1 Mục lục

2 Làm quen với visual basic 6.0

2.1 Xây dựng ứng dụng ban đầu

2.1.1 Viết ứng dụng ban đầu

Cách tốt nhất để học lập trình là viết chương trình. Vậy hãy thử viết chương trình hiển thị lịch biểu, trong đó cho phép người sử dụng:

Hiển thị lịch biểu của tháng hiện hành

Duyệt qua các tháng

Hiển thị đồng hồ báo giờ hiện hành

Nếu bạn cho rằng chương trình này có vẻ nặng nề cho người mới học, đừng lo lắng. Visual basic làm hết mọi việc cho bạn. Khác với ngôn ngữ C++, bạn phải viết mỗi thứ một ít, Visual basic cung cấp mức độ cao hơn của lập trình tự động. Như vậy, bạn có thể làm nhiều thứ mà không phải lập trình nhiều.

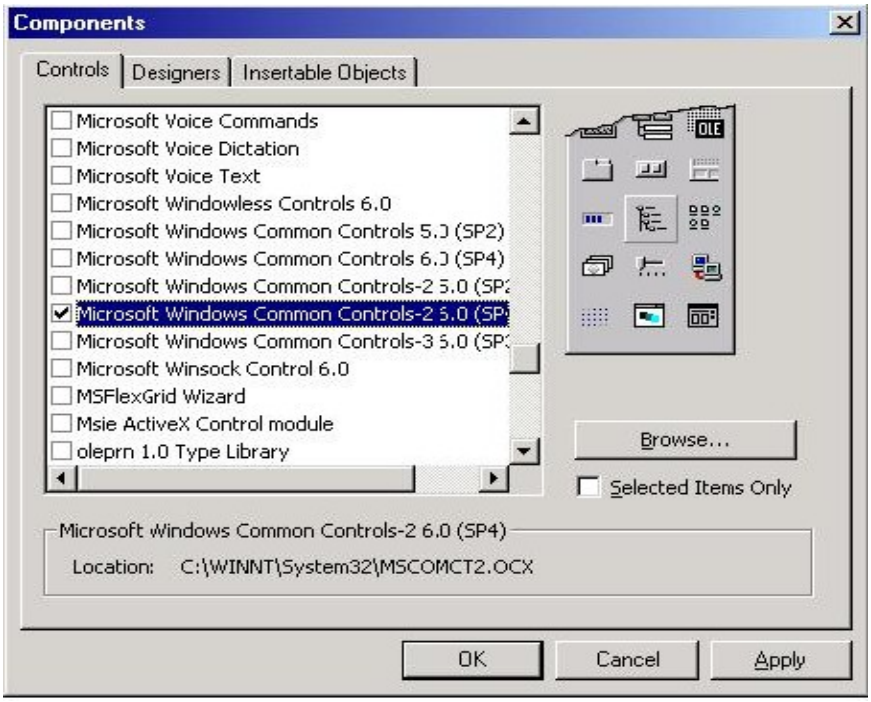
Tuy nhiên, đừng hiểu sai "không lập trình nhiều" nghĩa là "không có nhiều tính năng mạnh" Visual basic là một ngôn ngữ rất mạnh. Ta có thể lập trình để làm "mọi thứ" nếu cần. Ta cũng có thể khai thác khả năng tự động của Visual basic để viết chương trình thật nhanh. Chọn lựa là ở người lập trình. Visual basic đủ linh hoạt để hỗ trợ cho người lập trình từ người mới học đến lập trình chuyên nghiệp.

2.1.2 Xây dựng tính năng Calendar

Bây giờ ta bắt đầu xây dựng các tính năng của ứng dụng. Đầu tiên, ta cần một lịch biểu. Ta có thể tự tạo nó hoặc sử dụng lịch biểu có sẵn của Visual basic (đây là một điều khiển ActiveX). Ta chọn cách thứ 2.

Từ menu Project, chọn Components. Bởi vì, mặc định tất cả các điều khiển ActiveX của Visual basic không được nạp tự động. Muốn dùng bạn phải chọn từ menu Components.

Trong hộp thoại Components chọn Windows Common Controls 2.6.0 và nhấn OK.



ActivateX là gì?

ActivateX là sản phẩm của Microsoft cho phép ta tạo những chương trình nhỏ, gọi là các thành phần(componet) và các điều khiển (control) để có thể thêm vào các chương trình lớn. Đó có thể là các chương trình độc lập (Standalone program) hay các chương trình chạy trên Internet. Ta có thể dùng Visual basic để tự tạo các điều khiển ActivateX. Phần này sẽ được trình bày trong một riêng.

Đến đây điều khiển lịch được nạp vào thanh công cụ. Tên chính thức của nó là điều khiển ActivateX MonthView.

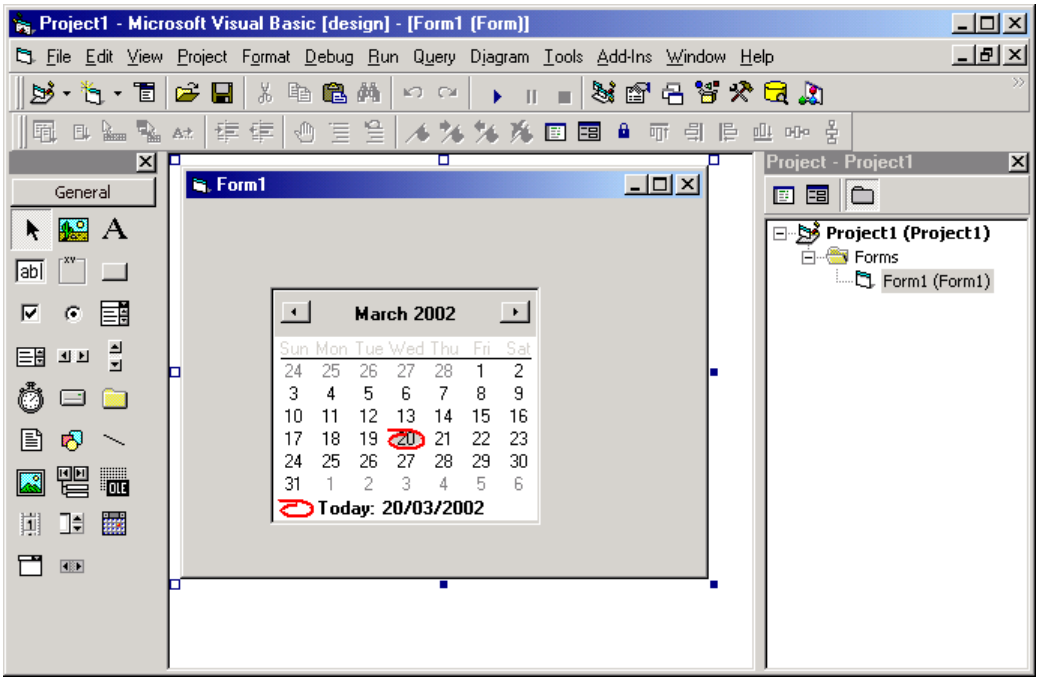
Kế tiếp ta đưa điều khiển vào biểu mẫu.

Thêm điều khiển MonthView vào biểu mẫu

Chọn biểu tượng điều khiển MonthView từ hộp công cụ

Nhấn đúp chuột lên biểu tượng điều khiển để đưa nó vào biểu mẫu.

Bạn vừa tạo xong chương trình nhấn F5 để chạy.



2.1.3 Thêm tính năng Clock

Tính năng kế tiếp là hiển thị giờ. Ta sẽ dùng điều khiển ActivateX Timer là một điều khiển nội tại của Visual basic. Điều khiển nội tịa luôn được nạp sẵn trong hộp công cụ.

Thêm Timer vào ứng dụng

Chọn vào biểu tượng trong hộp công cụ

Nhấn chuột lên điều khiển Timer trên hộp công cụ để đưa nó vào biểu mẫu.

Để hiển thị thời gian ta lập trình điều khiển Timer. Ta dùng thuộc tính Interval của timer để quy định việc đếm thời gian. Một đơn vị của Interval là 1/1000 giây. Do đó, để quy đinh nhịp đếm là nửa giây ta đặt Interval là 500. Cứ mỗi nửa giây chương trình sẽ làm một việc gì đó. Ở đây, ta muốn hiển thị thời gian hiện hành. Ta sẽ dùng nhãn (label) để hiển thị thời gian.

Hiển thị Timer

Thêm Label vào biểu mẫu.

Chọn điều khiển nhãn từ hộp công cụ và kéo nó vào biểu mẫu.

Thủ tục sự kiện là gì?

Một thủ tục sự kiện là một đoạn chương trình sẽ thi hành khi sự kiện đó xảy ra. Ví dụ, khi người sử dụng nhấn vào nút CommandButton, sự kiện click() sẽ được sinh ra. Visual basic cung cấp thủ tục CommandButton_Click cho ta lập trình để ứng dụng phản ứng đối với việc nhấn nút CommandBuuton. Tương tự với Timer. Khi đúng nhịp đếm Timer sự kiện Timer() sẽ phát ra.

Để hiển thị thời gian trên nhãn label ta đưa dòng lệnh sau vào thủ tục sự kiện của timer: Label1.Caption = time

```
Private Sub Timer1_Timer()
   Label1.Caption = Time
End Sub
```

2.2 Các tính năng mới trong Visual basic 6.0

2.2.1 Khái quát vắn tắt về Visual basic 6.0

Nếu bạn chưa quen với Visual basic, tựa đề của phần này có thể làm nhầm lẫn đôi chút. Rõ ràng rằng nếu bạn là người mới học ngôn ngữ, mọi thứ về Visual basic đều mới cả. Dù vậy, bạn không nên bỏ qua phần này, nhất là các điều khiển ActivateX mới. Đối với các bạn đã quen thuộc các phiên bản Visual basic trước thì phần này thật hữu ích.

Visual basic 6.0 có rất nhiều tính năng mới. Các điều khiển mới cho phép ta viết chương trình ứng dụng kết hợp giao diện, cách xử lý và tính năng của Office 97 và trình duyệt WEB internet explorer. Không nhất thiết phải có một instance của điều khiển trên biểu mẫu, Visual basic 6 cho phép ta lập trình để thêm điều khiển vào đề án tự động và ta có thể tạo ra các điều khiển ActivateX hiệu chỉnh.

Một vài cải tiến cho phép làm việc với các ứng dụng truy cập dữ liệu ở tầm cỡ vĩ mô liên quan đến hàng trăm hàng nghìn người sử dụng qua mạng Internet.

2.2.2 Khai thác thế mạnh của các điều khiển mở rộng

2.2.2.1 sự kiện Validate và thuộc tính CausesValidation

Phần mở rộng đầu tiên liên quan đến tất cả các điều khiển ActivateX cơ bản là việc thêm vào sự kiện Validate và thuộc tính CausesValidation. Trước các phiên bản Visual basic 6, nếu bạn kiểm tra tính hợp lệ của một từ nhập vào hộp văn bản, bạn phải viết thủ tục sự kiện LostForcus của TextBox. Nếu nhập sai bạn phải gọi phương thức SetForcus để buộc người dùng nhập lại dữ liệu đúng. Thỉnh thoảng logic của lập trình này làm người dùng khó chịu khi họ không bao giờ nhập đúng dữ liệu, họ có thể bị khoá chặt ở điều khiển đó họ cũng cũng không nhấn cả nút help để xem hướng dẫn chi tiết. Sự kiện Validate và thuộc tính CausesValidation giải quyết vấn đề đó.

2.2.2.2 Các cải tiến đồ hoạ mới làm ứng dụng thêm sinh động

Visual basic luôn cho phép bạn sử dụng đồ hoạ để làm chương trình sống động và Microsoft có khả năng đồ hoạ mở rộng cho nhiều điều khiển. Đầu tiên, điều khiển ImageList giờ đây hỗ trợ các tập tin.gif. Phần mở rộng này rất quan trọng bởi vì nó liên quan đến các điều khiển có sử dụng điều khiển ImageList.

Các điều khiển ListView và TabStrip có phần mở rộng cho phép sử dụng hình ảnh và biểu tượng để trang trí và minh hoạ. Điều khiển listView cho phép tạo một ảnh nền cho vùng làm việc. Ảnh nền có thể được đặt giữa trải đều hoặc đặt ở một góc bất kỳ.

