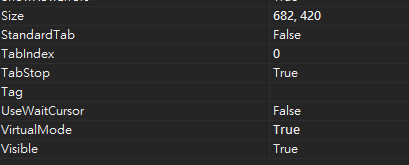
C# tips :

1. 盡量不 Class包class(DLL) ，若改寫可Orverride或另設一函數 巢狀類別不在這範圍(If也能用其他方式替代)
2. Datagridview 可以用虛擬化(virtualmode)方式來實現List對應資料視窗Cell的對應管理，記得開啟CellValueNeeded值更新對應功能與CellValuePushed資料視窗更動值能夠回推回List的資料





List<> lReadData(修改矩陣)

=>

Dgv\_ReadDataGrid\_CellValueNeeded(自己不能控制值)

1. 程式中結構以變數為準，若後方修改可避免程式錯位問題
2. List的格式嚴謹所以能輸入格式限制方式也嚴謹，避免跳出範圍外或有不相干字元(要限制輸入格式問題)
3. 陣列格式
   * 1. [] ：特定型別，固定長度的陣列，長度需事先宣告。

Int arr[5]

* + 1. List<>：特定型別，不固定長度的陣列。

List<string>

* + 1. Array：不特定型別，固定長度的陣列，長度需事先宣告。

Array arr[5]

* + 1. ArrayList：不特定型別，不固定長度的陣列。

ArrayList

補充：

Array與ArrayList是靠儲存 Object(弱型別)來達成非特定型態儲存，所以在執行階段調用時，會有轉型的成本跟風險，盡量避免使用。

Array是抽象類別，不能用new array建立。必須呼叫Array.CreateInstance() 來實作。

ArrayList的命名空間是System.Collections 預設不會載入，必須自行引用。

//===================================================================

Git和Subversion的命令比較表

|  |  |  |
| --- | --- | --- |
| **操作** | **Git** | **Subversion** |
| 複製數據庫 | git clone | svn checkout |
| 提交 | git commit | svn commit |
| 查看提交的詳細記錄 | git show | svn cat |
| 確認狀態 | git status | svn status |
| 確認差異 | git diff | svn diff |
| 確認記錄 | git log | svn log |
| 增加 | git add | svn add |
| 移動 | git mv | svn mv |
| 刪除 | git rm | svn rm |
| 取消修改 | git checkout / git reset | svn revert (※1) |
| 建立分支 | git branch | svn copy (※2) |
| 切換分支 | git checkout | svn switch |
| 合併 | git merge | svn merge |
| 建立標籤 | git tag | svn copy (※2) |
| 更新 | git pull / git fetch | svn update |
| 上傳到遠端 | git push | svn commit (※3) |
| 忽視檔案清單 | .gitignore | .svnignore |

※1. SVN的 revert 是用來取消修改，但Git的 revert 是用來取消提交。即使是同樣的命令，在SVN和Git裡的含義是不同的。

※2. SVN的分支與標籤在構造上是相同的，但在Git上明顯是不一樣的。

※3. SVN沒有本地端數據庫/遠端數據庫的概念，所以一提交馬上就會上傳到在遠端裡。但Git的本地端數據庫和遠端數據庫的表現方法是不一樣的。

//===================================================================

資料庫建置

Server用法

1. 安裝SSMS的windows的SQL Server
2. 以SQL Server mansgement studio 編輯資料表

指令相關:

select colA,colB,colC into tNewTable from tOldTable where colA=’x’ and colB is not null；

select後面是一個個的欄位，要哪個選哪個

into意味著放到一張新表裡面，沒有就是查詢出來

where之後的就是我們的條件，等於某個值，或者是不是空值

is表示”是否為”

not表示”不是”

null表示空值

//-----

select colA from oldtable group by colA

//----

增加欄位：alter tTable sTableName add sColName sVarChar;

減少欄位：alter tTable sTableName drop column colname;

//==========================================================

路徑:

