Build School

Build School 課程 – C# 基礎課程 04

Bill Chung
Build School 講師
V2021.2 新竹尖兵班

Build School

linq 再進階

運算

- Sum
  - 加總
- Min
  - 取得最小值
- Max
  - 取得最大值
- Count
  - 取得數量
- Average
  - 取得平均值

LAB

使用 linq 運算

新增一個方案及專案

- 方案名稱：ExLinqSamples
- 專案名稱：ExLinqSample001
- 範本：Console Application

加入 MyData Class，並建立其內容

```
class MyData
 {
     public string Name
     { get; set; }

     public int Age
     { get; set; }
 }
```

在 Program Calss 加入產生 List<MyData> 的方法

```
static List<MyData> CreateList()
{
    return new List<MyData>()
    {
        new MyData() { Name = "Bill" , Age = 47 },
        new MyData() { Name = "John" , Age = 37 },
        new MyData() { Name = "Tom"  , Age = 48 },
        new MyData() { Name = "David", Age = 36},
        new MyData() { Name = "Bill" , Age = 35 },
    };
}
```

在 Main Method 加入 程式碼

```
static void Main(string[] args)
{
    var list = CreateList();
    // 計算 list 中，所有 Age 的總和
    int total = list.Sum((x) => x.Age);
    Console.WriteLine($"年齡的總和為: {total}");
    // 取得 list 中, Age 最小的值
    var minAge = list.Min((x) => x.Age);
    Console.WriteLine($"最小的年齡為 : {minAge}");
    // 取得 list 中, Age 最大的值
    var maxAge = list.Max((x) => x.Age);
    Console.WriteLine($"最大的年齡為 : {maxAge}");
    // 取得 list 中的數量
    //請注意 Count 和 Count() 是不一樣的
    int count = list.Count();
    Console.WriteLine($"list 總個數為 : {count}");
    int countOfBill = list.Count((x) => x.Name == "Bill");
    Console.WriteLine($"list 中的 Bill 總數量為 : {countOfBill}");
    // 取得所有年齡的平均值
    var average = list.Average((x) => x.Age);
    Console.WriteLine($"年齡的平均值為 : {average}" );
    Console.ReadLine();
}
```

執行

LAB

複合查詢
這個練習做有條件的 Min，Sum 與 Average

在 LinqSamples 加入新專案

- 方案名稱： ExLinqSamples
- 專案名稱：ExLinqSample002
- 範本：Console Application

加入 MyData Class，並建立其內容

```
class MyData
 {
     public string Name
     { get; set; }

     public int Age
     { get; set; }
 }
```

在 Program Calss 加入產生 List<MyData> 的方法

```
static List<MyData> CreateList()
{
    return new List<MyData>()
    {
        new MyData() { Name = "Bill" , Age = 47 },
        new MyData() { Name = "John" , Age = 37 },
        new MyData() { Name = "Tom"  , Age = 48 },
        new MyData() { Name = "David", Age = 36},
        new MyData() { Name = "Bill" , Age = 35 },
    };
}
```

在 Main Method 加入 程式碼

```
static void Main(string[] args)
{
    var list = CreateList();
    // 找出名稱為 Bill 中的最小 Age
    var min = list.Where((x) => x.Name == "Bill").Min((x) => x.Age);
    Console.WriteLine($"所有 Bill 中最小的年齡是 : {min}");

    // 計算名稱為 Bill 的年齡總和
    var total = list.Where((x) => x.Name == "Bill").Sum((x) => x.Age);
    Console.WriteLine($"所有 Bill 的年齡總和是 : {total}");

    var average = list.Where((x) => x.Name == "Bill").Average((x) => x.Age);
    Console.WriteLine($"所有 Bill 的年齡平均是 : {average}");


    Console.ReadLine();
}
```