2.2.2.3 Ngày tháng với điều khiển MonthView và DataTimepicker

VB6 có 2 điều khiển đưa ra giải pháp mới để xem xét và chọn lựa ngày tháng, MonthView và DataTimePicker. Điều lý thú của các điều khiển là chúng cho phép ta xem và chọn ngày trong ngữ cảnh lịch biểu. Điều khiển MonthView trình bày một lịch biểu đầy đủ để ta có thể duyệt theo từng ngày hoặc từng tháng. Điều

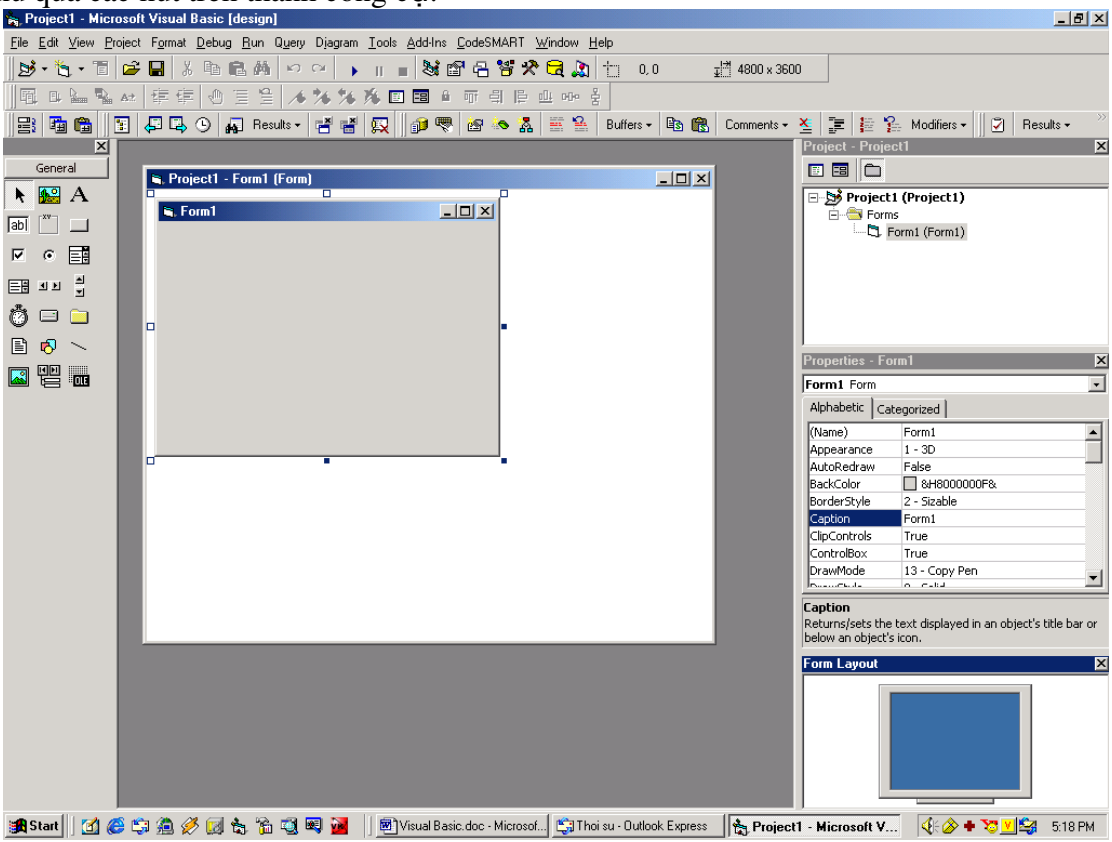
khiển DateTimePicker tương tự như MonthView, nhưng có điểm khác là lịch biểu sẽ thả xuống khi người dùng nhấn vào mũi tên xuống của điều khiển.

2.3 Làm việc với môi trường lập trình trong Visual basic

2.3.1 Tìm hiểu các phần của IDE

IDE là tên viết tắt của môi trường phát triển tích hợp (Inegreated Development Enviroment). IDE là nơi tạo ra các chương trình VB.

IDE của VB là nơi tập trung các menu, thanh công cụ và cửa sổ để tạo ra chương trình. Mỗi phần của IDE có các tính năng anhe hưởng đến các hoạt động lập trình khác nhau. Thanh menu cho phép bạn tác động cũng như quản lý trực tiếp trên toàn bộ ứng dụng. Thanh công cụ cho phép truy cập các chức năng của thanh menu qua các nút trên thanh công cụ.



Các biểu mẫu (Form) - khối xây dựng các chương trình Visual basic - xuất hiện trong cửa sổ form. Hộp công cụ để thêm các điều khiển vào các biểu mẫu của đề án. Project explorer hiển thị các đề án mà bạn đang làm cũng như các thành phần của các đề án. Bạn duyệt và cài đặt các thuộc tính của điều khiển, biểu mẫu và module trong cửa sổ property. Cuối cùng, bạn bố trí và xem xét một hoặc nhiều biểu mẫu trên màn hình thông qua cửa sổ form layout.

2.3.2 Thêm và xoá các thanh công cụ trong IDE của Visual basic

Thanh công cụ là tập hợp các nút bấm mang biểu tượng chứa trong một thanh thường đặt dưới thanh menu. Các nút này đảm nhận các chức năng thông dụng

trong cấu trúc menu của Visual basic. Thanh công cụ rất hửu ích, thay vì phải lần lượt chọn qua menu và menu con, ta nhấn một nút bấm nào đó trong thanh công cụ để gọi một chức năng tương tự trên menu.

Sử dụng thanh công cụ debug: Thanh công cụ debug dùng để kiểm tra chương trình và giải quyết các lỗi có thể xảy ra. Khi gỡ rối chương trình, ta làm một số việc như chạy từng dòng lệnh chương trình, kiểm tra giá trị các biến, và dừng chương trình tại một điểm nghi ngờ hoặc dưới những điều kiện nào đó.

Sử dụng thanh công cụ Edit: Thanh công cụ Edit được dùng để viết chương trình trong cửa sổ code. Các tính năng của thanh công cụ Edit tương tự như các tính năng khác ở menu edit. Bạn có thể Cut, Paste văn bản...

Một tính năng lý thú của IDE là thanh công cụ Edit dùng tính năng Coplete Word, tự động hoàn tất từ khoá. Tính năng Complete Word rất hữu ích để tránh các lỗi cú pháp.

Sử dụng thanh công cụ Form Editor: Thanh công cụ form editor dùng để kéo giãn, di chuyển và sắp xếp các điều khiển trên biểu mẫu. Thanh công cụ Form editor có các tính năng như menu Format.

Sử dụng thanh công cụ chuẩn(Standard): Là thanh công cụ trọng yếu trong IDE. Thanh công cụ chuẩn cung cấp nhiều tính năng trong menu **file, Project, Debug,** và **Run.**

2.3.3 Thêm các điều khiển vào hộp công cụ

Hộp công cụ là bảng chứa các điều khiển và ta thiết kế giao diện người sử dụng bằng cách chọn các điều khiển từ hộp công cụ và đưa chúng vào các biểu mẫu.

Một số điều khiển có sẵn trong Visual basic và không thể gỡ bỏ khỏi hộp công cụ. Một số khác nằm bên ngoài Visual basicvà chứa trong các tập tin mà có phần mở rộng là. ocx. Các điều khiển này có thể được thêm vào hoặc gỡ bỏ khỏi thanh công cụ.

Chúng ta sẽ trở lại chi tiết về các loại điều khiển trong Visual basic trong một chương riêng.

2.3.4 Định hướng thông qua cửa sổ form và code

Nếu điều khiển là những khối bê tông mà ta tập hợp trong ứng dụng thì biểu mẫu là nền móng để ta xây dựng các khối này.

Các biểu mẫu chứa trong cửa sổ Thiết kế biểu mẫu. Ta sẽ làm việc trong cửa sổ này để thêm các điều khiển vào biểu mẫu.

Đối với từng cửa sổ thiết kế mẫu, ta cũng có thể mở cửa sổ code. Cửa sổ code là nơi ta viết các đoạn chương trình chạy bên dưới biểu mẫu. Ta có thể mở cửa sổ code bằng cách nhấn đúp lên biểu mẫu hoặc điều khiển, hoặc chọn code từ menu.

2.3.5 Quản lý ứng dụng với project explorer

Project explorer trong Visual basic giúp quản lý và định hướng nhiều đề án. Visual basic cho phép tổ chức nhiều đề án trong một nhóm gọi là project group. Ta có thể lưu tập hợp các đề án trong Visual basic thành một tập tin nhóm đề án. Các tập tin này có phần mở rộng là.vbg.

2.3.6 Cửa sổ properties

Mỗi thuộc tính có một hoặc nhiều giá trị. Cửa sổ properties giúp bạn xem sửa đổi và điều khiển các thuộc tính của các điều khiển ActivateX trong chương trình.

2.3.7 Hiển thị IDE

Ta có thể xem IDE của Visual basic bằng 2 cách: MDI hoặc SDI. Hiển thị kiểu MDI(Multiple document interface) cho phép trình bày tất cả các cửa sổ thành phần trong IDE như là các cửa sổ con chứa trong một cửa sổ lớn.

Trái lại đối với hiện thị SDI(single document interface), các cửa sổ thành phần hiển thị một cách độc lập với nhau. Không có một cửa sổ chính để chứa và thống nhất các thành phần.

Chuyển đổi từ hiển thị MDI sang SDI

- Chọn Tools\option\
- Trên tang Advance, chọn hộp đánh dấu SDI development Enviroment; nhấn OK. IDE của Visual basic sẽ định lại cấu hình cho hiển thị SDI trong lần khởi động tiếp sau của Visual basic.
- Nhấn OK, thoát và khởi động lại Visual basic

2.3.8 Trợ giúp

Không những làm chủ ngôn ngữ lập trình Visual basic, bạn cũng cần phải sử dụng thuần thục môi trường Visual basic cũng như hiểu các thông điệp mà Visual basic gửi ra. Microsoft cung cấp một trong những hệ thống trợ giúp tốt nhất cho các công cụ phát triển ứng dụng.

Trợ giúp nhạy với ngữ cảnh

Tại một vị trí bất kỳ trong Visual basic, bạn nhấn phím F1, nút trợ giúp. Nó sẽ kích hoạt hệ thống trợ giúp của Visual basic, nơi có thể giải thích hoặc đưa ra những lời khuyên, cũng như các đoạn chương trình mẫu.

Visual basic có hệ thống trợ giúp là hệ thông thư viện MSDN được sử dụng rộng rãi cho các công cụ phát triển của Microsoft để cung cấp truy cập đến sổ tay hướng dẫn sử dụng sản phẩm trực tuyến.

3 Tìm hiểu Visual basic 6

3.1 Thuộc tính phương thức và sự kiện

3.1.1 Đối tượng

Trong VB, đối tượng là những thành phần tạo nên giao diện giữa người sử dụng cho ứng dụng. Các điều khiển là những đối tượng. Những nơi chứa (container) như biểu mẫu(form), khung(frame), gay hộp ảnh (picture box) cũng là một đối tượng.

VB 6 hỗ trợ một cách lập trình tương đối mới, lập trình hướng đối tượng (Object Oriented Programming).

Trong lập trình cổ điển, ta có kiểu lập trình theo cấu trúc. Nếu như ứng dụng được thiết kế để giải quyết một vấn đề lớn, thì lập trình viên có thể chia thành nhiều vấn đề nhỏ và viết các đoạn chương trình nhỏ để giải quyết riêng từng cái.

Với lập trình hướng đối tượng, lập trình viên sẽ chia nhỏ vấn đề cần giải quyếtthành các đối tượng. Từng đối tượng sẽ có đời sống riêng của nó. Nó có các đặc điểm mà ta gọi là thuocc tính và những chức năng riêng biệt mà ta gọi là phương thức. lập trình viên cần đưa ra các thuộc tính và phương thức mà các đối tượng cần thể hiện.

3.1.2 Thuộc tính

Nói một cách đơn giản, thuộc tính mô tả đối tượng.

