Build School

Build School 課程 – C# 基礎課程 02

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

Build School

集合

常用的集合型別

- Dictionary<TKey, TValue> 類別
- List<T> 類別
- ArrayList 類別 (大部分的狀況都不該用這個)
- 其他集合型別參考
  - System.Collections.Generic 命名空間
  - System.Collections 命名空間

Dictionary<TKey, TValue>

- 表示索引鍵和值的集合。
- 可以透過索引鍵取得對應的值。

你可以這樣新增資料給一個 Dictionary

當然，你得先建立這個 Dictionary 的執行個體

```
_dictionary = new Dictionary<string, DictionarySample.MyData>();
MyData data1 = new MyData();
data1.X = 10;
data1.Y = 10;
_dictionary.Add("D1", data1);
```

前面的寫法有點囉嗦，所以也可以這樣寫

```
_dictionary = new Dictionary<string, DictionarySample.MyData>();
_dictionary.Add("D1",new MyData() { X = 10, Y = 10 });
_dictionary["D5"] = new MyData() { X = 15, Y = 15 };
```

關於這段程式碼

```
new MyData() { X = 10, Y = 10 };
```

這種寫法被稱為『物件初始設定式』(Object Initializer)，甚至連 () 都可以省略

```
new MyData { X = 10, Y = 10 };
```

從 Dictionary 中移除資料

移除某個索引鍵的資料

```
_dictionary.Remove("D1");
```

從 Dictionary 中取得資料

(1)先確認索引鍵是否存在

```
MyData result = null;
if (_dictionary.ContainsKey("D1"))
{
    result = _dictionary["D1"];
}
```

(2)再使用此索引鍵取得對應的值

註：在 C# 中 [] 常常代表索引鍵 (index)

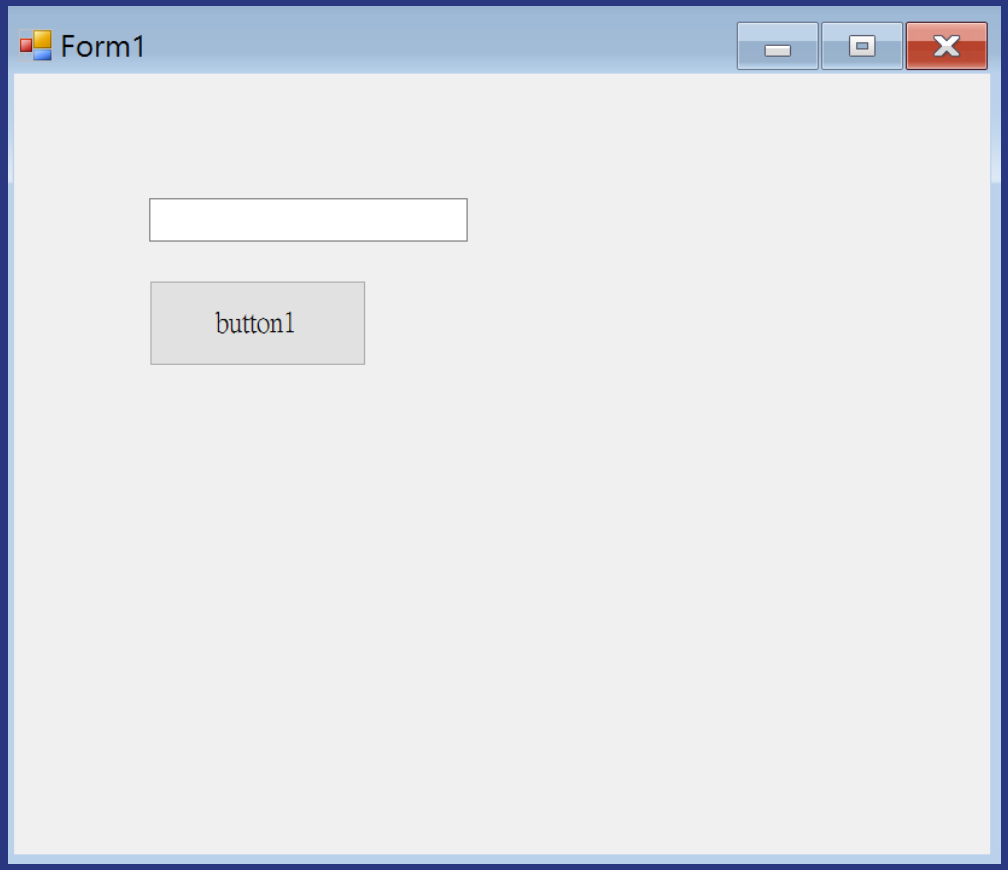
LAB

使用 Dictionary

新增一個方案及專案

- 方案名稱：DictionarySamples
- 專案名稱：DictionarySample001
- 範本：Windows Forms Application

畫面配置



一個 TextBox
一個 Button

先建立一個簡單的類別

```
class MyRectangle
{
    public int Width { get; set; }
    public int Height { get; set; }

    public int GetArea()
    {
        return Width * Height;
    }
}
```

在 Form1 Class 中宣告一個 Dictionary
索引鍵型別：string
值型別：MyRectangle

```
public partial class Form1 : Form
{
    private Dictionary<string, MyRectangle> _dictionary;
```

在 Form1 Class 中建立一個方法來初始化 _dictionary

```
private void CreateDictionary()
{
    _dictionary = new Dictionary<string, MyRectangle>();
    _dictionary.Add("D1", new MyRectangle { Width = 5, Height = 5 });
    _dictionary.Add("D2", new MyRectangle { Width = 10, Height = 10 });
    _dictionary.Add("D3", new MyRectangle { Width = 20, Height = 20 });
    _dictionary.Add("D4", new MyRectangle { Width = 100, Height = 100 });
}
```

註：範例中還有其他不同方式

在 Form1 Class的建構式呼叫剛剛建立的方法

```
public Form1()
{
    InitializeComponent();
    CreateDictionary();
}
```

為 Button 的 Click 事件掛上一個委派方法

```
private void button1_Click(object sender, EventArgs e)
{
    string key = textBox1.Text;
    if (_dictionary.ContainsKey(key))
    {
        int area = _dictionary[key].GetArea();
        MessageBox.Show($"{key} 的面積為： {area}");
    }
    else
    {
        MessageBox.Show("查無資料");
    }
}
```

執行

List<T>

- 表示可以依照索引存取的強型別物件清單。 提供搜尋、排序和管理清單的方法。
- 註：我們很常用 `List<T>`，但是陣列的效率是比較好的

List<T> 的宣告方式

這是 List 的變數名稱

```
List<string> _myList;
```

宣告一個以
string 為其內容元素的
的 List

你可以這樣新增資料給一個 List

當然，你得先建立這個 List 的執行個體

```
_myList = new List<string>();
_myList.Add("Cat");
_myList.Add("Dog");
```

```
_myList = new List<string>
{
    "dog", "cat"
};
```

匯入陣列的資料

```
string[] array = { "Cat", "Dog", "Fly" };
_myList.AddRange(array);
```