執行

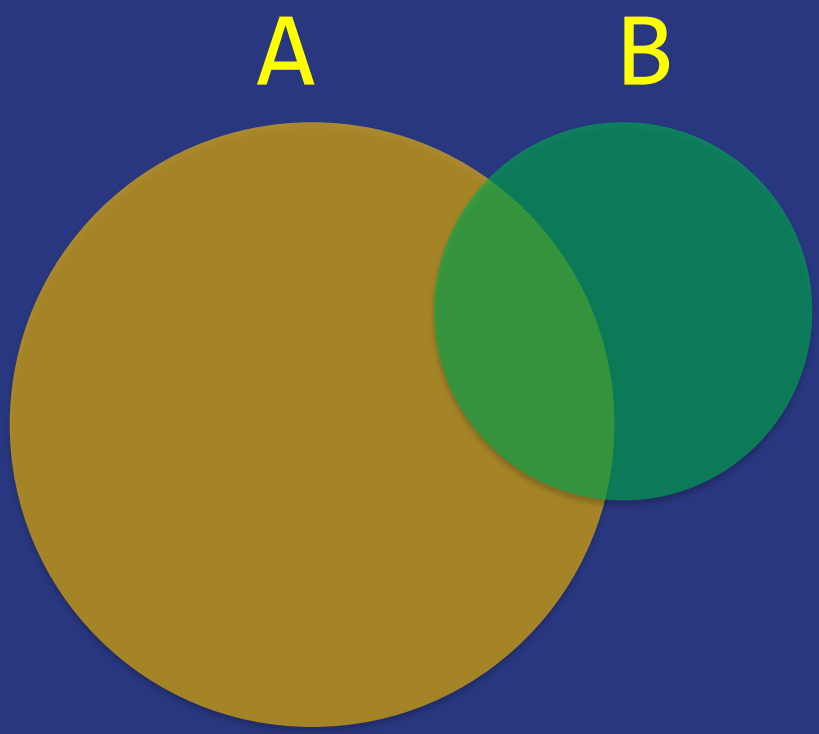
討論

- 是否了解如何使用複合的查詢方式呢？

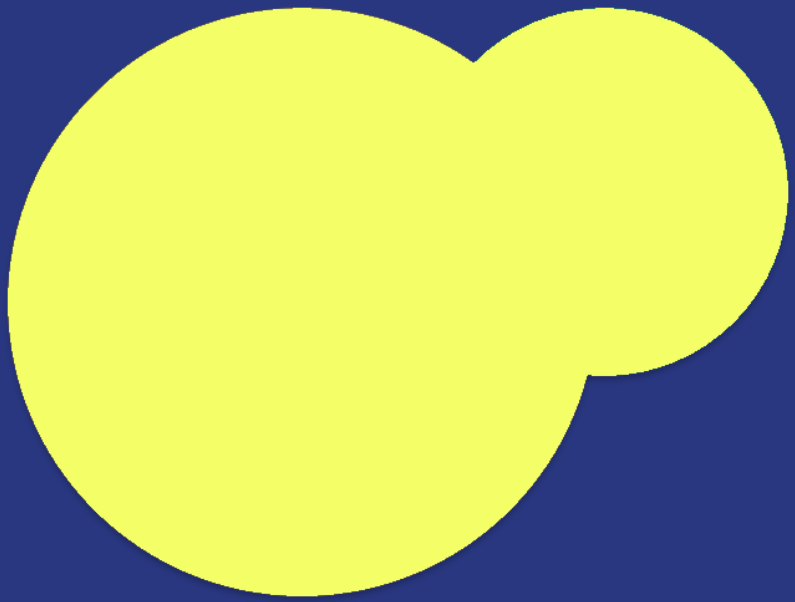
聯集、交集與差集

- Union
  - 取得兩個 IEnumerable <T> 的聯集
- Intersect
  - 取得兩個 IEnumerable <T> 的交集
- Except
  - 取得兩個 IEnumerable <T> 的差集
  - 【A 差集 B】 和 【B 差集 A】是不一樣的