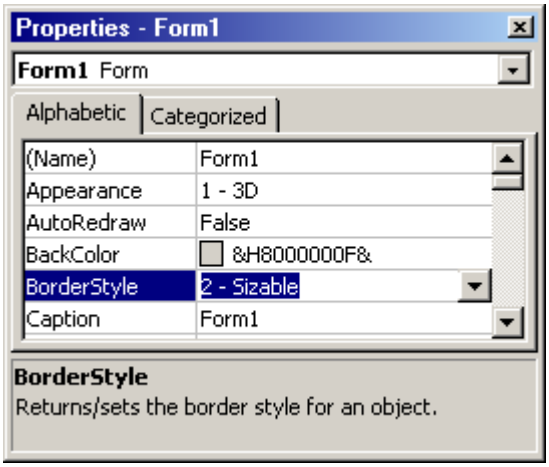
Mỗi đối tượng cộng đều có một bộ thuộc tính mô tả đối tượng. Biểu mẫu và điều khiển đều có thuộc tính. Thậm chí màn hình và máy in là những đối tượng chỉ cho phép can thiệp lúc thi hành cũng có thuộc tính.

Mặc dù mỗi đối tượng có những bộ thuộc tính khác nhau, nhưng trong đó vẫn còn một số thuộc tính thông dụng cho hầu hết các điều khiển.(bạn có thể xem toàn bộ thuộc tính của một điều khiển bằng cách chọn vào điều khiển và mở cửa sổ PROPERTIES trong Visual Basic)

Các thuộc tính thông dụng:

Thuộc tính	**Giải thích**
Left	Vị trí cạnh trái của điều khiển so với vật chứa nó
Top	Vị trí cạnh trên của điều khiển so với vật chứa nó
Hieght	Chiều cao của điều khiển
Width	Chiều rộng của điều khiển
Name	Một giá trị chuỗi được dùng để nói đến điều khiển
Enable	Giá trị logic (True hoặc False) quyết định người sử dụng có được làm việc với điều khiển hay không
Visible	Giá trị logic (True hoặc False) quyết định người sử dụng có thấy điều khiển hay không

Một thuộc tính quan trọng khác là BorderStyle, quyết định các thành phần của cửa sổ (như thanh tiêu đề, nút phóng to thu nhỏ…) mà một biểu mẫu sẽ có



Bảng sau đây liệt kê 6 giá trị của thuộc tính này.

Giá trị	Hiệu ứng trên biểu mẫu
0 – None	Không có cạnh viền, không thanh tiêu đề, không được di chuyển. Giá trị này thường được dùng cho cửa sổ khởi động chương trình
1 – Fixed Single	không thể co giãn cửa sổ bằng cách kéo rê cạnh viền, nhưng có thể dùng nút phóng to hoặc thu nhỏ. Giá trị này được dùng cho những cửa sổ có kích cỡ cố định nhưng vẫn xuất hiện trên thanh Taskbar
2 – Sizable	Có thể co giãn cửa sổ bằng cách kéo rê cạnh viền và dùng nút phóng to hoặc thu nhỏ. Giá trị dùng cho những cửa sổ thông dụng
3 – Fixed Dialog	Không thể co giãn và không có thể dùng nút phóng to hoặc thu nhỏ,. Giá trị này dùng cho các cửa sổ đơn giản như mật khẩu
4- Fixed Tool Window	tương tự Fixed Dialog nhưng thanh tiêu đề ngắn hơn. Font trên thanh tiêu đề và nút Close cũng nhỏ hơn. giá trị này dùng cho các thanh công cụ di động.
5 – Sizable Tool Window	Tương tự như Fixed Tool Window nhưng có thể co giãn được. Giá trị này dùng cho những cửa sổ Properties của Visual Basic

3.1.3 Phương thức

Là những đoạn chương trình chứa trong điều khiển, cho điều khiển biết cách thức để thực hiện một công việc nào đó, chẳng hạn dời điều khiển đến một vị trí mới trên biểu mẫu. Tương tự thuộc tính, mỗi điều khiển có những phương thức khác nhau, nhưng vẫn có một số phương thức rất thông dụng cho hầu hết các điều khiển..

Các phương thức thông dụng

Phương thức	Giải thích
Move	Thay đổi vị trí một đoói tượng theo yêu cầu của chương trình

Drag	Thi hành hoạt động kéo và thả của người sử dụng
SetFocus	Cung cấp tầm ngắm cho đối tượng được chỉ ra trong lệnh gọi phương thức
ZOrder	quy định thứ tự xuất hiện của các điều khiển trên màn hình

3.1.4 Sự kiện

Nếu như thuộc tính mô tả đối tượng, phương thức chỉ ra cách thức đối tượng hành động thì sự kiện là những phản ứng của đối tượng.

Tương tự thuộc tính và phương thức, mỗi điều khiển có những bộ sự kiện rất thông dụng với hầu hết các điều khiển. Các sự kiện này xảy ra thường là kết quả của một hành động nào đó, như là di chuyển chuột, nhấn nút bàn phím, hiặc gõ vào hộp văn bản. kiểu sự kiện này được gọi là sự kiện khởi tạo bởi người sử dụng, và ta sẽ phải lập trình cho chúng.

các sự kiện thông dụng

Sự kiện	**Xảy ra khi**
Change	Người sử dụng sửa đổi chuỗi ký tự trong hộp kết hợp hoặc hộp văn bản
Click	Người sử dụng dùng chuột click lên đối tượng
Dblclick	Người sử dụng dùng chuột click đúp lên đối tượng
DragDrop	Người sử dụng kéo rê một đối tượng sang nơi khác
DragOver	Người sử dụng kéo rê một đối tượng ngang qua một điều khiển khác
GotFocus	Đưa một đối tượng vào tầm ngắm của người sử dụng
KeyDown	Người sử dụng nhấn một nút trên bàn phím trong khi một đối tượng đang trong tầm ngắm
KeyPress	Người sử dụng nhấn và thả một nút trên bàn phím trong khi một đối tượng đang trong tầm ngắm
KeyUp	Người sử dụng thả một nút trên bàn phím trong khi một đối tượng đang trong tầm ngắm
LostFocus	Đưa một đối tượng ra khỏi tầm ngắm
MouseDown	Người sử dụng nhấn một nút chuột bất kỳ trong khi con trỏ chuột đang nằm trên một đối tượng
MouseMove	Người sử dụng di chuyển con trỏ chuột ngang qua một đối tượng
MouseUp	Người sử dụng thả nút chuột trong khi con trỏ chuột đang nằm trên một đối tượng

3.1.5 Mối quan hệ giữa phương thức, thuộc tính và sự kiện

Mặc dù thuộc tính, phương thức và sự kiện có vai trò khác nhau nhưng chúng thường xuyên liên hệ với nhau. ví dụ nếu ta di chuyển một điều khiển bằng phương thức Move (thường đáp ứng một số sự kiện) một số thuộc tính như Top, Height, Left, Width sẽ thay đổi theo. Bởi vì khi kích cỡ của điều khiển thay đổi, sự kiện Resize sẽ sảy ra.

Phụ thuộc lẫn nhau còn có nghĩa là ta có thể đạt được mjục đích công việc bằng nhiều cách: xử lýu trên thuộc tính hoặc phương thức. Ví dụ, ta có 2 cách để di chuyển nút lệnh:

a. thuộc tính
   cmdMove.Left=100
   cmdMove.Top=100
b. phương thức
   cmdMove.Move 100,100

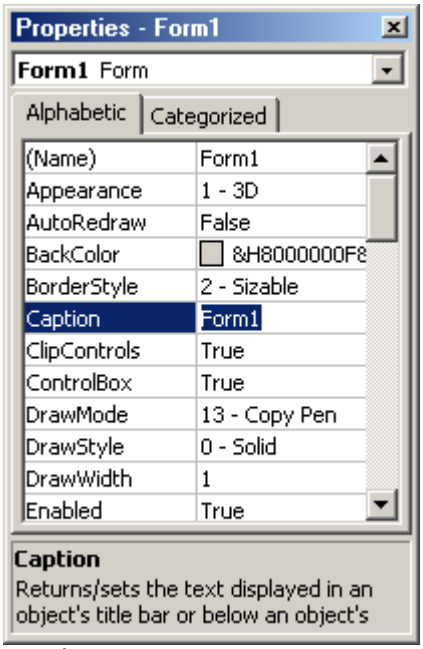
Một ví dụ khác, làm một biểu mẫu xuất hiện và biến mất trên màn hình

c. thuộc tính
   'xuất hiện
   frmMyForm.Visible =True
   'Biến mất
   frmMyForm.Visible =False
d. phương thức
   'xuất hiện
   frmMyForm.Show
   'Biến mất
   frmMyForm.Hide

3.1.6 Cửa sổ Properties

cửa sổ này cho phép lập trình viên xem xét và sửa đổi các thuộc tính của biểu mẫu và các điều khiển trong lúc thiết kế

Phần trên cửa sổ là các danh sách đối tượng, đối tượng được chọn trong danh sách này có các thuộc tính của nó hiển thị trong phần bên dưới của cửa sổ.



Thuộc tính Caption được đánh dấu, nghĩa là ta có thể sửa đổi thuộc tính này.

từng thuộc tính có một hía trị mặc định. ta có thêr sửa đổi bằng tay trong lúc thiết kế, hoặc bằng chương trình trong lúc thi hành.

một biểu mẫu có khoảng 40 thuộc tính được hiển thị trong lúc thiết kế, nhưng ta có thể truy cập một số thuộc tính khác vào trong lúc thi hành.

Ta có thể xem toàn bộ thuộc tính xứp xếp theo thứ tự bảng chữ cái bằng cách chọn vào tab Alphabetic, hoặc xem theo từng nhóm bằng cách chọn vào tab Categozized.

ta có thể mở cửa sổ Properties bằng nhiều cách:

a. nhấn chuộtvào biểu mẫu để chpnj nó như một đối tượng hiện hành, nhấn phím F4 để hiển thị cửa sổ Properties
b. Hoặc là từ menu_View, chọn Properties
c. hoặc là nhấn nút phải chuột lên biểu mẫu, ta sẽ thấy một menu hiển thị. chọn Properties.

3.1.7 Viết chương trình sử dụng thuộc tính, phương thức và sự kiện

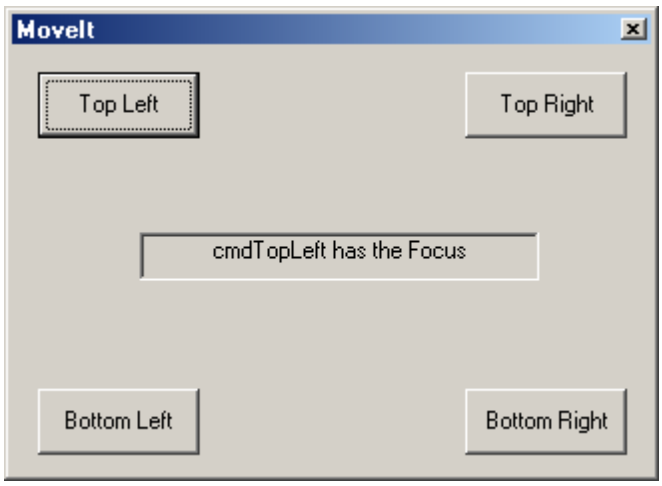
Ta thử viết chương trình Movelt dùng để di chuyển cửa sổ. Movelt có một cửa sổ tên là frmMove, chứa 4 nút lệnh ở 4 góc màn hình. khi thi hnàh nhấn vào moọt trong các nút này sẽ làm cửa sổ di chuyển tới góc màn hình tương ứng. giữa màn hình sẽ là một nhãn hiệu sẽ thông báo tức thời các di chuyển của chuột cũng như nút lệnh nào được Focus.

các bước tổng quát dể tạo Movelt:

a. Tạo giao diện người sử dụng (GUI)
b. Viết thủ tục Form_Load()
c. Viết thủ tục click()
d. thêm các thông báo sự kiện

