可在一開始就做初始設定。

//可直接在多線程增加資料，幾乎都可以在DataGridView上正常更新

Task.Run(() =>

{

dataTable.Rows.Add(new object[]

{

name,

accountStatus,

updateTime,

orderCount,

ticketCount,

effiectedCount,

"查看"

});

});

//若無法正常更新，可將增加資料的操作移到UI線程，應該可以解決問題

Task.Run(() =>

{

dataGridView1.Invoke(new Action(() =>

{

dataTable.Rows.Add(new object[]

{

name,

accountStatus,

updateTime,

orderCount,

ticketCount,

effiectedCount,

"查看"

});

}));

});

Visual Studio 環境設定，參考至[伊果的沒人看筆記本](https://igouist.github.io/post/2020/03/visualstudio/)

1. 字體推薦使用Consolas
2. 擴充套件
3. [CodeMaintainability 2022](https://marketplace.visualstudio.com/items?itemName=ognjen-babic.CodeMaintainability2022)

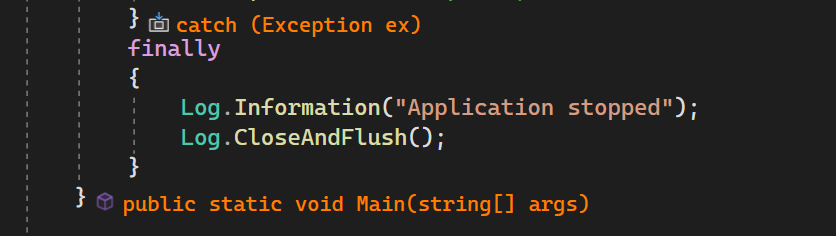
查看程式碼複雜度

1. [Editor Guidelines](https://marketplace.visualstudio.com/items?itemName=PaulHarrington.EditorGuidelinesPreview)

在特定字元數位置畫輔助線，避免程式碼過長，若長度過長，請為程式碼換行。

1. [CodeBlockEndTag](https://marketplace.visualstudio.com/items?itemName=KhaosPrinz.CodeBlockEndTag)

括弧的結束部分會顯示出這個括弧所屬的區塊，如下圖所示



圖片來源：[CodeBlockEndTag](https://marketplace.visualstudio.com/items?itemName=KhaosPrinz.CodeBlockEndTag)

1. Visual Studio 設定

開啟內嵌提示，可增加程式碼的可讀性。例如呼叫方法時顯示參數名稱，詳情可參考[伊果的沒人看筆記本](https://igouist.github.io/post/2020/03/visualstudio/)

在開發期間，常常會將變數輸出至Console查看，但Visual Studio輸出視窗還有其他系統訊息，常常找不到自己輸出的數值。這時可以點擊專案的Properties，修改輸出類型為「主控台應用程式」，這時執行程式將會多出一個Console視窗，方便調適。

在開發完成後，或是比賽最後需要繳交檔案時，請將輸出類型修改回原本的類型。

測試時間可測試API串接、加入參考是否可正常運行

# Android Studio (Java)

## 推薦網站

1. [開發人員指南 | Android Developers](https://developer.android.com/guide?hl=zh-tw)
2. [Android API reference | Android Developers](https://developer.android.com/reference)
3. [Android SDK | Android Developers](https://stuff.mit.edu/afs/sipb/project/android/docs/reference/android/package-summary.html)
4. [Android 教學](https://chris930921.gitbooks.io/android-traveling-gitbook/content/index.html)

## 教學影片

請參考原始文件

## 程式碼

以下程式碼皆為「Java」語言，不過Google漸漸轉用「Kotlin」進行Android開發。但應該不會相差太多。

專案類型為：Empty Views Activity

### 串接API GET Method 取得字串

要串接API之前，APP必須取得網際網路權限

<!--AndroidManifest.xml-->

<manifest>

<uses-permission android:name="android.permission.INTERNET"/>

</manifest>

Android預設不支持「http://」，只支援「https://」，所以必須增加

<!--AndroidManifest.xml-->

<application

android:usesCleartextTraffic="true"

