Chương 24: Liên kết giữa Excel với Visual Basic 6.0

Chương này được trích từ cuốn **Lập trình VBA trong Excel (Nhà xuất bản thống kê)**
http://www.giaiphapexcel.com/forum/showthread.php?t=22105

Trước khi tìm hiểu mối quan hệ này, bạn biết rằng Visual Basic 6.0 (VB 6.0) được xây dựng bởi tập đoàn Microsoft. Hiện này VB 6.0 đã được thay thế bởi VB.NET, tuy nhiên VB 6.0 vẫn là phần mềm được nhiều nhà lập trình sử dụng rộng rãi nhất trên thế giới. VB 6.0 mạnh mẽ hơn người anh em VBA vì VB 6.0 là ngôn ngữ lập trình hoạt động mang tính độc lập. Nhìn chung nội dung, cấu trúc, mã lệnh trong thủ tục của VB 6.0 và VBA rất giống nhau. Vì vậy, những người am hiểu về VBA thì có thể nhanh chóng tiếp cận và sử dụng VB 6.0.

Chương này sẽ chỉ dẫn bước đầu tạo mối liên kết giữa Excel với VB 6.0 và những lý do tại sao sử dụng VB 6.0 cho các dự án VBA của bạn. VB 6.0 có thể tạo ra hơn 6 kiểu ứng dụng, nhưng chỉ có hai kiểu liên kết được với Excel là ActiveX DLL và Standard EXE. Mục này sẽ hướng dẫn bạn cách sử dụng ActiveX DLL và Standard EXE trong VB 6.0 cho Excel với ứng dụng đơn giản “Hello World”. Chúng ta sẽ khám phá sự liên kết giữa Excel và Standard EXE trong mục tiếp theo.

Như vậy bạn sẽ thắc mắc tại sao cần phải sử dụng VB 6.0 trong khi đó VBA sẵn có trong Excel. Dưới đây là những nét chính để bạn quyết định có nên sử dụng VB 6.0 để liên kết với Excel hay không?!

- Khả năng bảo mật code: VBA có chức năng bảo mật code để chống người khác có thể xem dự án VBA của bạn (xem mục 11.2). Điều đó là cần thiết khi bạn đã bỏ công sức để xây dựng sản phẩm của mình. Tuy nhiên hiện nay có rất nhiều chương trình có thể dò tìm và phá được khóa. Chương trình của bạn lúc đó rất dễ bị phân tán để mọi người sử dụng ngoài tầm kiểm soát. Nguyên nhân là do VBA không biên dịch hay mã hoá được code, điều đó không thể ngăn chặn được người khác truy cập vào. VB 6.0 khả năng biên dịch thành thư viện liên kết động (Dynamic Link Library - viết tắt là DLL), chương trình tạo ra từ VB 6.0 hoạt động độc lập (Standard EXE), do đó khả năng bảo mật sẽ cao hơn.

- Sử dụng VB 6.0 Form nâng cao: Điều khiển xây dựng trong Form của VB 6.0 phong phú hơn so với VBA. Đối tượng Form được tạo trong ứng dụng Excel được gọi là MSForm, còn trong VB 6.0 thì được hiểu là Ruby Form. Sự giống nhau về bên ngoài giữa chúng là giao diện chương trình, Form để bạn có thể xây dựng, điều khiển các đối tượng trong Form. Ngoài ra các điều khiển trong cả hai kiểu đều sử dụng lập trình sự kiện xảy ra với chúng để

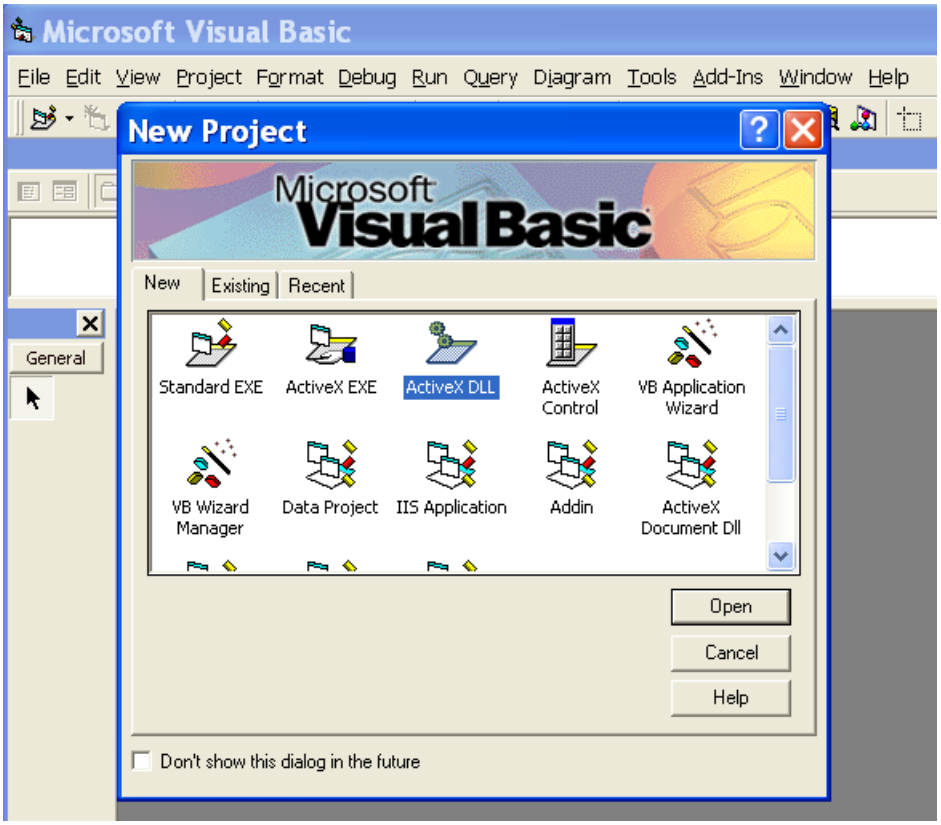
thi hành thủ tục. Đương nhiên giữa chúng có sự khác nhau, ví dụ như thuộc tính, phương thức, sự kiện và kể cả các điều khiển của VB 6.0 phong phú hơn,...

- Hỗ trợ điều khiển ActiveX tốt hơn: Không chỉ mỗi VB 6.0 Form cung cấp khả năng hỗ trợ tốt hơn so với UserForm của Excel. VB 6.0 còn cung cấp hàng trăm điều khiển ActiveX nhóm 3 mà không có đầy đủ ở UserForm của Excel. Ngoài ra khả năng điều khiển mảng dữ liệu liên liên kết được tăng cường hơn so với VBA.

Ứng dụng Active DLL "Hello World" dưới đây sẽ giải thích một cách tạo liên kết từ Excel tới DLL. Excel sẽ liên kết với DLL và DLL sẽ liên kết trở lại với Excel. Tiếp theo chúng ta sẽ nghiên cứu cách sử dụng Form của VB 6.0 như là UserForm sẵn có trong Excel.

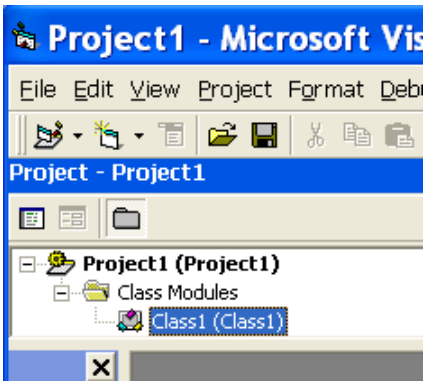
24.1. Tạo dự án ActiveX DLL

Khi mở chương trình VB 6.0 thì cửa sổ New Project hiện ra như hình 24-1. Trong trường hợp cửa sổ New Project không hiện ra, bạn vào menu File/New Project. Tại cửa sổ New Project chọn kiểu dự án ActiveX DLL. Sau khi bạn lựa chọn kiểu dự án và bấm vào nút Open, VB 6.0 sẽ tạo ra một dự án mới ActiveX DLL.

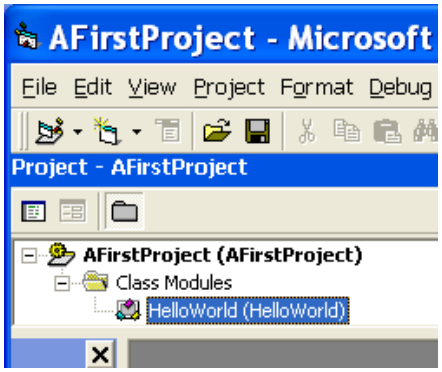


Hình 24-1: Cửa sổ New Project của VB 6.0

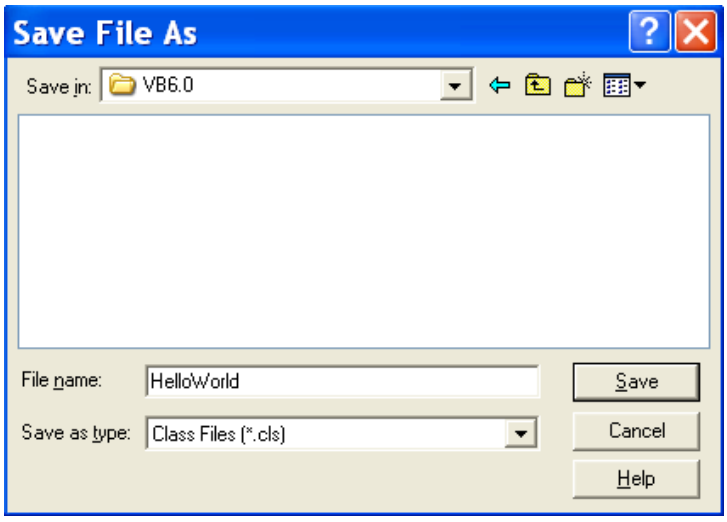
Chúng ta sẽ xây dựng ứng dụng "Hello World" đầu tiên, ActiveX DLL sẽ chỉ chứa hai phần Project (dự án) và Class Module (hình 24-2). Dự án đó sẽ xác định tên của ứng dụng trong DLL, và Class Module sẽ thể hiện các tính năng của DLL trong các chương trình khác. Hình 24-2 thể hiện cấu trúc của một dự án mới ActiveX DLL trong cửa sổ Project. Nếu thấy cửa sổ Project xuất hiện trông rất quen thuộc, điều đó không có ảnh hưởng gì cả. Bạn sẽ tìm thấy môi trường giữa VB 6.0 và VBA giống nhau đến mức dễ gây nên sự xáo trộn khi làm việc với chúng. Điều đó sẽ giúp bạn dễ dàng làm việc với VB 6.0 khi bạn đã có kinh nghiệm về VBA. Chúng ta sử dụng cửa sổ Project trong VB 6.0 gần như tương tự với VBA.