從 List<T> 中移除資料

移除某個特定的資料

```
_myList.Remove("Cat");
```

移除某個特定索引(位置)的資料

```
_myList.RemoveAt(1);
```

在 List<T> 中搜尋某個物件

```
string result = _list.Find((x) => x == "Cat");
```

這個玩意被稱為 Lambda 表示式
請先記得這個意思代表在 _list 中搜尋 “Cat” 這個字串
後面的課程會講這個原理

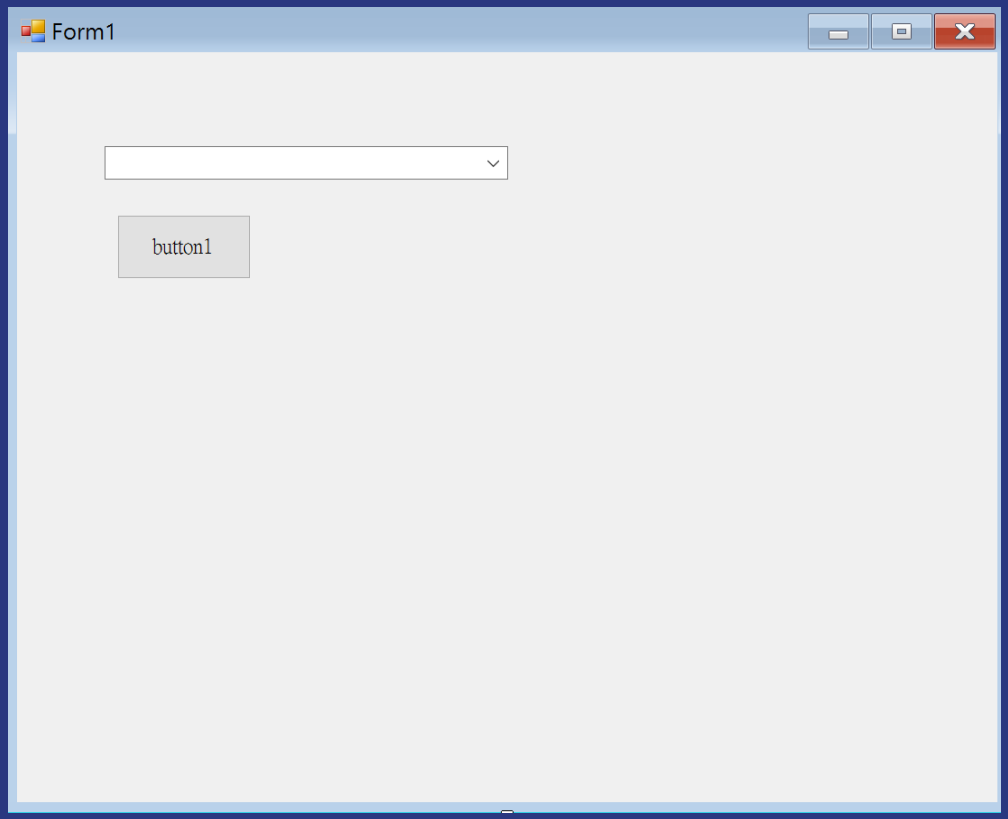
LAB

使用 List<T> 與 ComboBox

新增一個方案及專案

- 方案名稱：ListSamples
- 專案名稱：ListSample001
- 範本：Windows Forms Application

畫面配置



一個 ComboBox
一個 Button

先建立一個簡單的類別

```
class MyRectangle
{
    public string Name { get; set; }
    public int Width { get; set; }
    public int Height { get; set; }

    public int GetArea()
    {
        return Width * Height;
    }
}
```

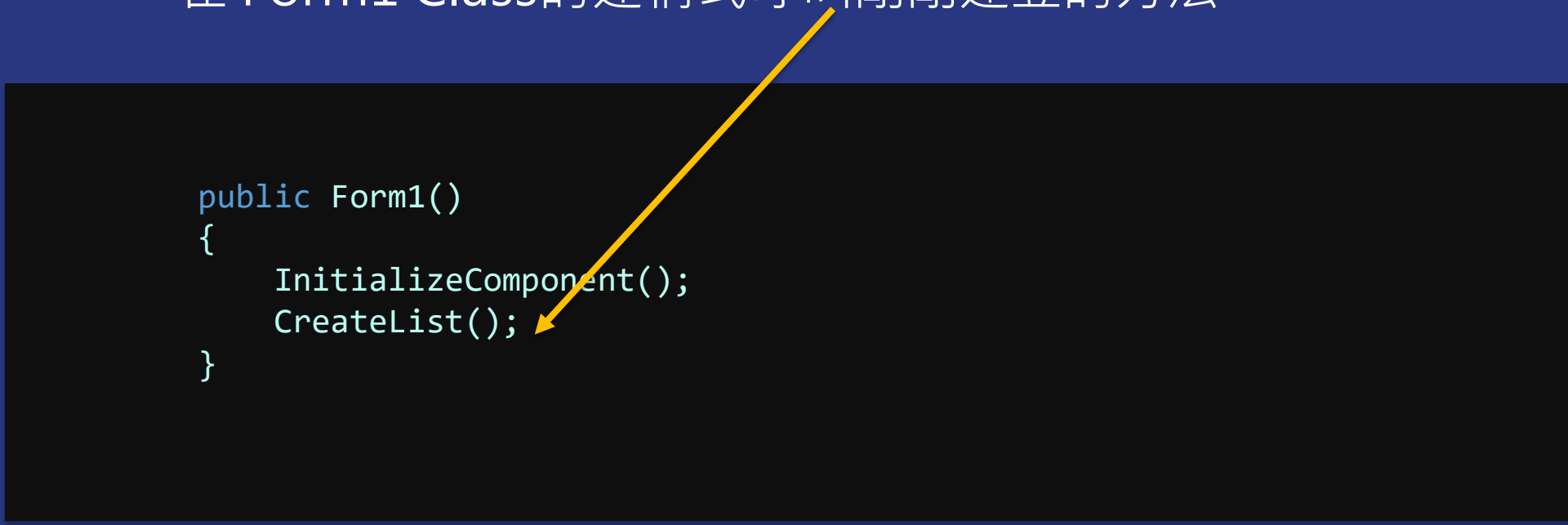
在 Form1 Class 中宣告一個 List<T>
元素型別：MyRectangle

```
public partial class Form1 : Form
{
    private List<MyRectangle> _list;
```

在 Form1 Class 中建立一個方法來初始化 _list

```
private void CreateList()
{
    _list = new List<MyRectangle>();
    _list.Add(new MyRectangle { Name = "D1", Width = 5, Height = 5 });
    _list.Add(new MyRectangle { Name = "D2", Width = 10, Height = 10 });
    _list.Add(new MyRectangle { Name = "D3", Width = 20, Height = 20 });
    _list.Add(new MyRectangle { Name = "D4", Width = 100, Height = 100 });
}
```