...>

</application>

使用HttpURLConnection串接API

URL url = new URL(YourUrl);

HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();

//設定方法"GET"

httpURLConnection.setRequestMethod("GET");

//要接收回傳值

httpURLConnection.setDoInput(true);

//設定Timeout為5000ms

httpURLConnection.setConnectTimeout(5000);

//開始連接

httpURLConnection.connect();

InputStream inputStream = httpURLConnection.getInputStream();

//InputStreamReader inputStreamReader = new InputStreamReader(inputStream, Charset.forName("UTF-8"));

InputStreamReader inputStreamReader = new InputStreamReader(inputStream);

BufferedReader bufferedReader = new BufferedReader(inputStreamReader);

//BufferedReader bufferedReader1 = new BufferedReader(new InputStreamReader(httpURLConnection.getInputStream()));

StringBuilder stringBuilder = new StringBuilder();

//斷開連接

httpURLConnection.disconnect();

if(httpURLConnection.getResponseCode() == HttpURLConnection.HTTP\_OK){

String line;

while ((line = bufferedReader.readLine()) != null){

stringBuilder.append(line).append("\n");

}

}

1. 取得InputStream後，可直接使用InputStream的read()方法搭配迴圈取得字元，連接成字串。具體可參考[InputStream](https://developer.android.com/reference/java/io/InputStream)
2. 為了讀取方便，最終可以使用BufferedReader讀取一整行，但new BufferedReader()只接收InputStreamReader，所以要將InputStream轉換成InputStreamReader。
3. InputStreamReader與InputStream類似。
4. 使用StringBuilder連接字串。
5. 可使用stringBuilder.toString();取得連接後的字串。

### 串接API POST Method Post json 字串

與GET方法一樣，必須在AndroidManifest.xml進行設定(同個專案只要設定一次)

使用HttpURLConnection串接API

URL url = new URL(YourUri);

HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();

httpURLConnection.setConnectTimeout(5000);

//設定可輸入

httpURLConnection.setDoInput(true);

//設定可輸出

httpURLConnection.setDoOutput(true);

//設定請求方法POST

httpURLConnection.setRequestMethod("POST");

//設定請求屬性

httpURLConnection.setRequestProperty("Content-Type","application/json; Charset=UTF-8");

//取得OutputStream

OutputStream outputStream = httpURLConnection.getOutputStream();

OutputStreamWriter outputStreamWriter = new OutputStreamWriter(outputStream, Charset.forName("UTF-8"));

BufferedWriter bufferedWriter = new BufferedWriter(outputStreamWriter);

//BufferedWriter bufferedWriter1 = new BufferedWriter(new OutputStreamWriter(httpURLConnection.getOutputStream()));

//寫入字串至緩衝區，以便稍後輸出

bufferedWriter.write(jsonString);

//將數據寫入目的地

bufferedWriter.flush();

//關閉BufferedWriter

bufferedWriter.close();

if(httpURLConnection.getResponseCode() == HttpURLConnection.HTTP\_OK){

//若有回傳值

InputStream inputStream = httpURLConnection.getInputStream();

InputStreamReader inputStreamReader = new InputStreamReader(inputStream, Charset.forName("UTF-8"));

BufferedReader bufferedReader = new BufferedReader(inputStreamReader);

//BufferedReader bufferedReader1 = new BufferedReader(new InputStreamReader(httpURLConnection.getInputStream()));

StringBuilder stringBuilder = new StringBuilder();

String line;

while ((line = bufferedReader.readLine()) != null){

stringBuilder.append(line).append("\n");

}

}

httpURLConnection.disconnect();