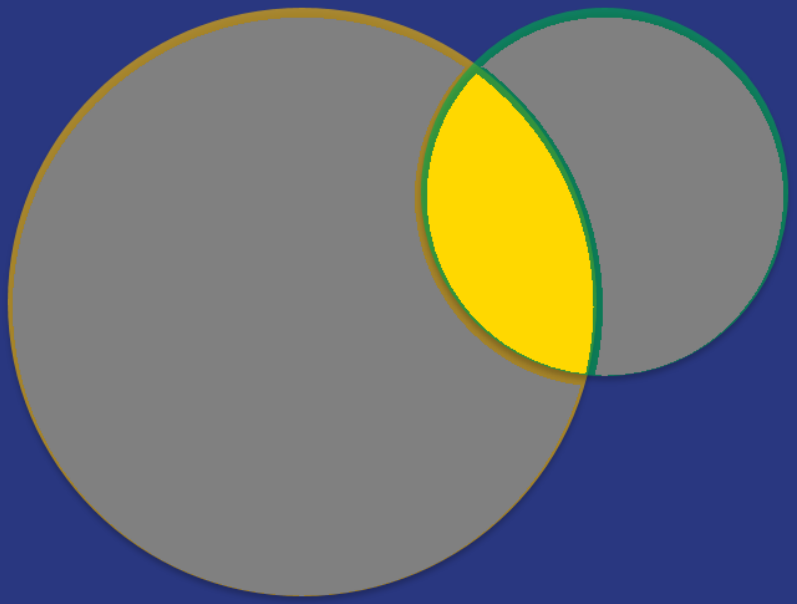
假設有兩個 IEmuerable<T>



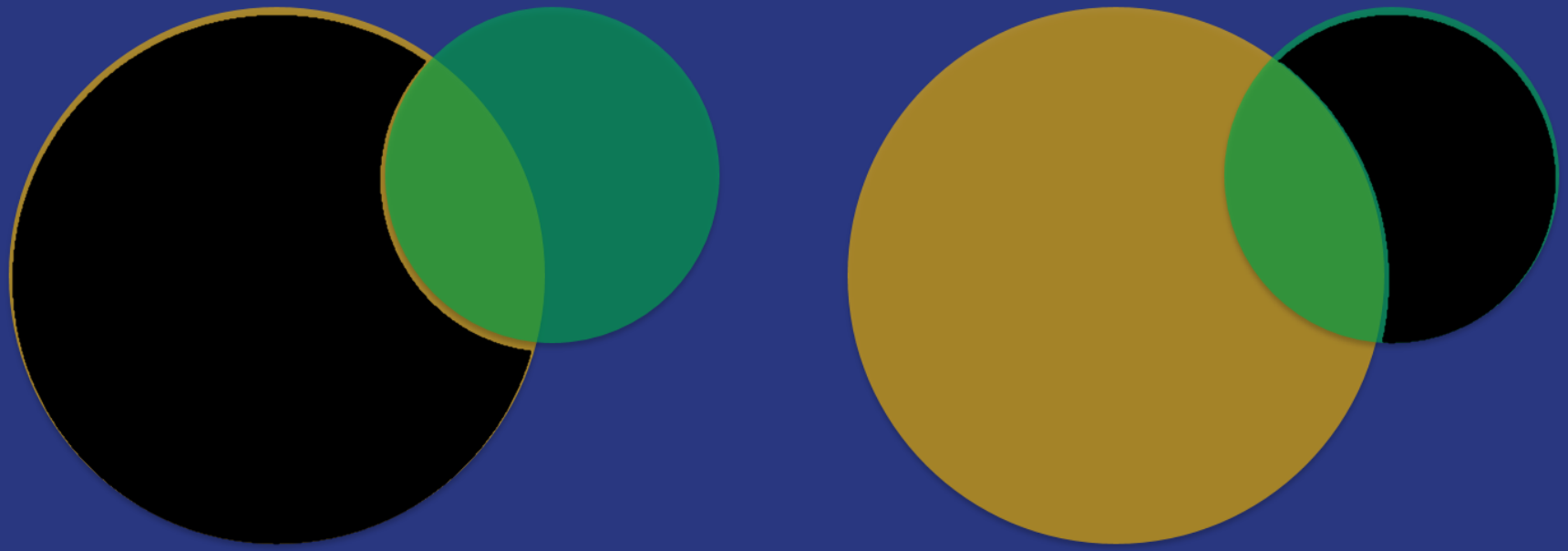
聯集



交集



差集



LAB

使用 Union 與 Intersect

在 LinqSamples 加入新專案

- 方案名稱： ExLinqSamples
- 專案名稱：ExLinqSample003
- 範本：Console Application

直接在 Main Method 加入 程式碼

```
static void Main(string[] args)
 {
     var list1 = new List<int> { 1, 2, 3, 4, 5, 6 };
     var list2 = new List<int> { 1, 3, 4, 7, 8, 9 };
     var union = list1.Union(list2);
     Console.WriteLine("聯集的結果為 :");
     foreach (var item in union )
     {
         Console.WriteLine(item);
     }

     var intersect = list1.Intersect(list2);
     Console.WriteLine("交集的結果為 :");
     foreach (var item in intersect)
     {
         Console.WriteLine(item);
     }

     Console.ReadLine();
 }
```