3.1.7.1 Tạo GUI

1. từ menu File, chọn New Project để mở hộp thoại đề án. Chọn kiểu standard EXE
2. vào cửa sổ Properties, sửa tên biểu mẫu thành frmMove
3. thêm 4 nút lệnh vào 4 góc biểu mẫu. ta sẽ xửa lại vị trí chính xác hiưn cho đíng bằng chương trình
4. nhấn đúp chuột lên biể mẫu để tạo thủ tục Form_Load()
5. Đổi thuộc tính BorderStyle của biểu mẫu thành 1- Fixed Single để cấm biểu mẫu co giãn khi chương trình thi hành. sau đó đổi các thuộc tính Alignment của nhãn thành 2- Center và BorderStyle của nhãn thành 1- Fixed Single
6. Lưu biểu mẫu với tên là frmmove và lưu đề án với tên là Movelt.vbp



3.1.7.2 Viết thủ tục Form _Load

Thủ tục này chuẩn bị một số khởi tạo cho biểu mẫu trước khi nó được hiển thị

a. Đặt thuộc tính Caption cho CommandButtion
b. Dặt chuỗi ký tự khởi tạo cho nhãn
c. Đặt chuỗi ký tự cho thanh tiêu đề của biểu mẫu
d. Đặt vị trí cho 4 nút lệnh nhãn và biểu mẫu trên màn hình
e. Đưa vào đoạn chương trình sau:

```
Private Sub Form_Load()

    'Set the Caption property of the CommandButtons
    cmdTopLeft.Caption = "Top Left"
    cmdTopRight.Caption = "Top Right"
    cmdBottomLeft.Caption = "Bottom Left"
    cmdBottomRight.Caption = "Bottom Right"

    'Clear the initial text of the label
    lblNotify.Caption = ""

    'Set the form's title bar text
    frmMove.Caption = "MoveIt"

    'The rest of the code centers the form on the
    'screen, sets the position of the four
    'CommandButtons, and sets the size and
    'position of the label.

    'Center the form on the screen. This works by
    'setting the Left side of the form to the center
    'of the screen, less half the width of the form.
    'Also, the Top of the form is set to the center
    'of the screen, less half the height of the form.
```

```
    frmMove.Left = (Screen.Width - frmMove.Width) / 2
    frmMove.Top = (Screen.Height - frmMove.Height) / 2

    'Set the Left edge of the buttons. The 200 setting
    'for the left buttons sets a space between the edge
    'of the form and the buttons. The right buttons are
    'set by subtracting the width of the button from
    'the width of the form, and subtracting 300 to
    'set a space between the button and the form edge.
    cmdTopLeft.Left = 200
    cmdBottomLeft.Left = 200
    cmdTopRight.Left = frmMove.Width - cmdTopRight.Width - 300
    cmdBottomRight.Left = frmMove.Width - cmdBottomRight.Width - 300

    'Set the Top edge of the buttons. This is done
    'similar to setting the Left edge.
    cmdTopLeft.Top = 200
    cmdBottomLeft.Top = frmMove.Height - cmdBottomLeft.Height - 500
    cmdTopRight.Top = 200
    cmdBottomRight.Top = frmMove.Height - cmdBottomRight.Height - 500

    'Set the size of the label
    lblNotify.Height = 360
    lblNotify.Width = 3000

    'Center the label within the form. This is done
    'similar to centering the form.
    lblNotify.Left = (frmMove.Width - lblNotify.Width) / 2
    lblNotify.Top = (frmMove.Height - lblNotify.Height) / 2 - 200

End Sub
```

3.1.7.3 Viết thủ tục Click

Dùng thủ tục này để di chuyển biểu mẫu xung quanh màn hình. Nhấn đúp chuột lên nút lệnh để mở cửa sổ Code. Đưa vào đoạn chương trình sau đay:

```
Private Sub cmdBottomLeft_Click()

    'Set the value of the form's TOP property
    'to the bottom of the screen but bring
    'it up the height of the screen so that the
    'bottom of the form is on the bottom of
    'the screen
    frmMove.Top = Screen.Height - frmMove.Height

    'Set the value of the form's LEFT property
```

```
    'to the left  most of the screen.
    frmMove.Left = 0

End Sub


Private Sub cmdBottomRight_Click()

    'Set the value for the form's TOP property to
    'the bottom of the screen, but bring the TOP
    'up the HEIGHT of the form so that the bottom
    'of the form is on the bottom of the screen.
    frmMove.Top = Screen.Height - frmMove.Height

    'Set the value of the form's LEFT property to
    'the  right of the screen but bring it across
    'the screen, the width of the form so that the
    'right side of the form is on the right
    'side of the screen
    frmMove.Left = Screen.Width - frmMove.Width

End Sub
```

```
Private Sub cmdTopLeft_Click()

   'Set the value of the form's TOP property
   'to the top of the screen.
   frmMove.Top = 0

   'Set the value of the form's LEFT property
   'to the left of the screen.
   frmMove.Left = 0

End Sub

Private Sub cmdTopRight_Click()

   'Set the value of the form's TOP property
   'to the top of the screen.
   frmMove.Top = 0

   'Set the value of the form's LEFT property to
   'the right of the screen but bring it back across
   'the screen the width of the form, so that the
   'right side of the form is on the right
   'side of the screen
   frmMove.Left = Screen.Width - frmMove.Width

End Sub
```

Đối tượng Screen sử trong đoạn chương trình trên là màn hình

Việc di chuyển biểu mẫu lên trên hoặc sang trái chỉ cần đổi thuộc tính Top hay Left thành 0. Giá trị này luôn đúngcho cạnh trên hay cạnh trái màn hình.

Cạnh phải hoặc cạnh dưới phức tạp hơn vì không có thuộc tính Right hay Bottom. Để canh phải biểu mẫu ta phải thay đổi thuộc tinh Left thông qua thuộc tính Width

Tương tự với cạnh dưới ta phải thay đổi thuộc tinh Top thông qua thuộc tính Height

3.1.7.4 Thêm thông báo sự kiện

Khi người sử dụng nhấn hoặc thả nút chuọt trên biểu mẫu chuỗi ký tự trong nhãn lblNotify sẽ thay đổi. Ngoài ra khi người sử dụng nhấn phím Tab hoặc chuột để di chuyển từ nút lện này sang nút lệnh khác, chuỗi ký tự của nhãn cũng thay đổi. như vậy ta phải chương trình cho 3 thủ tục khác nhau. MouseUp, mouseDown cho biểu mẫu và GostFocus cho từng nút lệnh.

Mở cửa sổ Code, chọn sự kiện MouseDown để mở thủ tục và đưa vào đoạn chương trình sau

Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)

```
    lblNotify.Caption = "MouseDown Event"

End Sub

Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)

    lblNotify.Caption = "MouseUp Event"

End Sub

Private Sub cmdBottomLeft_GotFocus()

    lblNotify.Caption = "cmdBottomLeft has the Focus"

End Sub

Private Sub cmdBottomRight_GotFocus()

    lblNotify.Caption = "cmdBottomRight has the Focus"

End Sub

Private Sub cmdTopLeft_GotFocus()

    lblNotify.Caption = "cmdTopLeft has the Focus"

End Sub

Private Sub cmdTopRight_GotFocus()

    lblNotify.Caption = "cmdTopRight has the Focus"

End Sub
```

3.2 Làm việc với một đề án

3.2.1 Định nghĩa

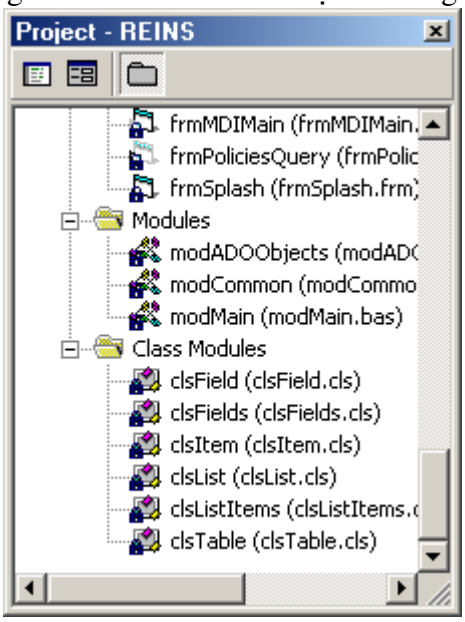
Một đề án gồm có :

a. 1 tệp tin đề án (.vbp) theo dõi toàn bộ các thành phần
b. 1 tệp tin cho biểu mẫu (.frm)
c. 1 tệp tin nhị phân (.frx) cho từng biểu mẫu. Người sử dụng không được sửa đổi. Các tập tin này được phát sinh tự động cho mỗi tập tin .frm bất kỳ và dùng để chứa các thuộc tính nhị phân như Picture hay Icon.
d. 1 tệp tin cho từng module lớp (.cls)- tuỳ chọn
e. 1 tệp tin cho từng module chuẩn (.bas)- tuỳ chọn

f. 1 hoặc nhiều tệp tin chứa các điều khiển Activex (.ocx)- tuỳ chọn
g. 1 tệp tin tài nguyên (.res)- tuỳ chọn

3.2.2 Cửa sổ Project Explorer

Cửa sổ này thường được hiển thị bên góc phải trên màn hình Visual Basic. Project Explorer giúp ta tổ chức các tệp tin trong đề án và truy cập chúng dưới dạng thiết kế biểu mẫu hoặc chương trình.



Để làm việc với cửa sổ này ta dùng menu nhạy với ngữ cảnh. menu này xuất hiện khi ta nhấn nút phải chuột vào một tệp tin trong cửa sổ. Khi đó, ta có thể:

a. xem một tệp tin dưới dạng thiết kế biểu mẫu hoặc chương trình
b. xem thuộc tính của tệp tin
c. Thêm một biểu mẫu hoặc module vào đề án
d. Lưu tệp tin hiện hành
e. Xoá một tệp tin khỏi đề án
f. in tệp tin
g. ghi hoặc thả cho cửa sổ Project Explorer di động trong màn hình Visual Basic
h. che cửa sổ Project Explorer

3.2.3 Tạo đề án

mỗi lần khởi động Visual Basic, ta sẽ thấy hộp thoại New Project. Từ đây, ta có thể chọn loại đề án mà ta muốn tạo và ấn Open.

Khi Visual Basic đã có sẵn, ta có thể tạo đề án bằng cách: từ menu File chọn New Project. Hộp thoại New Project xuất hiện, ta chọn loại đề án cần thiết và nhấn OK.

3.2.4 Đổi thuộc tính đề án

Một số thông tin liên quan đề án như tên đề án, số phiên bản, chuỗi ký tự dùng hiển thị trên thanh tiêu đề khi ứng dụng hoàn thành. Ta có thể xem các thông tin khác trong hộp thoai Project Properties

3.2.4.1 Hộp thoại Project Properties

1. trong cửa sổ Project Explorer, nhấn nút phải chuột lên tệp tin đề án
2. trong menu ngữ cảnh, Chọn Project Name Properties. Hộp thoại Project Properties xuất hiện
3. Hoặc là từ menu Properties, chọn Project Name Properties

3.2.5 Lưu và đặt tên đề án

3.2.5.1 Lưu đề án

Khi lưu đề án từng tập tin trong đề án sẽ được lưu trước kế tiếp là tệp tin đề án. Trong lần lưu đề án đầu tiên, Visual Basic đề nghị tên cho từng tệp tin, thường nó lấy tên biểu mẫu và có phần mở rrộngt uỳ thuộc vào loại tệp tin

1. từ menu file chọn Save Project
2. nếu đây là lần đầu lưu đề án hoặc ta vừa thêm một biểu mẫu hoặc module, hộp thoại save File as xuất hiện lần lượt cho từng tệp tin

3.2.5.2 Đổi tên

Ta không nhất thiết dùng tên mà Visual Basic đề nghị, mà có thể đặt tuỳ ý. tuy nhiên nên dặt tên sao cho gợi nhớ

1. Đưa vào một tên và ấn nút save
2. Tập tin cuối cùng được lưu là tệp tin đề án Nếu ta đã đặt tên cho đề án thông qua hộp thoại project properties, Visual Basic sẽ tự động đề nghị Project_Name.vbp. Lúc này, ta có thể đổi lại tên khác tuỳ thích, ví dụ như SaveTest.vbp

3.2.6 Mở đề án có sẵn

Ta có một số đề án đang làm việc. Khởi động Visual Basic, chọn menu File. Phần dưới menu liệt kê danh sách các đề án mới nhất mà ta đã làm việc, chọn đề án cần mở. nếu đề án không xuất hiện trong danh sách, ta phải chỉ ra đường dẫn.