在 Form1 Class的建構式呼叫剛剛建立的方法

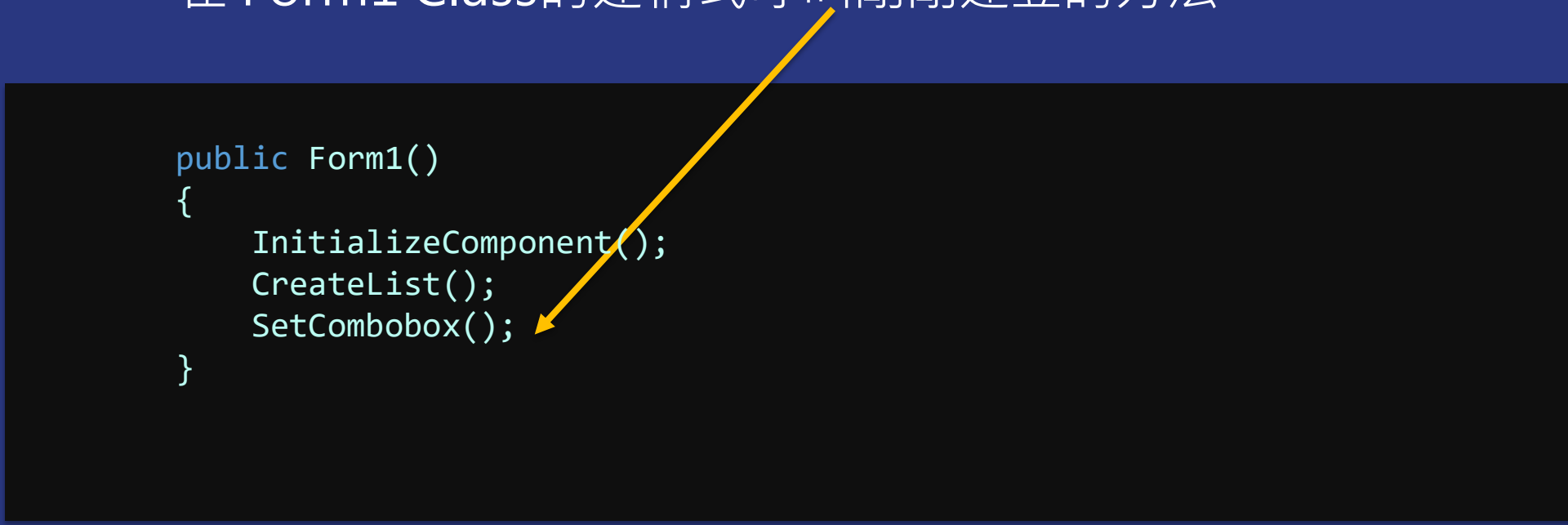


```
public Form1()
{
    InitializeComponent();
    CreateList();
}
```

建立一個方法把 _list 指派給 Combobox 的 DataSource 並且指定要在畫面顯示的屬性是哪一個

```
private void SetCombobox()
{
    comboBox1.DataSource = _list;
    comboBox1.DisplayMember = "Name";
}
```

在 Form1 Class的建構式呼叫剛剛建立的方法



```
public Form1()
{
    InitializeComponent();
    CreateList();
    SetCombobox();
}
```

執行

LAB

繼續前面未完成的部分

為 Button 的 Click 事件掛上一個委派方法

```
private void button1_Click(object sender, EventArgs e)
{
    MyRectangle item = (MyRectangle)comboBox1.SelectedItem;
    MessageBox.Show($" {item.Name} 的面積為: {item.GetArea()}");
}
```

執行

LAB

前面例子的變化題

想想看

- 如果我的目的只是要取得『面積』，有沒有其他的作法？
- 提示：Combobox 有一個 ValueMember 屬性
- 到 MSDN 文件庫看看 ValueMember 屬性的說明
  - ListControl.ValueMember 屬性

屬性值
Type: System.String
A String 代表單一屬性名稱的 DataSource 屬性值或句號分隔屬性名稱的解析成最終資料繫結物件的屬性名稱的階層。 預設為空字串 ("")。

Property Value
Type: System.String
A String representing a single property name of the DataSource property value, or a hierarchy of period-delimited property names that resolves to a property name of the final data-bound object. The default is an empty string ("").

討論

- 對照前一個 lab 中的 DisplayMember 的使用方式。
  你在 ValueMember 的說明看到了甚麼重點？

在 ListSamples 加入新專案

- 方案名稱：ListSamples
- 專案名稱：ListSample002
- 範本：Windows Forms Application
- 畫面配置同 ListSample001

一樣要建立 MyRectangle 類別

但這次不一樣，請把 GetArea 方法改成 Area 屬性

```
class MyRectangle
{
    public string Name { get; set; }
    public int Width { get; set; }
    public int Height { get; set; }

    public int Area
    {
      get { return Width * Height; }
    }
}
```

Why get only?

在 Form1 Class 中宣告一個 List<T>
元素型別：MyRectangle

```
public partial class Form1 : Form
{
    private List<MyRectangle> _list;
```

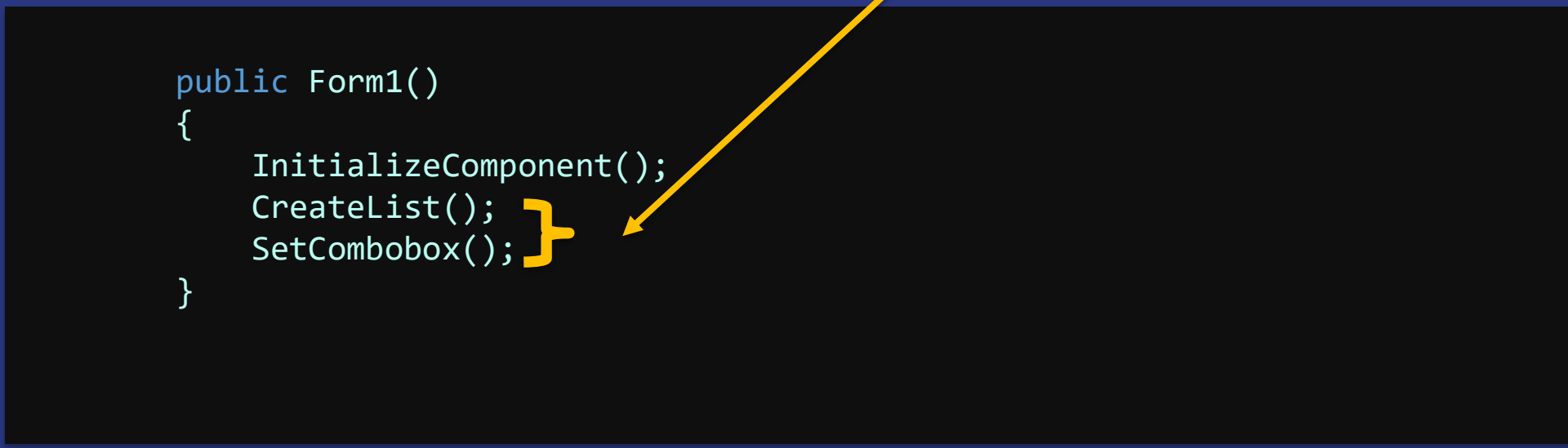
在 Form1 Class 中建立一個方法來初始化 _list

```
_list = new List<MyRectangle>()
{
    new MyRectangle { Name = "D1", Width = 5, Height = 5 },
    new MyRectangle { Name = "D2", Width = 10, Height = 10 },
    new MyRectangle { Name = "D3", Width = 20, Height = 20 },
    new MyRectangle { Name = "D4", Width = 100, Height = 100 }
};
```

建立一個方法把 _list 指派給 Combobox 的 DataSource
並且指定
(1)要在畫面顯示的屬性是哪一個
(2)選項值的屬性是哪一個

```
private void SetCombobox()
{
    comboBox1.DataSource = _list;
    comboBox1.DisplayMember = "Name";
    comboBox1.ValueMember = "Area";
}
```

在 Form1 Class的建構式呼叫剛剛建立的兩個方法



```
public Form1()
{
    InitializeComponent();
    CreateList();
    SetCombobox();
}
```

為 Button 的 Click 事件掛上一個委派方法
但是我們這次直接取 SelectedValue

```
private void button1_Click(object sender, EventArgs e)
{
    // SelectedItem 還是 MyRectangle
    int area = (int)comboBox1.SelectedValue;
    MessageBox.Show(area.ToString());
}
```