執行

討論

- 了解 Union，Intersect 的意義和其使用方式了嗎？
- 如果蓋上電腦，你是否能算出兩個集合的聯集與交集？

LAB

使用 Except

在 LinqSamples 加入新專案

- 方案名稱： ExLinqSamples
- 專案名稱：ExLinqSample004
- 範本：Console Application

直接在 Main Method 加入 程式碼

```
static void Main(string[] args)
{
    var list1 = new List<int> { 1, 2, 3, 4, 5, 6 };
    var list2 = new List<int> { 1, 3, 4, 7, 8, 9 };
    var aEXb = list1.Except(list2);
    Console.WriteLine("A 差集 B 的結果為 :");
    foreach (var item in aEXb)
    {
        Console.WriteLine(item);
    }

    var bEXa = list2.Except(list1);
    Console.WriteLine("B 差集 A 的結果為: ");
    foreach (var item in bEXa)
    {
        Console.WriteLine(item);
    }

    Console.ReadLine();
}
```

執行

討論

- 了解 Except 的意義和其使用方式了嗎？
- 如果蓋上電腦，你是否能算出兩個集合的差集？

猜猜這兩行在做甚麼？

```
list1.Except(list2).Union(list2.Except(list1));

list1.Union(list2).Except(list1.Instersect(list2));
```

Distinct

- Distinct
  - 排除重複，如果有兩個以上相同的資料，只會取一個

LAB

使用 Distinct

在 LinqSamples 加入新專案

- 方案名稱： ExLinqSamples
- 專案名稱：ExLinqSample005
- 範本：Console Application

直接在 Main Method 加入 程式碼

```
static void Main(string[] args)
{
    var list =
        new List<string> { "台北", "台北", "洛杉磯", "紐約", "紐約", "台北" };
    var result = list.Distinct();
    foreach (var item in result)
    {
        Console.WriteLine(item);
    }

    Console.ReadLine();
}
```

執行

Skip 與 Take

- Skip
  - 跳過幾筆
- Take
  - 取得幾筆

LAB

使用 Skip 與 Take

在 LinqSamples 加入新專案

- 方案名稱： ExLinqSamples
- 專案名稱：ExLinqSample006
- 範本：Console Application

```
static void Main(string[] args)
{
    var list = new List<string> {"A","B","C","D","E","F","F"};
    var resultOfSkip = list.Skip(3);
    Console.WriteLine("Skip(3) 的結果 ");
    Display(resultOfSkip);

    var resultOfTake = list.Take(3);
    Console.WriteLine("Take(3) 的結果 ");
    Display(resultOfTake);

    var resultOfSkipTake = list.Skip(2).Take(2);
    Console.WriteLine("Skip(2).Take(2) 的結果 ");
    Display(resultOfSkipTake);

    Console.ReadLine();
}

static void Display(IEnumerable <string> source)
{
    foreach (var item in source)
    {
        Console.WriteLine(item);
    }
}
```

執行

複製成另一個集合

- ToArray
- ToList
- ToDictionary

LAB

使用 ToArray、ToList 與 ToDictionary

在 LinqSamples 加入新專案

- 方案名稱： ExLinqSamples
- 專案名稱：ExLinqSample007
- 範本：Console Application

加入 MyData Class，並建立其內容

```
class MyData
 {
     public string Name
     { get; set; }

     public int Age
     { get; set; }
 }
```

在 Program Calss 加入產生 List<MyData> 的方法

```
static List<MyData> CreateList()
{
    return new List<MyData>()
    {
        new MyData() { Name = "Bill" , Age = 47 },
        new MyData() { Name = "John" , Age = 37 },
        new MyData() { Name = "Tom"  , Age = 48 },
        new MyData() { Name = "David", Age = 36},
    };
}
```