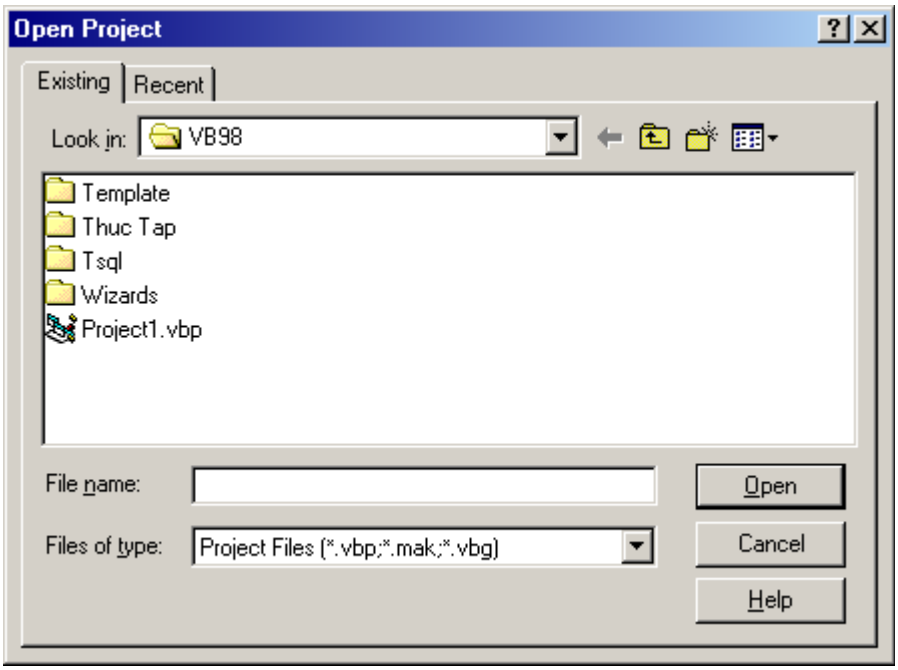
3.2.6.1 mở tệp tin vào lúc khởi động Visual Basic

Lần đầu khởi động Visual Basic, hộp thoại New project xuất hiện. ta có thể chọn mở đề án mới nhất hoặc có sẵn trên đĩa ngay từ hộp thoại này. nếu không muốn hộp thoại này xuất hiện mỗi lần khởi động Visual Basic, xoá đánh dáu trên hộp đánh dấu (checkbox) nằm ở bên dưới hộp thoại.

3.2.6.2 Mở đề án có sẵn

a. Từ menu File, chọn Open Project.

b. trên trang Existing, chuyển đến thư mục chứa đề án. nếu đề án cần mở là đề án lưu gần nhất, chuyển sang trang Recent
c. chọn tên tệp tin đề án và nhấn OK



3.2.7 Thêm xoá và lưu tập tin trong đề án

3.2.7.1 Thêm mới tệp tin

Thông thường, một biểu mẫu cần nhiều biểu mẫu hợăc module. Ví dụ muốn thêm hộp thoại About, ta cần thêm một biểu mẫu.

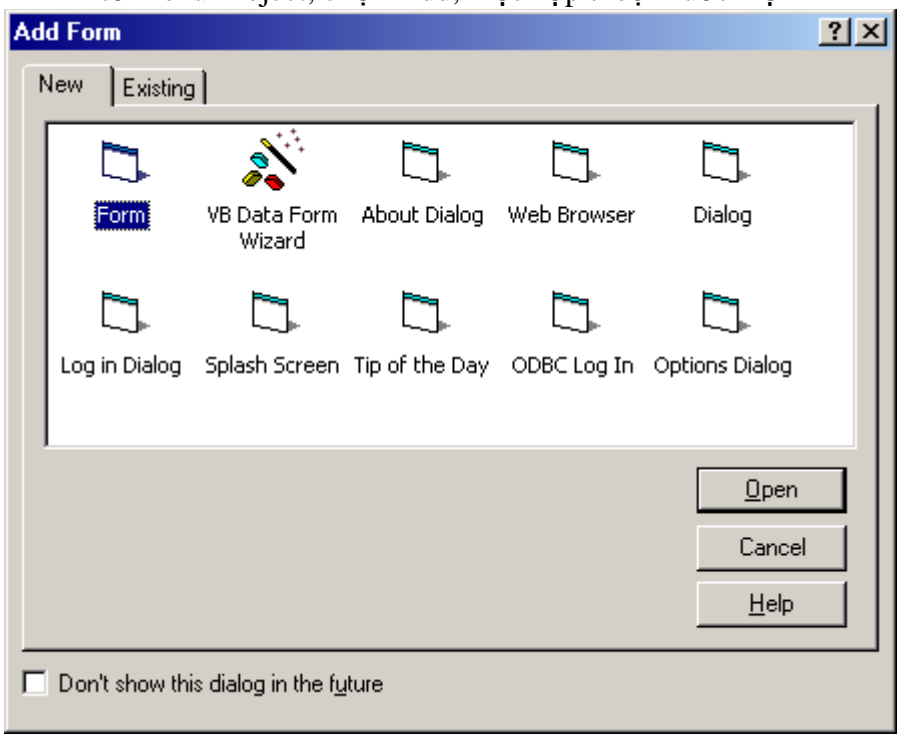
Dưới đây là các dạng tệp tin có thể thêm vào đề án :

a. biểu mẫu (form): tệp tin.frm chứa môtả của một biểu mẫu và các điều khiển, kể cả các thuộc tính của chúng. Nó cũng chứa khai báo các hằng, biến và thủ tục
b. Lớp (Class) là một trong những tính năngquan trọng nhất của Visual Basic, được dùng trong lập trình hướng đối tượng để định nghĩa các khuôn mẫu cho các đối tượng.
c. module chuẩn chứa các khai báo kiểu, hằng, biến, thủ thuộc phạm vi public hoặc ở mức module
d. tệp tin tài nguyên; chứa hình ảnh, chuỗi ký tự và các dữ liệu khác ta có thể soạn thảo mà không cần sửa lại mã nguồn
e. tài liệu ActiveX (.dob) tương tự biểu mẫu nhưng được hiển thị trong trình duyệt xét WEB, như là Internet Explorer.
f. Module điều khiển (.ctl) và module Property page (.pag) tương tự biểu mẫu, nhưng được dùng để tạo điều khiển ActiveX và danh sách các thuộc tính của chúng để hiển thị khi thiết kế
g. điều khiển ActiveX (.ocx) có thể được thêm vào hộp công cụ để dùng trong biểu mẫu. Khi Visual Basic được cài đặt, một số tệp tin dạng này kèm theo Visual Basic sẽ được chép vào trong máy

h. đối tượng, như là Worksheet của Excel
i. tham chiếu (Reference)
j. trình thiết kế ActiveX : là công cụ dùng để thiết kế các lớp cho đối tượng. giao diện thiết kế biểu mẫu là một trình thiết kế mặc định
k. các điều khiển thông dụng là nút lệnh điều khiển khung được Visual Basic cung cấp sẵn.ẫnt có thể thêm tệp tin bằng hai cách;

cách 1:

- từ menu Project, chọn Add, một hộp thoại xuất hiện



- nếu muốn tạo mới, chọn tab new. Visual Basic cung cấp các danh sách có sẵn
- nếu muốn dùng một tệp tin có sẵn, chọn Tab Existing, chọn tên tệp tin, nhấn Open

cách 2:

- Nhấn nút chuột trong cửa sổ Project Explorer
- Trong menu ngữ cảnh chọn Add
- xuất hiện hộp thoại như trên

3.2.7.2 Xoá tệp tin

a. Chọn tệp tin trong cửa sổ Project Explorer
b. Từ menu Project chọn Remove
c. Tham chiếu tệp tin bị xoá trong đề án (thực chất nó vẫn được lưu trên đĩa)

Khi một tệp tin trong đề án bị xoá Visual Basic sẽ cập nhật những thay đổi này trong tập tin.vbp khi ta lưu đề án. Do đó nếu ta xoá tệp tin bên ngoài Visual Basic, tệp tin đề án sẽ không được cập nhật. khi ta mở lại đề án Visual Basic sẽ báo lỗi là thiếu tệp tin

3.2.7.3 Lưu tệp tin

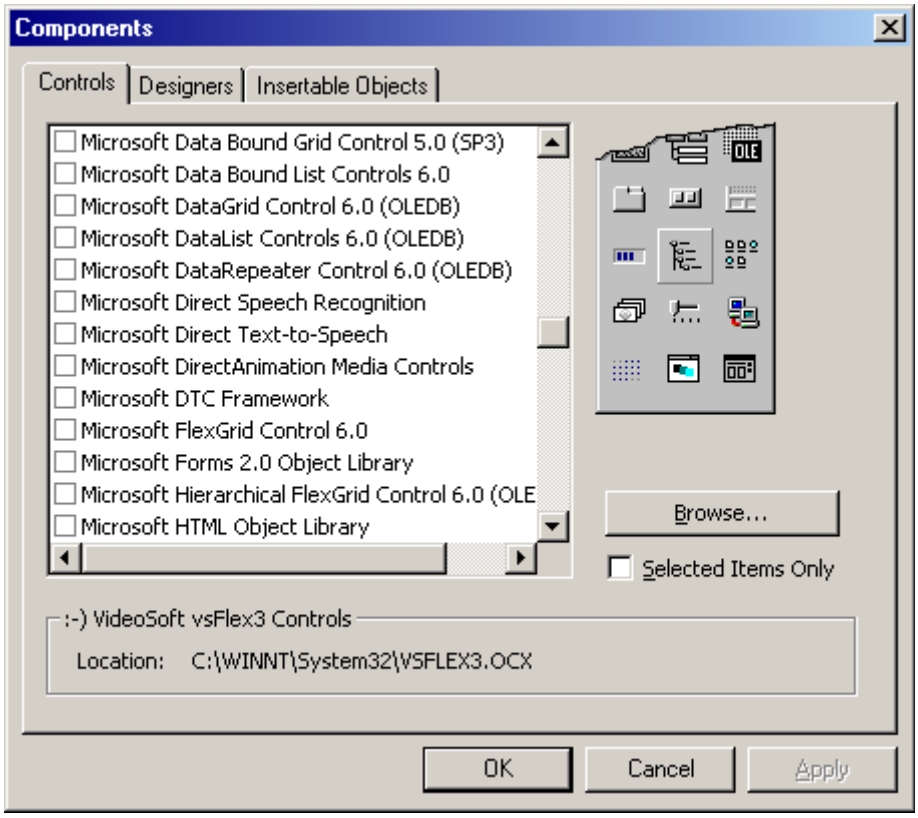
a. Chọn tệp tin trong cửa sổ Project Explorer
b. từ menu chọn Save

3.2.8 Thêm điều khiển vào đề án

3.2.8.1 Thêm điều khiển ActiveX

Ta có thể thêm vào đề án một điều khiển ActiveX và các đối tượng nhúng được bằng cách thêm nó vào hộp công cụ

1. Từ menu Project chọn components
2. Để thêm một điều khiển (.ocx) hoặc thêm một đối tượng nhúng vào hộp công cụ, chọn vào hộp đánh dấu bên trái tên điều khiển
3. Chọn OK để đóng hộp thoại. Các điều khiển đánh dấu sẽ hiển thị trên hộp công cụ.
4.