執行

ComboBox 其他的重要成員

- 屬性
  - SelectedIndex
- 事件
  - SelectedIndexChanged

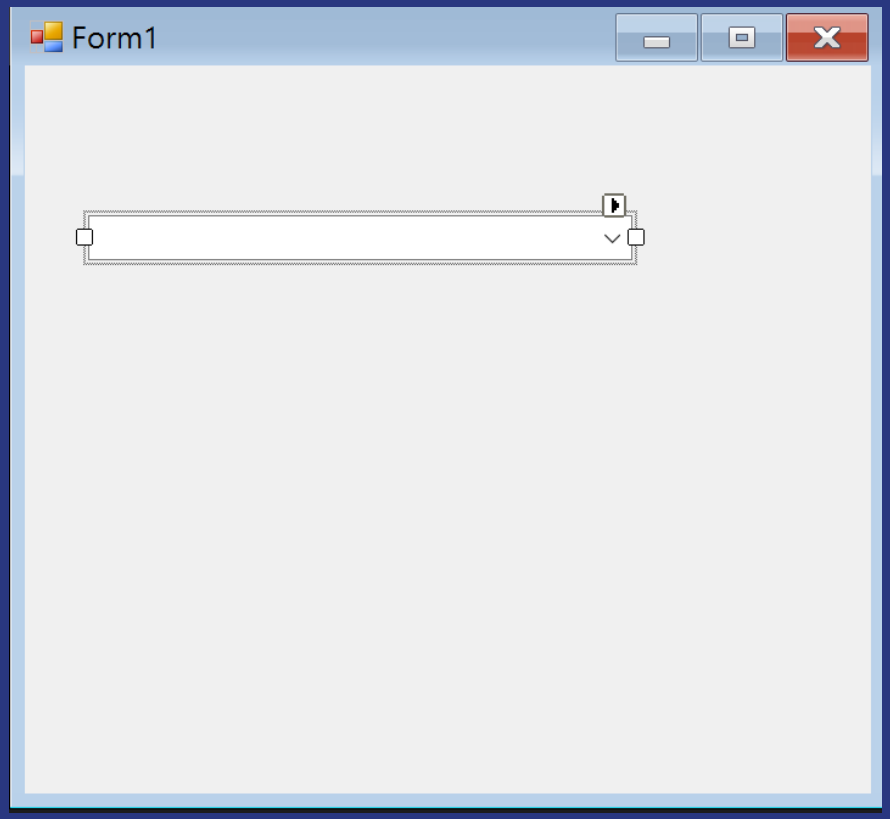
LAB

SelectedIndex
與
SelectedIndexChanged

在 ListSamples 加入新專案

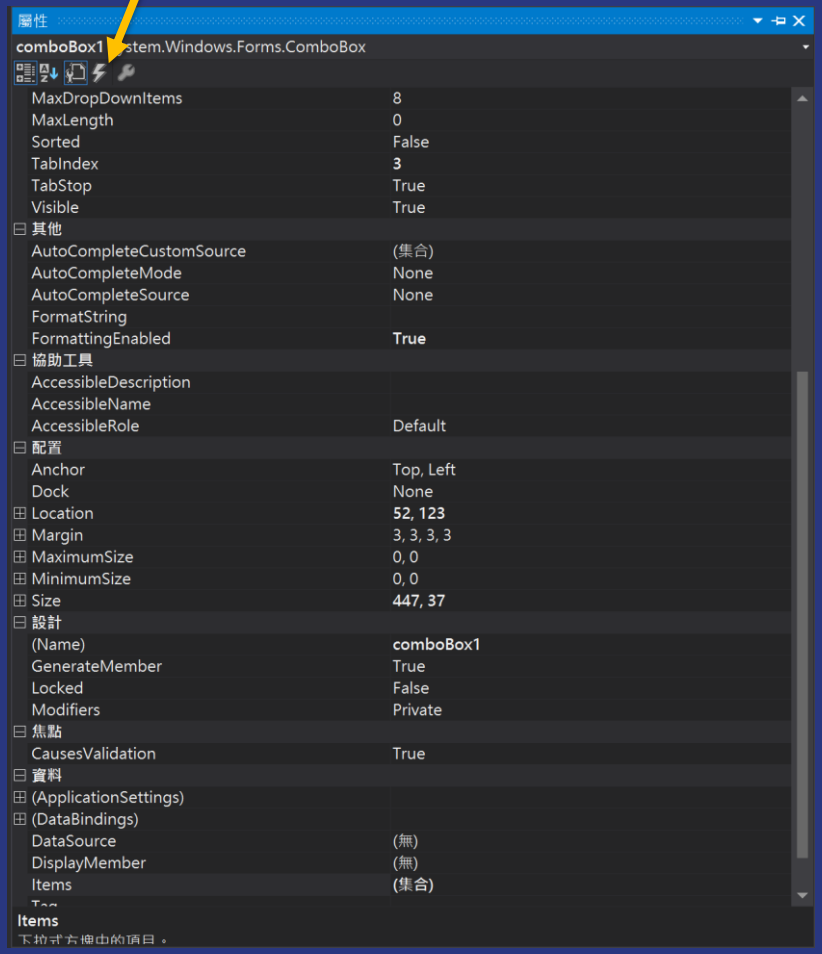
- 方案名稱：ListSamples
- 專案名稱：ListSample003
- 範本：Windows Forms Application