Hình 24-2: Dự án lúc đầu



Hình 24-3: Dự án đã đặt tên

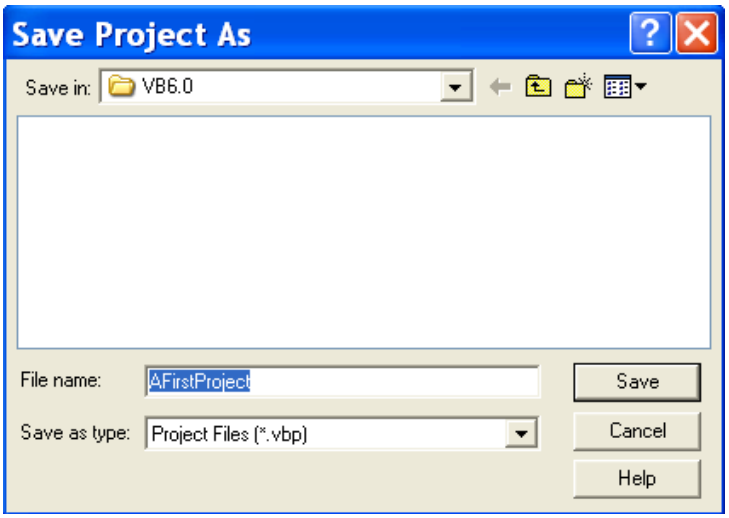


Hình 24-4: Lưu Class Module "HelloWorld" dưới dạng file Class Files (*.cls)

Hãy thay đổi tên của Project thành AFirstProject, tên của Class1 thành HelloWorld bằng cách gõ vào trong hộp Name của cửa sổ Properties của chúng như hình 24-6. Kết quả thay

đổi thể hiện tại hình 24-3. Bạn sẽ thấy tên file trong ngoặc đơn nằm phía bên ngoài là thành phần trong dự án, đó là một sự khác nhau giữa VBA và VB 6.0. Trong VBA, các thành phần của dự án được lưu bên trong file riêng của Excel. Trong VB 6.0, tất cả các thành phần được lưu giữ trong các file riêng biệt của dự án.

Sau đó bạn lưu giữ tên dự án đó lại với hai phần Project (dự án) và Class Module (hình 24-6) với tên mặc định đã đặt. Bạn sẽ thấy khi dự án đã được lưu thì tên dự án với đuôi xác định sẽ hiển thị trong cửa sổ Project (hình 24-5).



Hình 24-5: Lưu dự án "AFirstProject" dưới dạng file Project Files (*.vbp)

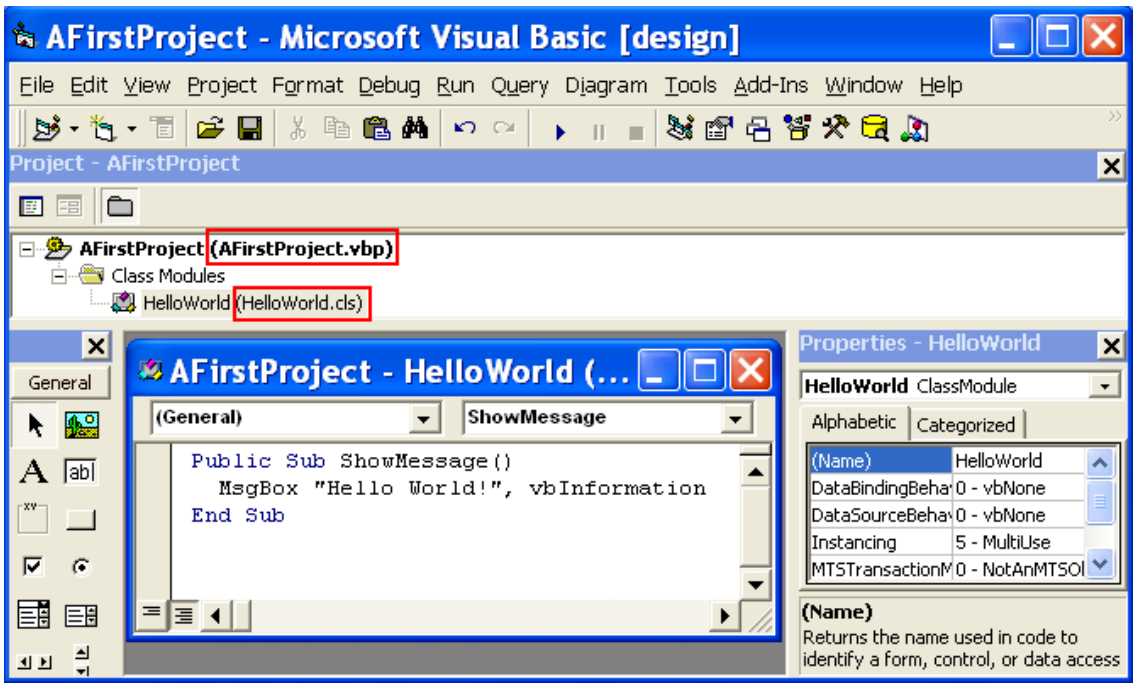
24.1.1. Cách tạo liên kết đơn giản ActiveX DLL một hướng

Bây giờ chúng ta sẽ tìm hiểu cấu trúc của dự án ActiveX DLL hoàn chỉnh, hãy bổ sung thủ tục để thực hiện công việc nào đó. Trong dự án "Hello World", DLL sẽ cho hiện hộp thông báo với nội dung "Hello World!" khi thực hiện lệnh. Việc đó sẽ được hoàn thành khi bạn thêm phương thức tới Class Module "HelloWorld", đó chính là thủ tục sẽ hiển thị hộp thông báo khi được gọi. Việc thêm phương thức tới VB 6.0 Class Module sẽ làm việc tương tự như trong VBA. Bạn bấm đúp chuột vào Class Module HelloWorld trong cửa sổ Project, cửa sổ soạn code hiện ra. Thủ tục ShowMessage như sau (hình 24-6):

```
Public Sub ShowMessage()
   MsgBox "Hello World!", vbInformation
End Sub
```

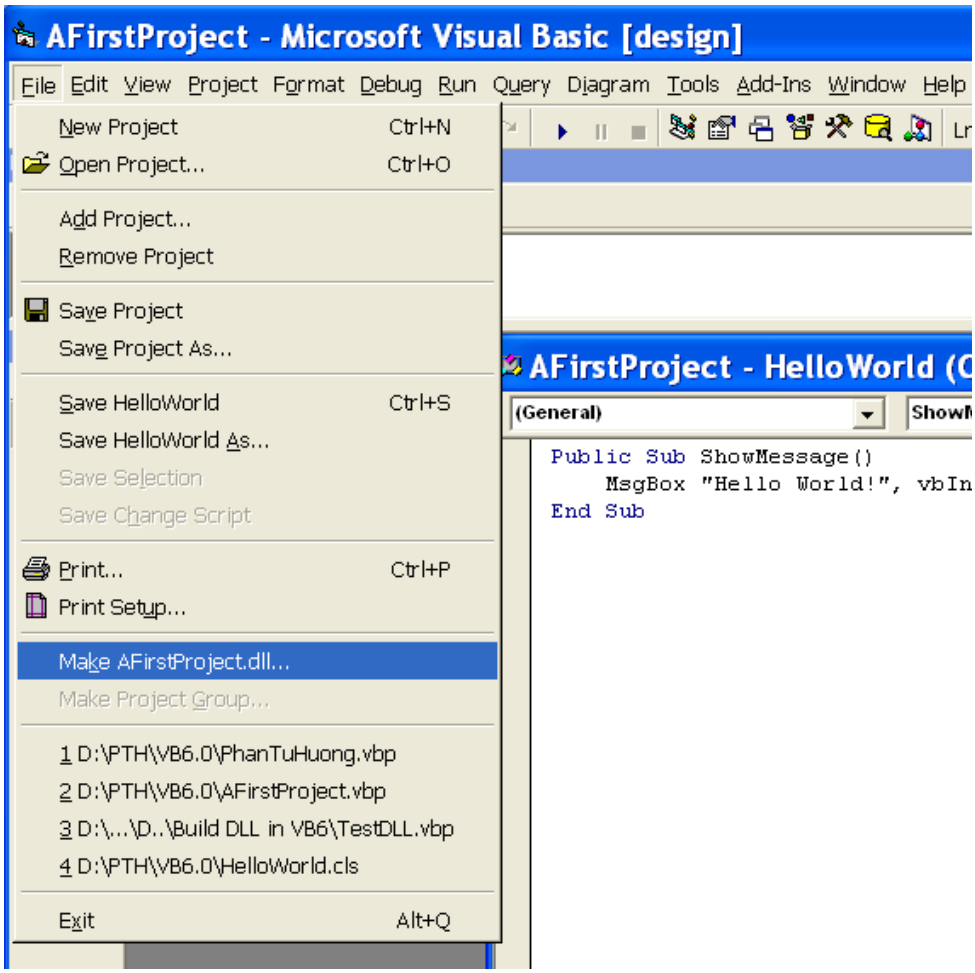
Công việc thật đơn giản! Bây giờ chúng ta cần phải biên dịch ActiveX DLL của bạn và gọi ra được từ ứng dụng Excel để hiển thị lời chào đó. Nhưng trước hết hãy biên dịch code đó.