1. 取得OutputStream後，也可以直接使用outputStream.write()，不過只能寫入字元。
2. 為了方便寫入資料，使用BufferedWriter寫入資料，但BufferedWriter()只接收OutputStreamWriter作為參數，則需再將OutputStream轉為OutputStreamWriter。
3. OutputStream與OutputStreamWriter相似。
4. 若POST方法沒有回傳值，則不需撰寫接收回傳值的部分。
5. 使用StringBuilder連接字串。
6. 可使用stringBuilder.toString();取得連接後的字串。

### 串接API GET Method 取得圖片

與[取得字串](#_串接API_GET_Method_1)相似

if(httpURLConnection.getResponseCode() == HttpURLConnection.HTTP\_OK){

InputStream inputStream = httpURLConnection.getInputStream();

Bitmap bitmap = BitmapFactory.decodeStream(inputStream);

}

### Json序列化 JSONObject to JsonString

Java中並沒有原生的方法將Java物件轉成Json字串。

不過Java中內建JSONObject，可達到類似的效果。

JSONObject jsonObject = new JSONObject();

jsonObject.put(key/name,value);

可多次put，創建許多鍵值對。

若要取得Json字串，可使用

jsonObject.toString();

取得序列化後的Json字串。

### Json反序列化 JsonString to JSONObject

Java中並沒有原生的方法將Json字串轉成Java物件。

不過Java中內建JSONObject，可達到類似的效果。

JSONObject jsonObject = new JSONObject(jsonString);

jsonObject.get(key/name);

jsonObject.getInt(key/name);

jsonObject.getString(key/name);

jsonObject擁有許多get方法，可依照需求取得所需的資料。

### try-catch Exception

在串接API時，避免API出現問題，造成程式不穩定，Android Studio要求處理例外情況，否則編譯器會報錯。例如，HttpURLConnection要求處理IOException，JSONObject要求處理JSONException

catch中可一次處理多個例外，每個例外使用「|」隔開。

catch (IOException | JSONException exception){

//Do something

}

### 多線程

在執行網路任務時，例如API串接，避免等待響應造成UI線程阻塞，網路任務必須執行在其他線程中，這並不是可選項，而是必要的，否則APP會拋出例外並且閃退。

在Android中，有許多方法可以達成多線程，這裡介紹較為簡單的方法「Thread」

new Thread(new Runnable() {

@Override

public void run() {

//Do something on other thread

}

}).start();

### runOnUiThread

顧名思義，將下列程式碼，執行在UI線程中。在Android Studio中，要更新UI畫面，也必須在UI線程中執行。

new Thread(new Runnable() {

@Override

public void run() {

//Do something on other thread

runOnUiThread(new Runnable() {

@Override

public void run() {

//Do something on UI thread

}

});

}

}).start();

Java中，Timer內的任務也屬於多線程。

而C#中的Timer\_Tick事件，依然屬於主線程。

### Context

Android Studio中的Context通常指的是當前的Activity，可使用「this」，指向當前的Activity物件。

因為Fragment是依附在Activity中的物件，因此在Fragment，可使用getContext()、getActivity() 取得父層的Activity。

### Activity、Fragment、Context

以下內容不會提到有關於頁面跳轉與資料傳遞的內容，若有需要，請參考教學影片。

以下部分內容為我個人的理解與猜測，並不能保證絕對正確。

某些顯示元件，例如Toast、AlertDialog…等等，都會需要傳入Context參數，這代表他們與Fragment一樣，都需要基於某個Activity才能工作，所以使的Context大多是Activity。

在Fragment中，若要使用Toast、AlertDialog等元件，則Context參數必須傳入其父層Activity，可由getContext()或是getActivity()取得。

而在Fragment的多線程，要回到UI線程時，也必須使用其父層Activity.runOnUiThread()方法。

## 經驗

在Android Studio中，有許多的顯示元件，這些元件的用法必須熟知。還有一些常用元件，例如 Spinner(下拉式選單)、ListView(清單)、RecyclerView(客製化的ListView)，如有需要，請上網查詢使用方法。