在 Main Method 加入 程式碼

```
static void Main(string[] args)
{
    var list = CreateList();
    var result1 = list.Where((x) => x.Age > 40).ToList();
    var result2 = list.Where((x) => x.Age > 40).ToArray();
    // 使用 Name 當群組分類的索引鍵，而值資料仍然是 MyData
    var result3 = list.Where((x) => x.Age > 40).ToDictionary((x) => x.Name);

    foreach (var item in result3 )
    {
        Console.WriteLine(item.Key);
        Console.WriteLine($"{item.Value.Name} -- {item.Value.Age}");
    }
    Console.WriteLine("--------------");

    // 使用 Name 當群組分類的索引鍵，而且用 Age 當值資料
    var result4 = list.ToDictionary((x) => x.Name, (y) => y.Age);
    foreach (var item in result4)
    {
        Console.WriteLine(item.Key);
        Console.WriteLine(item.Value);
    }
    Console.ReadLine();
}
```

執行

討論

- ToList、ToArray 和 ToDictionary 的用法

群組

- GroupBy
  - 依據條件將資料分成群組
  - 回傳型別是 `IEnumerable<IGrouping<TKey,TSource>>`

LAB

GroupBy – Method Expression

在 LinqSamples 加入新專案

- 方案名稱： ExLinqSamples
- 專案名稱：ExLinqSample008
- 範本：Console Application

加入 MyData Class，並建立其內容

```
class MyData
{
    public string City
    { get; set; }

    public string Name
    { get; set; }
}
```

在 Program Calss 加入產生 List<MyData> 的方法

```
static List<MyData> CreateList()
{
    return new List<MyData>()
    {
        new MyData() { Name = "Bill" , City = "台北" },
        new MyData() { Name = "John" , City = "台北" },
        new MyData() { Name = "Tom"  , City = "高雄" },
        new MyData() { Name = "David", City = "台南" },
        new MyData() { Name = "Jeff" , City = "台南" },
    };
}
```

在 Main Method 加入 程式碼

```
static void Main(string[] args)
{
    var list = CreateList();
    var result = list.GroupBy((x) => x.City);
    foreach (var item in result )
    {
        Console.WriteLine($"住在 : {item.Key}");
        foreach (var p in item)
        {
            Console.WriteLine(p.Name);
        }
        Console.WriteLine("--------");
    }
    Console.ReadLine();
}
```

執行

LAB

GroupBy – Query Expression

在 LinqSamples 加入新專案

- 方案名稱： ExLinqSamples
- 專案名稱：ExLinqSample009
- 範本：Console Application

加入 MyData Class，並建立其內容

```
class MyData
{
    public string City
    { get; set; }

    public string Name
    { get; set; }
}
```

在 Program Calss 加入產生 List<MyData> 的方法

```
static List<MyData> CreateList()
{
    return new List<MyData>()
    {
        new MyData() { Name = "Bill" , City = "台北" },
        new MyData() { Name = "John" , City = "台北" },
        new MyData() { Name = "Tom"  , City = "高雄" },
        new MyData() { Name = "David", City = "台南" },
        new MyData() { Name = "Jeff" , City = "台南" },
    };
}
```

在 Main Method 加入 程式碼

```
static void Main(string[] args)
{
    var list = CreateList();
    var result =
        from o in list
        group o by o.City into gp
        select gp;

    foreach (var item in result)
    {
        Console.WriteLine($"住在 : {item.Key}");
        foreach (var p in item)
        {
            Console.WriteLine(p.Name);
        }
        Console.WriteLine("--------");
    }
    Console.ReadLine();
}
```