Đầu tiên vào menu File trong cửa sổ VB 6.0, chọn Make AFirstProject.dll... (hình 24-7). Cửa sổ Make Project hiện ra như hình 24-8, chọn nút OK để tạo AFirstProject.dll trong thư mục chứa dự án của bạn.

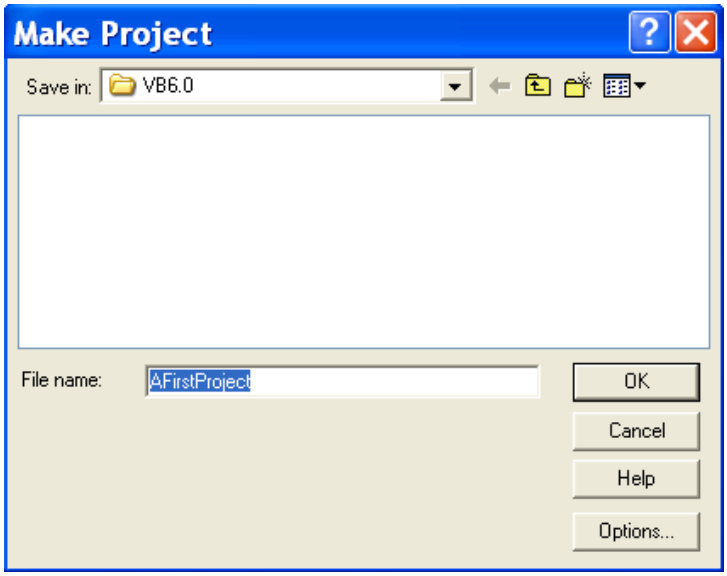


Hình 24-6: Tạo thủ tục ShowMeesage trong Class Modules HelloWorld

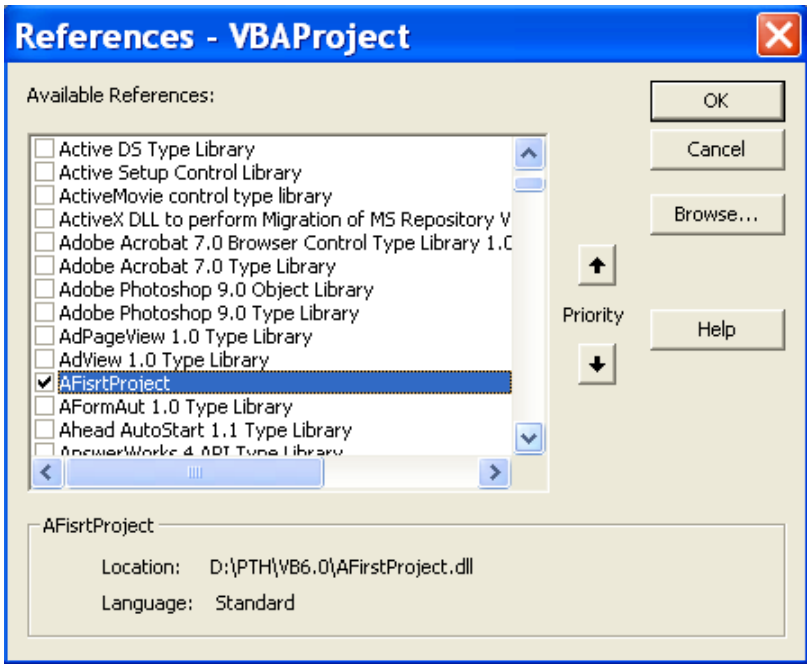
Tiếp theo bạn sẽ làm gì trong Excel? Công việc đó rất dễ dàng khiến bạn phải ngạc nhiên. Bước đầu tiên khởi động Excel, sau đó bạn mở cửa sổ VBE. Vào menu Tools/References..., cửa sổ References - VBAProject hiện ra như hình 24-9. Tìm dự án AFirstProject.dll trong danh sách Available References và chọn. Nếu dự án đó không xuất hiện trong danh sách thì bạn bấm nào nút Browse... để tìm kiếm nơi lưu trữ, sau đó bấm OK để xác nhận.



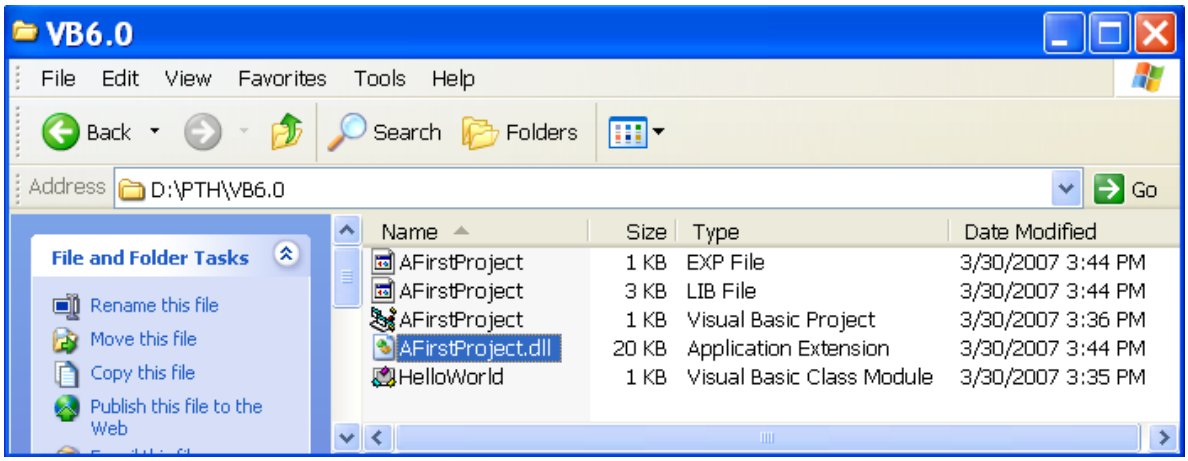
Hình 24-7: Tạo DLL cho dự án AFirstProject



Hình 24-8: Đặt tên cho dự án DLL của bạn



Hình 24-9: Xác nhận dự án AFirstProject trong file



Hình 24-10: Dự án của bạn được tạo trong thư mục VB6.0 với đầy đủ các thành phần

Sau đó lưu workbook với tên Hello.xls trong thư mục chứa dự án đó, việc lưu trong cùng thư mục giúp bạn dễ dàng tìm kiếm và sử dụng chứ không phải bắt buộc. Trong thực tế, DLL có thể sử dụng ở mọi nơi trong máy tính. Công việc tiếp theo là thêm module vào trong cửa sổ VBE và xây dựng thủ tục ShowDLLMessage như sau:

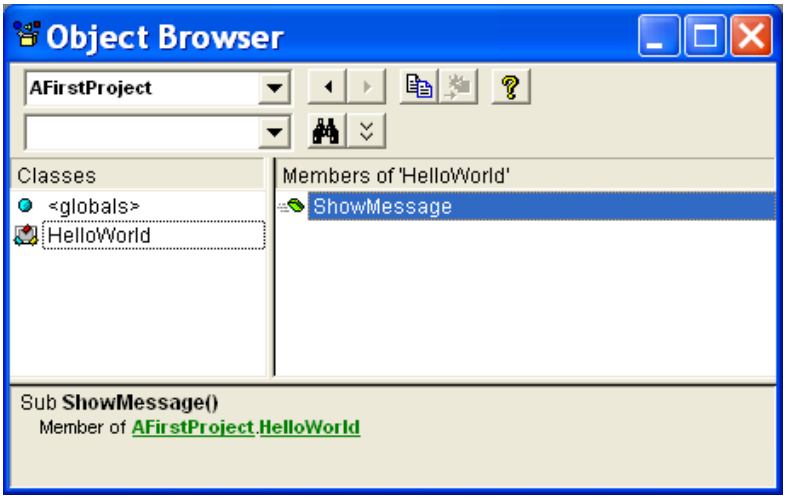
```
Public Sub ShowDLLMessage()
```

```
    'Khai báo biến đối tượng
    Dim HelloWorld As AFirstProject.HelloWorld

    Set HelloWorld = New AFirstProject.HelloWorld
    'Thực hiện thủ tục ShowMessage trong class HelloWorld
    HelloWorld.ShowMessage
    Set HelloWorld = Nothing
End Sub
```