畫面配置

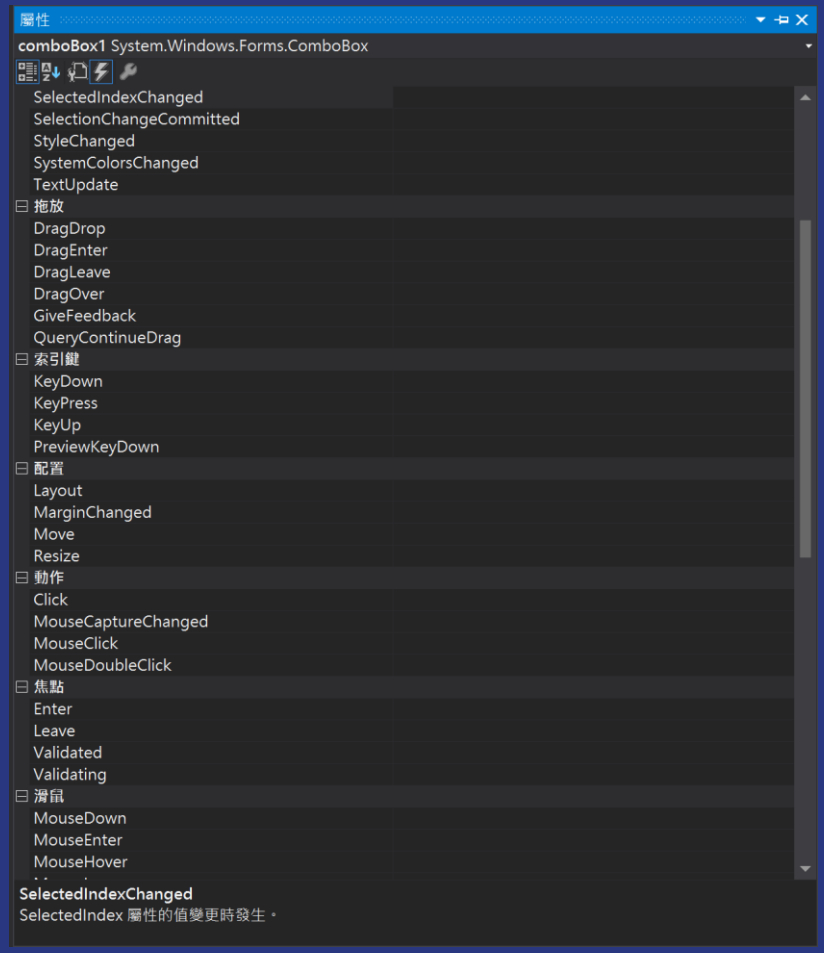


一個 ComboBox

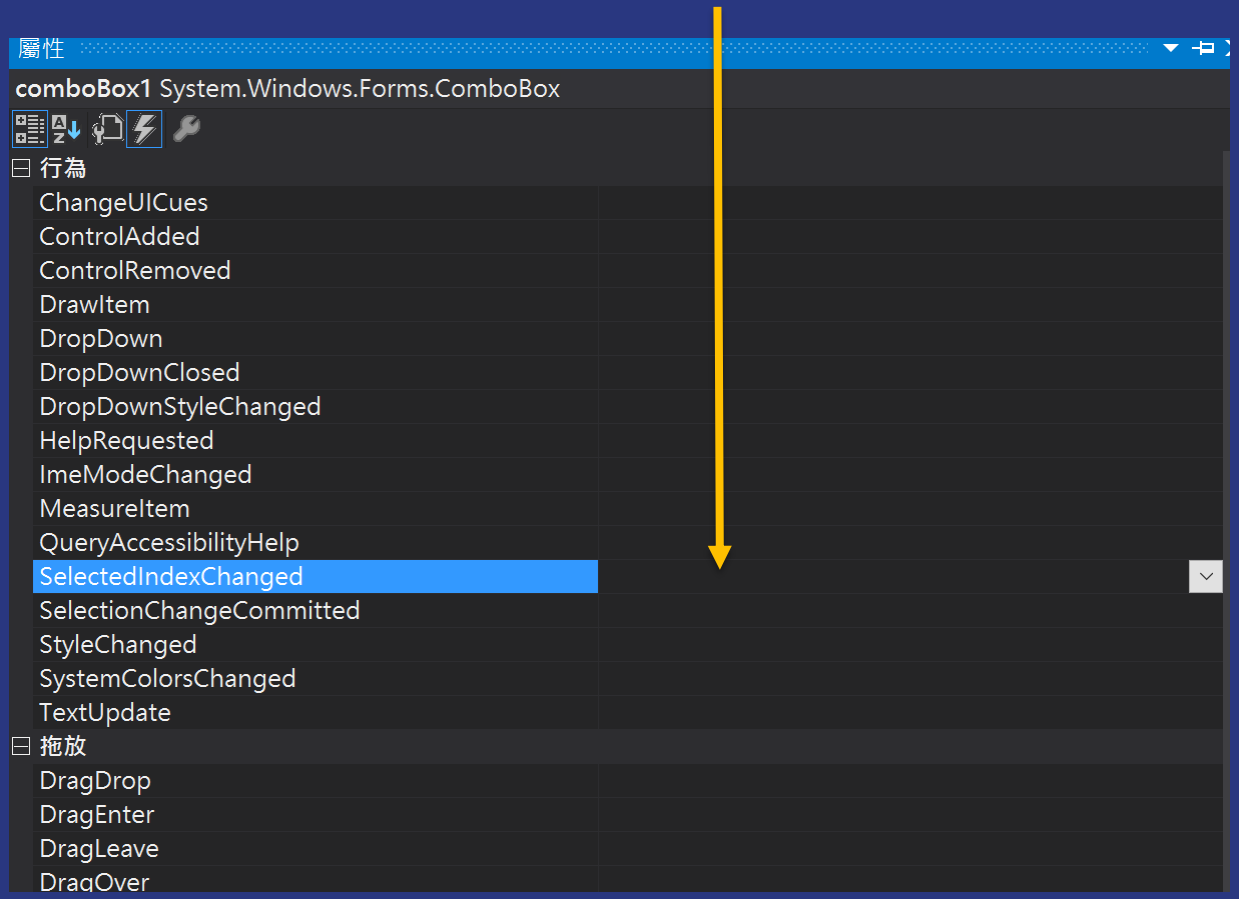
選擇畫面上的 ComboBox
開啟屬性視窗點選那個【閃電】



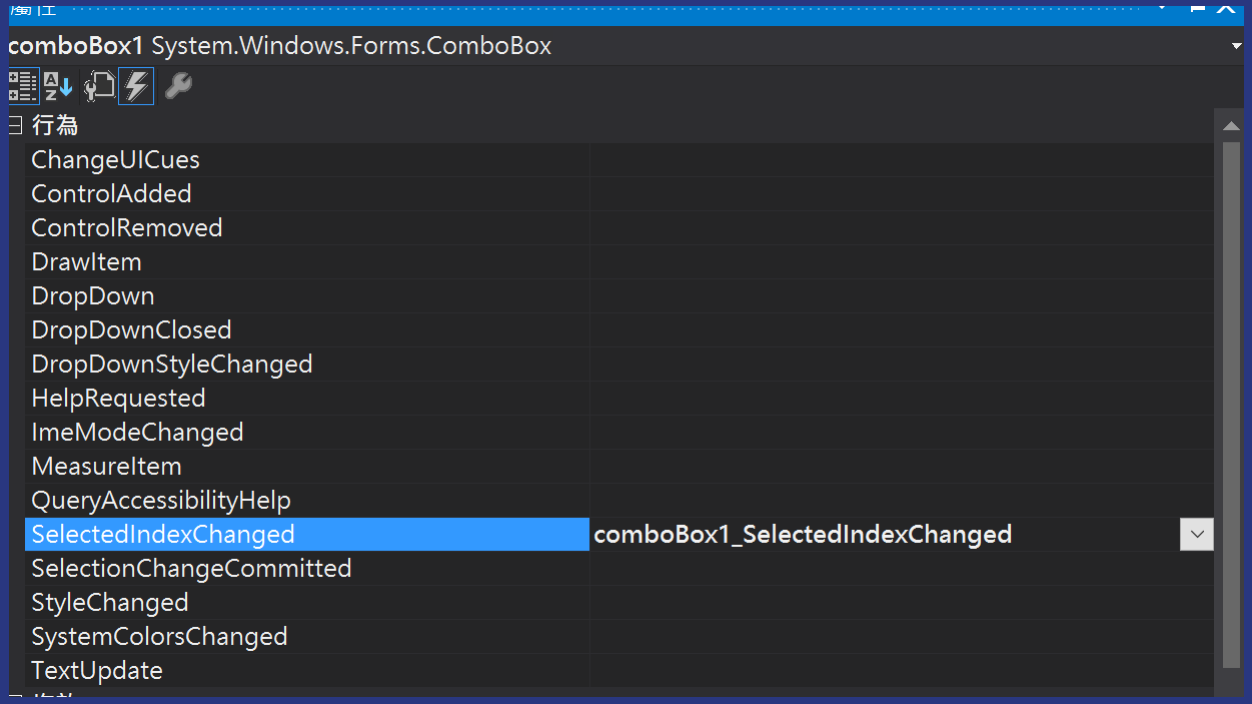
現在的內容就是 ComboBox 的所有事件



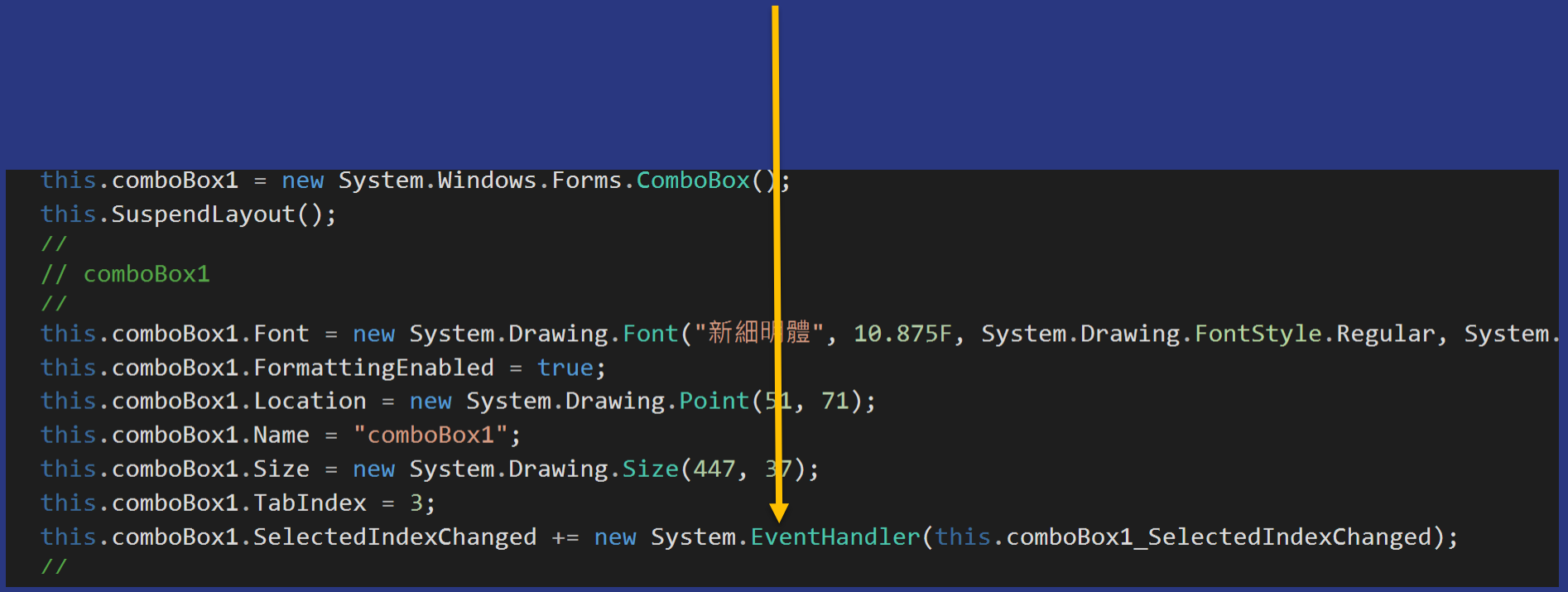
找到【SelectedIndexChanged】
並在它右邊的空格雙擊滑鼠左鍵



這個畫面代表 Visual Studio 幫你建立了一個 comboBox1_SelectedIndexChanged 方法並將它掛在 SelectedIndexChanged 事件上

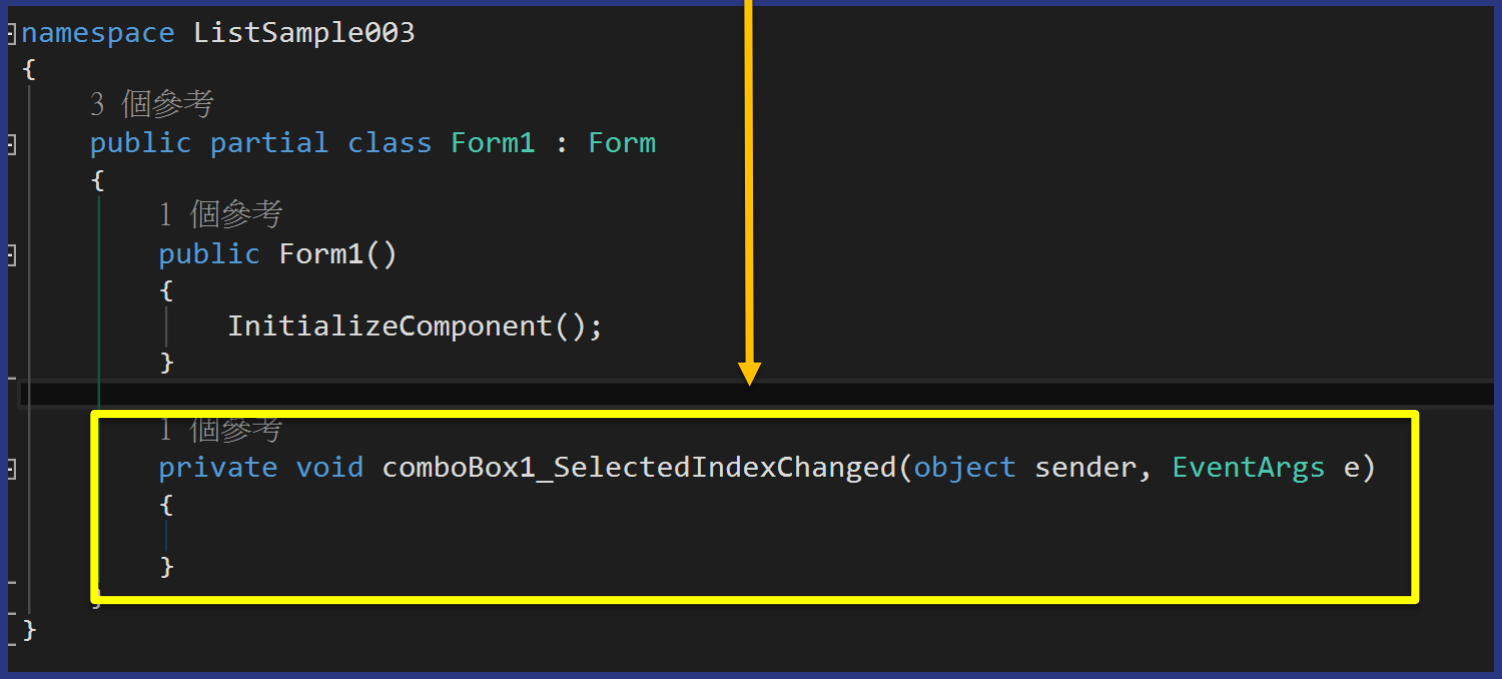


觀察 Form1.Designer.cs 是不是有多這一行？



```
this.comboBox1 = new System.Windows.Forms.ComboBox();
this.SuspendLayout();
//
// comboBox1
//
this.comboBox1.Font = new System.Drawing.Font("新細明體", 10.875F, System.Drawing.FontStyle.Regular, System.
this.comboBox1.FormattingEnabled = true;
this.comboBox1.Location = new System.Drawing.Point(51, 71);
this.comboBox1.Name = "comboBox1";
this.comboBox1.Size = new System.Drawing.Size(447, 37);
this.comboBox1.TabIndex = 3;
this.comboBox1.SelectedIndexChanged += new System.EventHandler(this.comboBox1_SelectedIndexChanged);
//
```

觀察 Form1. cs 是不是有自動產生這個方法



再度建立 MyRectangle 類別

```
class MyRectangle
{
    public string Name { get; set; }
    public int Width { get; set; }
    public int Height { get; set; }

    public int Area
    {
      get { return Width * Height; }
    }
}
```