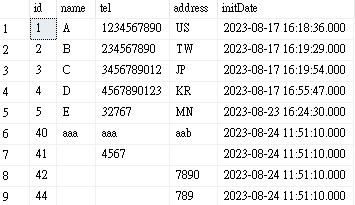
1. 當前路徑:
   * 1. string str = System.Environment.CurrentDirectory;
     2. Result: C:\xxx\xxx
2. 應用程序的可執行文件的路徑:
   * 1. string str = System.Windows.Forms.Application.StartupPath;
     2. Result: C:\xxx\xxx
3. 主模塊的完整路徑，包含文件名:
   * 1. string str = System.Diagnostics.Process.GetCurrentProcess().MainModule.FileName;
     2. Result: C:\xxx\xxx\aaa.exe
4. 獲取應用程序的當前工作目錄:
   * 1. string str = System.IO.Directory.GetCurrentDirectory();
     2. Result: C:\xxx\xxx

SQL資料庫檔案的實際路徑

指令:

SELECT name, physical\_name

FROM sys.master\_files;



SQL指令:

1.查詢項目

SELECT TOP 1000 [id]

,[name]

,[tel]

,[address]

,[initDate]

FROM [MVC\_TestDB].[dbo].[Table\_Customers]

//-------------------------------------------------

2.更新項目

DECLARE @name varchar(200)--先定義

DECLARE @tel varchar(200)

DECLARE @address varchar(200)

DECLARE @id INT

SET @name = 'aaa'--給值

SET @tel = 'aaa'

SET @address = 'aaa'

SET @id = 40

UPDATE [MVC\_TestDB].[dbo].[Table\_Customers] SET name = @name, tel = @tel,

address = @address WHERE id = @id

//---------------------------------------------------------------

3.刪除

DECLARE @id INT

SET @id = 43

DELETE [MVC\_TestDB].[dbo].[Table\_Customers] WHERE id = @id

4.新增(其中id與initDate會自己生成不用加)

DECLARE @name varchar(200)--先定義

DECLARE @tel varchar(200)

DECLARE @address varchar(200)

SET @name = 'aaa'--給值

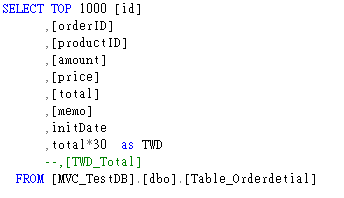
SET @tel = 'aaa'

SET @address = 'aaa'

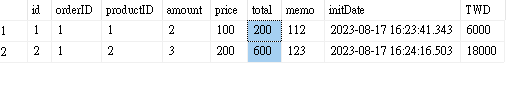
INSERT INTO [MVC\_TestDB].[dbo].[Table\_Customers] (name, tel, address)

VALUES (@name, @tel, @address)

語法很活 要多看怎麼應用



Result---



下面的例子表示有 Customers(顧客) 與 Orders(總交易)與Orderdetails(交易細項)

目標為: 來自US的顧客在”2023-08-25”這天工作幾筆交易包括總金額(其中總金額拿交易細項的做處理)

中間會用到

SELECT: 查詢

SUM: 加總內部項目，另外還有AVG(),COUNT(),MAX(),MIN()

AS: 將後方自定義名稱指為前者

{JOIN…ON…GROUP BY}

JOIN: 將JOIN前後資料表關聯

ON: 表示JOIN的條件 通常是對應鍵

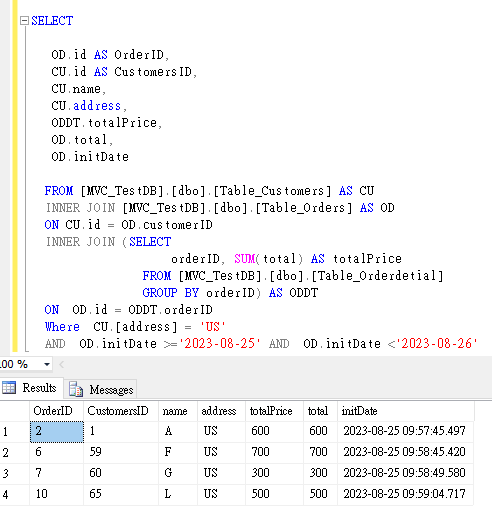
GROUP BY: 以特定Column做合併組的相關依據 有這個就必須使SELECT標籤變形