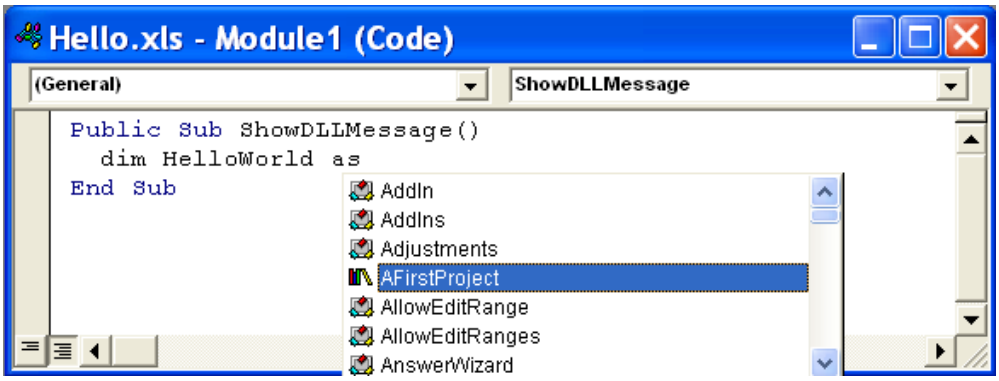
Ban đầu, bạn sẽ thấy công việc khó khăn khi sử dụng các thủ tục xây dựng trong DLL. Tuy nhiên khi hiểu rõ phương thức sử dụng DLL bằng VBA, bạn sẽ dễ dàng thực hiện được. Trong ví dụ trên, các đối tượng trong DLL được nhận biết trong Excel như sau:

Bảng 24-1: Mô tả đối tượng và chức năng của nó trong DLL

Đối tượng	Mô tả
AfirstProject	Là thư viện quản lý các đối tượng trong dự án AFirstProject, dự án này quản lý các Classe Module. VBA tự động nhận biết bằng chức năng Auto List Members (hình 24-11, 24-12)
HelloWorld	Là Classe Module chứa trong dự án AFirstProject (hình). Một dự án sẽ có một hay nhiều Classe Module. VBA tự động nhận biết bằng chức năng Auto List Members (hình 24-13).
ShowMessage	Là thủ tục chứa trong Classe Module HelloWorld. Mỗi Classe Module có thể chứa một hay nhiều thủ tục.VBA tự động nhận biết bằng chức năng Auto List Members (hình 24-14).

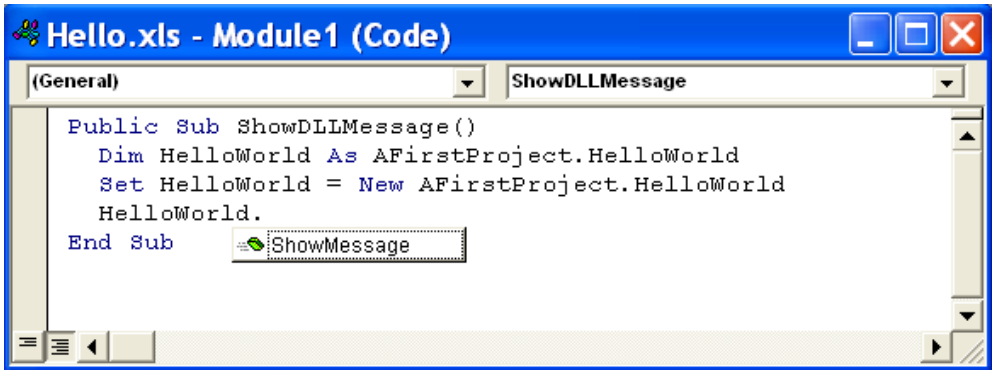


Hình 24- 11: Thư viện AFirstProject và các đối tượng chứa bên trong



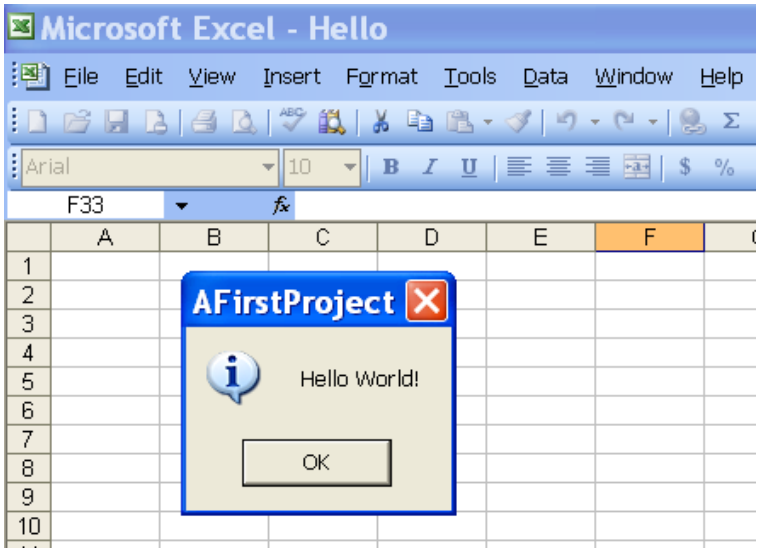
Hình 24-12: VBA tự nhận biết thư viện AFirstProject bằng chức năng Auto List Memmbers

Hình 24-13: VBA tự nhận biết Classe HelloWorld bằng chức năng Auto List Memmbers



Hình 24-14: VBA tự nhận biết thủ tục ShowMessage trong Classe HelloWorld

Khi thực thi thủ tục ShowDLLMessage, kết quả thể hiện như tại hình 24-15. Bạn sẽ thấy kết quả tạo hộp nhắn tin này tương tự khi sử dụng VBA.



Hình 24-15: Thông báo được thể hiện khi bạn chạy thủ tục ShowDLLMessage

Và một điều đáng quan tâm nữa là tiêu đề (title) trong hộp nhắn tin. Nếu không cung cấp tên tiêu đề, hộp nhắn tin sẽ thể hiện tên tiêu đề mặc định, là tên của dự án đang sử dụng (AFirstProject). Ví dụ nếu bạn muốn hộp nhắn tin hiển thị "Control Excel" ở thanh tiêu đề thì bạn bổ sung nội dung "Control Excel" như dưới đây:

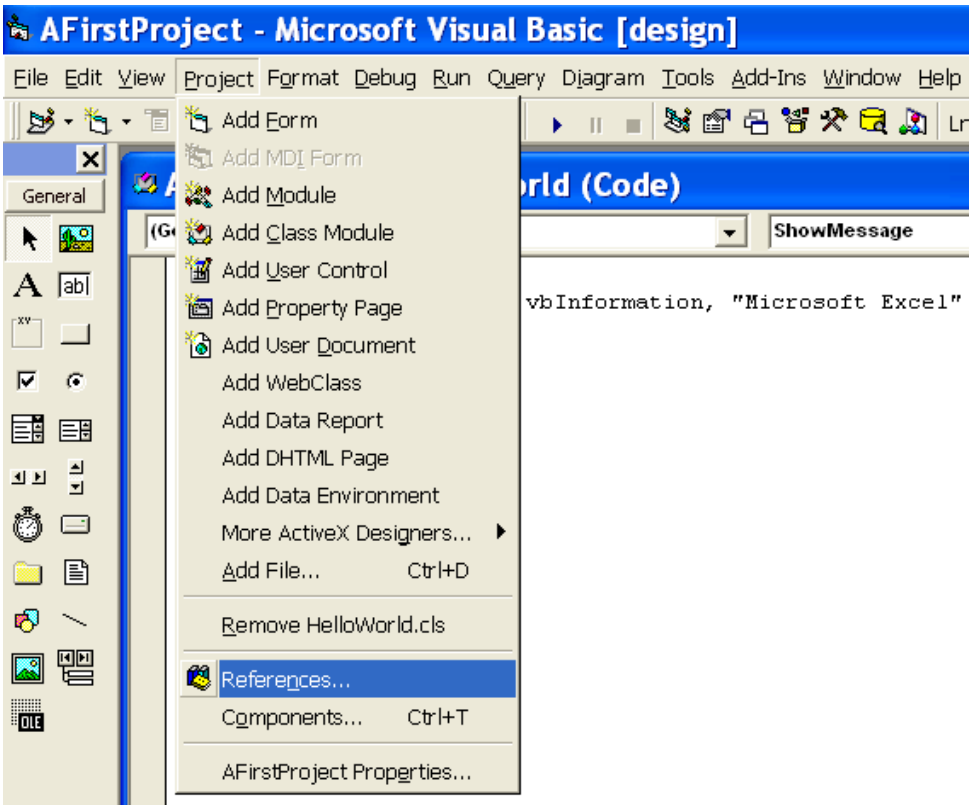
```
MsgBox "Hello World!", vbInformation, "Control Excel"
```