在 Form1 Class 中宣告一個 List<T>
元素型別：MyRectangle

```
public partial class Form1 : Form
{
    private List<MyRectangle> _list;
```

在 Form1 Class 中建立一個方法來初始化 _list

```
private void CreateList()
{
    _list = new List<MyRectangle>()
    {
        new MyRectangle { Name = "D1", Width = 5, Height = 5 },
        new MyRectangle { Name = "D2", Width = 10, Height = 10 },
        new MyRectangle { Name = "D3", Width = 20, Height = 20 },
        new MyRectangle { Name = "D4", Width = 100, Height = 100 }
    };
}
```

完成必要的初始化程式碼

```
public Form1()
{
    InitializeComponent();
    CreateList();
    SetCombobox();
}

private void CreateList()
{
    _list = new List<MyRectangle>()
    {
        new MyRectangle { Name = "D1", Width = 5, Height = 5 },
        new MyRectangle { Name = "D2", Width = 10, Height = 10 },
        new MyRectangle { Name = "D3", Width = 20, Height = 20 },
        new MyRectangle { Name = "D4", Width = 100, Height = 100 }
    };
}
private void SetCombobox()
{
    comboBox1.DataSource = _list;
    comboBox1.DisplayMember = "Name";
    comboBox1.ValueMember = "Area";
}
```

撰寫委派方法的內容

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    // 來點變化，從參數中取得觸發該事件的物件
    ComboBox theComboBox = (ComboBox)sender;
    int index = theComboBox.SelectedIndex;
    MyRectangle item = _list[index];
    MessageBox.Show($"取得索引 {index} 的面積為: {item.Area}");
}
```

執行

討論

- 你執行後有沒有發現哪裡不對勁？

LAB

再來玩一次

在 ListSamples 加入新專案

- 方案名稱：ListSamples
- 專案名稱：ListSample004
- 範本：Windows Forms Application
- 畫面配置同 ListSample003

再度建立 MyRectangle 類別

```
class MyRectangle
{
    public string Name { get; set; }
    public int Width { get; set; }
    public int Height { get; set; }

    public int Area
    {
      get { return Width * Height; }
    }
}
```

在 Form1 Class 中宣告一個 List<T>
元素型別：MyRectangle

```
public partial class Form1 : Form
{
    private List<MyRectangle> _list;
```

完成必要的初始化程式碼

```
public Form1()
{
    InitializeComponent();
    CreateList();
    SetCombobox();
}

private void CreateList()
{
    _list = new List<MyRectangle>()
    {
        new MyRectangle { Name = "D1", Width = 5, Height = 5 },
        new MyRectangle { Name = "D2", Width = 10, Height = 10 },
        new MyRectangle { Name = "D3", Width = 20, Height = 20 },
        new MyRectangle { Name = "D4", Width = 100, Height = 100 }
    };
}
private void SetCombobox()
{
    comboBox1.DataSource = _list;
    comboBox1.DisplayMember = "Name";
    comboBox1.ValueMember = "Area";
}
```

這一次，要在指派資料給 comboBox1 後
才將方法掛在事件上

```
public Form1()
 {
     InitializeComponent();
     CreateList();
     SetCombobox();
     comboBox1.SelectedIndexChanged +=
 }
```

請先加上以上的程式碼，在 += 後面按下【TAB】鍵

是不是看到這樣的結果？

```
public Form1()
{
    InitializeComponent();
    CreateList();
    SetCombobox();
    comboBox1.SelectedIndexChanged += ComboBox1_SelectedIndexChanged;
}

private void ComboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    throw new NotImplementedException();
}
```

完成委派方法的內容

```
private void ComboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    ComboBox theComboBox = (ComboBox)sender;
    int index = theComboBox.SelectedIndex;
    MyRectangle item = _list[index];
    MessageBox.Show($"取得索引 {index} 的面積為: {item.Area}");
}
```

執行

一些小提醒

- 有些時候，事件委派方法指派給事件的時機會影響程式的正確性。
- 事件委派方法也是可以移除的

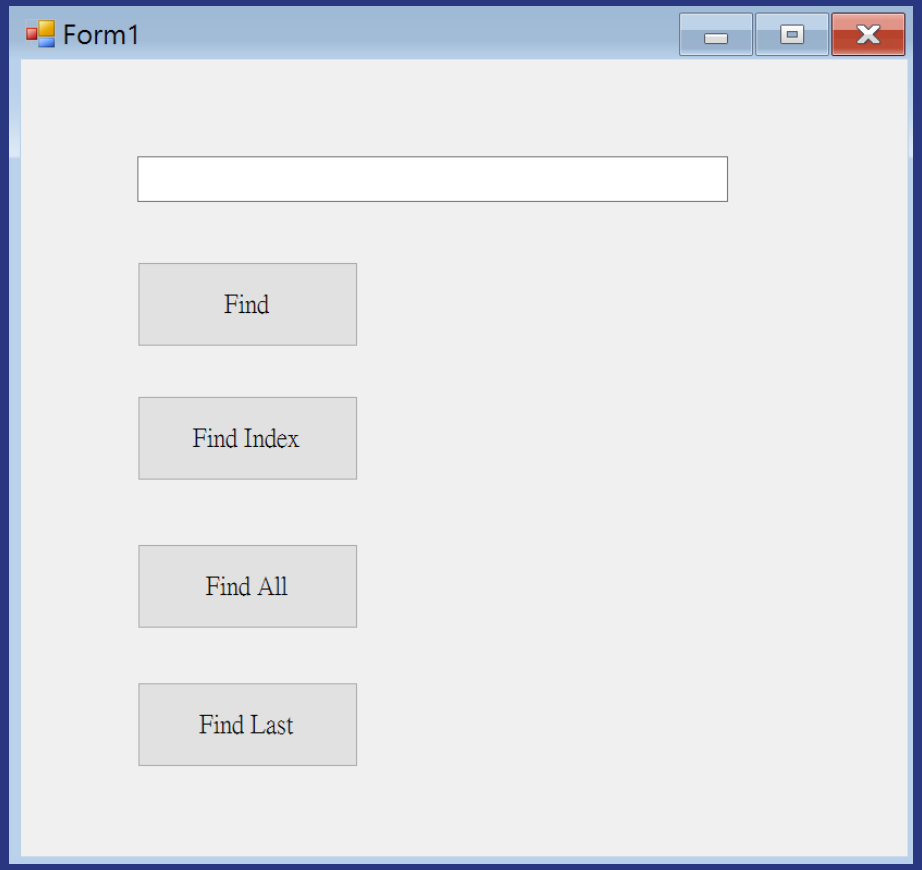
LAB

搜尋 List<T>

在 ListSamples 加入新專案

- 方案名稱：ListSamples
- 專案名稱：ListSample005
- 範本：Windows Forms Application

畫面配置



一個 TextBox
四個 Button
(1) Find
(2) Find Index
(3) Find All
(4) Find Last

完成必要的初始化程式碼

```
private List<string> _list;
public Form1()
{
    InitializeComponent();
    CreateList();
}

private void CreateList()
{
    _list = new List<string>()
    {
      "Dog", "Cat", "Monkey",
      "Fly",  "Donkey", "Dog2"
     };

}
```

為每個 Button 的 Click 事件掛上委派方法

```
private void button1_Click(object sender, EventArgs e)
{
    string result = _list.Find((x) => x == textBox1.Text);
    MessageBox.Show($"Find: {result}");

}
private void button2_Click(object sender, EventArgs e)
{
    int index = _list.FindIndex((x) => x == textBox1.Text);
    MessageBox.Show($"Find Index: {index}");
}
```

請查詢 Microsoft Docs，看一下 String.Contains 在做甚麼？

```
private void button3_Click(object sender, EventArgs e)
{
    List<string> results = _list.FindAll((x) => x.Contains(textBox1.Text));
    string result = string.Empty;
    foreach (string item in results )
    {
        result = result + item + ",";
    }
    MessageBox.Show($"Find All: {result}");
}

private void button4_Click(object sender, EventArgs e)
{
    string result = _list.FindLast((x) => x.Contains(textBox1.Text));
    MessageBox.Show($"Find Last: {result}" );
}
```

執行

討論

- 剛剛執行的四個不同的搜尋方式，分別代表甚麼樣的意義

ListBox 控制項

- 表示要顯示項目清單的 Windows 控制項。
- 很類似 ComboBox，但是它會直接顯示選項清單，而且可以多選。
- 當清單比畫面配置的高度還要多的時候，會出現捲軸。

ListBox 的 SelectionMode 屬性

- Microsoft Docs ListBox.SelectionMode 屬性
- MultiExtended
  - 可選取多個項目，而且使用者可以使用 SHIFT、 CTRL 和方向鍵做選擇
- MultiSimple
  - 可以選取多個項目
- None
  - 無法選取項目
- One
  - 單選

LAB

各種不同的 SelectionMode

- 這次不需要寫程式
- 直接開啟 ListBoxSelectionModeSample
- 執行這個範例
- 對照前面關於 SelectionMode 屬性的說明，分別操作這四個 ListBox

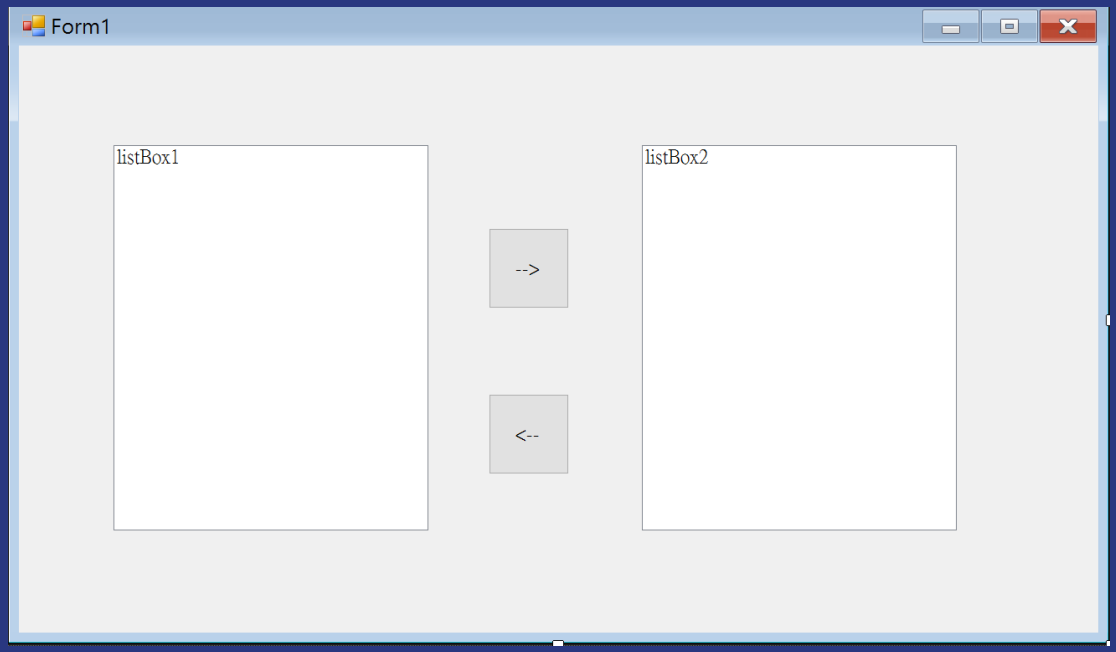
LAB

在兩個 ListBox 中移動資料

在 ListSamples 加入新專案

- 方案名稱：ListSamples
- 專案名稱：ListSample006
- 範本：Windows Forms Application

畫面配置



兩個 ListBox
兩個 Button

資料來源

- 有兩個 ListBox，所以會有兩個資料來源

```
public partial class Form1 : Form
{
    private List<string> _leftList;
    private List<string> _rightList;
    public Form1()
    {
        InitializeComponent();
    }
}
```

初始化這兩個 List<string>

```
private void CreateList()
{
    _leftList = new List<string>
    {
        "A","B","C","D"
    };
    _rightList = new List<string>();
}
```

設定兩個 ListBox 的 SelectionMode

```
private void SetListBoxDataSource()
{
    listBox1.SelectionMode = SelectionMode.One;
    listBox2.SelectionMode = SelectionMode.One;

}
```

設定兩個 ListBox 的資料來源

```
private void ChangeData()
{
    listBox1.DataSource = null;
    listBox2.DataSource = null;
    listBox1.DataSource = _leftList;
    listBox2.DataSource = _rightList;
}
```

在建構式呼叫剛剛那些方法，以完成初始化的動作

```
public Form1()
{
    InitializeComponent();
    CreateList();
    SetListBoxDataSource();
    ChangeData();
}
```

撰寫 button1 的 Click 事件委派方法

```
/// <summary>
/// 從左移到右
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void button1_Click(object sender, EventArgs e)
{
    if (listBox1.SelectedItem != null)
    {
        string item = (string)listBox1.SelectedItem;
        _leftList.Remove(item);
        _rightList.Add(item);
        ChangeData();
    }
}
```

撰寫 button2 的 Click 事件委派方法

```
/// <summary>
/// 從右移到左
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void button2_Click(object sender, EventArgs e)
{
    if (listBox2.SelectedItem != null)
    {
        string item = (string)listBox2.SelectedItem;
        _rightList.Remove(item);
        _leftList.Add(item);
        ChangeData();
    }
}
```

執行

討論

- 剛剛在 ChangeData 方法中為什麼要先設定兩個 ListBox 的 DataSource = null 呢
- 試著把那兩行移除，然後執行看看結果會如何

以下是你們的回家作業

農夫渡河

- 農夫要帶著狼、羊、菜過河。
- 小船不夠大，因此農夫每次只能帶一樣東西過河。
- 當農夫在的時候，狼、羊、菜都不會有事情。
- 當農夫不在時，狼會吃羊，羊會吃菜

別急著寫程式

- 在這個階段，我們來學習如何從分析需求開始到完成程式碼的過程。
- 先構思這個程式需要甚麼，如何安排邏輯
  - 畫面如何安排
  - 需要甚麼樣的資料來源
  - 甚麼樣的狀況可以過河的邏輯
  - 甚麼樣的狀況會導致遊戲失敗的邏輯
  - 甚麼樣的狀況會讓遊戲結果是成功的

討論

- 請各小組討論出關於以下需求的可行的方案，並寫下它
  - 畫面如何安排
  - 需要甚麼樣的資料來源
  - 甚麼樣的狀況可以過河的邏輯

LAB

完成剛剛所設計的需求程式碼

討論

- 請各小組討論出關於以下需求的可行的方案，並寫下它
  - 甚麼樣的狀況會導致遊戲失敗的邏輯
  - 甚麼樣的狀況會讓遊戲結果是成功的

LAB

完成剛剛所設計的需求程式碼

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