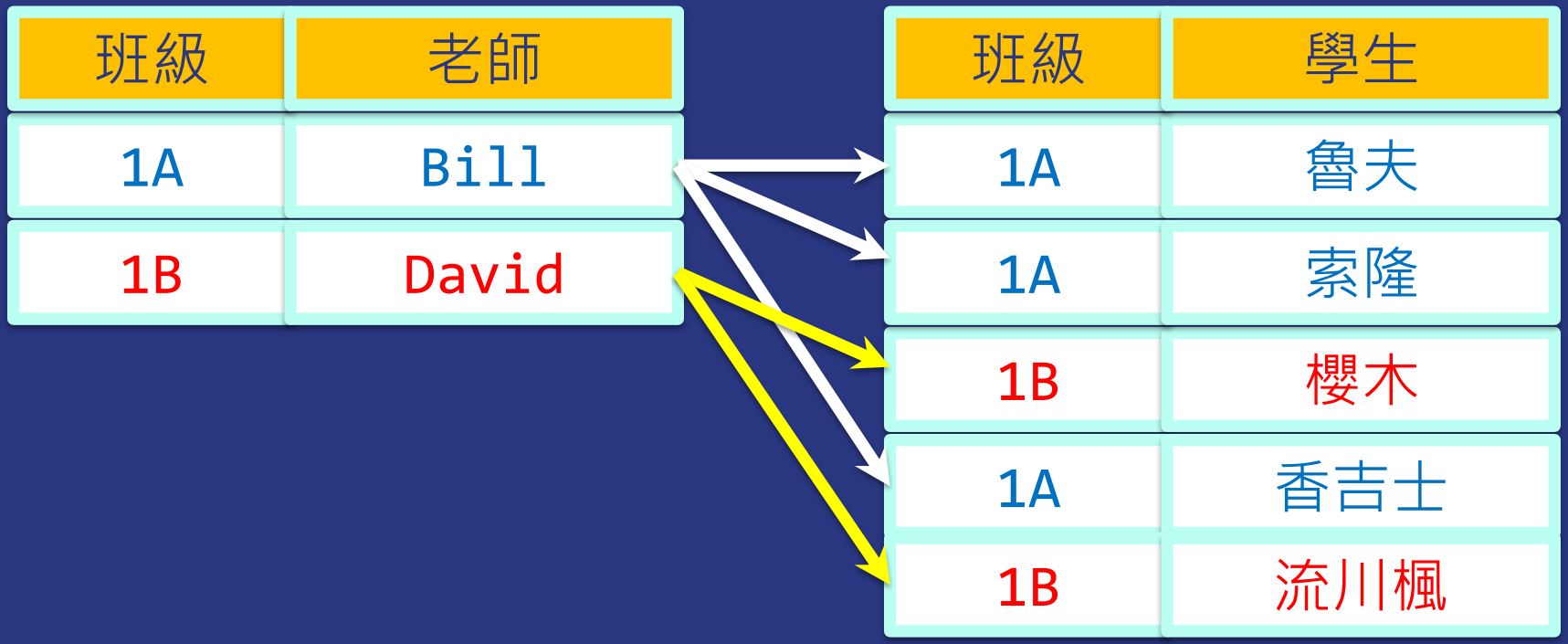
執行

討論

- 解釋 GroupBy 的用法

關聯

- Join
  - 根據相符索引鍵的兩個序列的項目相互關聯
  - Join 通常用 Query Expression 寫


班級
老師
1A
Bill
1B
David
班級
學生
1A
魯夫
1A
索隆
1B
櫻木
1A
香吉士
1B
流川楓

班級	老師	學生
1A	Bill	魯夫
1A	Bill	索隆
1A	Bill	香吉士
1B	David	櫻木
1B	David	流川楓

LAB

Join

在 LinqSamples 加入新專案

- 方案名稱： ExLinqSamples
- 專案名稱：ExLinqSample010
- 範本：Console Application

加入 TeacherInfo Class ， StudentIfo Class和 ResultInfo Class，並建立其內容

```
class TeacherInfo
{
    public string ClassName { get; set; }
    public string Teacher { get; set; }
}

class StudentInfo
{
    public string ClassName { get; set; }
    public string Student { get; set; }
}

class ResultInfo
{
    public string ClassName { get; set; }
    public string Teacher { get; set; }
    public string Student { get; set; }
}
```

在 Program Calss 加入產生來源資料的方法

```
static List<TeacherInfo> CreateTeachers()
{
    return new List<TeacherInfo>()
    {
       new TeacherInfo { ClassName ="1A" , Teacher ="Bill" },
       new TeacherInfo { ClassName ="1B" , Teacher ="David"}
    };

}

static List<StudentInfo> CreateStudents()
{
    return new List<StudentInfo>()
    {
       new StudentInfo { ClassName ="1A" , Student ="魯夫"  },
       new StudentInfo { ClassName ="1A" , Student ="索隆"  },
       new StudentInfo { ClassName ="1B" , Student ="櫻木"  },
       new StudentInfo { ClassName ="1A" , Student ="香吉士"},
       new StudentInfo { ClassName ="1B" , Student ="流川楓"}
    };
}
```