Để thêm điều khiển ActiveX vào hộp thoại Components, nhấn nút Browse để tìm đường dẫn cho tệp tin.OCX

mỗi điều khiển ActiveX có kèm theo một tệp tin mở rộng là.OCA Tệp tin này chứa các thông tin cấu thư viện kiểu lưu trữ và các dữ liệu liên quan đến điều khiển. Các tệp tin.OCA chứa trong cùng thư mục với điều khiển ActiveX và được tạo lại khi cần

3.2.8.2 Xoá điều khiển khỏi đề án

1. từ menu project, chọn Components

2. hộp thoại Components xuất hiện, chọn điều khiển mà ta muốn xoá, xoá hộp đánh dấu kế bên nó

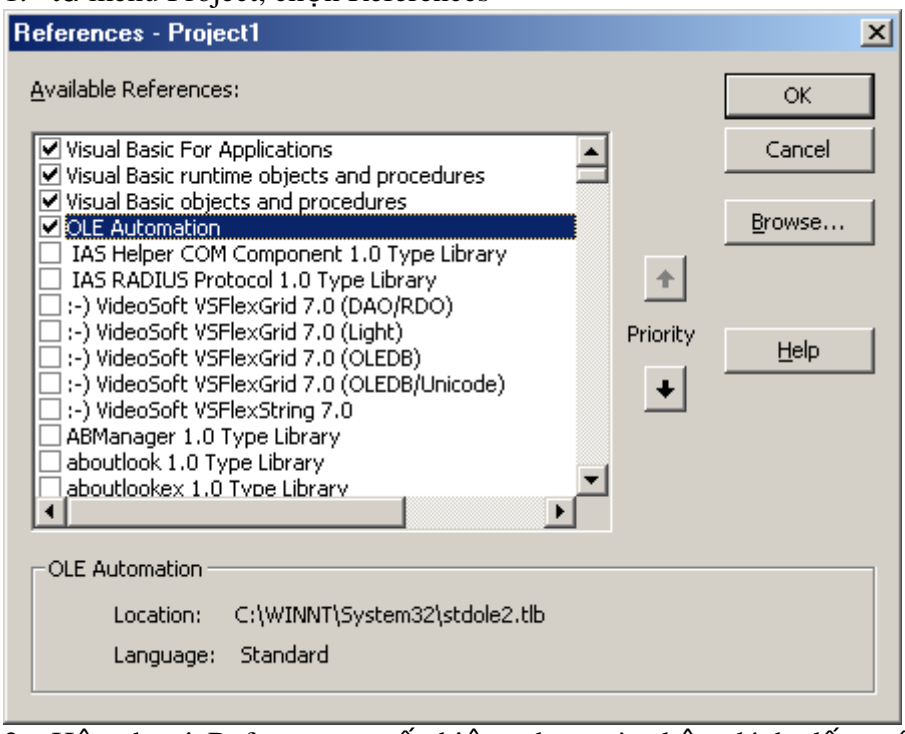
Điều khiển sẽ bị xoá khỏi hộp công cụ

3.2.8.3 Sử dụng đối tượng của ứng dụng khác

Để lấy đối tượng từ một ứng dụng nào đó, ví dụ ta muốn sử dụng thư viện đối tượng của Microsoft Excel, đặt tham chiếu đến thư viện đối tượng của ứng dụng đó

Thêm tham chiếu đến thư viện đối tượng của ứng dụng khác

1. từ menu Project, chọn References



2. Hộp thoại References xuất hiện, chọn vào hộp đánh dấu. nếu tên tham chiếu chưa có sẵn trong danh sách, nhấn Browse vào ứng dụng và ấn OK

Nếu không muốn tiếp tục sử dụng đối tượng của thư viện tham chiếu. Ta nên xoá đánh dấu tham chiếu để giảm số tham chiếu mà Visual Basic đang quản lý, giảm được thời gian biên dịch đề án,.

Khi có tham chiếu đến thư viện đối tượng, ta có thể lấy được đối tượng cùng với các thuộc tính và phương thức của nó bằng cách vào menu view, chọn Object Browser

3.2.8.4 Thêm tệp tin tài nguyên vào đề án

Tệp tin tài nguyên chứa toàn bộ các hình ảnh, biểu tượng, chuỗi văn bản hiển thị trên màn hình và các thành phần khác liên quan đến việc địa phương hoá ứng dụng.

a. từ menu project, chọn Add File
b. Chọn tệp tin tài nguyên có sẵn (.RES) và chọn Open

Một đề án đơn giản chỉ có một tệp tin tài nguyên, nếu thêm một tệp tin.RES thứ hai, Visual Basic sẽ báo lỗi

3.2.9 Tạo tệp tin EXE

Các ví dụ trên đây được thi hành thông qua nút Start của Visual Basic hoặc nhấn F5. Tuy nhiên khi chương trình hoàn tất, ta cần có một tệp tin thi hành, hay tập tin EXE

3.2.9.1 So sánh trình biên dịch và trình thông dịch

ngôn ngữ lập trình chia làm hai trường phái: thông dịch và biên dịch. người sử dụng ngôn ngữ biên dịch hay xem thường ngôn ngữ thông dịch. ngôn ngữ thông dịch cách ly người sử dụng với hệ thống, tạo một lớp che chắn để lập trình dễ dàng. chúng rất chậm và thiếu chiều sâu so với ngôn ngữ biên dịch

máy tính chỉ hiểu được các tín hiệu 0 và 1. trình biên dịch tập hợp các lệnh từ khoá rồi chuyển chúng thành các tín hiệu 0 và 1 để máy có thể hiểu được

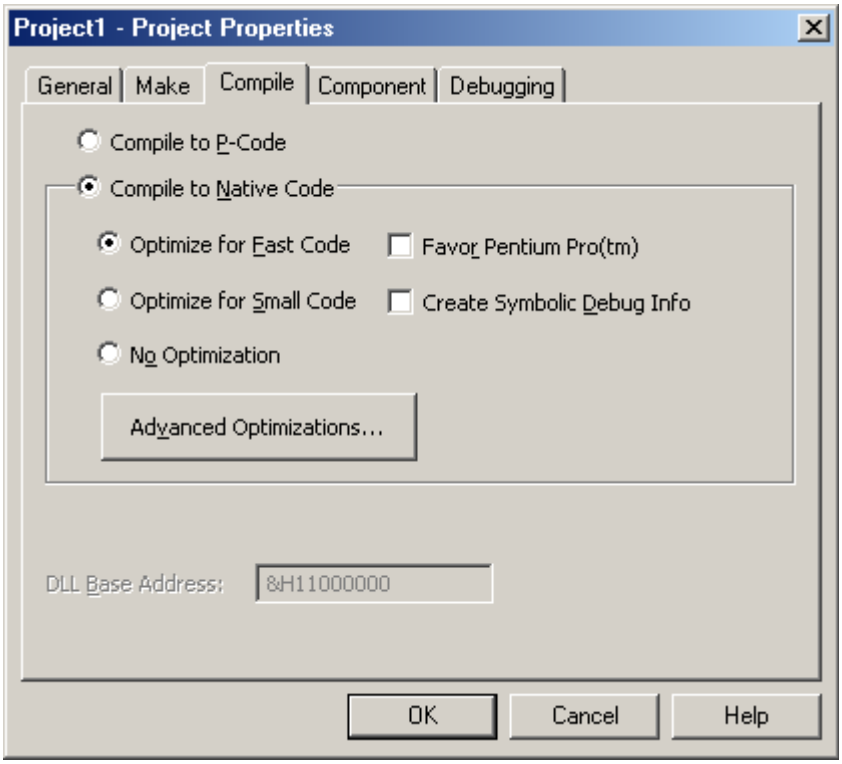
trình thông dịch không làm thế. Nó là một chương trình chen giữa máy tính và các ứng dụng. khi thi hành ứng dụng, trình thông dịch sẽ duyệt qua từng dòng chương trình, chuển đổi chúng thành mã máy. Vì vậy quá trình này rất chậm chạp.

3.2.9.2 Ngôn ngữ giả biên dịch

Trong các phiên bản trước của Visual Basic, khi ta biên dich ứng dụng, chúng được chuyển sang một loại mã để dễ dàng sử lý, gọi là P- code. về phương diện kỹ thuật, có thể gọi đó là biên dịch. Tuy nhiên ta cần kèm theo mmột số tập tin của Microsoft vì máy tính vẫn chưa hiểu ngôn ngữ P- Code. những tập tin gửi kèm theo ứng dụng sẽ thông dịch nó.

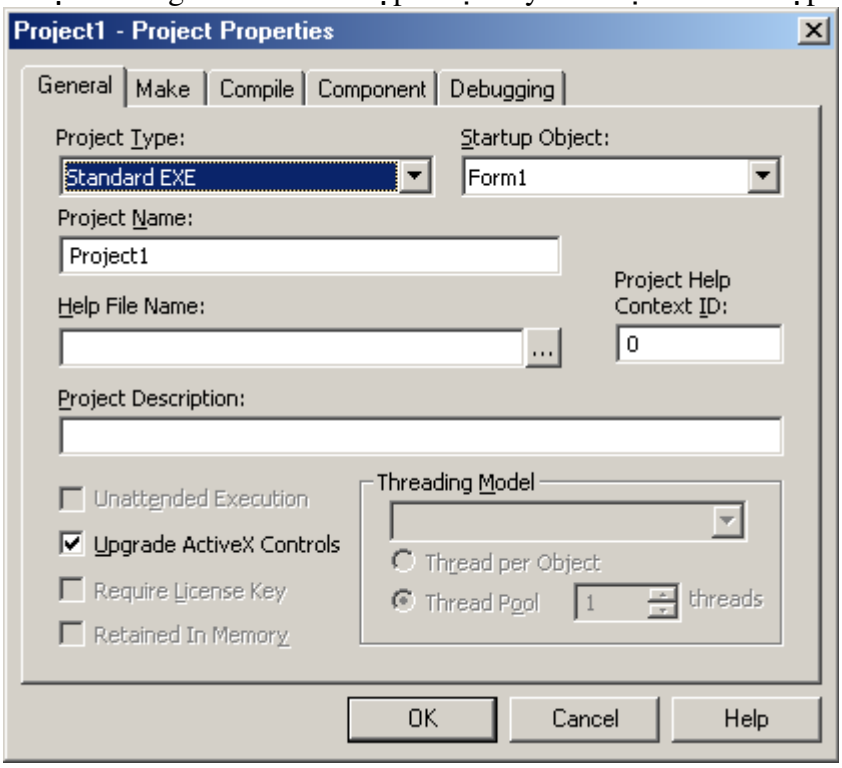
Từ Visual Basic 5 trở về sau, chúng ta có thể biên dịch thực sự trên các chương trình Visual Basic. không còn những thông dịch thi hành ẩn: chương trình được biên dịch thẳng thành ngôn ngữ máy.

Ta có thể bật hay tắt chức năng này bằng cách mở hộp thoại Project Properties từ menu Project



3.2.10 Sửa đổi thuộc tính đề án

Từ menu Project, chọn <tên đề án > Properties. Hộp thoại Project Properties xuất hiện. những sửa đổi trên hộp thoại này sẽ được lưu trên tệp tin.vbp



Chọn lựa	Giải thích
Startup Object	Tên biểu mẫu được hiển thị hoặc tên thủ tục được thi hành đầu tiên khi chương trình được khởi động

Project Name	Tên đề án, nó không được chứa dấu chấm, khoảng trốn và phải bắt đầu bằng chữ cái. Tên đề án không được qua s 37 ký tự
Help File	Tên tệp tin hỗ trợ kèm theo đề án
Project Help Context ID	số ID của chủ đề Help được hiển thị khi người sử dụng click vào nút "?"
Project Description	Tên gợi nhớ của đề án. Nó được hiển thị trong References và Object Browser

3.3 Làm việc với nhiều đề án

3.3.1 Sử dụng Project Group

Visual Basic cho phép ta làm việc với nhiều đề án cùng lúc. Để theo dõi ta dùng cửa sổ Project Explorer và nhóm đề án (Project Groups) là tập hợp đề án. Nhóm đề án có thể được lưu thành tập tin, tựpng tự module, biểu mẫu, hay đề án. Phầm mở rộng là.vbg

3.3.2 Thêm đề án vào nhóm đề án

1. từ menu File chọn Add Project
2. trong hộp thoại chọn Add Project, mở Tab New, chọn kiểu đề án cần thêm, hoặc chọn đề án có sẵn
3. nhấn OK, Visual Basic tự động tạo nhóm đề án và thêm mới đề án

3.3.3 Xoá đề án trong nhóm đề án

1. Trong cửa sổ Project Explorer, chọn đề án cần xoá
2. Từ menu file, chọn REMOVE Project

Ta chỉ dùng nhóm đề án khi tạo các điều khiển ActiveX, vốn đòi hỏi nhiều đề án mở cùng một lúc.