24.1.2. Cách tạo liên kết hoàn thiện ActiveX DLL hai hướng

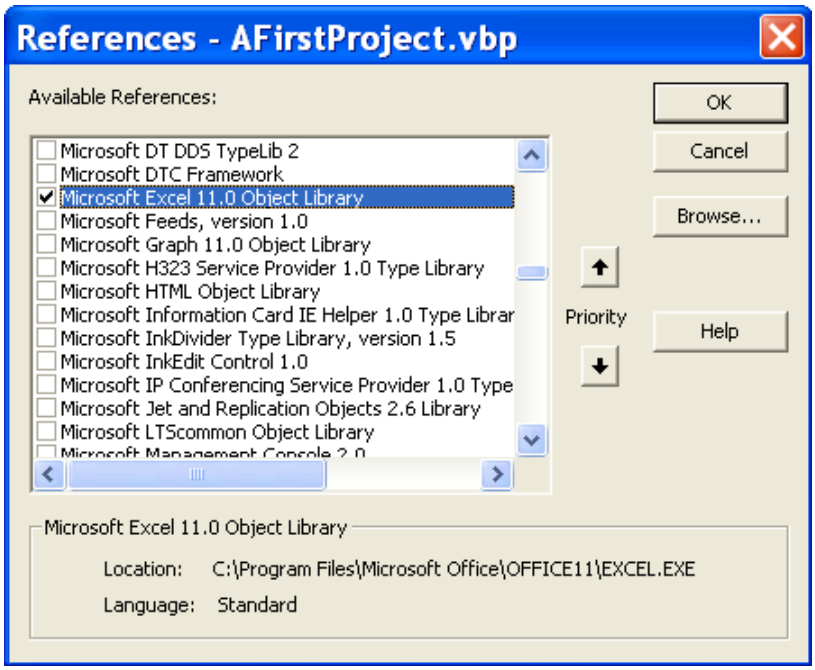
Dự án đầu tiên cho thấy cách tạo ActiveX DLL khá đơn giản. Kết quả Excel có thể thực liên kết với VB6 qua DLL. Nhưng việc trao đổi trên bị giới hạn bởi vì tất cả các liên kết chỉ là một hướng, Excel gọi DLL. Để nâng cao khả năng liên kết giữa Excel với VB 6.0, phải tạo ra cấu trúc mà cho phép liên kết được truyền theo cả hai hướng. Trong dự án "Hello World" ở trên, chúng ta mở rộng để ứng dụng có thể liên kết theo cả hai hướng, trong đó DLL là đối tượng chính để các thành phần tham chiếu tới. DLL không thể tạo ra các hành động trong nó mà chỉ đáp ứng những yêu cầu từ các ứng dụng khác sử dụng nó để làm việc. Sau đó yêu cầu sẽ được thực hiện bởi vì DLL có thể liên kết trực tiếp với ứng dụng đã gọi nó ra. Việc đầu tiên chúng ta sẽ làm cho có thể liên kết hai hướng giữa Excel và DLL bằng cách cung cấp DLL với hướng để nhận biết được chương trình nào gọi.

Bước đầu tiên cài đặt đối tượng tham chiếu Excel trong VB 6.0, gần như giống hệt với cách cài đặt tham chiếu trong VBA tại nội dung 23.4.2. Chỉ có điều khác là vị trí của thực

đơn. Để thiết lập tham chiếu tới Excel từ VB 6.0 ActiveX DLL, vào menu Project và chọn References... (hình 24-16). Hình 24-16 thể hiện tham chiếu từ dự án VB 6.0 tới thư viện đối tượng Excel 11.0 (tương ứng Office 2003), còn nếu làm việc với phiên bản trước hoặc sau đó thì số hiệu sẽ khác đi (giảm hoặc tăng lên).



Hình 24-16: Tạo tham chiếu trong VB 6.0



Hình 24-17: Chọn thư viện đối tượng Excel

Bây giờ DLL đã tham chiếu tới thư viện đối tượng trong Excel, bạn bổ sung thêm code trong VBA để cho phép DLL liên kết với Excel đã được tham chiếu đến.

Để giải thích quá trình xử lý trong dự án "Hello World", bạn sẽ lập thêm một phương thức mới để khi gọi ra, chuỗi "Hello World!" sẽ xuất hiện tại ô đang được chọn của worksheet hiện hành. Bạn tạo ra biến mới để tham chiếu và điều khiển đối tượng bên trong Excel, cùng những thuộc tính mới mà Excel có thể sử dụng từ DLL.

Bởi vì DLL hiện tại phải mượn tham chiếu tới ứng dụng bên ngoài nên bạn phải chắc chắc rằng tham chiếu đó sẽ bị mất đi khi kết thúc chương trình để giải phóng bộ nhớ. Sử dụng sự kiện Class_Terminate để hoàn thành điều này. Bạn thêm đoạn mã lệnh vào Class Module như sau:

```
Option Explicit              'Yêu cầu toàn bộ biến phải khai báo
'Khai báo biến mxlApp, là biến sẽ gọi ứng dụng Excel
   Private mxlApp As Excel.Application
'Đặt thuộc tính của class ExcelApp
Public Property Set ExcelApp(ByRef xlApp As Excel.Application)
   Set mxlApp = xlApp
End Property
'Xoá bỏ biến mxlApp sau khi kết thúc công việc
```

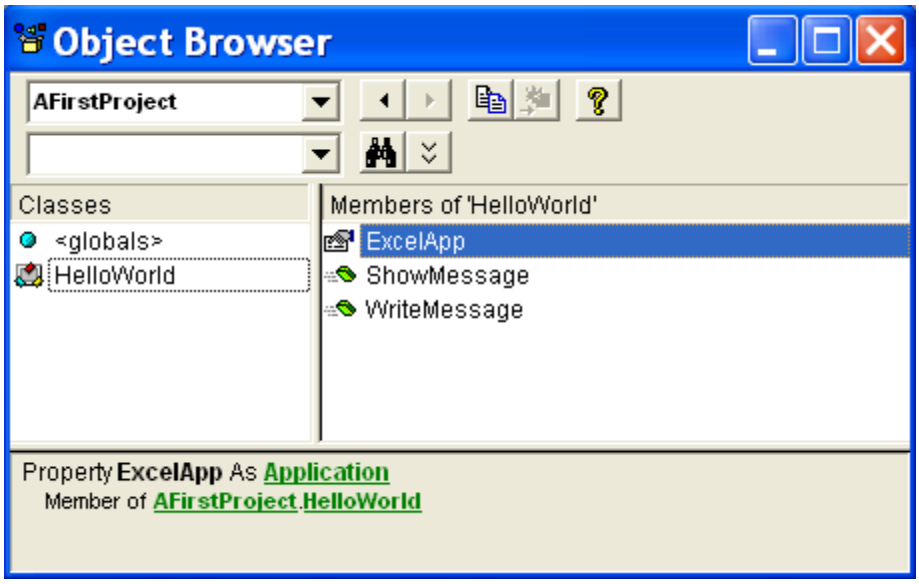
```
Private Sub Class_Terminate()
   Set mxlApp = Nothing
End Sub

'Các phương thức trong Class hoạt động
Public Sub ShowMessage()
   MsgBox "Hello World!", vbInformation
End Sub

Public Sub WriteMessage()
   mxlApp.ActiveCell.Value = "Hello World!"
End Sub
```

Tôi giải thích các biến như sau:

- Biến mxlApp sẽ chứa đựng tham chiếu tới đối tượng trong Excel qua DLL.
- Thuộc tính ExcelApp trên được sử dụng bởi ứng dụng Excel gọi DLL cung cấp tham chiếu tới nó.
- Sự kiện Class_Terminate thực hiện khi chấm dứt làm việc với Class Module.
- Phương thức WriteMessage (bản chất là thủ tục được xây dựng trong Class Module "HelloWorld") sẽ gán nội dung "Hello World!" vào ô hiện hành trong worksheet của Excel khi gọi phương thức đó từ VBA.

Và bây giờ, bạn biên dịch lại DLL trên. Trong trường hợp cửa sổ Excel chứa DLL cũ vẫn đang mở thì VB 6.0 sẽ không cho phép biên dịch cho đến khi cửa sổ Excel được đóng. Biểu hiện là hộp báo lỗi "Permission denied:" khi bạn biên dịch. Sau khi Excel đã tải được DLL, DLL luôn tồn tại trong bộ nhớ cho đến khi bạn thoát khỏi Excel. Nếu chỉ đóng workbook thì vẫn chưa được mà phải thoát ra khỏi hẳn Excel.

Sau khi bạn đã đóng Excel, vào menu File và chọn Make AFirstProject.dll như hình 24-7. Khi đó AFirstProject.dll sẽ biên dịch lại và hỏi có thay thế bản cũ hay không, bấm vào Yes để thay thế dự án cũ. Cửa sổ trình duyệt đối tượng sẽ thể hiện các đối tượng trong thư viện AFirstProject như hình 24-18 dưới đây.