Android Studio 建議先將重要的教學影片看完，接著直接開始練習歷屆試題，Android理論部分我覺得較少，大部分都是基於前人的成果進行修改。我認為這部分最重要的是了解有哪些工具可以使用，使用方法又是如何。

在Android Studio 操作SQLite資料庫時，有時需撰寫SQL語法，可在撰寫字串時(雙引號中間)，按下鍵盤「Alt + Enter」，選擇「Inject Language or reference」，搜尋SQL，選擇「SQLITE-SQL」，即可擁有SQL Intellisense與標示關鍵字的功能。

(此選項為新版擁有的選項，比賽現場與選手室電腦不一定會有，在測試環境時可先測試)

若無「SQLITE-SQL」選項，也可選擇「RoomSql」，雖無Intellisense功能，但仍然會標示關鍵字。

若需要撰寫其他語言，例如Json，也可以使用該方法，看有沒有支援。

Java中，物件屬性的getter、setter往往使用「get屬性名」、「set屬性名」方法。在初次使用物件時，可使用先撰寫「物件.get」，來查看此物件有什麼屬性。

Android 建議使用實機進行開發，避免效能瓶頸，造成開發速度緩慢。

API的部分，使用虛擬機，若是localhost網址，則需將localhost改成10.0.2.2。

實機連接loaclhost API

若要連接自己建立的API，可將API發布後，部署到IIS，實機與電腦連接到同一個網際網路。開啟防火牆後，即可使用http(https)://電腦IP:連接埠/路徑，至於使用http或是https，取決於一開始發布在IIS使用的類型。

要取得電腦IP，可在cmd中撰寫指令「ipconfig」，查看IPv4地址。

使用上述方法時，必須增加防火牆輸入規則，允許特定的連接埠輸入規則。

輸入規則🡪新增規則🡪連接🡪TCP🡪建議使用特定本機連接埠🡪輸入IIS中設定API的連接埠🡪允許連接🡪輸入可辨識的名稱🡪完成。

歷屆試題中的API，若仍然需要使用實機進行連接，可使用「[ngrok](https://ngrok.com/)」，讓外部的網址指向api的localhost地址，即可使用實機連結到API，此方法建議只在開發時期使用。

若是自己建立的API，也可以發布後，部署到IIS，使用[ngrok](https://ngrok.com/)讓外網連接。

使用[ngrok](https://ngrok.com/)指向localhost地址，可以不需要連接到同一個網際網路，只要能夠連上網際網路，即可連接到API。

比賽測試時間可先測試虛擬機是否能夠正常開啟，Android Studio撰寫頁面設計的部分有沒有Intellisense，應該會有，也應該要有。

# Web API (全國賽)

此章節講解內容為全國賽考科。

以下內容使用專案類型為：ASP.NET Web 應用程式 (.NET Framework)

使用架構為：.NET Framework 4.8。或許未來會更換架構為.NET Core

若需更多詳細資料，可參考[Web API](https://learn.microsoft.com/zh-tw/aspnet/web-api/)

## 前言

API常用的方法與資料庫相同。增刪查改 (CRUD)。

對應到API則為

**查詢 (Read) 🡪 GET Method**

**增加 (Create) 🡪 Post Method**

**改正 (Update) 🡪 Put Method**

**刪除 (Delete) 🡪 Delete Method**

**API操作常常與資料庫有關，但API操作並不只用在資料庫上。**

## 程式碼

### 新增控制器

Controllers資料夾上按右鍵🡪加入🡪控制器🡪Web API🡪具有讀取/寫入動作的Web API 2 控制器。

### 實體資料模型

創建實體資料模型，方便後續操作。創建實體資料模型時，驗證方式建議使用SQL Serve驗證。使用Windows驗證，部署到IIS會有權限不足的問題。

SQL Server驗證中也需選擇「在連接字串中包含敏感性資料」

### 使用Json格式作為回傳值

//Global.asax

//新增以下程式碼，清除Xml格式。

GlobalConfiguration.Configuration.Formatters.XmlFormatter.SupportedMediaTypes.Clear();