4 Làm việc với các điều khiển

4.1 Các loại điều khiển

Trong Visual Basic có ba nhóm điều khiển

Điều khiển nội tại, ví dụ như là các điều khiển nút lệch và khung. Các điều khiển này được chứa trong các tập tin. EXE của Visual Basic. Các điều khiển nội tại luôn chứa sẵn trong hộp công cụ, ta không thể gỡ bỏ hay thêm chúng vào hộp công cụ

Điều khiển ActiveX, tồn tại trong các tập tin độc lập có phần mở rộng là. OCX. Chúng có thể đưa ra các điều khiển hiện diện trong mọi ấn bản của Visual Basic(ví dụ DataCombo, Datalist,....) hoặc là các điều khiển chỉ hiện diện trong ấn bản Professional và Enterprise(như ListView,Toolbar,Animation). Ngoài ra, còn rất nhiều điều khiển ActiveX do các nhà cung cấp thứ ba đưa ra

Đối tượng chèn được, ví dụ như đối tượng bảng tính (Worksheet) của Microsoft Excel chứa một danh sách các nhân viên của một công ty hay đối tượng lịch biểu (Calendar) của Microsoft Project chứa việc lập biểu thông tin cho một đề án. Bởi vì chúng có thể thêm vào hộp công cụ, chúng có thể là các điều khiển được chuẩn bị chu đáo. Một vài đối tượng kiểu này cũng cung cấp phần Automation lập trình với các đối tượng sinh ra từ những ứng dụng khác ngay trong ứng dụng của Visual Basic. Xem phần "Lập trình với các đối tượng" để biết thêm thông tin về Automation

4.1.1 Thao tác với điều khiển

4.1.1.1 Hộp công cụ

Để đặt một hộp văn bản hay nút lệch vào biểu mẫu, đơn giản là trỏ và nhấn chuột. Tất cả các điều khiển nội tại chứa trong **hộp công cụ** (toolbox) thường hiển thị ở bên trái màn hình.

Muốn hiển thị hộp công cụ, từ menu View, chọn Toolbox hoặc là nhấn chuột trên biểu tượng (icon). Khi hộp công cụ hiển thị, ta có thể dịch chuyển hộp công cụ xung quanh màn hình bằng cách nhấn thanh tiêu đề của nó(title bar) rồi giữ chuột và kéo tới nơi ta muốn và thả ra

Muốn đóng hộp công cụ, nhấn chuột lên nút đóng(nằm trên góc phải của thanh tiêu đề).

Ngoài hộp công cụ, ta cũng cần xem một số cửa sổ tương tự phục vụ cho việc thiết kế ứng dụng như Gỡ rối chương trình(Debug), viết chương trình (Edit), thiết kế biểu mẫu(Form Editor). Để hiển thị các cửa sổ này, nhấn nút phải chuột trên thanh công cụ(tool bar), ta sẽ thấy một menu theo ngữ cảnh(context sensitive menu), chọn trong menu cửa sổ mà ta muốn xem.

Các cửa sổ này có thể hiển thị theo hai cách: trôi nổi và cố định. Hai cách này có thể chuyển đổi qua lại bằng cáh nhấn đúp chuột trên thanh tiêu đề của cửa sổ đó.

4.1.1.2 Đưa điều khiển vào biểu mẫu

Ta lấy nút lệch làm ví dụ.

a. Từ menu File, chọn New Project để tạo một đề án mới.

b. Trong hộp thoại New Project, chọn Standard EXE

c. Một biểu mẫu trống hiển thị. Để đưa nút lệch vào biểu mẫu, ta nhấn chuột vào biểu tượng nút lệch trên hộp công cụ.

Khi chọn trong hộp công cụ, nếu không nhớ tên điều khiển, ta có thể đưa chuột ngang qua từng biểu tượng, tên của nó sẽ hiện ra.

Dời con trỏ màn hình tới vị trí ta muốn, vẽ điều khiển bằng cách giữ nút trái chuột và rê nó đi. Một hình chữ nhật xuất hiện, thể hiện kích cỡ của điều khiển. Khi ta đã vừa ý, ta thả chuột và điều khiển được vẽ trên biểu mẫu.

Ta có thể nhấn vào điều khiển và rê nó đến vịt trí ta muốn

*Nếu muốn hiệu chỉnh vị trí của điều khiển, ta giữ nút **Ctrl** và dùng các phím mũi tên trên bàn phím. Mỗi lần nhấn phím, điều khiển dịch chuyển đi một đơn vị màn hình(một điểm trên biểu mẫu)*

4.1.1.3 Điều chỉnh kích cỡ điều khiển

Thông thường, khi ta thả một điều khiển vào biểu mẫu, ta có thể điều chỉnh kích cỡ điều khiển bằng cách chọn vào nó rồi nhấn chuột lên cạch biên và rê chuột đi. Tuy nhiên, một vài điều khiển không thể co giãn, ví dụ như hộp kết hợp(combo box)

Có thể nhấn đúp chuột lên biểu tượng trong hộp công cụ, Visual Basic sẽ tự động thả điều khiển vào biểu mẫu với kích thước mặc định của nó.

*Nếu muốn hiệu chỉnh kích cỡ của điều khiển, ta giữ phím **Shift** và dùng các phím mũi tên trên bàn phím*

4.1.1.4 Lưới(grid) điểm trong biểu mẫu

Để tạo sự thuận tiện cho lập trình viên khi thiết kế các điều khiển, Visual Basic hiển thị biểu mẫu với các khung kẻ thẳng hàng bằng các điểm nhỏ. Ta có thể sửa lại kích cỡ hoặc là loại bỏ hẳn các ô này bằng cách: từ menu Tool, chọn Option, chon tab General

4.1.1.5 Khoá(Lock) điều khiển

Để giữ các điều khiển cố định tại vị trí của nó, ta dùng tính năng **Lock.**

Chọn điều khiển, sau đó, từ menu Format, chọn Lock Controls hoặc là nhấn chuột vào biểu tượng vào biểu tượng ô khoá trên Form Editor.

Khi đó, ta không thể dùng chuột để điều chỉnh kích cỡ điều khiển. Tuy vậy, ta vẫn có thể dùng tổ hợp phím.

4.1.1.6 Thuộc tính và sự kiện

a. Thuộc tính (*Property*): là bộ các thông số mà ta có thể gán cho điều khiển, ví dụ như tên, chiều rộng, chiều cao,.... Ta có thể xem toàn bộ thuộc tính của điều khiển bằng cách chọn vào nó và nhấn F4 để mở cửa sổ thuộc tính
b. Phương thức(*Method*): là những phản ứng của điều khiển
c. Sự kiện(*Event*): là những tín hiệu mà điều khiển có thể hiểu để phản ứng

Thế mạnh của Visual Basic là sử dụng các điều khiển và tận dụng tối đa khả năng lập trình của chúng

Một điều khiển thực chất là một cửa sổ được lập trình sẵn bên trong. Không có gì khác nhau giữa một ứng dụng và một điều khiển. Để thi hành một ứng dụng, ta mở một cửa sổ. Ứng dụng sẽ chiếm điều khiển trên cửa sổ đó và hoạt động thông qua giao diện cũng như các chức năng của nó. Một điều khiển cũng thực hiện tương tự như vậy.

Một điều khiển chứa đựng một một chương trình được lập sẵn và chương trình này có thể tích hợp một cách dễ dàng vào ứng dụng có sử dụng điều khiển. Để thi hành một ứng dụng, ta mở cửa sổ. Ứng dụng sẽ chiếm điều khiển trên cửa sổ đó và hoạt động thông qua giao diện cũng như các chức năng của nó. Một điều khiển cũng thực hiện tương tự như thế.

Một điều khiển chứa đựng một chương trình được lập sẵn và chương trình này có thể tích hợp một cách dễ dàng vào ứng dụng có sử dụng điều khiển. Trước đây,

lập trình viên thường phải tự xây dựng toàn bộ mo-dun cần thiết cho chương trình. Điều này có nghĩa là các lập trình viên khác cũng phải lặp lại công việc đó. Trong khi đó, PC được câu tạo từ vô số thành phần được cung cấp bởi nhiều nhà sản xuất khác nhau, mỗi thành phần có một công dụng đặc biệt. Khái niệm điều khiển của Visual Basic cũng mang ý tưởng như thế. Từng điều khiển có thể được hiệu chỉnh và được tích hợp lại với nhau tạo thành một ứng dụng.

So với các điều khiển có sẵn trong hộp công cụ, một điều khiển hiệu chỉnh (*custom control*), hay một điều khiển ActiveX là một thành phần có khả năng phát huy cao hơn và sâu hơn các tính năng hiện tại của môi trường. Bằng cách thêm một điều khiển ActiveX vào hệ thống, ta đã mở rộng năng lực và tiện ích của môi trường Visual Basic. Chỉ cần cài đặt một bản Visual Basic duy nhất, mỗi lập trình viên có quyền thêm những điều khiển mà họ thích vào hộp công cụ.

Vì là những điều khiển ActiveX nên chúng có thể được dùng lại một cách dễ dàng bởi các ứng dụng ActiveX như là bộ Office, trình duyệt Web Internet Explorer,...Các điều khiển này được cung cấp bởi các nhà sản xuất phần mềm. Chúng có thể là một sản phẩm thương mại hoặc được tải xuống miễn phí từ Internet.

4.2 Các điều khiển nội tại

Các điều khiển nội tại gồm có:

Điều khiển	**Mô tả**
Label	Hiển thị chuỗi ký tự không đổi trên biểu mẫu
Frame	Cho phép người sử dụng chọn hoặc không chọn một khả năng nào đó.
CheckBox	Cho phép người sử dụng chọn hoặc không chọn một khả năng nào đó
ComboBox	Cho phép người sử dụng chọn từ danh sách các chọn lựa hay nhập liệu mới
HscrollBar	Cho phép người dùng sử dụng cuộn ngang qua một điều khiển chứa dữ liệu khác
Timer	Cho phép chương trình tự động thi hành một công việc nào đó vào một thời điểm, không cần tương tác của người sử dụng.
DirListBox	Cho phép người sử dụng chọn một thư mục
Shape	Hiển thị một dạng hình học trên biểu mẫu
Image	Hiển thị hình ảnh đồ hoạ trên biểu mẫu nhưng không thể làm nơi chứa
OLE Container	Cho phép thêm chức năng lập trình của một điều khiển vào ứng dụng
PictureBox	Hiển thị hình anh trên biểu mẫu và có thể dùng làm nơi chứa.
TextBox	Dùng trình bày văn bản, nhưng cũng cũng cho phép người sử dụng sửa đổi hay thêm mới văn bản
CommandButton	Cho phép người sử dụng thực hiện một hành động
OptionButton	Cho phép người sử dụng chọn lựa từ một nhóm có hai hay nhiều khả năng trở lên.
ListBox	Cho phép người sử dụng chọn từ danh sách các phần tử
VscrollBar	Cho phép người sử dụng cuộn dọc qua một điều khiển chứa

	dữ liệu khác
DriveListBox	Cho phép người sử dụng chọn ổ đĩa
FileListBox	Cho phép người sử dụng chọn một tập tin
Line	Hiển thị một đoạn thẳng trên biểu mẫu
Data	Cho phép lập trình để kết nối dữ liệu

Sau đây, ta sẽ tìm hiểu về các điều khiển nội tại phổ biến nhất. Các điều khiển không được đề cập đến trong chương này, do những khả năng đặc biệt riêng của nó, sẽ được dành trình bày trong các chương riêng phía sau.

4.2.1 Nút lệnh

a. **Phương thức:** *Click*
b. **Sự kiện:** MouseDown, KeyDown
c. **Thuộc tính:** Height,Font,BackColor,Caption, ShortcutKey

Đặt tên(thuộc tính **Name**) cho nút lệch thường bắt đầu bằng **cmd.** Ví dụ như cmdQuit, tương tự với hộp văn bản là **txt,** với biểu mẫu là **frm**, với nút tuỳ chọn là **opt**, v.v..Trong trường hợp dùng mảng điều khiển, tất cả các nút lêch có cùng tên.