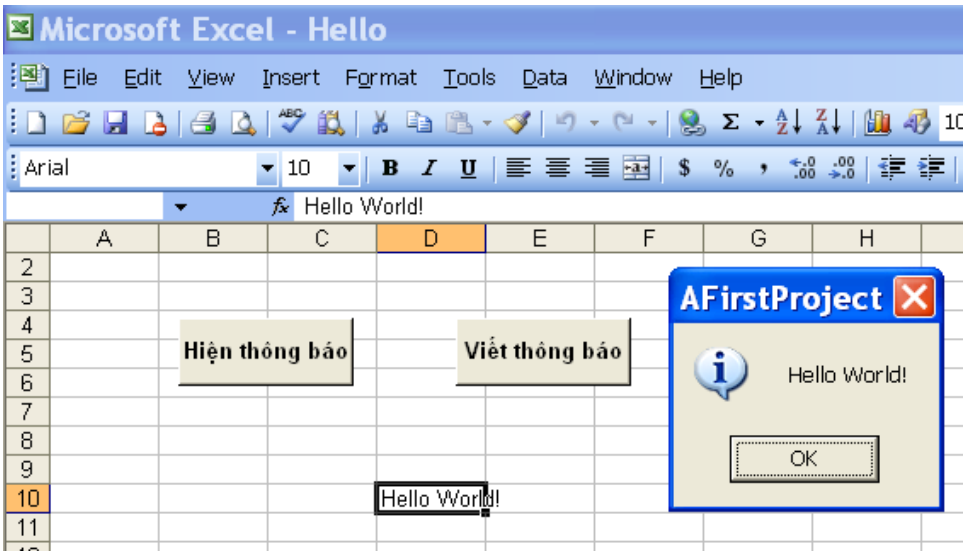
Hình 24-18: Các đối tượng trong thư viện mới AFirstProject

Bây giờ bạn đã có biên dịch mới DLL, cần phải bổ sung thủ tục trong Excel để thi hành phương thức mới WriteMessage. Bạn mở file Hello.xls đã lưu dự án trước đó và bổ sung thêm thủ tục dưới đây:

```
Public Sub WriteDLLMessage()
   Dim HelloWorld As AFirstProject.HelloWorld
   Set HelloWorld = New AFirstProject.HelloWorld
   Set HelloWorld.ExcelApp = Application
   HelloWorld.WriteMessage
   Set HelloWorld = Nothing
End Sub
```

Sự khác nhau cơ bản giữa thủ tục WriteDLLMessage và thủ tục trước ShowDLLMesage là sử dụng thuộc tính mới ExcelApp trong Class Module "HelloWorld" để cho phép tham chiếu tới đối tượng Excel vào trong Class. Điều đó nói lên Class mà ứng dụng gọi tới và ứng dụng thực hiện công việc đó sẽ liên kết trực tiếp trở lại.

Sau đó, trong Sheet1 của Hello.xls bạn xây dựng hai nút điều khiển. Nút điều khiển "Hiện thông báo" gắn với thủ tục ShowDLLMessage, nút "Viết thông báo" gắn với thủ tục WriteDLLMessage. Kết quả thể hiện trên hình 24-19.



Hình 24-19: Kết quả chạy hai thủ tục WriteDLLMessage và ShowDLLMessage

```
(General)
Public Sub ShowDLLMessage()
  Dim HelloWorld As AFirstProject.HelloWorld
  Set HelloWorld = New AFirstProject.HelloWorld
  HelloWorld.ShowMessage
  Set HelloWorld = Nothing
End Sub
Public Sub WriteDLLMessage()
  Dim HelloWorld As AFirstProject.HelloWorld
  Set HelloWorld = New AFirstProject.HelloWorld
  Set HelloWorld.ExcelApp = Application
  HelloWorld.WriteMessage
  Set HelloWorld = Nothing
End Sub
```

Hình 24-20: Hai thủ tục xây dựng trong Hello.xls

24.1.3. Hiển thị Form của VB 6.0 trong Excel

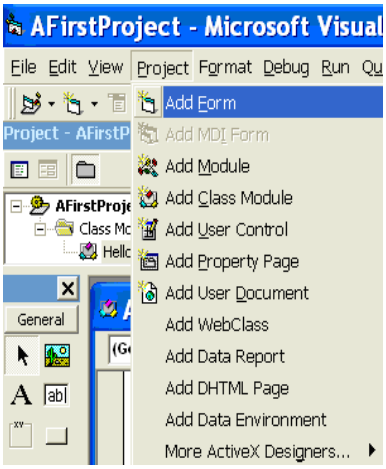
Tiếp theo bạn có thể thực hiện những thao tác hoàn thiện hơn trong ứng dụng VB 6.0 Hello World, đó là hiển thị Form của VB 6.0 trong Excel giống như là UserForm của VBA. Bây giờ bạn hãy để những rắc rối sang một bên và xem cách tối thiểu nhất để hiển thị Form của VB 6.0 trong Excel.

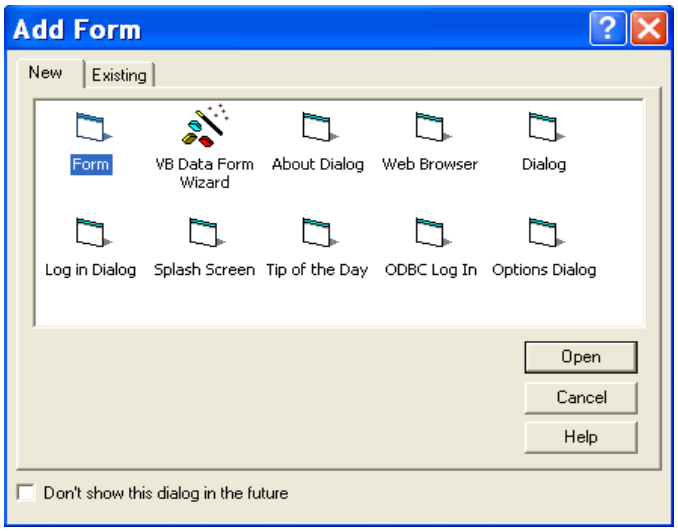
Bước đầu tiên mở dự án "Hello World" (AFirstProject) trong VB 6.0, sau đó bổ sung thêm Form bằng cách vào menu Project và chọn Add Form, sau đó bấm Open. Đối tượng Form mới sẽ được bổ sung vào dự án của bạn. Trước khi lưu giữ dự án của bạn với Form mới

bên trong, cần thiết phải thay đổi một số thuộc tính trong Form. Nội dung một số thay đổi như bảng 24-2 dưới đây:

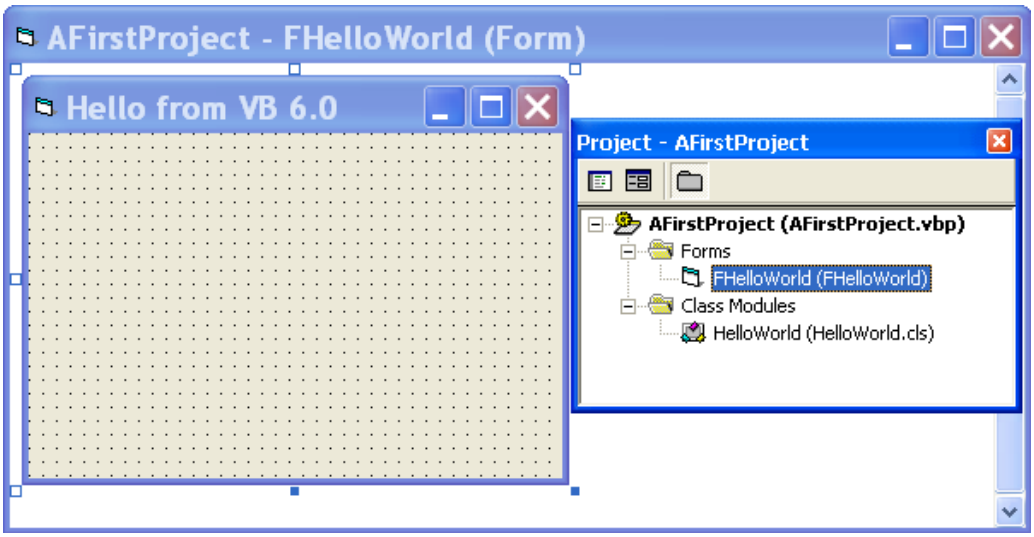
Bảng 24-2: Khai báo thuộc tính của đối tượng Form FHelloWorld

Thuộc tính	Đổi thành
(Name)	FHelloWorld
BorderStyle	3 - Fixed Dialog
Caption	Hello From VB 6.0
Icon	(None) (bạn xoá giá trị mặc định)
StartupPosition	1 - Center Owner





Hình 24-21: Tạo Form trong VB 6.0

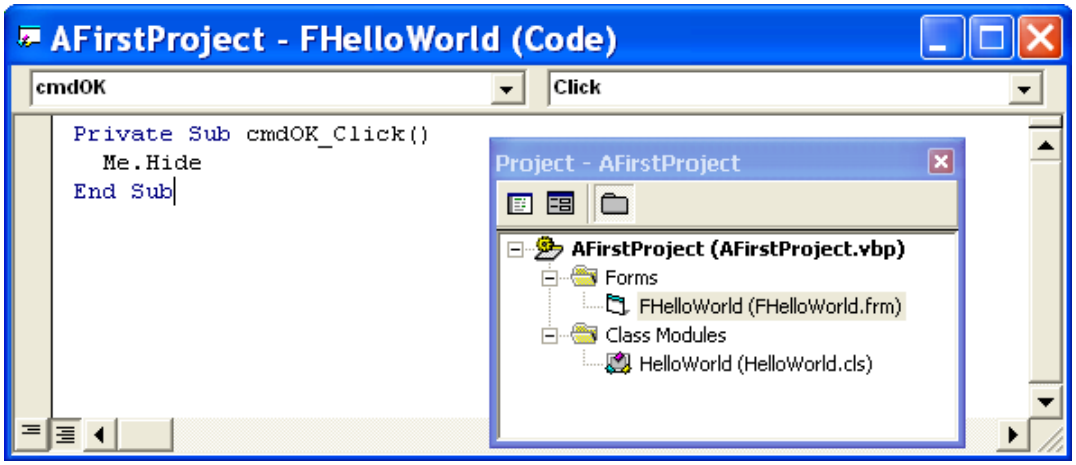


Hình 24-22: Form được tạo ra và cấu trúc mới của dự án AFirstProject

Sau khi đã thay đổi một số thuộc tính trên, hãy lưu dự án lại. Vì Form chưa được lưu nên sẽ có hộp thông báo lưu lại với tên mới (mặc định trùng với tên trong thuộc tính Name). Bạn hãy giữ nguyên tên và chọn vị trí cùng thư mục với dự án trước. Kết quả thực hiện như hình 24-22. Tiếp theo bổ sung điều khiển CommandButton và Label trong Form mới. Tên (Name) của CommandButton đặt là cmdOK, thuộc tính Caption đặt là OK. Thuộc tính Caption của Label đặt là “Hello World!”, thuộc tính Alignment đặt là 2 - Center, thuộc tính Font đặt là 24. Bạn đã hoàn thành công việc xây dựng Form giống như hình 24-23.