WHERE 查找結果的依據條件

AND 條件的連結方式 另外還有OR,NO,NOT這些運算子

下圖中的totalPrice 為Orderdetails中對應orderID關係所GROUP BY的方式建立

可以對應到Orders中的total 是相同的



Code:

SELECT

\*--全輸出

--OD.id AS OrderID,

--CU.id AS CustomersID,

--CU.name,

--CU.address,

--ODDT.totalPrice,

--OD.total,

--OD.initDate

FROM [MVC\_TestDB].[dbo].[Table\_Customers] AS CU

INNER JOIN [MVC\_TestDB].[dbo].[Table\_Orders] AS OD

ON CU.id = OD.customerID

INNER JOIN (SELECT

orderID, SUM(total) AS totalPrice

FROM [MVC\_TestDB].[dbo].[Table\_Orderdetial]

GROUP BY orderID) AS ODDT

ON OD.id = ODDT.orderID

Where CU.[address] = 'US'

AND OD.initDate >='2023-08-25' AND OD.initDate <'2023-08-26'

LIST:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | id | name | tel | address | initDate | id | customerID | total | initDate | orderID | totalPrice |
|  | 1 | A | 1234567890 | US | 2023-08-17 16:18:36.000 | 2 | 1 | 600 | 2023-08-25 09:57:45.497 | 2 | 600 |
|  | 59 | F | 123 | US | 2023-08-25 09:26:32.890 | 6 | 59 | 700 | 2023-08-25 09:58:45.420 | 6 | 700 |
|  | 60 | G | 123 | US | 2023-08-25 09:26:35.320 | 7 | 60 | 300 | 2023-08-25 09:58:49.580 | 7 | 300 |
|  | 65 | L | 123 | US | 2023-08-25 09:26:49.547 | 10 | 65 | 500 | 2023-08-25 09:59:04.717 | 10 | 500 |

另一解法

SELECT

OD.id AS OrderID,

CU.id AS CustomerID,

CU.name,

CU.address,

OD.total,

OD.initDate,

ODDT.orderID,SUM(ODDT.total) AS totalPrice

FROM [MVC\_TestDB].[dbo].[Table\_Customers] AS CU

INNER JOIN [MVC\_TestDB].[dbo].[Table\_Orders] AS OD

ON CU.id = OD.customerID

INNER JOIN [MVC\_TestDB].[dbo].[Table\_Orderdetial]AS ODDT

ON OD.id = ODDT.orderID

Where CU.[address] = 'US'

AND OD.initDate >='2023-08-25' AND OD.initDate <'2023-08-26'

GROUP BY ODDT.orderID,OD.id,CU.id,CU.name,OD.total,CU.address,OD.initDate

將SELECT中元素全部丟進GROUP BY當中，使每一項的聚合資料表行列相同能夠放入輸出 (想想 SUM是 好幾筆合成1筆，所以當然對不齊)

也可以以以下思路進行

SELECT

MAX(OD.id)AS OrderID,

MAX(CU.id) AS CustomersID,

MAX(CU.name)AS name,

MAX(CU.address)AS address,

MAX(OD.total)AS total,

MAX(OD.initDate) AS initDate,

ODDT.orderID,SUM(ODDT.total) AS totalPrice

FROM [MVC\_TestDB].[dbo].[Table\_Customers] AS CU

INNER JOIN [MVC\_TestDB].[dbo].[Table\_Orders] AS OD

ON CU.id = OD.customerID

INNER JOIN [MVC\_TestDB].[dbo].[Table\_Orderdetial]AS ODDT

ON OD.id = ODDT.orderID

Where CU.[address] = 'US'

AND OD.initDate >='2023-08-25' AND OD.initDate <'2023-08-26'

