# 第 9 章簡易通訊錄

**簡介：**

這一章要製作一個簡易的通訊錄，技術面來說就是一個精簡的資料庫程式。資料庫的應用範圍極廣，功能需求也非常複雜，如果將完整的理論架構與標準作業程序說清楚大概就是一整個「資訊管理系」的課程了！所幸我們的 C#程式提供了許多的工具，讓我們可以相當省力的製作出資料庫程式，只要善用它們，其實不必讀完整個資管系應該也可以寫出很多資料庫應用程式！

不過，即使如此，對於資料庫運作的理論架構與概念還是必須盡量弄清楚，不然會連現成的物件或功能都找不到的！所以本章的程式碼雖然很少，但是操作程序與理論說明卻很多，希望這一切都是值得的！

**9-1 C#資料庫操作概念簡述**

[資料庫引擎]

要寫資料庫程式當然必須先有一個資料庫。廣義來說，任何資料檔案簡單到一個記事本寫出的文字檔都可以視為資料庫。但是在實務上，資料庫必須可以讓一般的高階程式語言(如 VB、C#或 Java 等等)可以用程式碼對它作檢視、更新、新增與刪除等基本操作，這四個動作所對應的英文關鍵字就是 Select, Update, Insert 以及 Delete。為此目的，各種資料庫都必須提供一個中介軟體，一般稱為「資料庫引擎」，讓高階程式語言下達如 Select 等簡單指令時可以對資料庫作出複雜而正確的動作。因為每種資料庫的格式不同，所以資料庫引擎也各自不同，至今為止資料庫引擎還是必不可少的東西！

[何謂 SQL？]

在早期，資料庫的發展相當混亂，各種資料庫操作動作的語法都各自不同，讓程式師非常困擾，稍後業界漸漸統一了高階語言層級的操作語法，稱之為 SQL，唸法類似中文的「西擴」，英文全名是 **S**tructured **Q**uery **L**anguage，翻譯是「結構化的查詢語言」。上面提到的 Select 等四個基本操作指令就是 SQL 最常用的語法。簡單說，就是現在的程式師面對「不同」的資料庫，可以用「相同」的語法下指令操作。話雖如此，程式設計過程之中還是不能省略指定資料庫種類(或者說指定「資料庫引擎」)的步驟，這常常是讓初學者感到迷惑的第一道關卡。

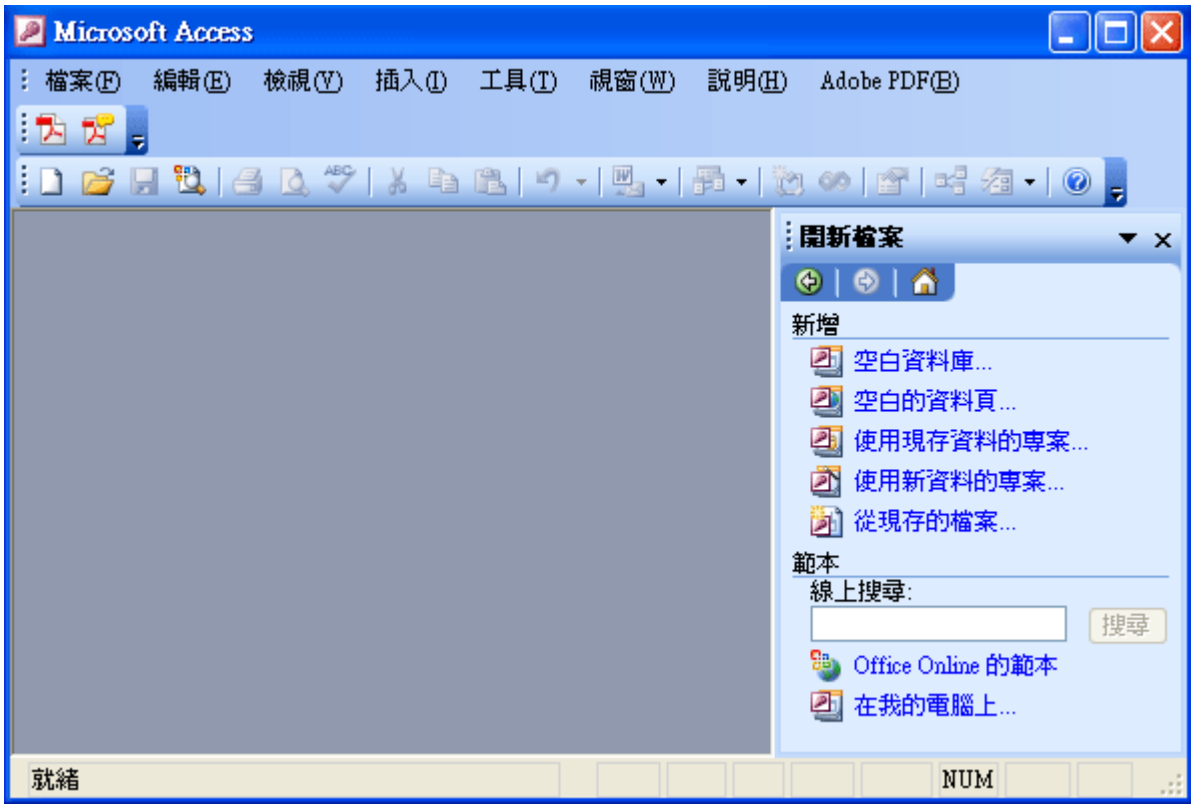
不過，真正讓多數設計者頭昏腦脹，最後決定放棄寫資料庫程式的原因卻是微軟公司非常體貼設計者，努力讓程式師連 SQL 語法都不必寫，而設計出來的一些資料庫連結物件！最主要是：DataSet，BindingSource 以及 Adapter 等三個物件。它們雖然將複雜的資料操作動作包括 SQL 語法封裝起來，只要作一些設定就可以操作資料庫。但是如果你腦袋裡根本沒有資料庫運作的清晰概念，這些物件只會讓你更加迷惑而已！稍後本章會詳細的說明這些物件，讓大家可以精準的使用它們。說實話，之前我看遍市面上

的所有的程式語言書籍範例，對此都還是一知半解！直到自己帶領學生實際開了發幾個實務專案之後才終於完全掌握，希望本書的說明會因此與一般書籍有所不同。

## 9-2 建立一個 Access 資料庫

[Access 資料庫建置程序]

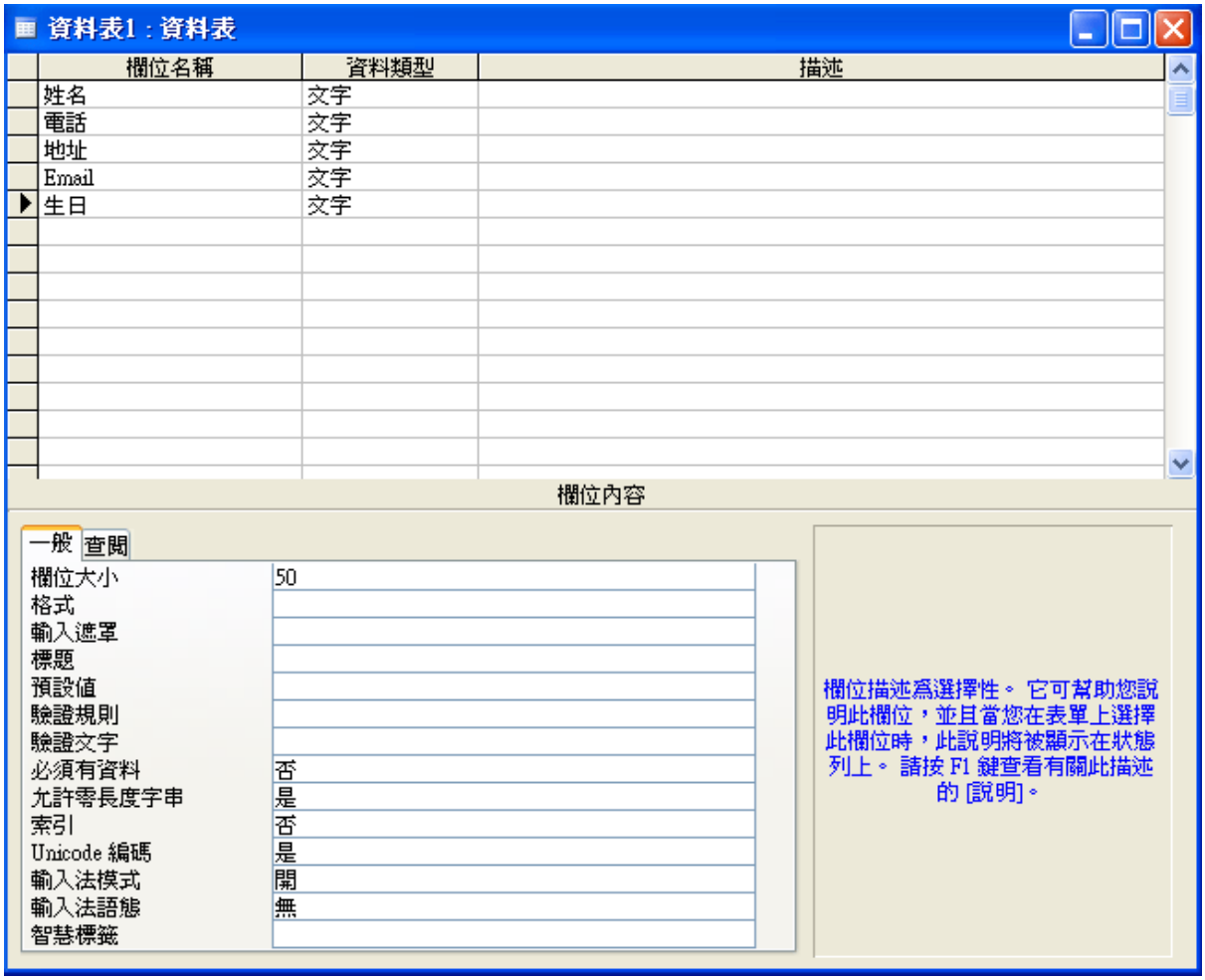
資料庫種類很多，一般來說，最簡單好用的是微軟 Office 系列軟體中提供的 Access 資料庫。雖然它被視為「簡易」資料庫，但是已經具備了相當完整的功能，足夠應付大多數的專案開發需求，不是僅供初學者用的玩具而已！現在請打開程式集中 Office 選項下的 Access(2003 版)軟體，選擇開新檔案，出現畫面如下，請點選「空白資料庫」，會出現將資料庫放在哪裡的選項，請選擇一個磁碟位置，預設檔案名稱是 db1.mdb，此名可以不改，除非與現有檔案衝突。




選好檔案之後出現畫面如下，較新的資料庫基本結構都是一個資料庫檔案，其中包含一或多個資料表，以及其他有關資料連結與處理過程的檔案。所以建立好資料庫檔案之後就是要在裡面建立一個資料表了！請選擇預設的「使用設計檢視建立資料表」：


[建置欄位]

出現如下畫面之後請建立資料表的項目如下：

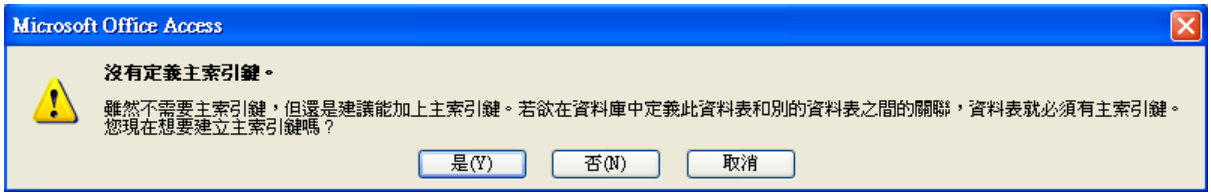



上面的視窗其實還有『資料類型』可以修改，預設都是文字，欄位大小是 50，表示這個欄位可以容納 50 個字，請注意到：這是指 Unicode(雙位元)的字元，中英文都會用兩個 Byte 儲存，所以確實可以容納 50 個中文字！如名字等欄位當然不會用到那麼長，但住址欄卻又可能不夠，你都可以在此加以調整。實際應用時很多欄位還必須指定為數字等資料，在此不作詳述，知道就好。使用預設值不作調整即可繼續本單元課程。

接下來應該要將此資料表的定義儲存，你可以按功能表的「儲存檔案」功能，也可以按下工具列的磁碟片圖示，會出現如下視窗，請填入「通訊錄」。

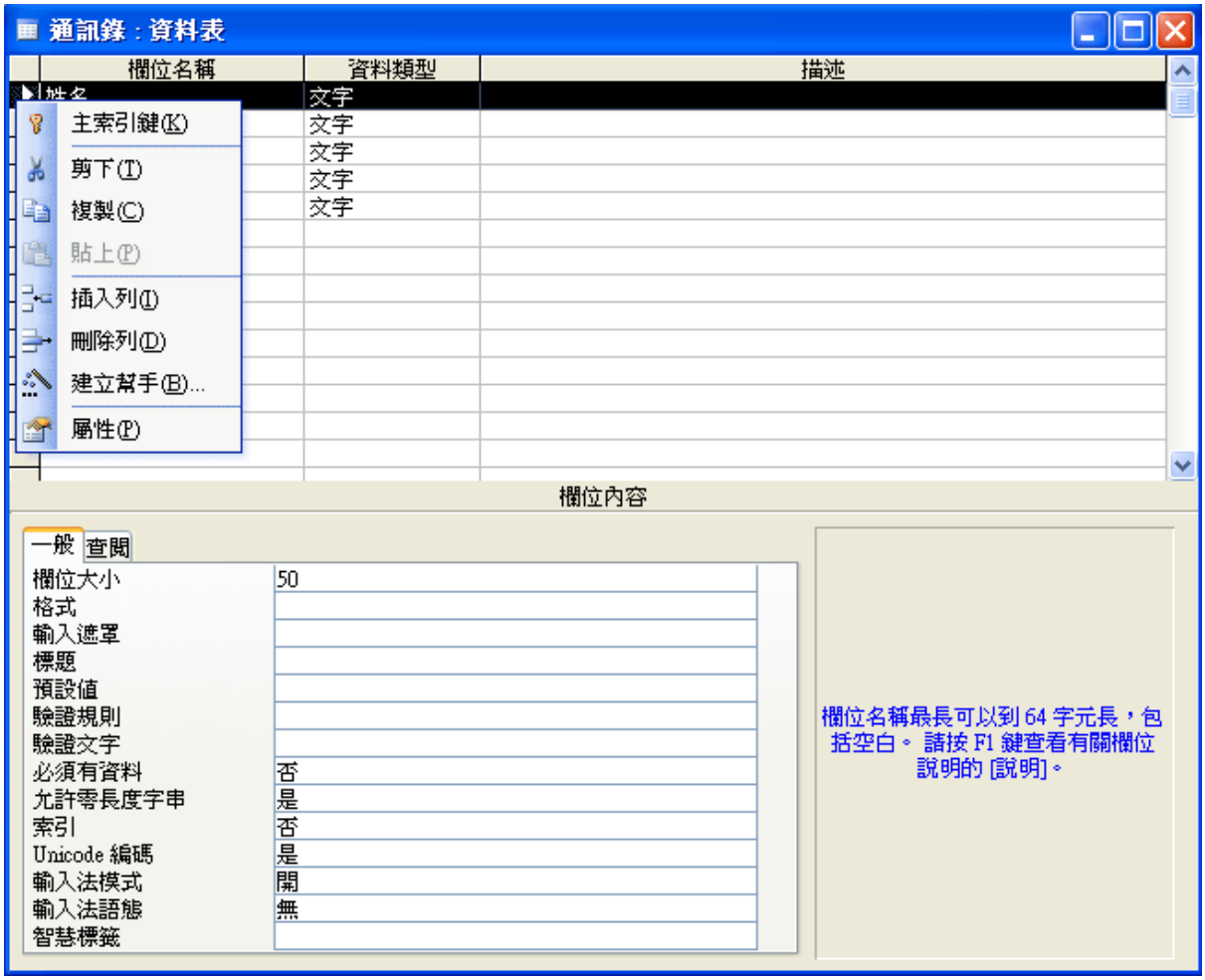

[建立主索引]

按下確定鍵應該會出現下面的警告標示：




如果選「是」！你的資料表中會多一個自動編號的項目作為「主索引鍵」，如果選否就是自己必須選擇一個項目作為主索引鍵。在此請選擇「否」，我們要自己選擇將「姓名」欄作為主索引鍵。操作畫面如下，對準「姓名」欄左邊的空格按下右鍵，選擇主索引鍵即可，完成後這個空格會出現一把鑰匙圖示。

這個訊息的意義是：任何一個資料表都應該有一個資料「唯一」(不重複)的欄位。譬如一間學校的學生資料表「主索引鍵」應該就是「學號」了！不應該有學生的學號是重複的，即使同名同姓也應該有不同的學號！這樣可以幫助資料庫進行很多辨識個別資料的工作，如果你堅持不設定主索引鍵也是可以的，不過可能在日後的操作會產生一些錯誤與限制，或者資料庫的效率變差。




[建置資料]

設好主索引鍵後請儲存資料表，再到視窗左上角，找到並按下「檢視」圖示切換到

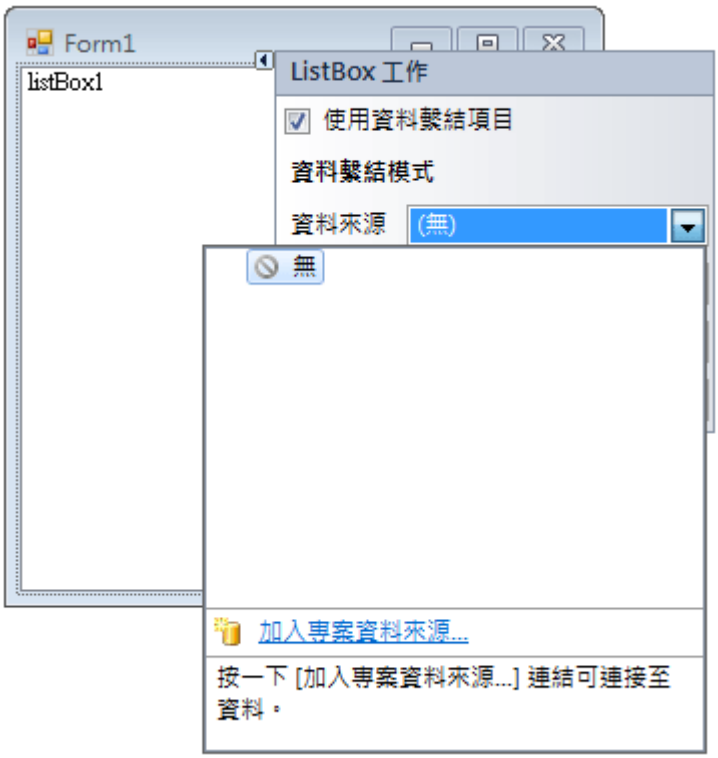
類似 Excel 外觀的資料表格，填入幾筆模擬資料，最後存檔並關閉 Access。如下圖：

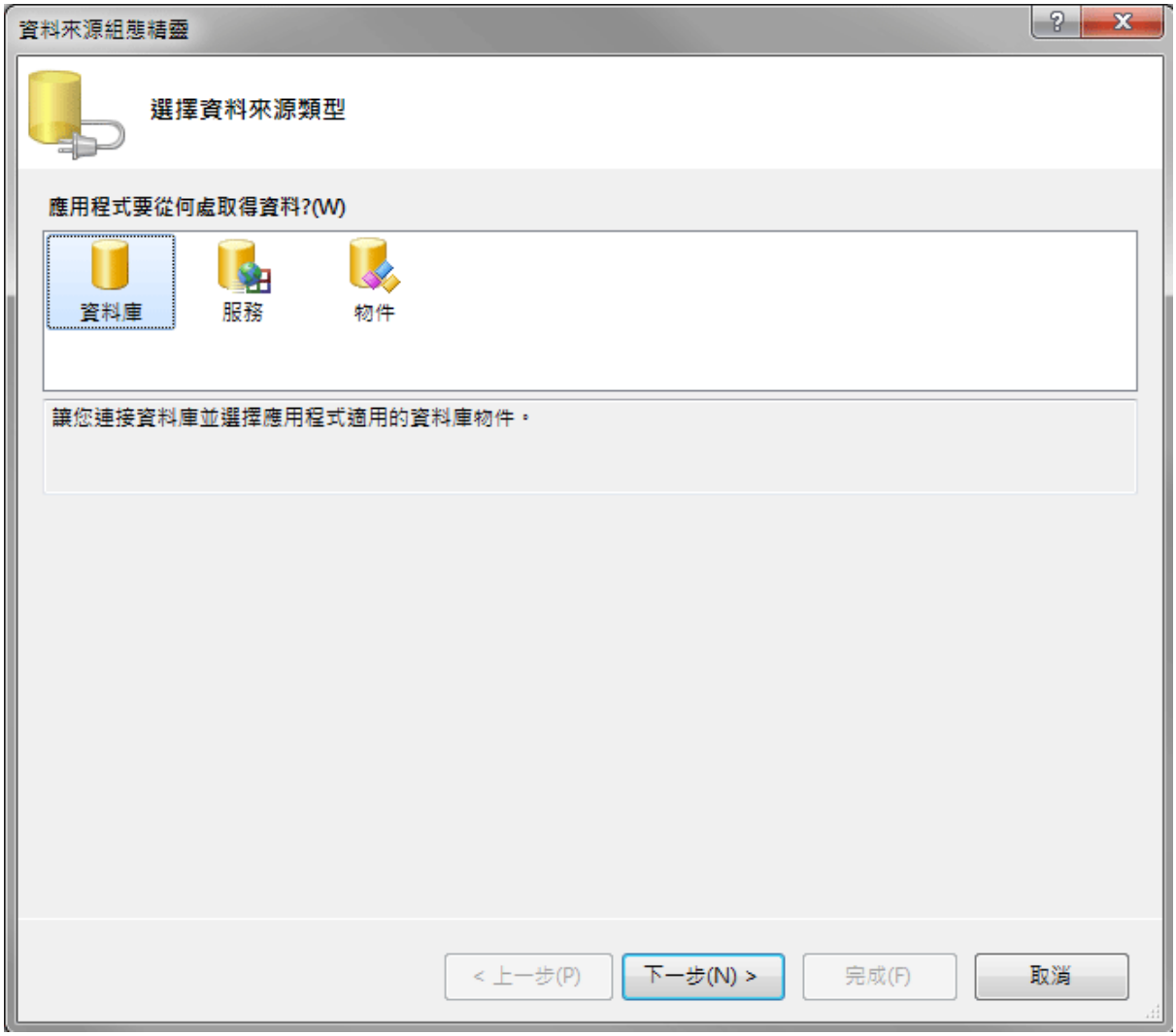


## 9-3 建立資料連結

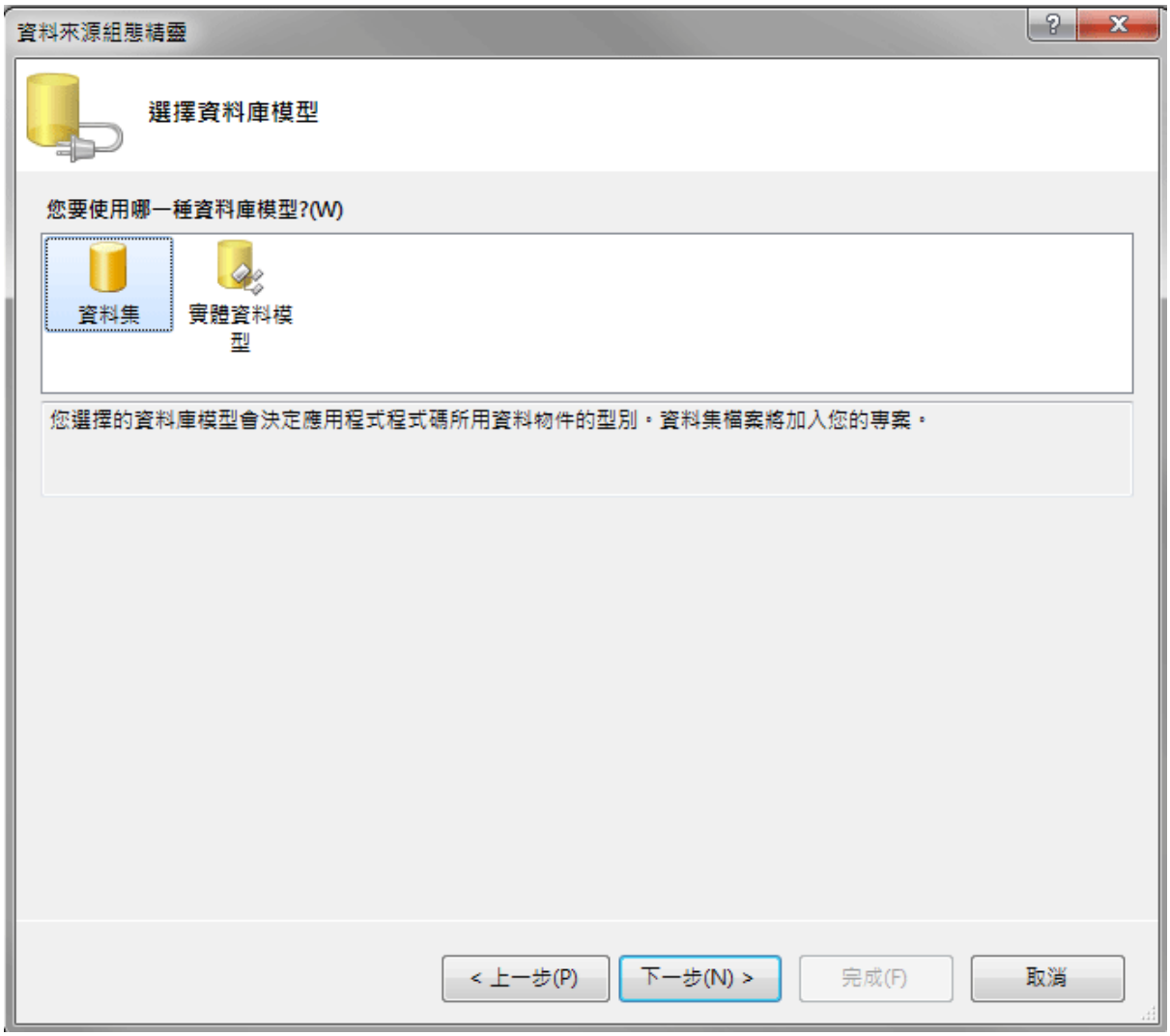
[物件的資料繫結]

資料庫與資料表有了，接下來就是要將它顯示在我們的程式之中了！請開啟一個新的 C#專案，在表單上置入一個 listBox1 物件，並將 Dock 屬性設定為靠左邊(Left)。請點選 listBox1 右上角的那個小三角形，出現一個小選單後勾選「使用資料繫結項目」，再點選「資料來源」選項右邊的向下箭頭，然後選擇「加入專案資料來源」，此時會跳出另一個包含資料庫選項的視窗，如下圖：

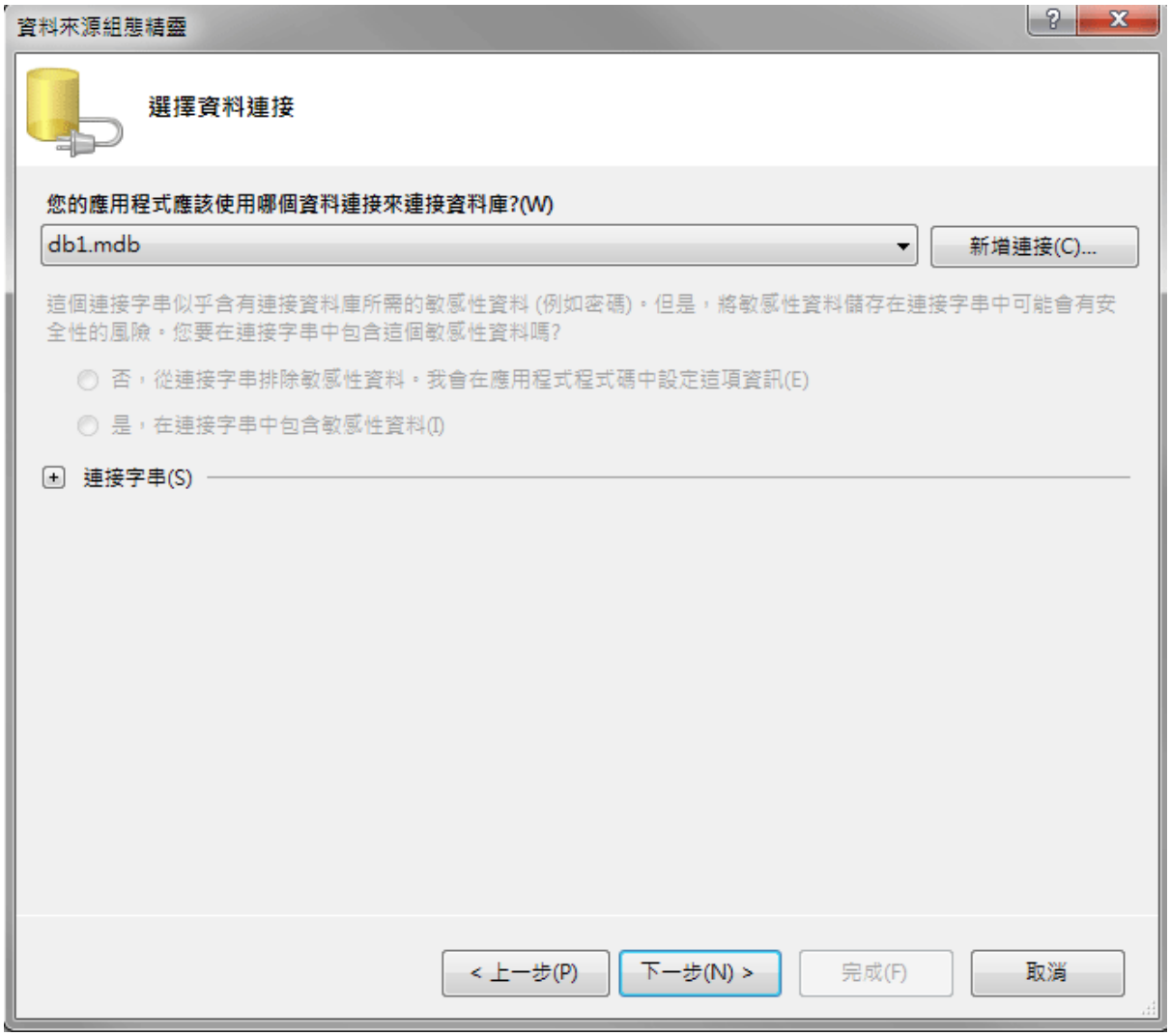

Form1
listBox1
ListBox 工作
使用資料繫結項目
資料繫結模式
資料來源
(無)
無
加入專案資料來源...
按一下 [加入專案資料來源...] 連結可連接至資料。



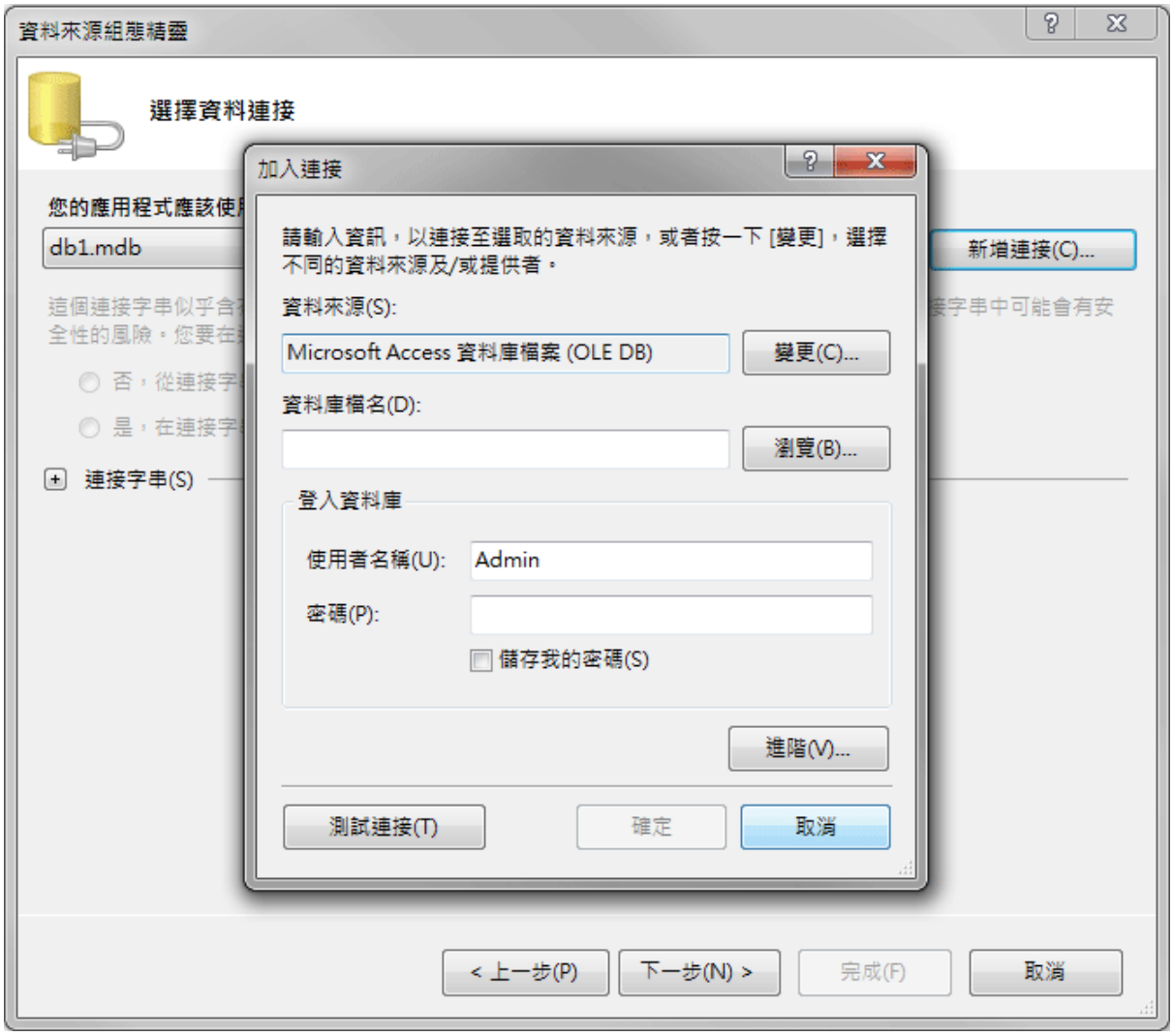

此處資料庫之外還有服務與物件兩個選項，服務通常是指網路服務，可以連結到一些特殊的網路服務網站，查詢資料或協助作計算等服務，物件則泛指非典型資料庫的資料檔案。請保持預設的「資料庫」選項按下一步，視窗內容改變後按下「新增連結」按鈕，會出現另一個小視窗如下圖：



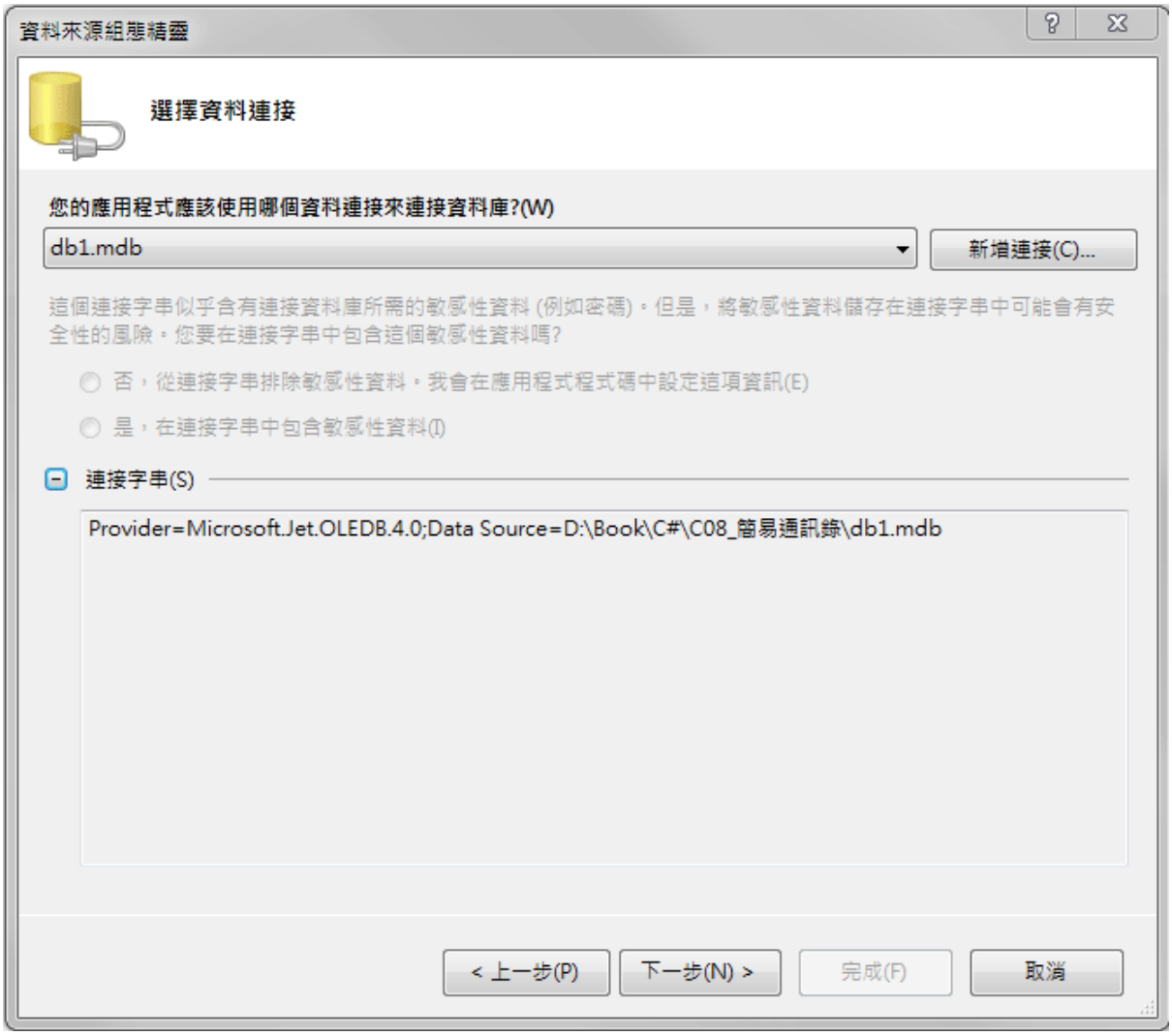

此處使用預設值，按下一步！




按新增連結！出現下一畫面：



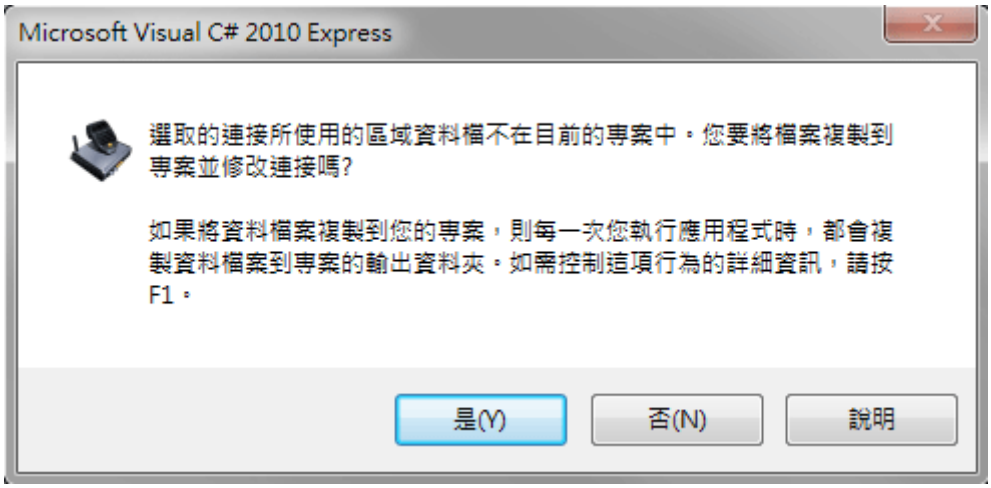

在上圖「變更」按鈕按下後可以選擇不同的資料庫種類，請選擇 Access 資料庫檔案(OLE DB)。在此我們只要知道自己使用哪種資料庫的檔案，引導我們的程式精靈就會自動找出合用的資料庫引擎軟體，接下來就是按下瀏覽選擇你的資料庫檔案了！因為我們並沒有在剛剛的 Access 資料庫設定密碼，所以使用者名稱與密碼不必去動它，可以按一下測試連接，應該會跳出一個「測試連接成功」的訊息。之後請按下確定繼續進行。




[資料庫引擎就是它了！]

回到如上畫面點開連接字串可以看到 Provider=Microsoft.Jet.OLEDB.4.0 這就是所謂的資料庫引擎軟體，也稱作資料提供者(Provider)，Data Source=…就是資料庫檔案了！

繼續按確定及下一步會出現如下的訊息視窗：

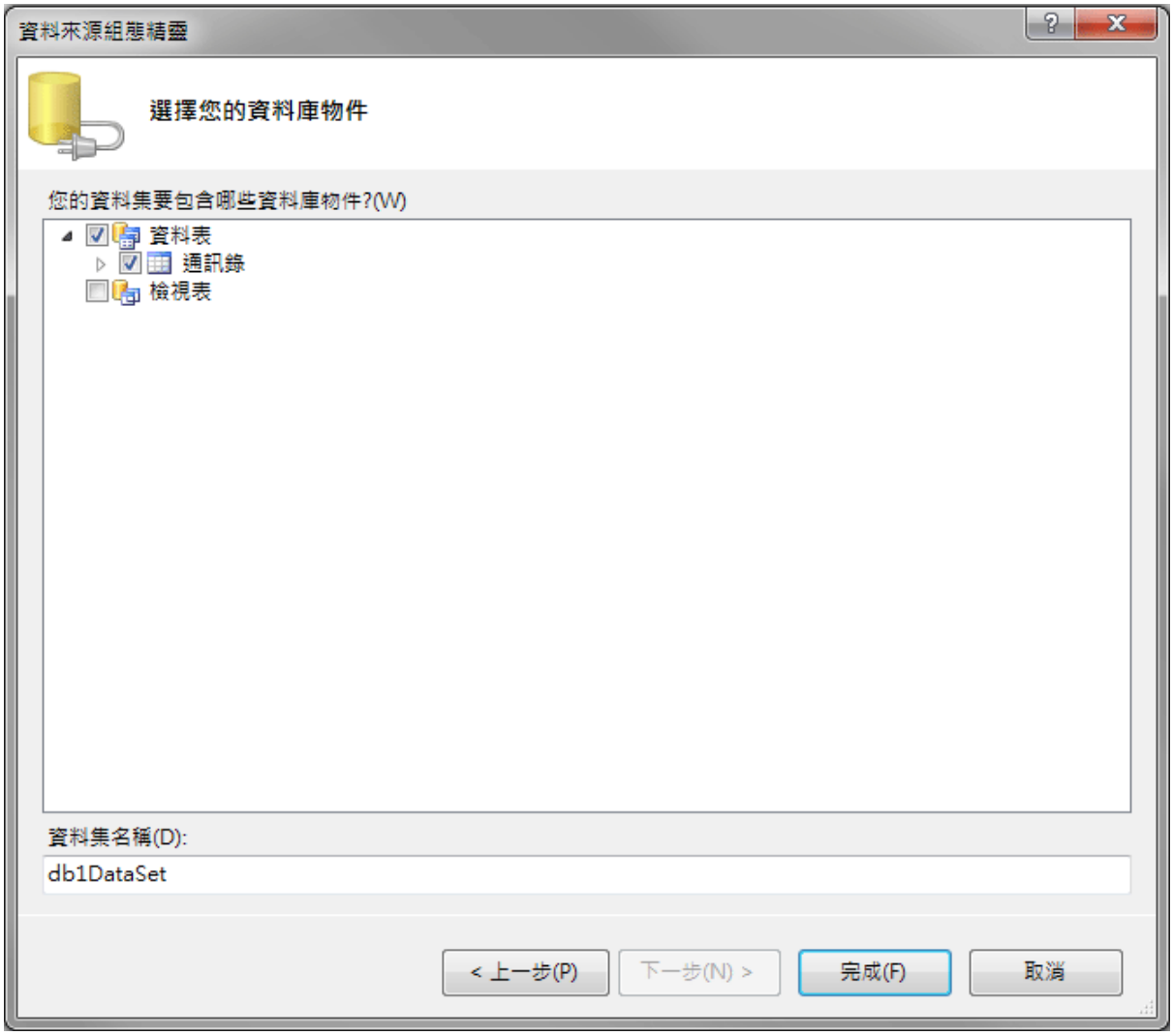



這個訊息問你是不是要將資料庫拷貝到專案之內？這個設計的原因應該是開發程式時害怕測試過程損毀(弄亂)了寶貴的範例資料庫檔案，所以讓你選擇可以用資料庫備份到專案檔內使用，就是上面「是」的選項。

[建議不要將資料庫附置到專案內]

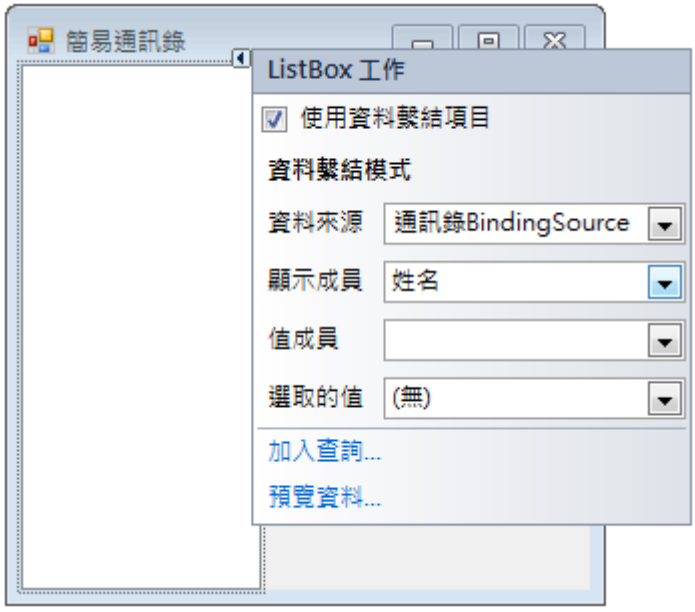
但是使用這種預設方式的問題很大！首先是每次執行程式時資料庫就會自動復原！你的程式輸出的資料庫(更改後)其實會在專案的 bin 目錄下，但是其上層目錄又有個原始備份，每次開始偵錯時又會從原始拷貝狀態開始。因此我們寫程式改資料時常常會迷惑程式是不是真的有效的改道了資料？為什麼剛剛改好，等一下又「變」回來呢？等到真的要將程式上線使用時當然也不能讓資料庫這樣子運作，永遠會**自動復原**的資料庫還寫程式幹嘛？所以最後還是要大幅改變資料庫的位置與連結方式才能真的使用！我認為發明這個機制的人不是太笨，就是想惡意阻擋外行人太快學會資料庫程式技術到市場上賺錢。所以在此絕對應該選「**否**」！如果怕損壞原來的資料庫自己拷貝一份就好，千萬不要在上面視窗選擇預設的「是」！那絕對是一場惡夢的開始。

之後經過幾個必然的「下一步」會引導你到如下的畫面，你應該選取資料表「通訊錄」，並按下完成鍵。

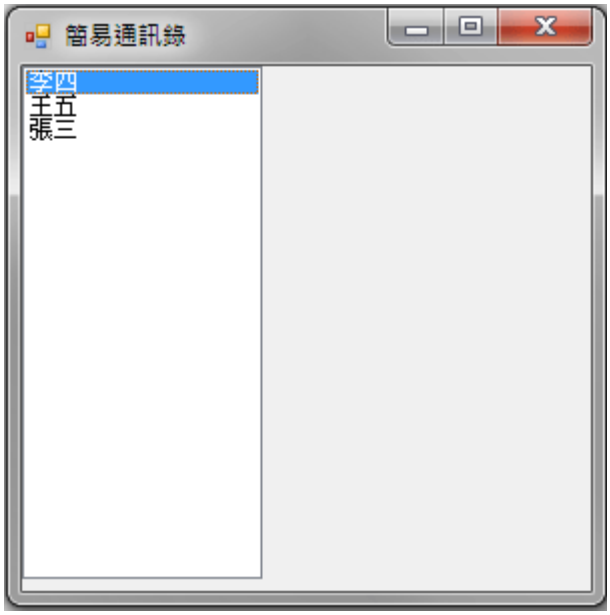



[顯示資料項目與值成員]

接下來又會回到 listBox1 的設定畫面，我們可以看到「資料來源」處出現了一個「**通訊錄 BindingSource**」的項目，表示本清單(listBox1)的資料由這個物件提供。但是下面的「顯示成員」必須由右邊的箭頭下拉選擇「姓名」，這就是程式執行時會出現在清單上的項目(對應於 Text 屬性)，底下還有一個「值成員」(Value)，是會隱藏在顯示項目之下的值，使用者看不見，但是程式師需要時可以用程式叫用這個值，譬如某人的姓名顯示於上，他的電話或地址則可以用「值成員」(對應於 Value 屬性)隱藏於物件內部，可以供程式設計時使用。



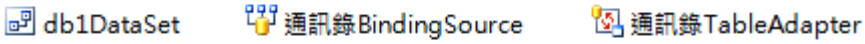

執行程式出現畫面應該如下。至此，我們已經成功的讓程式的一個表單物件連結資料庫，並在執行階段可以顯示資料庫內的部分資料了！




**9-4 資料連結物件解析**

[DataSet, BindingSource, Adapter]

或許前面你跟著老師點選資料連結點得很高興，一下子就連好了！又可以抽空打電動去了！但是提醒你，如果你不認真去了解過程，最終還是一場空！對資料庫還是一無所知。現在就暫停腳步，看看剛剛過程中 C#程式已經幫我們做的事情吧！或許已經有人注意到，當我們成功連結資料庫物件後，設計頁面下方出現了三個新的東西，如下圖：




事實上不只如此，切換到程式碼設計頁面還會發現專案自動產生了 Form_Load 事件副程式框架，以及一行程式碼，如下：

```
private void Form1_Load(object sender, EventArgs e)
{
// TODO: 這行程式碼會將資料載入 'db1DataSet.通訊錄' 資料表。您可以視需要進行移動或移除。
    this.通訊錄TableAdapter.Fill(this.db1DataSet.通訊錄);
}
```

在這裡有許多故事必須一次說個清楚！在 C#正確地將資料庫的資料連結到表單物件上面的「路途」之中需要經過三個物件，就是：

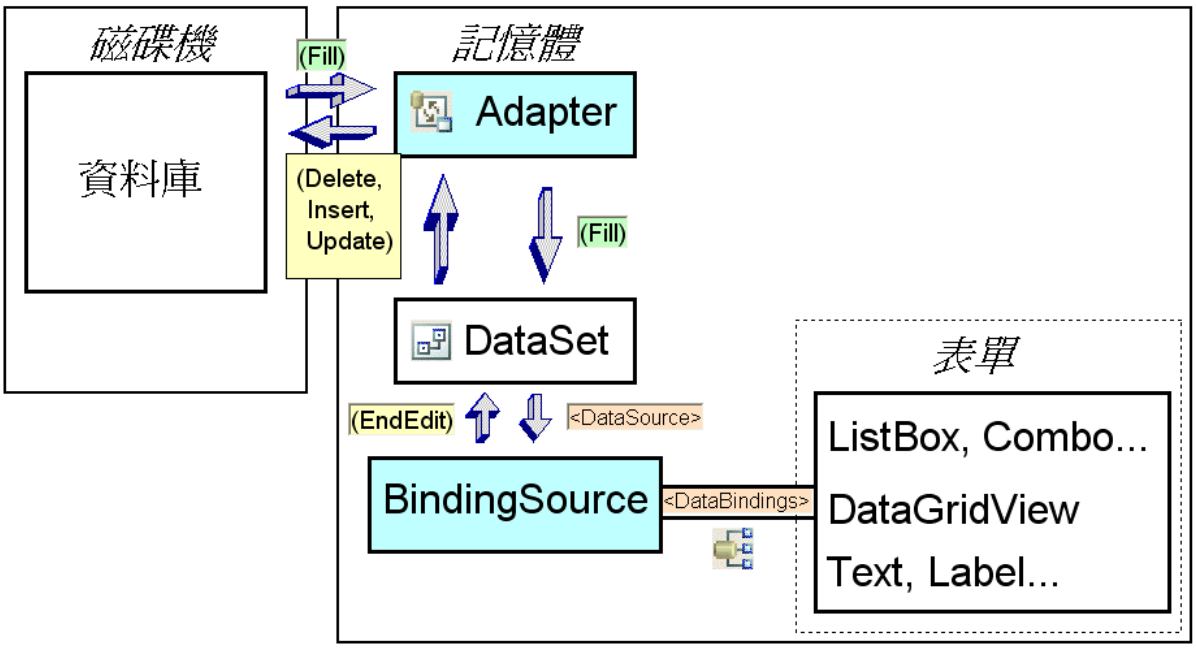
1. **Adapter**：這是專案與磁碟內實體資料庫接觸的最前線，負責封裝 SQL 操作指令，直接負責「讀出」或「寫入」磁碟資料庫的資料。當我們在前面辛苦的設定完要顯示的資料庫資料表時，出現了一個「通訊錄 Table**Adapter**」，就是它將資料從磁碟資料庫檔案讀出來的！稍後如果資料有修改，我們還必須靠它將修改的資料寫回資料庫。要用它讀或寫資料庫還是必須用程式碼去驅動它的！所以 C#很貼心的在程式一開始(Form_Load 事件中)就主動幫我們寫好「讀入資料」的程式碼，使用 Adapte 物件的 Fill 指令將資料讀進專案的某個 **DataSet** 物件裡面。C#也知道你未必在程式一開始就要讀入資料，所以用註解說「**您可以視需要進行移動或移除**」。

2. **DataSet**：這是專案存放從資料庫讀取資料的物件，你可以將它視為資料庫資料的暫存區，為何需要暫存區？原因是效率問題，如果每一個資料處理動作都直接存取磁碟，尤其是網路資料庫還必須長途跋涉，那動作一定很慢。而且如果每個使用者都保持連線資料庫，一定會塞車的！就像吃自助餐吧？一次拿好要吃的菜就到旁邊用餐吧！因此，你在程式介面中操作的資料都會先存到這裡，如有修改時再經過 Adapte 物件回存到實體資料庫中。

3. **BindingSource**：這是一個 DataSet 與表單上面的操作介面物件之間的聯繫物件，譬如剛剛的 listBox1 並不是直接連到 DataSet 物件或 Adapte 物件，而是連到一個「通訊錄 **BindingSource**」的物件。它的作用是因為 DataSet 只是一個靜態的資料表，當表單上的物件，譬如 listBox1 只需要連接 DataSet 資料的一部分，如某資料表中的一個欄位，或者只是其中一個儲存格的時候，BindingSource 物件就可以代為管理這些連結資訊，包括記得目前正處理到哪一筆資料等等。它是與 DataSet 緊密相連的，所以如果除了 listBox1 物件之外，我們也建立了其他的資料顯示介面要連到一樣的資料表，只要將物件的資料繫結(**DataBinding**)屬性指向這個物件就好了！不必再大費周章執行前一節的完整資料庫連結動作，這只會耗費資源，產生一大堆功能重複的 Adapter、DataSet 以及 BindingSource 物件。

[資料連結物件的流程結構]

頭開始昏了嗎？這是一定會的，不過請盡量耐心地弄清楚這幾個物件的意義，這是你學會資料庫程式，打通任督二脈的關鍵時刻！希望下面這張流程圖可以有所幫助：




整個流程是：Adapter 物件經過 Fill 指令將資料庫的部分實體資料載入專案的 DataSet 物件，接著 BindingSource 物件建立表單上面的顯示資料物件與 DataSet 之間的聯繫資訊，你可以去檢視 BindingSouce 物件的 DataSource 屬性就知道它連到哪一個 DataSet；同樣的，表單物件如 listBox1 的屬性 DataBindings 會顯示它連結到哪一個 BindingSource。

當使用者在操作表單介面的時候，譬如將某個顯示出來的原始資料修改了，暫存資料會存在 BindingSource 物件裡面，呼叫 BindingSource 物件的「**EndEdit**」指令時，會將暫存資料回存到 DataSet，之後再呼叫 Adapter 的 **Update**(更新)，**Insert**(新增)或 **Delete**(刪除)指令時就可以將資料真的寫回資料庫了！對於初學者最大的陷阱是：在使用一大堆設定精靈與表單編輯之後，我們常常忘了「EndEdit」與「Update, Delete...」等指令還是必須用程式碼執行！沒有這些關鍵程式碼資料庫就不會真的修改。

如果你已經頭很昏了，下面這一段可以不看。個人有點不滿意的是其實在同樣微軟公司出品的 ASP.NET 架構中，這個與資料庫連結的過程就簡化很多，概略來說，就是以上三個物件(Adapte, DataSet & BindingSource)簡化為一個稱為 DataSource 的物件，為什麼在視窗程式中就不能也簡化一點？總之目前使用視窗程式專案寫資料庫程式還是必須將以上理論架構弄得很清楚才行，寫 ASP.NET 的資料庫程式則相對簡單很多。

[ADO.NET 才是技術核心]

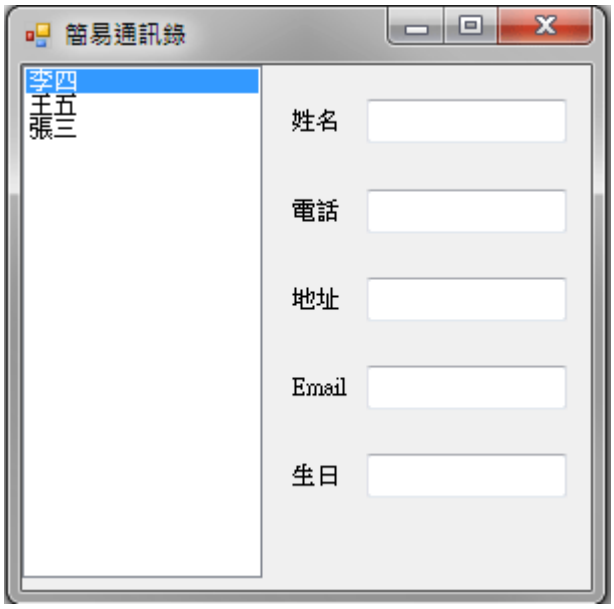
這些微軟公司程式軟體中的物件其實都是架構於所謂的 ADO.NET 程式庫之上。在 ADO.NET 架構中，資料連結的程序是先建立一個 Connection(連結)物件，裡面指定資

料庫的位置與使用哪個資料庫引擎。之後建立一個 Command(命令)物件，裡面指定使用哪個 Connection(資料連結)，以及要作甚麼操作(SQL 指令)，最後執行這個 Command。如果執行的動作是取出資料，就必須建立 DataSet 來承接了！前面提到的 Adapter 物件事實上是封裝了 ADO.NET 的 Connection 與 Command 物件。總之，如果你願意下功夫研究 ADO.NET 其實可以更有效率的直接用程式碼完成很多資料庫程式的功能。完全不使用到 Adapter 等三個物件也是可能的！

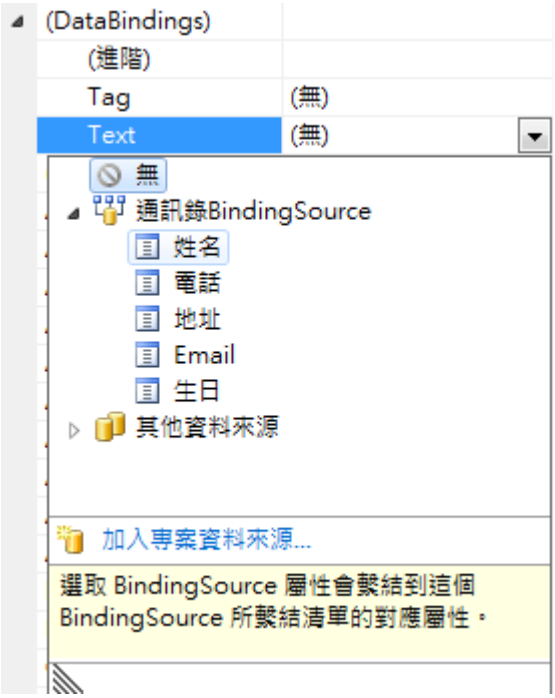
**9-5 顯示單筆資料**

[資料繫結物件的同步化]

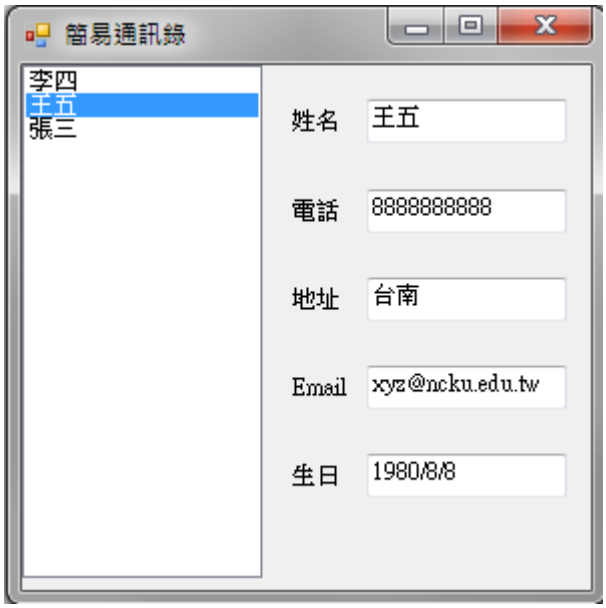
接下來我們希望點選 listBox1 的某一個名字時顯示該表資料的所有內容。請先建立好以下介面，文字顯示部分使用 Label 物件，資料顯示部份則使用 TextBox 物件，以便稍後可以修改。




在此，你一定要知道的是：「所有資料應該聯繫到 BindingSource 物件」。請先點選姓名欄的 textBox1，到屬性視窗打開「(DataBindings)」選項，在 Text 欄位先拉開「通訊錄 BindingSource」的選項，選擇其中的「姓名」選項，畫面如下：




其餘的 TextBox 資料選項也是類似的操作，試試看執行程式的畫面應該如下，在 listBox1 中點選不同筆資料，他們的詳細內容會出現在右邊的五個 TextBox 中。




## 9-6 修改、新增與刪除

[資料編修介面]

資料當然不能只是看看而已，好的程式應該可以修改資料，具體的說，所謂「修改」應該包括「修改」現有資料以及「新增」或「刪除」資料，所以請先建立好三個按鈕在上述的幾個 TextBox 之下。如圖：


[修改(Update)的程式]

首先寫入「修改」按鍵的程式碼如下：

```
private void button1_Click(object sender, EventArgs e)
{
    this.通訊錄BindingSource.EndEdit();
    this.通訊錄TableAdapter.Update(this.db1DataSet);
}
```

其意義是：先呼叫 bindingSource 物件，請它結束編輯(EndEdit)，就是將資料寫入 DataSet 物件，接著呼叫 Adapter 物件，要它使用 Update(更新)指令將 DataSet 裡面的資料確實寫到磁碟的實體資料庫檔案之中。要證明你的更新確實有效，可以到程式外面打開 Access 資料庫看看修改的結果，或者關閉程式再開啟一次，程式會再次讀取資料庫，有沒有改成功馬上可以證實。

[新增(Insert)的程式]

接下來請按下「新增」鍵寫程式如下：

```
   private void button2_Click(object sender, EventArgs e)
{
    this.通訊錄TableAdapter.Insert(textBox1.Text, textBox2.Text, textBox3.Text,
       textBox4.Text, textBox5.Text);
    this.通訊錄TableAdapter.Fill(this.db1DataSet.通訊錄);
}
```

「新增」的 SQL 指令是 Insert，因為新增的資料並不在 DatSet 裡面，因此也不必請求 BindingSource 物件結束編輯存入 DataSet，只要直接請 Adapter 物件寫一筆新資料到資料庫就好了！在此我們的程序是使用表單上的五個 TextBox 寫一筆新資料，Insert 指令會有五個參數欄，接受新資料的五個欄位值(textBox1~textBox5)。這樣處理之後資料是寫進資料庫了，但是使用者無法立即看到資料變化，所以必須再加一行程式，使用如同 Form_Load 事件中的方式再次載入(Fill)資料到 DataSet 物件之中，當然 BindingSource 物件就會自動接力運作，讓所有表單物件都顯示出最新的資料狀態了！

[刪除(Delete)的程式]

最後寫入 Delete 按鍵程式如下：

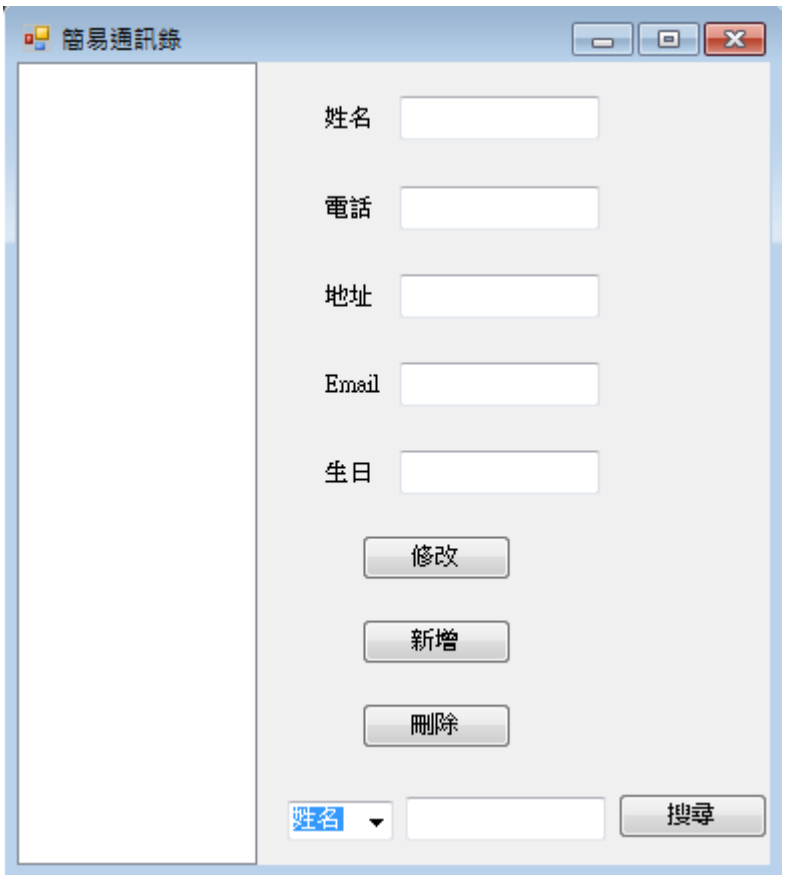
```
private void button3_Click(object sender, EventArgs e)
{
    this.通訊錄TableAdapter.Delete(textBox1.Text, textBox2.Text, textBox3.Text,
        textBox4.Text, textBox5.Text);
    this.通訊錄TableAdapter.Fill(this.db1DataSet.通訊錄);
}
```

基本上與 Insert 類似，直接請 Adapter 物件使用 Delete 指令刪除「條件符合五個 TexBox 資料內容」的那一筆資料，之後重新載入資料並正確顯示出來。在此必須小心資料欄位的順序必須與資料庫裏面的順序一樣，譬如電話與地址如果互換位置，結果是找不到完全一樣的資料，Delete 就會沒有作用，找不到自然也殺不掉嘛！

**9-7 資料查詢**

[資料搜尋的程式]

我們自己的通訊錄可能只有數十筆資料，直接使用 ListBox 選取就可以了，但是一般的資料庫可能有數百、數千甚至數萬筆的資料，此時搜尋資料的功能就很重要了！前面的幾個物件中 BindingSource 就有搜尋的功能，本單元我們就先來試試。請在表單案件下方加入如下圖的一個 comboBox1，一個 textBox6 以及一個搜尋按鍵 botton4，在 ComboBox 內建立通訊資料的五個項目(姓名、電話、地址等等)，接著建立 button4_Click 程式如後：




[多個項目搜尋的程式]

```
private void button4_Click(object  sender, EventArgs e)
{
    switch (comboBox1.Text)
    {
        case "姓名":
            this.通訊錄BindingSource.Position  =
                this.通訊錄BindingSource.Find("姓名",  textBox6.Text);
            break;
        case "電話":
            this.通訊錄BindingSource.Position  =
                this.通訊錄BindingSource.Find("電話",  textBox6.Text);
            break;
        case "地址":
            this.通訊錄BindingSource.Position  =
                this.通訊錄BindingSource.Find("地址",  textBox6.Text);
            break;
        case "Email":
            this.通訊錄BindingSource.Position  =
                this.通訊錄BindingSource.Find("Email",  textBox6.Text);
            break;
        case "生日":
            this.通訊錄BindingSource.Position  =
                this.通訊錄BindingSource.Find("生日",  textBox6.Text);
            break;
    }
}
```

程式基本流程是以 comboBox1 選擇搜尋的項目，再輸入搜尋內容，最後以 switch 語法分流，在不同的項目中依據鍵入的 textBox6.Text 找資料。上述 Find 指令找到的結果其實是資料的索引值，你可以想像一下為何前面的程式中，當使用者改變 listBox1 選項時，右邊的電話地址等欄位會跟著變化？這暗示著 BindingSource 物件內應該有個指標記錄目前選定的項目，事實上就是它的 Position 屬性，所以當我們把搜尋(Find)結果定義給 Position 時，所有的 TextBox 就會顯示該筆資料的內容了！

[Find 方法的限制]

但是請注意 Find 方法只能找出內容完全一樣的目標，用部分資料，如姓氏搜尋是會失敗的！而且如果有多筆一樣的資料它會在找到第一筆之後就不找了！所以真正聰明的搜尋程式會比這個範例複雜許多，同學們可以繼續思考怎麼製作？

**9-8 進階挑戰**

一、如何建立顯示整張資料表的介面？
　　提示：使用工具箱「資料」分類中的 DataGridView 物件連結資料庫。

二、如何讓生日的輸入變成點選而非打字輸入？
　　提示：使用工具箱的 MonthCalendar 物件。

三、如何建立部分字串資料的搜尋功能？(如搜尋姓張的朋友)
　　提示：使用 IndexOf 函數及迴圈搜尋 listBox1 之內容。

**課後閱讀**

**資料庫為就業之本**

[資料庫非萬能，但沒資料庫萬萬不能]

雖然多數程式設計的書籍都要到最後才稍稍提到資料庫的相關程式技術，但事實上如果你不懂得資料庫程式設計，應該就不必想靠著寫程式的專長就業了！可以說：任何的應用程式都必須搭配某些資料的運作，甚至很多軟體就是為處理大量資料而設計的！資料庫在任何資訊系統裡面都非常重要，也相當複雜。要配合程式軟體的運作，還要考慮網路聯繫時，會使它們更加複雜！很多程式設計的初學者因此選擇跳過這一部分的學習，事實上是絕對錯誤的想法！不能操作資料庫，你的程式功力再好，實務上能做的事情也會非常的少。

[漸進學習資料庫]

不過任何東西都有簡易版與完整版，資料庫也一樣！本單元設計的目的就在此，要學好資料庫的竅門還是要由簡單的範例入門。目前最簡單又可以完整經由高階程式操作的資料庫就是 Access，在本單元同學已經體驗到建立與修改 Access 與修改 Excel 表格非常相似，事實上只要用 Access 就可以處理非常多實務問題了！包括我曾用 Access 非常順利的建立並實驗運作一個全校層級的點名系統好幾年！這個系統的基礎資料是全校的選課紀錄，在本校一學期約三萬筆！程式相對簡單，效能卻一點都不比使用高階資料庫系統的正式校務系統差！

[先學資料結構與編輯再學資安與網路]

事實上所謂「高級」的資料庫系統與「簡單」資料庫的差異多半在登入使用的「安全性」、資料衝突處理的能力、網路連結的架構，以及大量搜尋的效能等等。這些額外的考慮通常讓資料連結與管理方式變得較為複雜，譬如 Access 可以直接允許程式使用資料庫檔案，多數高階資料庫在程式與資料庫檔案之間還必須有個「資料庫伺服器」的軟體，光是這個軟體的設定與管理就可以嚇跑大多數的初學者。

在此建議大家學習資料庫的適當階梯就是：先跳過「資料庫伺服器」的這個關卡，學習不必使用資料庫伺服器的簡易資料庫系統，當你逐步熟悉直接讀寫修改資料庫的技術，也就是 SQL 語法的操作之後，再逐步的涉獵資料庫的安全機制與網路架構等等高階議題。這樣應該就不會因為學習門檻太高而永遠作個資料庫程式設計的門外漢了！