### GET Method 回傳資料

//原始方法

// GET api/values

public IEnumerable<string> Get()

{

return new string[] { "value1", "value2" };

}

// GET api/values/5

public string Get(int id)

{

return "value";

}

可與實體資料模型做搭配，假設我要回傳所有帳號的姓名組合

// GET: api/DataBase

public IEnumerable<string> Get()

{

using(For\_TestEntities db = new For\_TestEntities())

{

var name = (from a in db.AccountData

select a.FirstName + " " + a.LastName).ToList();

return name;

}

}

### GET Method 回傳圖片

// GET api/values

public HttpResponseMessage Get()

{

Bitmap bitmap = ImageResource.it\_software\_solutions\_for\_business;

HttpResponseMessage response = new HttpResponseMessage();

using(MemoryStream memoryStream = new MemoryStream())

{

bitmap.Save(memoryStream, ImageFormat.Png);

response.Content = new ByteArrayContent(memoryStream.ToArray());

response.Content.Headers.ContentType = new MediaTypeHeaderValue("image/png");

return response;

}

}

若需要串接API取得圖片，可依照

[C#串接API取得圖片](#_串接API_GET_Method_2)

[Android Studio串接API取得圖片](#_串接API_GET_Method_3)

### POST Method 新增資料

//原始方法

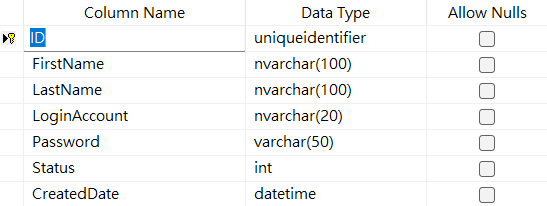
// POST api/values

public void Post([FromBody] string value)

{

}

FromBody，強制 Web API 從要求本文讀取類型，Web API會從文本讀取字串，並且存入value變數。

以下表為例，若需要使用API為AccountData資料表增加資料，則：

public class PostClass

{

public string FirstName { get; set; }

public string LastName { get; set; }

public string LoginAccount { get; set; }

public string Password { get; set; }

}

我設定使用者只需傳入以上四個屬性。

// POST: api/DataBase

//若無回傳值，使用void

public AccountData Post([FromBody] PostClass postClass)

{

using(For\_TestEntities db = new For\_TestEntities())

{

AccountData accountData = new AccountData();

Guid id = Guid.NewGuid();

accountData.ID = id;

accountData.FirstName = postClass.FirstName;

accountData.LastName = postClass.LastName;

accountData.LoginAccount = postClass.LoginAccount;

accountData.Password = postClass.Password;

accountData.Status = 1;

accountData.CreatedDate = DateTime.Now;

db.AccountData.Add(accountData);

db.SaveChanges();

//回傳

var account = (from a in db.AccountData

where a.ID == id

select a).FirstOrDefault();

return account;

}

}

### Put Method 修改資料

//原始方法

// PUT api/values/5

public void Put(int id, [FromBody] string value)

{

}

接續POST案例，若我想做個修改密碼的API功能，則：

使用者只需輸入三個參數，帳號、密碼、欲更改的密碼。

API回傳成功、失敗兩種狀態之一。若是成功修改，則附帶更改後的帳號資訊。

public class PutClass

{

public string LoginAccount { get; set; }

public string Password { get; set; }

public string NewPassword { get; set; }

}

public class PutResponse : AccountData

{

public string ResponseStatus { get; set; }

}

// PUT: api/DataBase

public PutResponse Put([FromBody]PutClass putClass)

{

using(For\_TestEntities db = new For\_TestEntities())

{

var accountData = (from a in db.AccountData

where a.LoginAccount == putClass.LoginAccount &&

a.Password == putClass.Password

select a).FirstOrDefault();

PutResponse response = new PutResponse();

if (accountData != null)