在 Main Method 加入 程式碼

```
static void Main(string[] args)
{
    var teachers = CreateTeachers();
    var students = CreateStudents();
    var result =
        from t in teachers
        join s in students
        on t.ClassName equals s.ClassName
        select
        new ResultInfo
      { ClassName = t.ClassName, Teacher = t.Teacher, Student = s.Student };

    foreach (var item in result)
    {
        Console.WriteLine($"{item.ClassName} : {item.Teacher} : {item.Student}");
    }

    Console.ReadLine();
}
```

執行

討論

- 解釋 Join 的用法

排序

- OrderBy
  - 依照條件由小到大排序
- OrderByDescending
  - 依照條件由大到小排序
- ThenBy
  - 在 Mtehod Expression 需要兩個條件排序使用
- ThenByDescending
  - 在 Mtehod Expression 需要兩個條件排序使用

LAB

OrderBy -- Method Expression

在 LinqSamples 加入新專案

- 方案名稱： ExLinqSamples
- 專案名稱：ExLinqSample011
- 範本：Console Application

加入 MyData Class，並建立其內容

```
class MyData
 {
     public string Name
     { get; set; }

     public int Age
     { get; set; }
 }
```

在 Program Calss 加入產生 List<MyData> 的方法

```
static List<MyData> CreateList()
{
    return new List<MyData>()
    {
        new MyData { Name = "Bill" , Age = 47 },
        new MyData { Name = "John" , Age = 37 },
        new MyData { Name = "Tom"  , Age = 48 },
        new MyData { Name = "David", Age = 36 },
        new MyData { Name = "Bill" , Age = 35 },
    };
}
```

在 Program Class 加入 程式碼

```
static void Main(string[] args)
 {
     var list = CreateList();
     var order1 = list.OrderBy((x) => x.Age);
     Display(order1);
     var order2 = list.OrderByDescending((x) => x.Age);
     Display(order2);
     var order3 = list.OrderBy((x) => x.Name).ThenBy((x) => x.Age);
     Display(order3);
     var order4 = list.OrderBy((x) => x.Name).ThenByDescending((x) => x.Age);
     Display(order4);
     Console.ReadLine();
 }

 static void Display(IOrderedEnumerable<MyData> source)
 {
     foreach (var item in source)
     {
         Console.WriteLine(item.Name + " : " + item.Age);
     }
     Console.WriteLine("-------------");
 }
```

執行

LAB

OrderBy -- Query Expression

在 LinqSamples 加入新專案

- 方案名稱： ExLinqSamples
- 專案名稱：ExLinqSample012
- 範本：Console Application

加入 MyData Class，並建立其內容

```
class MyData
 {
     public string Name
     { get; set; }

     public int Age
     { get; set; }
 }
```

在 Program Calss 加入產生 List<MyData> 的方法

```
static List<MyData> CreateList()
{
    return new List<MyData>()
    {
        new MyData { Name = "Bill" , Age = 47 },
        new MyData { Name = "John" , Age = 37 },
        new MyData { Name = "Tom"  , Age = 48 },
        new MyData { Name = "David", Age = 36 },
        new MyData { Name = "Bill" , Age = 35 },
    };
}
```

在 Program Class 加入 程式碼

```
static void Main(string[] args)
{
    var list = CreateList();
    var order1 =
        from o in list
        orderby o.Name, o.Age
        select o;
    Display(order1);
    var order2 =
      from o in list
      orderby o.Name descending , o.Age descending
      select o;
    Display(order2);
    Console.ReadLine();
}

static void Display(IOrderedEnumerable<MyData> source)
{
    foreach (var item in source)
    {
        Console.WriteLine($"{item.Name} : {item.Age}");
    }
    Console.WriteLine("-------------");
}
```

執行

討論

- 解釋排序的用法

linq 的方法
是可以混在一起用的

在這裏你將學到 ....

Learn How to Learn

- **學新東西、新技術的能力**
- **尋找解答的能力**
- **隨時吸取新知識的能力**

跟著你一輩子的能力 ...

Build School