GROUP BY ODDT.orderID

就是每個都去聚合所以就沒問題 原理與丟進GROUP BY當中類似都是進行聚合

//-----------------------------------------------------------------------------------------------

在ShareMemory中測試了兩個方案的傳輸值

1. ShareMemory: 在記憶體中MemoryMappedFile一個區塊，使單一執行續能對區塊寫入(要定義區塊)，要利用WaitOne的方式去鎖定or解除執行續使用，反之讀取一樣的方式。
2. SendMessage: 以winuser.h的函數庫做視窗的訊息傳遞，因為對應數值皆為包裝完成的方式，基本上是借用FindWindow找出要傳送之視窗代碼、以SendMessage附帶用COPYDATASTRUCT包裝輸出訊息 與WndProc 的listener方式讀取COPYDATASTRUCT的資料解為輸出資料

//---------------------------------------------------------------------------------------

Volatile

三大特性

1. volatile修飾的變數具有可見性

2. volatile避免指令優化重排

3. volatile不保證Atomic

//------------------------------------------------------------------------------------------------------

Class繼承改泛型的內部Property需修改如下，才不會有Property繼承錯誤的問題

//BaseClass不變

public class BaseClass

{

public object Property { get; set; }

}

//改成存取BaseClass

public class GenericClass<T> : BaseClass

{

public new T Property

{

get

{

return (T)base.Property;

}

set

{

base.Property = value;

}

}

}

<https://dotblogs.com.tw/wadehuang36/2010/10/09/auto-implemented-property-bug>

//-------------------------------------------------------------------------------------

LINQ實作:

1. 使用LINQ查詢集合：

有一LIST

List<int> numbers = new List<int> { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

要取出條件(X%2==0) 以下方式:

var evenNumbers = from num in numbers

where num % 2 == 0

select num;

2.使用LINQ查詢資料庫：

有一資料庫

var DBContext = new YourDbContext();

//將資料庫中產品價錢大於100取出

var productsOVER100 = from p in DBContext.Products

where p.Price > 100

select p;

3. 使用LINQ查詢集合並排序：

var sortedNames = from name in names

orderby name descending //降冪排列

select name;

foreach (var name in sortedNames)

{

Console.WriteLine(name);

}

4. 使用LINQ查詢XML資料：

有一XML檔

XDocument xmlDoc = XDocument.Load("data.xml");

//將XML中產品價錢大於50取出

//並加入或鍵入元素(Title,Price)進books中

var books = from book in xmlDoc.Descendants("Book")

where (int)book.Element("Price") > 50

select new

{

//可能格式

Title = book.Element("Title").Value,

Price = (int)book.Element("Price")

};

5.使用LINQ進行集合操作：

var numbers1 = new List<int> { 1, 2, 3 };

var numbers2 = new List<int> { 3, 4, 5 };

var union = numbers1.Union(numbers2); // 合併兩個集合，去除重複項

var intersection = numbers1.Intersect(numbers2); // 取兩個集合的交集

var difference = numbers1.Except(numbers2); // 取兩個集合的差集

LINQ查詢集合並使用群組：

var groupedNames = from name in names

group name by name[0] into nameGroup

select new {

Initial = nameGroup.Key, Names = nameGroup

};

foreach (var group in groupedNames)

{

Console.WriteLine($"Initial: {group.Initial}");

foreach (var name in group.Names)

{

Console.WriteLine($" - {name}");

}

}

與方法語法結合使用：

var evenNumbers = numbers.Where(num => num % 2 == 0).ToList();

var sortedNames = names.OrderBy(name => name.Length).ToList();

var groupByInitial = names.GroupBy(name => name[0]).ToList();

First,Single,Any

var firstCustomer = customers.First();

var singleCustomer = customers.Single(c => c.Id == 2);

bool anyHighPricedOrder = orders.Any(order => order.Amount > 100);

使用 Distinct 去除重複的項目：

var uniqueNames = names.Distinct();

//-------------------------------------------------------------------------

Action用法: