

# Excel 2010 的 VBA 快速入門

Office 2010

**摘要：**簡介 Excel 2010 中的 Visual Basic for Applications (VBA)。適用對象為尚未成為程式設計師的 Excel 進階使用者。本文將概略介紹 VBA 語言，以及指導您如何在 Excel 2010 中存取 VBA，同時詳細實際上 Excel VBA 程式編寫問題的解決方法，並提供您一些程式設計與偵錯的小秘訣 (共 14 個列印頁面)。

前次修改時間：2009年12月11日



Ben Chinowsky, SDK Bridge

## 本文內容

為何要在 [Excel 2010 中使用 VBA](#)？

[VBA 程式設計 101](#)

[巨集與 Visual Basic 編輯器](#)

[實例](#)

[修改錄製好的程式碼](#)

[更多 VBA 的用途](#)

[再來呢？](#)

[其他資源](#)

2009 年 11 月

*適用於：Excel 2010 / Office 2010 / SharePoint Server 2010 / VBA / Visual Basic for Applications (VBA)*

## 內容

- [為何要在 Excel 2010 中使用 VBA](#)？
- [VBA 程式設計 101](#)
- [巨集與 Visual Basic 編輯器](#)
- [實例](#)
- [修改錄製好的程式碼](#)
- [更多 VBA 的用途](#)
- [再來呢？](#)
- [其他資源](#)

## 為何要在 Excel 2010 中使用 VBA？

Microsoft Excel 2010 是一個非常強大的工具，您可用以操作、分析及展示資料。雖然標準 Excel 使用者介面 (UI) 中已具有相當豐富的功能，但有時候您可能還是希望能找到更簡單的方式，來處理重複的日常工作的，或是執行一些 UI 所不處理的一些工作。幸好 Office 應用程式如 Excel 等都含有 Visual Basic for Applications (VBA)，它是一種程式語言，能讓您具有擴充應用程式的能力。

VBA 以執行「巨集」的方式運作。而所謂的巨集就是使用 Visual Basic 語言撰寫之循序漸進的步驟流程。學習撰寫程式聽起來好像很難，但只要您有耐心再配合一些範例 (像是本文中所提供之範例)，許多使用者都會發現，即使只學會一點點 VBA 程式碼，都對他們的工作有莫大的助益，讓他們可以在 Office 內達成之前做不到的事。只要您學會一部分的 VBA，要學會其他的就簡單多了。所以之前的不可能的現在都變成可能了。

到目前為止，在 Excel 中使用 VBA 最常見的原因就是為了自動化重複性的工作。舉例來說，假設您有幾十份活頁簿，而每一份當中都有幾十張工作表，且每張工作表內又有一些地方需要修改。這些修改，小如套用新的格式到某幾個固定範圍的儲存格，大到找出每張工作表資料中具統計特性的地方，選擇最合適的圖表類型來呈現這些特性的資料，再依序建立、格式化圖表。

無論是何者，您都不太可能希望要手動執行那些工作，至少不要一直重複做。反之，您可以使用 VBA 撰寫詳盡的指令供 Excel 遵循，讓工作自動化。

VBA 不僅僅是能應付重複性的工作而已。您也能運用 VBA 為 Excel 打造全新功能 (例如，可以寫一套用以分析資料的演算法，然後利用 Excel 的圖表功能顯示結果)，或是執行一些整合了 Excel 和其他 Office 應用程式的工作，如 Microsoft Access 2010 等等。事實上，在所有的 Office 應用程式中，Excel 是最常被拿來當作類似一般開發平台的項目。除了清單、會計等普遍用途外，開發人員還可利用 Excel 處理各種不同的工作，從資料視覺化到軟體原型設計等等。

雖然在 Excel 2010 中使用 VBA 有許多好處，但別忘了，解決問題的最佳方案往往根本不必用到 VBA。因為即使不含 VBA，Excel 本身便已具有非常廣泛的功能，就算是進階使用者也未必對它們完全熟悉。在您決定採用 VBA 解決方法之前，請先徹底搜尋 [說明] 與線上資源，說不定其中已經有更簡單的方法了。

## VBA 程式設計 101

### 使用程式碼讓應用程式工作

您可能認為撰寫程式碼既神祕又困難，但它的基本原則其實就像一般的常識邏輯一樣淺顯易懂。Office 2010 應用程式的運作基礎，是它們都含有一些可以接收指令的項目，稱為「物件」。您只要傳送指令給應用程式中各式不同的物件，即可和應用程式進行互動。這些物件為數眾多且各不相同，同時具備彈性，但它們也有各自的限制。它們只能進行其原始設計之用途，而且也只會執行您指示的動作。

### 物件

程式設計物件在稱為應用程式的「物件模型」階層結構中，有系統地與彼此相關聯。物件模型大致和您看到的使用者介面相符；例如，Excel 的物件模型包含 **Application**、**Workbook**、**Sheet** 及 **Chart** 物件，以及其他許多內容。物件模型就是應用程式與其功能的概念相對圖。

### 屬性與方法

您可設定物件的「屬性」並呼叫其「方法」，以操控物件。設定屬性會改變物件的某些本質；呼叫方法則可能會讓物件執行某些動作。舉例來說，**Workbook** 物件的 **Close** 方法可以關閉活頁簿；而 **ActiveSheet** 屬性又可指明目前活頁簿中有哪些工作表正在運作中。

### 集合

許多物件同時存在於單一或多個版本中，如單一活頁簿和多個活頁簿、單一工作表及多個工作表等等。多個版本稱為「集合」。當您必須在集合中的多個項目上執行同一個動作時，會使用到集合物件。本文稍後將探討如何使用 **Worksheets** 集合，變更單一活頁簿中每張工作表的名稱。

## 巨集與 Visual Basic 編輯器

現在您對 Microsoft Excel 2010 物件模型的方式有所認識，不妨嘗試呼叫物件方法並設定物件屬性看看。首先，必須已撰寫好您的程式碼，同時要是 Office 能理解的程式碼；一般而言，都會使用 Visual Basic 編輯器加以撰寫。雖然它預設會加以安裝，但許多使用者在將它啟用於功能區之前，都不知道它的存在。

### 開發人員索引標籤

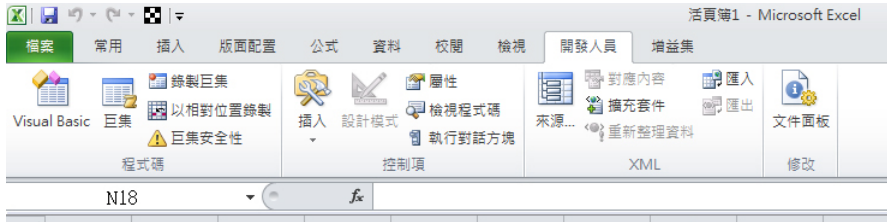
所有的 Office 2010 應用程式都使用功能區。其中一個功能區上的索引標籤為 [開發人員] 索引標籤，您可利用此索引標籤存取 Visual Basic 編輯器以及其他開發人員工具。由於 Office 2010 預設並未顯示 [開發人員] 索引標籤，您必須利用以下程序加以啟用：

### 啟用開發人員索引標籤

1. 在 [檔案] 索引標籤上，選擇 [選項] 開啟 [Excel 選項] 對話方塊。
2. 按一下對話方塊左側的 [自訂功能區]。
3. 在對話方塊左側的 [由此選擇命令] 底下，選取 [常用命令]。
4. 在對話方塊右側的 [自訂功能區] 底下，選取 [主要索引標籤]，然後選取 [開發人員] 核取方塊。
5. 按一下 [確定]。

Excel 顯示出 [開發人員] 索引標籤之後，請記下 [Visual Basic]、[巨集] 和 [巨集安全性] 按鈕在索引標籤上的位置。

圖 1：Excel 2010 中的 [開發人員] 索引標籤



## 安全性問題

按一下 [巨集安全性] 按鈕，指定可以執行的區集以及執行巨集的時機。雖然惡意的巨集程式碼有可能嚴重損害您的電腦，但設了安全性條件之後也可能會讓您無法執行一些相當有用的巨集，而大幅降低您的生產力。巨集安全性是一個很複雜的問題，在您使用 Excel 巨集之前請務必詳閱相關主題加以了解。

為配合本文所需，當您開內含巨集的活頁簿時，如果在功能區與工作表之間出現 [安全性警告：已經停用巨集]，可以按一下 [啟用內容] 按鈕啟用該巨集。

此為為安全起見，請勿將巨集儲存為預設的 Excel 檔案格式 (.xlsx)，而應一律將巨集儲存為另一種特殊的副檔名 .xlsm。

## Visual Basic 編輯器

下列程序將為您示範如何建立用以儲存巨集的空白活頁簿。之後您可將活頁簿儲存為 .xlsm 格式。

### 建主新的空白活頁簿

1. 按一下 [開發人員] 索引標籤上的 [巨集] 按鈕。
2. 在出現的 [巨集] 對話方塊中，於 [巨集名稱] 底下輸入 **Hello**。
3. 按一下 [建立] 按鈕，以開啟 Visual Basic 編輯器，其中會有所輸入之新巨集的大綱。

VBA 是一套功能完整的程式設計語言，搭配上功能完整的程式設計環境。本文只會介紹程式設計入門階段會使用到的工具，Visual Basic 編輯器中還有許多其他工具，在此暫不會探討。因此建議您關閉 Visual Basic 編輯器左側的 [屬性] 視窗，且無需理會程式碼上方的兩個下拉式清單。

圖 2. Visual Basic 編輯器



Visual Basic 編輯器包含下列程式碼。

```
VB

Sub Hello()

End Sub
```

「Sub」的意思是「副程式」，以現在的用途來看，您可以稱它為「巨集」。執行 **Hello** 巨集，就會執行從 **Sub Hello()** 到 **End Sub** 之間所有的程式碼。

現在請將您的巨集編輯成如下的程式碼。

```
VB
```

```
Sub Hello()  
    MsgBox ("Hello, world!")  
End Sub
```

回到 Excel 的 [開發人員] 索引標籤，然後再按一下 [巨集] 按鈕。

從出現的清單中選取 [Hello] 巨集，然後按一下 [執行]，您會看到一個小訊息方塊內含 "Hello, world!"

現在即已在 Excel 中建立並執行了自訂的 VBA 程式碼。按一下該訊息方塊中的 [確定] 關閉它，即會結束執行此巨集。

如果未出現訊息方塊，請檢查您的巨集安全性設定，然後重新啟動 Excel。

### 讓巨集可供存取

從 [檢視] 索引標籤也能打開 [巨集] 對話方塊，但如果您會經常用到某個巨集，使用鍵盤快速鍵或 [快速存取工具列] 按鈕加以存取，可能會更為方便。

若要在 [快速存取工具列] 上建立 [Hello] 巨集的按鈕，請使用下列程序。

以下程序說明如何為巨集在快速存取工具列上製作一個按鈕：

#### 在快速存取工具列上建立巨集的按鈕

1. 按一下 [檔案] 索引標籤。
2. 按一下 [選項] 開啟 [Excel 選項] 對話方塊，然後按一下 [快速存取工具列]。
3. 在 [由此選擇命令] 底下的清單中，選擇 [巨集]。在出現的清單中找到類似於 [Book1!Hello] 的文字，然後選取該文字。
4. 按一下 [新增 >>] 按鈕，將巨集新增至右側清單，再按一下 [修改...] 按鈕，選擇與該巨集相關的按鈕圖像。
5. 按一下 [確定]。您應該會在 [檔案] 索引標籤上方的 [快速存取工具列] 上看到您的新按鈕。

現在您可以隨時快速執行巨集，而不需使用 [開發人員] 索引標籤了，快試一下。

### 實例

假設您有一份活頁簿，在數量龐大的工作表上內含一些清單。您想一一修改工作表的名稱，好讓它們和該工作表上的清單標題一致。並非每張工作表上都有清單，但如果有清單的話，標題會位於儲存格 B1；若沒有的話，儲存格 B1 會是空白。不含清單的工作表，名稱就不必修改。

通常這會是件很複雜的工作，您必須一一瀏覽每張工作表，看看它是否含有清單，如果有的話就複製其名稱，再按一下工作表索引標籤，然後貼上新的名稱。與其手動操作這麼多個步驟，不如改用 Excel VBA 自動重新命名。

### 深入了解物件

若要解決 VBA 程式設計問題，首先必須知道該程式碼會操控哪些物件。深入了解這項資訊最基本的工具是 [Excel 物件模型參考資料 \(可能為英文網頁\)](#)，其位於 Microsoft Developer Network (MSDN) 上的 < Excel 2007 開發人員參考 > 中。

這些參考資料在公開發行 Excel 2010 之後會進行更新，但「Excel 2007 開發人員參考」的內容也適用於 Excel 2010 的大多數用途。

圖 3：MSDN 上的 Excel 物件模型參考資料



第一步是先了解如何操控完成工作時所需之特定物件，例如工作表、工作表名稱、儲存格及儲存格內容。在 Excel 中至少有兩種方法可以解決此問題：

- 直接查閱「物件模型參考資料」。
- 錄下一部分您想要自動化的動作，看看錄下的程式碼是如何操控物件的，然後再進入「物件模型參考資料」查詢更多詳情。

用什麼方法錄製比較好的解答不一，我們現在先試試「巨集錄製」。

### 使用巨集錄製

有時候您只是需要簡單錄製一個巨集便大功告成；在這種情況下，您甚至不必查看程式碼。但在更多情況下光是錄製本身還不夠，它只不過是下列程序的開端而已。

### 使用巨集錄製作為您解決方法的起點

1. 錄下您想要寫成程式碼的動作。
2. 檢閱程式碼，找出執行這些動作的程式碼。
3. 刪除其他的程式碼。
4. 修改錄製好的程式碼。
5. 加入變數、控制結構以及其他「巨集錄製」無法錄製的程式碼。

錄下一個會將工作表重新命名為 **New Name** 的巨集，開始您的調查。接著您可以使用錄製好的巨集，自行開發一個根據工作表內容重新命名多張工作表的巨集。

### 錄製重新命名工作表的巨集

1. 按一下 [開發人員] 索引標籤上的 [錄製巨集]。
2. 將該巨集命名為 **RenameWorksheets**，將 Sheet1 重新命名為 **New Name**，再按一下 [停止錄製]。
3. 前往 [開發人員] 或 [檢視] 索引標籤，按一下 [巨集] 按鈕，然後選擇 [編輯] 開啟 Visual Basic 編輯器。

Visual Basic 編輯器中的程式碼看起來應該類似如下。

```
VB
```

```
Sub RenameWorksheets()  
,  
' RenameWorksheets Macro  
,  
,  
  
    Sheets("Sheet1").Select  
    Sheets("Sheet1").Name = "New Name"  
End Sub
```

**Sub** 這一行之後的前四行都是註解。任何開頭為單引號 (') 的句子都是註解，不會影響巨集的運作。註解的主要用途如下：

- 讓程式碼更容易理解，不只是為了您，也方便未來可能會修改此程式碼的人。
- 為了暫時停用程式碼的某一行 (稱為「加註」)。

這段錄好的巨集內的四個註解不是上述兩種功用，所以加以刪除。

下一行使用 **Select** 方法選取 **Sheets** 集合物件中的 **Sheet1** 成員。在 **VBA** 程式碼中，通常不必在操控物件之前先加以選取，雖然「巨集錄製」會如此做。換句話說，這一行程式碼是多餘的，您也可以加以刪除。

所錄製之巨集的最後一行，會修改 **Sheets** 集中 **Sheet1** 這個成員的名稱屬性。這行應該保留。

改好後，錄製的程式碼看起來應如下。

VB

```
Sub RenameWorksheets()  
    Sheets("Sheet1").Name = "New Name"  
End Sub
```

手動將稱為 **New Name** 的工作表改回 **Sheet1**，然後執行巨集。它的名稱應該會再改回 **New Name**。

## 修改錄製好的程式碼

現在我們該來研究一下「巨集錄製」所使用過的 **Sheets** 集合。「物件模型參考資料」中的「工作表」主題包含下列文字。

"**Sheets** 集合可包含 **Chart** 或 **Worksheet** 物件。若您必需使用單一類型的工作表，請參閱該工作表類型的物件主題。"

您只用到 **Worksheets**，所以請將程式碼變更如下：

VB

```
Sub RenameWorksheets()  
    Worksheets("Sheet1").Name = "New Name"  
End Sub
```

## 迴圈

此程式碼截至目前為止有一個限制，就是它只能對一份工作表進行變更。您也可以為每份想要重新命名的工作表新增一行程式碼，但要是您不知道總共有多少工作表呢？或者您不知道這些工作表目前的名稱為何，該怎麼辦？您需要有一種方法，能將某些規則分別套用到活頁簿中的每張工作表。

**VBA** 有一種句法叫做 **For Each** 迴圈，相當實用。**For Each** 會檢查集合物件 (例如 **Worksheets**) 中的每個項目，而且可用以對部分或所有項目執行動作，比方說變更名稱。

如需 **For Each** 迴圈的詳細資訊，請參閱 [VBA 語言參考資料 \(可能為英文網頁\)](#)。按一下 <Visual Basic 概念主題>，再按一下 <使用 For Each...Next 陳述式>。同時請記住，「**VBA 語言參考資料**」就和「物件模型參考資料」一樣，是個尋找靈感的好去處，當您在設計程式碼但遇到瓶頸時，記得上去逛一逛，保證會有豐富的收穫。

參考 <使用 For Each...Next 陳述式> 主題中的第三個範例，將巨集編輯為像下列程式碼一般。

VB

```
Sub RenameWorksheets()  
For Each myWorksheet In Worksheets  
    myWorksheet.Name = "New Name"  
Next  
End Sub
```

`myWorksheet` 是變數；意思就是說，它所代表的東西是會變動的。在此案例中，`myWorksheet` 變數所代表的就是 `Worksheets` 集合中每張工作表的名稱。它不一定要取名為 `myWorksheet`，您也可以改用 "x"、"ws"、"WorksheetToRenameAfterTheContentsOfCellB1" 或幾乎任何您想要的名稱 (含有某些限制)。最好的命名法則，是使用長度足以讓您想起該變數所代表之意義的名稱，但不要長到讓程式碼擠成一團。

若以目前的狀態執行此巨集，它會產生錯誤，因為 Excel 需要活頁簿中的每張工作表都具有唯一的名稱，但下列這一行卻指示 Excel 為所有工作表提供相同的名稱。

VB

```
myWorksheet.Name = "New Name"
```

為了讓您能確認 For Each 迴圈是否運作正確，請將該行變更如下：

VB

```
myWorksheet.Name = myWorksheet.Name & "-changed"
```

如此一來，程式碼就不會再為每張工作表提供相同的名稱，而是會在每張工作表目前的名稱 (`myWorksheet.Name`) 後面加上 "-changed"。

### 好用的重新命名法

現在這個巨集差不多可以實際解決眼前的問題了。您現在需要的是一個可以從工作表本身擷取資訊 (特別是從每張工作表的儲存格 B1 擷取)，並將該資訊插入工作表名稱的方法。

這一次，我們不用「巨集錄製」尋找參照儲存格的方法，猜猜看如果使用 Cell 物件是不是可行？這是個好問題，但如果您開啟「物件模型參考資料」並搜尋 Cell 物件，您會發現沒有 Cell 物件這東西！但倒是有一個 [CellFormat 物件 \(可能為英文網頁\)](#)。

CellFormat 物件主題的第一個範例中包含下列程式碼。

VB

```
' Set the interior of cell A1 to yellow.  
Range("A1").Select
```

原來您應該要用 `Range` 指定一個範圍的儲存格或單一儲存格。而且，您不需要 `.Select` 這部分，但您需要了解該如何參照 `Range` 的內容，而不是參照 `Range` 物件本身。如果您前往 `Range` 物件的主題，會讀到 `Range` 具有 `Methods` 以及 `Properties`。`Range` 內容是一件事，但不是一項動作，所以它可能是 `Property`。從清單一路往下找，會看到 `Value` 屬性。所以就讓我們試試以下動作。

VB

```
Sub RenameWorksheets()  
For Each myWorksheet In Worksheets  
    myWorksheet.Name = myWorksheet.Range("B1").Value  
Next  
End Sub
```

如果您在內含工作表的活頁簿上執行此程式，且工作表的 B1 是空的，就會出現錯誤。這是因為空白 `Range` 的 `Value` 會是 "" (空白字串)，而這樣的工作表名稱並非法。但沒關係，現在剛好該建立一些範例資料了。接下來請將活頁簿中的三張工作表改成像下圖一般，然後執行巨集。

圖 4：RenameWorksheets 巨集的範例資料

B1	
A	B
1	Occupation: Doctors
2	
3	

B1		
	A	B
1	Occupation:	Lawyers
2		
3		

B1		
	A	B
1	Occupation:	Engineers
2		
3		

工作表名稱應會隨之變更。

### 檢查是否存在空的儲存格

如上所述，如果活頁簿中有任何 B1 儲存格是空的，巨集就會失敗。與其手動檢查每張工作表，還是寫一句程式碼讓巨集幫你完成吧。在 `myWorksheet.Name` 這一行之前，新增下列程式碼。

VB

```
If myWorksheet.Range("B1").Value <> "" Then
```

然後在 `myWorksheet.Name` 這一行之後新增下列文字。

VB

```
End If
```

這就叫做 If...Then 陳述式。If...Then 陳述式會指示 Excel 執行 If 行到 End If 行之間的所有指令，但只有在符合 If 行中的條件時才會成立。在範例中下列程式碼的作用，是指定要符合的條件。

VB

```
myWorksheet.Range("B1").Value <> ""
```

`<>` 是「不等於」的意思。如果引號之間沒有任何東西，代表它是一個空的字串，也就是完全沒有任何文字。因此，在 If 和 End If 之間無論有什麼程式碼，都只會在 B1 的值不等於無的條件下才會執行；也就是說，儲存格 B1 內有必須有文字。

如需有關 If...Then 陳述式的詳細資訊，請參閱 VBA 語言參考資料 (它的完整名稱是 "If...Then...Else 陳述式"，*Else* 是一個選用的元件)。

### 變數宣告

此巨集還有一個可以加強的地方，那就是您應該在巨集的開頭加上一段 `myWorksheet` 變數的宣告。

VB

```
Dim myWorksheet As Worksheet
```

**Dim** 是 "Dimension" (維度) 的縮寫，而 **Worksheet** 代表此特定變數的類型。這段陳述式會告知 VBA `myWorksheet` 所代表的是哪種實體。請注意，當您輸入 **As** 之後，Visual Basic 編輯器會顯示快顯清單，列出所有可用的變數類型。這即是展現 IntelliSense 技術的一個好例子，也就是說 Visual Basic 編輯器會自動回應它判別您想要做的事，並提供適用選項的清單。您可以選擇清單中的選項，或繼續輸入內容。

雖然 VBA 中並不需要變數宣告，但極力建議您使用！有了變數宣告能讓追蹤您的變數以及找出程式碼中的錯誤，變得容易得許多。此外請留意，假如您利用物件類型 (例如 Worksheet) 宣告變數，則當您稍後在巨集中使用該物件變數時，IntelliSense 會顯示與該物件相關的適合屬性與方法的清單。

### 註解

此巨集現在已經相當複雜，應該加些註解以免忘記這些程式碼在做什麼。而該加多少註解基本上依個人習慣而定，但一般來說，寧可過多也不要太少。程式碼經常會需要隨時修改與更新，要是沒有註解，我們要搞清楚程式碼究竟在做什麼就會變得十分困難，特別是當修改者和程式碼的原作者不是同一人時，就更麻煩了。所以我們應該為 If 條件，以及重新命名工作表的程式碼加上註解，結困如以下的程式碼。



VB

```
Sub RenameWorksheets()  
Dim myWorksheet As Worksheet  
For Each myWorksheet In Worksheets  
    'make sure that cell B1 is not empty  
    If myWorksheet.Range("B1").Value <> "" Then  
        'rename the worksheet to the contents of cell B1  
        myWorksheet.Name = myWorksheet.Range("B1").Value  
    End If  
Next  
End Sub
```

若要測試巨集，請先將工作表的名字重新命名回 **Sheet1**、**Sheet2** 和 **Sheet3**，並刪除一或多份工作表上儲存格 B1 的內容。接著執行巨集，確定它會將儲存格 B1 內有文字的工作表重新命名，且不會變動其他的工作表。巨集適用於不限數量的工作表，無論有字和空白的 B1 儲存格以任何不同的組合形式，都可正確運作。

### 更多 VBA 的用途

本節將會探討幾種 VBA 在 Excel 2010 中的用途。本節中所設計的範例，是為了讓您體驗一下 VBA 的強大能力，而不是針對現實生活中特定的案例所寫。您在練習這些範例步驟時，不妨一邊查閱「物件模型參考資料」複習一下每個物件的相關資訊，會很有幫助。

### 敞開學習之門

學習所有程式設計的竅門就在於多方嘗試，學習 Excel VBA 更是如此。當您終於試成功了以後，不妨問問自己：

- 再來我還可以嘗試什麼？
- 以我的用途來看，我應該先學 VBA 的什麼？
- 還有什麼有趣的內容是值得了解一下？
- 我最好奇的是什麼？

我們強烈鼓勵讀者敞開學習大門，鑽研各種知識。

### 圖表

Excel 有一項很常見的工作，就是根據某範圍內的儲存格建立圖表。請建立稱為 **AssortedTasks** 的新巨集，然後在 Visual Basic 編輯器中輸入下列文字。

VB

```
Dim myChart As ChartObject
```

新增一行建立圖表物件的程式碼，然後指派 **myChart** 變數給它。

VB

```
Set myChart = ActiveSheet.ChartObjects.Add(100, 50, 200, 200)
```

括弧內的數字會決定了圖表的位置與大小。前兩個數字是左上角的座標，後兩個數字是寬度與高度。

新建一張空白的工作表，然後執行巨集。它所建立的圖表沒有用處，因為圖表中沒有任何資料。刪除您剛剛建立的圖表，然後在巨集結尾處新增下列程式碼。

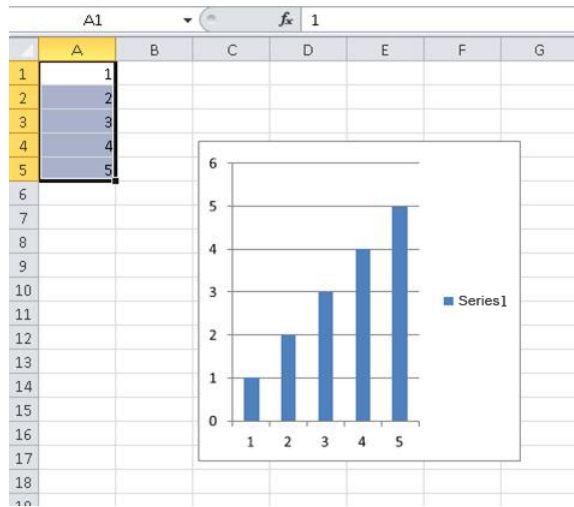
VB

```
With myChart
    .Chart.SetSourceData Source:=Selection
End With
```

這就是一種在 VBA 程式設計中常用的典型。首先您建立一個物件，並將其指派給一個變數，然後使用 With...End With 句法建構方式運作此物件。程式碼範例会指示該圖表使用目前選取的資料範圍。(Selection 是 SetSourceData 方法 Source 參數的值，而不是物件屬性的值，因此 VBA 語法會要求您使用一個冒號加一個等號 (:=) 而不要只使用等號 (=) 來指派該值。)

在儲存格 A1:A5 中輸入一些數字，選取那些儲存格，然後執行巨集。出現的圖表是預設的樣式：橫條圖。

圖 5：使用 VBA 建立的橫條圖



如果您不喜歡橫條圖，也可以使用類似如下的程式碼，將其變更為其他種類的圖表。

VB

```
With myChart
    .Chart.SetSourceData Source:=Selection
    .Chart.ChartType = xlPie
End With
```

xlPie 是內建常數的一種例子，也稱為列舉常數。Excel 內建有非常多的常數，相關說明也十分詳盡。如需內建常數的詳細資訊，請參閱「物件模型參考資料」的〈列舉〉一節。例如，圖表類型的常數會列在「xlChartType 列舉」之下。

您也可以修改資料。例如，嘗試緊接在變數宣告之後，新增此程式碼。

VB

```
Application.ActiveSheet.Range("a4").Value = 8
```

您可以取得使用者輸入的內容，然後用該筆輸入資料修改資料。

VB

```
myInput = InputBox("Please type a number:")
Application.ActiveSheet.Range("a5").Value = myInput
```

最後，在巨集結尾處新增下列程式碼。

VB

```
ActiveWorkbook.Save
ActiveWorkbook.Close
```

現在這個巨集完整的樣子應該類似如下。

VB

```
Sub AssortedTasks()
Dim myChart As ChartObject
Application.ActiveSheet.Range("a4").Value = 8
myInput = InputBox("Please type a number:")
Application.ActiveSheet.Range("a5").Value = myInput
Set myChart = ActiveSheet.ChartObjects.Add(100, 50, 200, 200)
With myChart
    .Chart.SetSourceData Source:=Selection
    .Chart.ChartType = xlPie
End With
ActiveWorkbook.Save
ActiveWorkbook.Close
End Sub
```

確定儲存格 A1:A5 仍是選取狀態，然後執行巨集，在輸入方塊中輸入一個數字，再按一下 [確定]。程式碼會儲存並關閉該活頁簿。接著請重新開啟活頁簿，看看是否已變更為圓形圖。

UserForms

前一節示範了如何使用簡單的輸入方塊，取得使用者的輸入內容。除了顯示資訊的相對應訊息方塊之外，VBA 還會提供了非常多元的功能，可讓您用以建立自訂對話方塊、編寫直接置於工作表中的控制項，或操控 Excel 內建的對話方塊。如需這些功能的詳細資訊，請參閱 Excel 2007 開發人員參考資料中的[控制項、對話方塊和表單 \(可能為英文網頁\)](#)。

本節中會快速探討一下 UserForms，為這篇 Excel VBA 簡介作個總結。

在 [開發人員] 索引標籤上，按一下 [Visual Basic] 按鈕，開啟 Visual Basic 編輯器，然後進入 [插入] 功能表中選擇 [UserForm]，以開啟 [UserForm 設計檢視]。

您會看到兩個視窗。其中一個是您正在建立的 UserForm，另一個是 [工具箱]，會顯示出各種您可以加入 UserForm 的控制項，例如命令按鈕、選項按鈕、核取方塊等等。您可以移動滑鼠遊標到 [工具箱] 控制項上方，了解可以建立什麼類型的控制項。

建立一個只包含單一按鈕的超極簡易 UserForm，而按鈕的作用為執行本文稍早的 Hello 巨集。在 [工具箱] 中，按住 **CommandButton** 控制項，然後將其拖曳到 UserForm 中，建立一個新的命令按鈕。在該命令按鈕上按一下滑鼠右鍵，然後選擇 [檢視程式碼]。

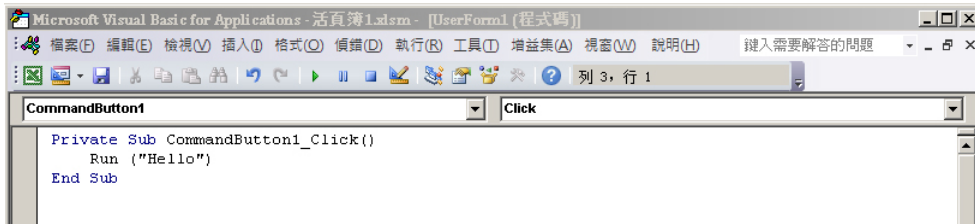
您所看到的 **Sub** 是一個「事件程序」(發生某特定事件時即會執行) 的基本結構。在此例中，如 **Sub** 的名稱所表達的，會執行程式碼的事件為在 **CommandButton1** 上 **Click**。請在事件程序中加入以下行。

VB

```
Run("Hello")
```

現在，Visual Basic 編輯器看起來應類似如下。

圖 6：CommandButton1\_Click 事件程序



儲存活頁簿，進入 [視窗] 功能表中選擇 [UserForm1 (UserForm)]，重新顯示 UserForm。按一下工具列上的綠色箭頭，執行 UserForm。出現對話方塊時，按一下命令按鈕，執行會顯示 "Hello, world!" 訊息方塊的 Hello 巨集。關閉訊息方塊回到執行中的 UserForm，然後關閉執行中的 UserForm 以返回 [設計檢視]。

## 再來呢？

本篇文章附上了一些實做，同時花了相當多的時間研究「物件模型參考資料」與「VBA 語言參考資料」，您一定覺得收穫匪淺，有信心完成一切驅使您開始學習 VBA 的工作了吧？如果是，太好了！若不是，沒關係，您需要的是進一步對 VBA 展開更全面的認識。

研究現成的程式碼是深入了解 VBA 的一個好方法。除了「物件模型參考資料」和「VBA 語言參考資料」之外，還有許多線上來源 (包括 MSDN、專長 Excel 的 Microsoft MVP 所主持的網站，以及其他來源) 都提供非常大量的 Excel VBA 程式碼。您只需上網快速搜尋一下，即可找到。

這些資源所提供的程式碼，能協助您快速處理眼前的編碼問題，甚至還可以替您未來的案子得到一些靈感。

如果您比較希望能有一套更有系統的 VBA 學習法，市面上能找到幾本討論 VBA 的好書，網路上也有幾篇寫得不錯的書評，您可參考後選擇一本最適合您學習風格的寶典。

## 其他資源

- [Excel 開發人員中心 \(可能為英文網頁\)](#)
- [Office 開發人員中心](#)
- [Microsoft MVP 獎 \(可能為英文網頁\)](#)
- [在 MVPs.org 上尋找 Excel MVP \(可能為英文網頁\)](#)
- [Office 2010 的 VBA 快速入門](#)