Hình 24-23: Form của dự án đã được thiết kế xong

Bạn tạo một thủ tục khi bấm chuột vào nút OK, Form sẽ biến mất. Cách thực hiện là bấm đúp chuột vào nút OK, thủ tục cmdOK_Click hiện ra, bạn thực hiện như hình dưới đây:



Hình 24-24: Gán thủ tục vào sự kiện bấm nút OK

Đối tượng Form trong VB 6.0 ActiveX DLL không sử dụng được trực tiếp ngoài ứng dụng. Bởi vậy cần phải thêm mã lệnh trong Class Module để cho phép ứng dụng Excel của bạn hiện được Form này. Cần phải tạo ra cái điểm ngắt tới Form để Excel cư xử giống như là UserForm. Thủ tục hoàn chỉnh của HelloWorld class được thể hiện với mã lệnh mới bổ sung.

```
Option Explicit

'Khai báo hằng số SetWindowLongA API
Private Const GWL_HWNDPARENT As Long = -8
'Khai báo biến mxlApp, là biến sẽ gọi ứng dụng Excel
Private mxlApp As Excel.Application
'Window truy cập vào bộ nhớ phát triển để gọi ứng dụng Excel
Private mlXLhWnd As Long
'Khai báo hàm API ở dưới đây
Private Declare Function FindWindowA Lib "user32"
   (ByVal lpClassName As String, ByVal lpWindowName As String) As Long
Private Declare Function SetWindowLongA Lib "user32" (ByVal hWnd As
  Long, ByVal nIndex As Long, ByVal dwNewLong As Long) As Long
'Đặt thuộc tính của class ExcelApp
```

```
Public Property Set ExcelApp(ByRef xlApp As Excel.Application)
   Set mxlApp = xlApp
'Lấy handle của cửa sổ Excel ngay khi vừa thực hiện thành công

   mlXLhWnd = FindWindowA(vbNullString, mxlApp.Caption)
End Property
'Xoá bỏ biến mxlApp sau khi kết thúc công việc
Private Sub Class_Terminate()
   Set mxlApp = Nothing
End Sub

'Các phương thức trong Class hoạt động
Public Sub ShowMessage()
   MsgBox "Hello World!"
End Sub

Public Sub WriteMessage()
   mxlApp.ActiveCell.Value = "Hello World!"
End Sub

Public Sub ShowVB6Form()
   Dim frmHelloWorld As FHelloWorld
   Set frmHelloWorld = New FHelloWorld
   Load frmHelloWorld        'Tải Form
   'Cửa sổ Form mẹ tới cửa sổ ứng dụng Excel
   SetWindowLongA frmHelloWorld.hWnd, GWL_HWNDPARENT, mlXLhWnd
   frmHelloWorld.Show vbModal
   Unload frmHelloWorld
   Set frmHelloWorld = Nothing
End Sub
```

Form trong VB 6.0 là cửa sổ được mặc định, có thể nói rằng đó cửa sổ con trên màn hình nền. Để tạo VB 6.0 Form hoạt động giống như UserForm trong Excel, bạn cần phải sử dụng Windows API để thay đổi cửa sổ mẹ của VB 6.0 Form từ màn hình nền tới cửa sổ đối tượng ứng dụng Excel. Để thay đổi cửa sổ mẹ của Form, bạn thực hiện 2 bước sau:

- Lấy lại handle (sự điều khiển) của cửa sổ mà bạn muốn sử dụng như cửa sổ mẹ của Form. Trong Microsoft Windows, vùng lưu trữ đặc biệt trong bộ nhớ (global heap) sẽ lưu các đối tượng trong bộ nhớ. Mỗi đối tượng đều được gán một handle.

- Thay đổi cửa sổ mẹ của Form bằng cách đặt handle của cửa sổ mà bạn đã lấy vào khu vực lưu trữ cửa sổ cấu trúc Form, mà được sử dụng để chỉ định cửa sổ mẹ của Form.

Bạn sẽ phải khai báo biến mới mlXLhWnd để giữ được handle của cửa sổ ứng dụng Excel. Sau đó phải khai báo thêm hai hàm API: FindWindowA định vị handle ở cửa sổ ứng dụng Excel, và SetWindowLongA để thay đổi cửa sổ mẹ của Form. Hằng số mới GWL_HWNDPARENT sẽ xác minh vị trí cửa sổ cấu trúc Form của bạn nằm ở đâu trong cửa sổ mẹ mới sẽ được thay đổi.

Như vậy, chúng ta đã tạo phương thức mới ShowVB6Form mà sẽ hiển thị trong cửa sổ Excel. Phương thức đó được sử dụng theo từng bước để hoàn thành công việc như sau:

- Tạo ra form mới FHelloWorld.

- Tải form FHelloWorld vào trong bộ nhớ.

- Thay đổi cửa sổ mẹ của form từ cửa sổ màn hình nền sang cửa sổ ứng dụng Excel bằng cách sử dụng hàm API SetWindowLongA.

- Hiện form bằng cách sử dụng cờ hiệu vbModal. Không giống như UserForm, Form của VB 6.0 sẽ hiện ra với mẫu được mặc định. Đây không phải là điều bạn thực hiện trong ví dụ này, bởi vì bạn sử dụng cờ hiệu vbModal của phương thức Show.

Khi đã bổ sung thêm mã lệnh trong dự án VB 6.0, bạn nhớ phải đóng cửa sổ ứng dụng trước khi biên dịch lại DLL của bạn. Trong file Hello.xls, bạn tạo thủ tục DisplayDLLForm và gán vào một nút điều khiển “Hiện Form VB 6.0”. Kết quả thủ tục này thực hiện như hình (hình 24-25).

```
Public Sub DisplayDLLForm()
   Dim HelloWorld As AFirstProject.HelloWorld
   Set HelloWorld = New AFirstProject.HelloWorld
   Set HelloWorld.ExcelApp = Application
   HelloWorld.ShowVB6Form
   Set HelloWorld = Nothing
End Sub
```

Bạn sẽ thấy mã lệnh trong DisplayDLLForm sẽ rất giống như trong WriteDLLMessage đã lập ở dự án lúc trước đó. Điều chỉ khác là sử dụng hai phương thức ShowVB6Form và WriteMessage trong class HelloWorld.

Hình 24-25: Form Hello From VB 6.0 hiện ra trong ứng dụng Excel

Ghi chú: Kết nối giữa hai ứng dụng khác nhau như VBA trong Excel và VB 6.0 sử dụng hai phương thức kết nối khác nhau. Hai phương thức đó được hiểu như là liên kết in-process (xử lý trong) và out-of-process (xử lý ngoài).

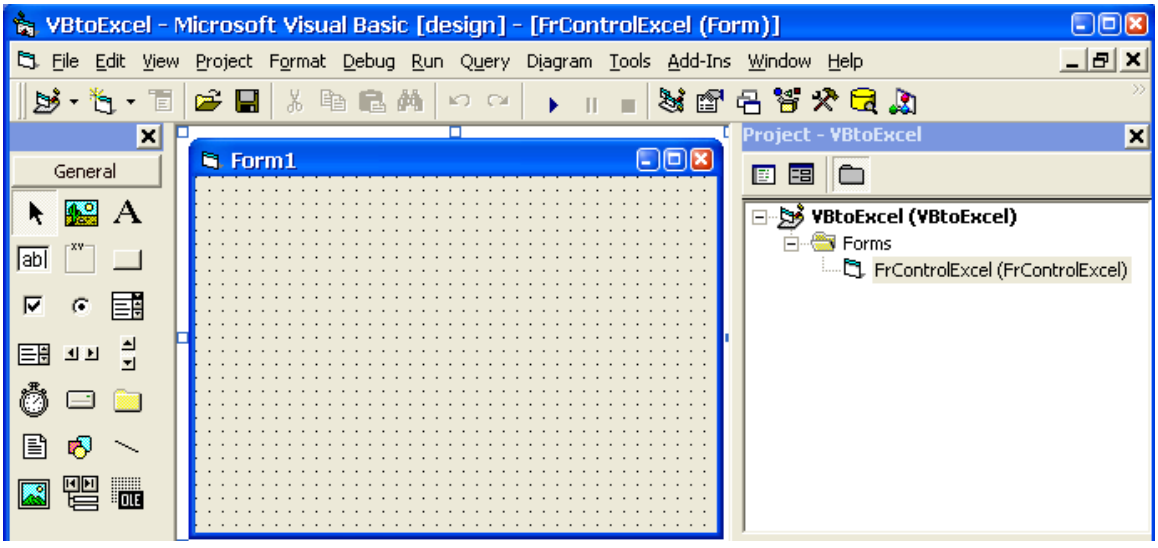
- Liên kết in-process: xuất hiện khi hai ứng dụng chạy trong vùng bộ nhớ gần như nhau được định phần bởi Window. ActiveX DLL là một dạng ứng dụng của VB 6.0 sẽ chạy cùng ứng dụng VBA trong Excel. Khi VBA gọi ActiveX DLL của VB 6.0, Window sẽ tải DLL vào trong vùng bộ nhớ giống nhau mà đã được định phần cho VBA. Khi hai ứng dụng chia sẻ bộ nhớ giống nhau, liên kết giữa chúng rất nhanh.
- Liên kết out-of-process: xuất hiện khi hai ứng dụng chạy trong vùng bộ nhớ khác nhau được định phần bởi Window. Đó chính là sự liên kết giữa hai ứng dụng Standard EXE. Window sẽ tải tất cả ứng dụng Standard EXE vào trong các vùng bộ nhớ tách rời. Điều đó không cho phép hai ứng dụng Standard EXE chia sẻ vùng bộ như nhau. Bởi vậy khi sử dụng Standard EXE của VB 6.0 với ứng dụng VBA của Excel, chúng sẽ chạy trong các

vùng bộ nhớ khác nhau. Kết quả là đường truyền dẫn thực thi sẽ bị mất đi. Liên kết out-of-process chạy vài kiểu với cường độ chậm hơn so với liên kết in-process.