{

accountData.Password = putClass.NewPassword;

db.SaveChanges();

response.ResponseStatus = "Success";

var changedAccountData = (from a in db.AccountData

where a.LoginAccount == putClass.LoginAccount &&

a.Password == putClass.NewPassword

select a).FirstOrDefault();

response.ID = changedAccountData.ID;

response.FirstName = changedAccountData.FirstName;

response.LastName = changedAccountData.LastName;

response.LoginAccount = changedAccountData.LoginAccount;

response.Password = changedAccountData.Password;

response.Status = changedAccountData.Status;

response.CreatedDate = changedAccountData.CreatedDate;

response.StaffData = changedAccountData.StaffData;

response.TravelAgencyUserData = changedAccountData.TravelAgencyUserData;

}

else

{

response.ResponseStatus = "Fail";

}

return response;

}

}

### Delete Method 刪除資料

//原始方法

// DELETE api/values/5

public void Delete(int id)

{

}

HTTP規範中，沒有明確禁止或允許Delete方法中是否可以包含Body。

但有不少人認為Delete方法應該要允許包含Body。

不過在某些伺服器中，包含Body可能會使伺服器無法正常運作。

在實務上，會偏向使用隱藏的方式，而不是直接刪除。

接續先前案例，使用API刪除一筆資料。

以下示範兩種方法，並且示範如何呼叫。

1. 使用Route(建議)

[HttpDelete]

[Route("api/DatabaseDelete/{LoginAccount}/{Password}")]

public object Delete(string LoginAccount, string Password)

{

using (For\_TestEntities db = new For\_TestEntities())

{

var accountData = (from a in db.AccountData

where a.LoginAccount == LoginAccount &&

a.Password == Password

select a).FirstOrDefault();

if (accountData == null)

{

var response = new { Status = "Fail" };

return response;

}

else

{

db.AccountData.Remove(accountData);

db.SaveChanges();

var response = new { Status = "Success" };

return response;

}

}

}

設定Route，從Uri中取得LoginAccount與Password，例如

<http://localhost:5147/api/database/wnilnay/NewPassword>

則，LoginAccount = wnilnay，Password = NewPassword

如何發出Delete請求

HttpClient httpClient = new HttpClient();

httpClient.BaseAddress = new Uri("http://localhost:5147");

httpClient.Timeout = new TimeSpan(0, 0, 5);

DeleteClass deleteClass = new DeleteClass();

deleteClass.LoginAccount = "wnilnay";

deleteClass.Password = "NewPassword";

HttpResponseMessage response = httpClient.DeleteAsync

($"/api/databaseDelete/{deleteClass.LoginAccount}/{deleteClass.Password}").Result;

if (response.IsSuccessStatusCode)

{

string result = response.Content.ReadAsStringAsync().Result;

Console.WriteLine(result);

}

else

{

Console.WriteLine(response.StatusCode);

Console.WriteLine(response.Content.ReadAsStringAsync()

.Result);

}

1. 使用Body

// DELETE: api/DataBase

public object Delete([FromBody] DeleteClass deleteClass)

{

using(For\_TestEntities db = new For\_TestEntities())

{

var accountData = (from a in db.AccountData

where a.LoginAccount == deleteClass.LoginAccount &&

a.Password == deleteClass.Password

select a).FirstOrDefault();

if(accountData == null)

{

var response = new { Status = "Fail" };

return response;

}

else

{

db.AccountData.Remove(accountData);

db.SaveChanges();

var response = new { Status = "Success" };

return response;

}

}

}

在HttpClient的DeleteAsync方法中，無法傳入Body。因此需要換個方法。

使用HttpRequestMessage可以達到我的需求。

DeleteClass deleteClass = new DeleteClass();

deleteClass.LoginAccount = "wnilnay";

deleteClass.Password = "NewPassword";

string json\_delete = JsonSerializer.Serialize(deleteClass);

HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Delete,

"http://localhost:5147/api/database");

request.Content = new StringContent(json\_delete,Encoding.UTF8,"application/json");

HttpClient httpClient = new HttpClient();

HttpResponseMessage response = httpClient.SendAsync(request).Result;

if (response.IsSuccessStatusCode)

{

string result = response.Content.ReadAsStringAsync().Result;

Console.WriteLine(result);

}

else

{

Console.WriteLine(response.StatusCode);

Console.WriteLine(response.Content.ReadAsStringAsync()

.Result);

}

### Body

Post、Put方法，需傳入Body，通常傳入Json格式，傳入的Json字串會自動反序列化為一個物件。

且若API方法回傳一個物件，則也會自動序列化為Json字串。

### 路由

預設路徑：[http://localhost:](http://localhost:r)連接埠/api/Controllers(/id)

主要變化在Controllers中，若新建一個Controllers，叫做BasicControllers，則預設路徑為/api/basic(/id)

在程式碼中，可以使用[Route("路由")]更改路徑，例如

[Route("api/BasicGet/{id}")]

public void Func(int id)

{

}

使用以下網址：http://localhost:5000/api/basicGet/5

不管是使用何種請求，都會導向Func函式，且id為5。

若要限定特定請求方法，可以使用[HTTP 方法](#_HTTP方法)

### HTTP方法

包含

[HttpGet]、[HttpPost]、[HttpPut]、[HttpDelete]...

等等。

可限定特定請求方法，可與[Route](#_路由)一起使用。

寫於函式前

[HttpGet]

[Route("api/BasicGet/{id}")]

public void Func(int id)

{

}

### 發佈 Web API

在專案上按右鍵🡪發佈🡪資料夾🡪選擇資料夾位置🡪完成

完成後，必須點擊發佈，才會將專案發佈至資料夾。

### 部署到IIS

1. 安裝IIS

開啟或關閉Windows功能🡪勾選「Internal Information Services」

Internal Information Services🡪World Wide Web服務🡪應用程式開發工具🡪勾選「ASP.NET 4.8」

1. 開啟IIS

開啟Internal Information Services (IIS) 管理員，左邊連線選擇

電腦名稱(電腦名稱\使用者)。若右側動作為啟動狀態，則可以至瀏覽器進入網站「[http://localhost:80」，若能成功進入](http://localhost:80」，若能成功進入網5)網站，說明已成功開啟IIS。

1. 部署API至IIS

左側選擇站台🡪右側新增網站🡪設定站台名稱、實體路徑、連接埠

實體路徑為先前[發佈](#_發佈_Web_API)至資料夾的資料夾路徑。

1. 測試是否部署成功

開啟瀏覽器，進入網址「<http://localhost>:連接埠」，若能進入網站，則說明部署成功。

## 經驗

以上Web API與請求專案(Windows Forms App)程式碼，我將放置在另一個[GitHub儲存庫](https://github.com/wnilnay/Web_API_and_Request_Project.git)中，不過專案在另一台電腦中可能因為環境不同而無法正常執行，不過程式碼均無錯誤，主要是對上述Web API程式碼內容進行總結。

我對於這個科目練習的時間不多，沒有特別的經驗可以分享。

主要經驗內容與[C#經驗分享](#_經驗)相似。

# 總結

商務軟體設計需要學習的技能很多，此文件內容短時間內不需要全部理解。

以實作為優先，了解這些工具該如何使用，該如何變化，如何運用在一起。

C#的部分可以先試著做出歷屆試題全功能，若有需要補充理論，可再觀看教學影片或其他介紹。

Android Studio建議先觀看教學影片重要的部分，例如畫面切換或是元件的使用，再去練習歷屆試題。若沒有足夠的基礎，在練習歷屆試題時容易處處碰壁。

無論是分區賽，還是全國賽，都建議自備鍵盤帶過去，分區賽可能會遇到拿青軸，專門干擾其他選手的注意力，得保持專注，不要被外界影響。