24.2. Điều khiển Excel từ Standard EXE của VB 6.0

Kiểu phổ biến nhất của ứng dụng là điều khiển Excel từ Standard EXE của VB 6.0. Trong mục này, chúng ta sẽ nghiên cứu sự liên kết giữa Standard EXE của VB 6.0 với ứng dụng Excel. Điều khiển Excel từ Standard EXE của VB 6.0 đơn giản hơn là sử dụng ActiveX DLL (đã trình bày ở mục trước). So với sự liên kết dữ liệu giữa các chương trình trong Window, thì việc điều khiển Excel từ VB 6.0 không khác gì điều khiển các chương trình khác từ Excel. Ví dụ như bạn có thể liên kết với Excel từ VBA của AutoCad, Word hay PowerPoint,... (Xem thêm tại mục 23.6).

Để bắt đầu với dự án này, bạn hãy khởi động VB 6.0 và chọn Standard EXE từ cửa sổ New Project (hình 24-1). Dự án mới đã sẵn có Form1 mặc định. Sau đó bạn đổi tên của dự án thành VBtoExcel, đổi tên form thành FrControlExcel và lưu lại dự án đó. Trong cửa sổ Project của VB 6.0, dự án của bạn trông như hình 24-26 dưới đây. Sau đó tiến thành thiết lập tham chiếu tới Excel bằng cách vào menu Project và chọn References..., di chuyển xuống và chọn Microsoft Excel xx.0 Object Libary (hình 24-16).



Hình 24-26: Dự án VBtoExcel với Form1

Dự án ban đầu có một form, trong form đó tiến hành xây dựng 7 điều khiển cùng với thuộc tính của chúng như sau:

Đối tượng	Thuộc tính	Đổi thành
Form1	Name	FrControlExcel
	Caption	Control Excel
	BorderStyle	1 – Fixed Single
Label	Caption	Nhập nội dung
	AutoSize	True
	Font	MS Sans Serif, cỡ chữ 8
TextBox1	Name	txtNoidung
	Text	Để trống
Frame	Name	frmLuachon
Checkbox1	Name	chkChudam
	Value	0 - Unchecked
Checkbox2	Name	chkNghieng
	Value	0 - Unchecked
CommandButton1	Name	cmdThuchien
	Caption	Thực hiện
	Default	True
	Font	MS Sans Serif, cỡ chữ 8, đậm
CommandButton2	Name	cmdHuybo
	Caption	Hủy bỏ
	Default	False
	Font	MS Sans Serif, cỡ chữ 8, đậm

Hình 24-27 dưới đây là kết quả công việc thực hiện xây dựng và thay đổi thuộc tính của các điều khiển ở trên.

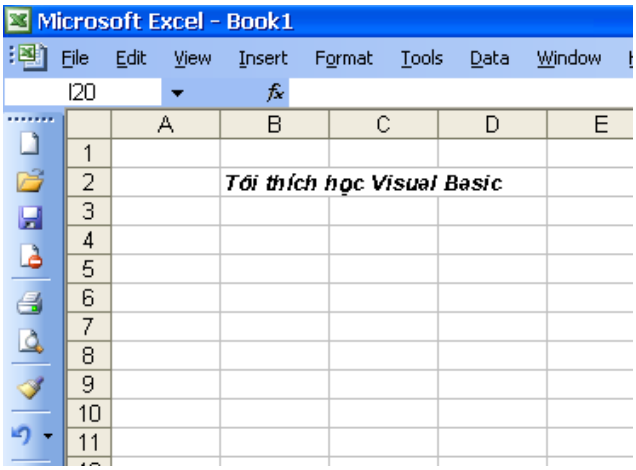
Hình 24-27: Form FrControlExcel được thiết kế

Sau đó tiến hành xây dựng thủ tục cho hai điều khiển là nút Thực hiện và Huỷ bỏ như sau (bằng cách bấm kép chuột vào biểu tượng điều khiển, tương tự như trong VBA):

```
Private Sub cmdThuchien_Click()
   Dim ExcelApp As Excel.Application
   Dim NewWB As Workbook, Ogiatri As Range

   On Error Resume Next  'Bỏ qua lỗi khi Excel chưa mở
   'Khi Excel đang mở
   Set ExcelApp = GetObject(, "Excel.Application")
   'Lỗi xuất hiện khi Excel chưa mở
   If Err Then
      Err.Clear
      Set ExcelApp = CreateObject("Excel.Application")
   End If
   'Thêm 1 workbook mới, đặt là NewWB
   Set NewWB = ExcelApp.Workbooks.Add
   'Nếu không nhập nội dung cho txtNoidung thì sẽ báo lỗi
   If Me.txtNoidung = vbNullString Then
        MsgBox "Yêu cầu nhập dữ liệu"
        Exit Sub
     Else
        Set Ogiatri = NewWB.Sheets(1).Range("B2")
        With Ogiatri
           'Gán giá trị txtNoidung vào ô B2
           .Value = Me.txtNoidung
```

```
            'Cho chữ trong ô đậm hoặc nhạt
            If Me.chkChudam = 1 Then
               .Font.Bold = True
              Else
               .Font.Bold = False
            End If
            'Cho chữ trong ô thẳng hoặc nghiêng
            If Me.chkNghieng = 1 Then
               .Font.Italic = True
               Else
               .Font.Italic = False
            End If
         End With
         Unload Me
   End If
   'Hiện cửa sổ Excel
   ExcelApp.Visible = True
   'Giải phóng biến đối tượng
   Set NewWB = Nothing
   Set ExcelApp = Nothing
End Sub


Private Sub cmdHuybo_Click()
   Unload Me
End Sub
```

Hình 24-28: Form Control Excel hiển thị

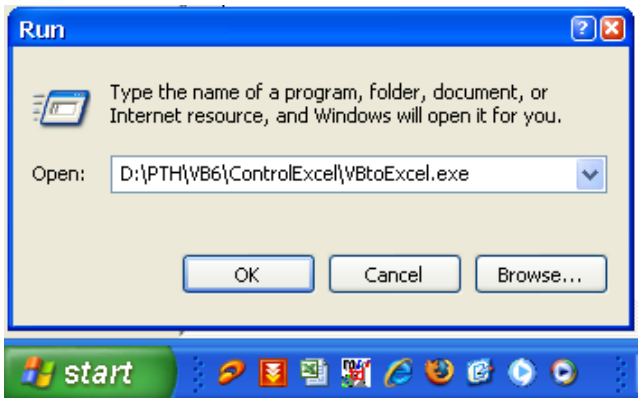


Hình 24-29: Kết quả thực hiện điều khiển Excel

Khi xây dựng thủ tục trên, bạn sẽ thấy môi trường thực hiện giống như trên VBA với đầy đủ sự hỗ trợ như Auto List Members và các công cụ khác của VB6. Bây giờ bạn bấm phím F5, Form điều khiển hiển thị như ở hình dưới:

Sau khi nhập nội dung và chọn kiểu thể hiện, bạn bấm nút Thể hiện hoặc phím Enter (đã mặc định). Cửa sổ Excel hiện ra với nội dung chữ và kiểu thể hiện như mong muốn.

Khi đã kiểm tra, chạy thử và vá lỗi, tức là chương trình đã hoàn thiện, bạn chọn menu File/VBtoExcel.exe... VB6.0 sẽ tạo ra một file chương trình VBtoExcel.exe thi hành một cách độc lập. Bạn có thể chạy file VBtoExcel.exe bằng cách bấm đúp chuột vào tại cửa sổ Window hoặc từ Run (hình 24-30). Kết quả thực hiện tương tự như trên.



Hình 24-30: Thực hiện VBtoExcel.exe bằng Run

Bạn có thể thắc mắc sử dụng file chương trình VBtoExcel.exe thực hiện ở các máy tính khác được không?. Câu trả lời là có, với điều kiện máy tính đó phải cài bộ Office 2003. Nếu

sử dụng Office phiên bản khác thì sẽ báo lỗi. Do vậy, để linh hoạt cho các phiên bản Office, khi đã xây dựng chương trình hoàn thiện, bạn nên khai báo các biến đối tượng là Object chung (kiểu Late Binding) thay vì các đối tượng cụ thể như Workbook, Range (kiểu Early Binding). Xem thêm tại mục 23.4.

Nguồn tài liệu này tôi tham khảo chủ yếu từ "**Professional Excel Development:** *The Definitive Guide to Developing Applications Using Microsoft® Excel and VBA® By Stephen Bullen, Rob Bovey, John Green*".