Khi đặt tên cho điều khiển, ta cần tuân theo một số quy tắc. Điều này sẽ giúp chương trình của ta trở nên sáng sủa, dễ đọc, nhất là khi cần gỡ rối chương trình hoặc ta cần đọc lại chương trình sau vài tháng

4.2.1.1 Phân biệt hai thuộc tính Caption và Text

a. **Caption:** Dùng cho các đối tượng như biểu mẫu, khung, nút lệnh, thường để hiển thị tiêu đề cho đối tượng.
b. **Text:** Dùng cho những điều khiển thuộc loại nhận dữ liệu do người dùng nhập vào, như hộp văn bản, hộp kết hợp.

Ngoài ra ta có thể quy định phím nóng cho các điều khiển có thuộc tính Caption, bằng cách đặt dấu & kế bên kí tự. Ví dụ &Thoát.

4.2.2 Hộp văn bản

Là một điều khiển rất thông dụng dùng để nhận dữ liệu từ người sử dụng cũng như hiển thị dữ liệu trên màn hình. Visual basic và Windows tự động xử lý những hoạt động như hiển thị ký tự khi Người sử dụng gõ vào, chèn và xoá ký tự, cuốn dữ liệu, đánh dấu văn bản, cắt dán,...

4.2.2.1 Kiểm tra giá trị nhập

Hộp dữ liệu không tự kiểm tra dữ liệu nhập vào, lập trình viên phải làm việc đó. Mặc định, hộp văn bản nhận và hiển thị mọi ký tự mà Người sử dụng nhập vào, kể cả khi ta muốn gõ mật khẩu hoặc hcỉ muốn nhận con số. Nếu ta đổi thuộc tính MaxLength thành một con số, ví dụ 5, ta chỉ nhập được 5 lý tự. Nếu đổi MaxLength về 0 thì ta có thể nhập tuỳ thích.

4.2.2.2 Sự kiện KeyPress

Sự kiện này được phát ra khi Người sử dụng gõ vào hộp văn bản.

Mỗi ký tự trên bàn phím có một con số duy nhất, gọi là mã ASCII. Ta có thể xem toàn bộ bảng mã này trong cửa sổ help.

4.2.3 Điều khiển thanh cuộn

Thanh cuộn(Scroll bar) cho phép duyệt dễ dàng qua một danh sách dài gômd nhiều phần tử hoặc một lượng lớn thông tin bằng cách cuộn ngang hoặc cuộn dọc ở trong ứng dụng hay điều khiển. Đây là một điều khiển thông dụng của Windows.

Điều khiển thanh cuộn dùng sự kiện Scroll và Change để theo dõi sự dịch chuyển của hộp cuộn trên thanh cuộn.

Sự kiện	Mô tả
Change	Xảy ra sau khi hộp cuộn dịch chuyển
Scroll	Xảy ra khi hộp cuộn dịch chuyển. Không xảy ra nếu mũi tên cuộn hoặc thanh cuộn được nhấn.

Sử dụng sự kiện Scroll cho phép truy cập đến giá trị thanh cuộn khi nó được kéo đi. Sự kiện Change xảy ra sau khi hộp cuộn được nhả hay là khi thanh cuộn hoặc mũi tên cuộn được nhấn.

4.2.3.1 Thuộc tính Value

Thuộc tính Value (mặc định là 0) là một số nguyên tương ứng với vị trí của hộp cuộn trong thanh cuộn. Khi hộp cuộn ở vào giá trị nhỏ nhất nó dịch chuyển về bên trái, hay phía trên cùng. Khi hộp cuộn vào giá trị lớn nhất, nó dịch chuyển về bên phải hoặc là phía dưới cùng. Tương tự, giá trị trung bình sẽ đặt hộp cuộn vào giữa thanh cuộn.

4.2.4 Điều khiển Timer

Các điều khiển timer đáp ứng với thời gian trôi qua, chúng độc lập với người sử dụng, và ta có thể lập trình với chúng để thi hành một hành động trong các khoảng thời gian đều đặn. Kiểu đáp ứng điển hình là kiển tra giờ hệ thống xem đã đến lúc thi hành nhiệm vụ nào đó chưa.

Mỗi điều khiển Timer có thuộc tính Interval chỉ ra số phần nghìn giây trôi qua giữa hai sự kiện timer. ngoại trừ khi nó bị vô hiệu hoá, timer tiếp tục nhận sự kiện tại các thời khắc bằng khoảng thời gian quy định.

4.2.5 Điều khiển nhãn

Thường đi kèm với hộp văn bản. Bởi vì hộp văn bản không có thuộc tính caption như nút lệnh, nên nhãn làm nhiệm vụ đó. Thường ta chỉ thao tác với nhãn qua vài thuộc tính như gán font chữ, Cption, BorderStyle....

4.2.6 *Checkbox:* 

4.2.7 Một số thuộc tinh thông dụng:

Enable
Viable

Focus

4.2.8 Thứ tự điều khiển (TabIndex)

Đôi khi dung tab để điều khiển thay vì dùng chuột. Thuộctính tabIndex thực hiện điều này.

4.2.8 4.2.9 Hộp danh sách (Listbox).

Biểu tượng danh sách listbox trong toolbox: 

Trong thực tế,danh sách rất cần thiết. Một hệ thống nhân sự cần liệt kê các nhóm công việc và tên các phòng ban để đưa các nhân viên vào hệ thống.

Người sử dụng chỉ thấy những gì họ được phép xem. Họ sẽ được phép chọn một hoặc một vài phần tử trong danh sách.

4.2.9.1 Sắp xếp

VB mặc định các phần tử được sắp xếp theo thứ tự mà chúng được nhập vào danh sách.

Muốn sắp xếp theo thứ tứ ABC ta đổi thuộc tính **Sorted** thành TRUE, thuộc tính này chỉ được đổi trong khi thiết kế không được đổi trong lúc thi hành.

Nhưng thuọc tính này làm chậm đi quá trình thêm phần tử vào danh sách. Ta có thể thêm phần tử và đồng thời chỉ ra vị trí mất muốn thêm:

List.AddItem "Zebra",3 (Thêm phần tử có tên là Zebra vào vị trí thứ 4 của danh sách và ListIndex là 3).

Để chắc chắn giá trị dung là hợp lệ ta dung **listcount:**

```
NNewPosition=6
If  Listcount > 6 Then
        List.AddItem "Zebra",nNewPosition
End If
```

4.2.9.2 Thêm một phần tử vào danh sách.

Sử dung lệnh:

```
List.AddItem <Tên phần tử,Index>
```

4.2.9.3Xoá một phần tử từ danh sách.

Sử dung lệnh:

```
List.RemoveItem <Ind>
```

4.2.10 Hộp kết hợp (Combo Box)

Biểu tượng hộp kết hợp Combo Box: 

4.2.11 Điều khiển OLE

Biểu tượng trong tool box: 

OLE là tên gọi tắt của Oject Linking and Embedding. Nó cho phép ta nhúng toàn bộ ứng dụng và dư liệu của nó vào chương trình của ta.

Các điều khiển mới

4.3 Các điều khiển M ới

Điều khiển ADO data
Điều khiển Coolbar
Điều khiển D ata grid
Điều khiển Datalist, DataCombo
Điều khiển DataRepeater
Điều khiển DataTimePicker
Điều khiển Flat Scollbar
Điều khiển Hierarchical FlexGrid
Điều khiển ImageComBo
Điều khiển Month View

5 Nhập môn lập trình

Các điều khiển trên biểu mẫu chỉ là một phần nhỏ của quá trình lập trình phát triển ứng dụng, nhằm tạo ra giao diện cho ứng dụng. Sau đó, bạn cần viết chương trình để ứng dụng hoạt động. Do đó, chương này sẽ đi sâu vào phần công việc chính của Visual Basic, viết chương trình.

Visual Basic là ngôn ngữ lập trình dựa trên đối tượng. Nếu bạn là người mới học, chương này sẽ giới thiệu các khối thiết kế cơ bản để xây dựng chương trình. Khi đã hiểu được các khái niệm cơ bản, bạn có thể tạo ra các ứng dụng rất mạnh bằng Visual Basic.

5.1 Chuẩn lập trình (Coding convention)

5.1.1 Coding conventions

Object Naming Conventions
Object name has 2 parts: prefix and description.

The prefix that makes it easy to identify the type of object, the description mentions name of objects.

- Conventions of description part are:

  + In English.

  + Can contain many words, each word is contiguous to others (No hyphen).

  + No acronym except listed in table Acronyms (see 4. Acronym).

  + Capitalize the first letter of each word.

(Note: These conventions will be applied to all of name types mentioned after in this document)

- Prefix conventions for some of the objects supported by Visual Basic are listed below (Sorted by control name):

Control type	Prefix	Example
3D Panel	Pnl	pnlGroup
ADO Data	Ado	adoBiblio
Animated button	Ani	aniMailBox
Check box	Chk	chkReadOnly
Combo box, drop-down list box	Cbo	cboEnglish
Command button	Cmd	cmdExit
Common dialog	dlg	dlgFileOpen
Communications	com	comFax
Control (used within procedures when the specific type is unknown)	ctr	ctrCurrent

Data	dat	datBiblio
Data-bound combo box	dbcbo	dbcboLanguage
Data-bound grid	dbgrd	dbgrdQueryResult
Data-bound list box	dblst	dblstJobType
Data combo	dbc	dbcAuthor
Data grid	dgd	dgdTitles
Data list	dbl	dblPublisher
Data repeater	drp	drpLocation
Date picker	dtp	dtpPublished
Directory list box	dir	dirSource
Drive list box	drv	drvTarget
File list box	fil	filSource
Flat scroll bar	fsb	fsbMove
Form	frm	frmEntry
Frame	fra	fraLanguage
Gauge	gau	gauStatus
Graph	gra	graRevenue
Grid	grd	grdPrices
Hierarchical flexgrid	flex	flexOrders
Horizontal scroll bar	hsb	hsbVolume
Image	img	imgIcon
Image combo	imgcbo	imgcboProduct
ImageList	ils	ilsAllIcons
Label	lbl	lblHelpMessage
Lightweight check box	lwchk	lwchkArchive
Lightweight combo box	lwcbo	lwcboGerman
Lightweight command button	lwcmd	lwcmdRemove
Lightweight frame	lwfra	lwfraSaveOptions
Lightweight horizontal scroll bar	lwhsb	lwhsbVolume
Lightweight list box	lwlst	lwlstCostCenters
Lightweight option button	lwopt	lwoptIncomeLevel
Lightweight text box	lwtxt	lwoptStreet
Lightweight vertical scroll bar	lwvsb	lwvsbYear
Line	lin	linVertical
List box	lst	lstPolicyCodes
ListView	lvw	lvwHeadings
MAPI message	mpm	mpmSentMessage
MAPI session	mps	mpsSession
MCI	mci	mciVideo
Menu	mnu	mnuFileOpen
Month view	mvw	mvwPeriod
MS Chart	ch	chSalesbyRegion
MS Flex grid	mfg	mfgClients
MS Tab	mst	mstFirst
OLE container	ole	oleWorksheet

Option button	opt	optGender
Picture box	pic	picVGA
Picture clip	clp	clpToolbar
ProgressBar	prg	prgLoadFile
Remote Data	rd	rdTitles
RichTextBox	rtf	rtfReport
Shape	shp	shpCircle
Slider	sld	sldScale
Spin	spn	spnPages
StatusBar	sta	staDateTime
SysInfo	sys	sysMonitor
TabStrip	tab	tabOptions
Text box	txt	txtLastName
Timer	tmr	tmrAlarm
Toolbar	tlb	tlbActions
TreeView	tre	treOrganization
UpDown	upd	updDirection
Vertical scroll bar	vsb	vsbRate

Prefix conventions for menus

Menu control prefixes will be extended beyond the initial "mnu" label by adding an additional prefix for each level of nesting, with the final menu caption at the end of the name string. The following table lists some examples.

Menu caption sequence	Menu handler name
File Open	mnuFileOpen
File Send Email	mnuFileSendEmail
File Send Fax	mnuFileSendFax
Format Character	mnuFormatCharacter
Help Contents	mnuHelpContents

Variable naming conventions

Variable name must describe data type, scope and identifier of a variable.

Variable data types

Data type	Prefix	Example
Boolean	bln	
Byte	byt	
Currency	cur	
Date (Time)	dtm	
Double	dbl	
Error	err	
Integer	int	
Long	lng	
Object	obj	
Single	sng	

String	str	
User-defined type	udt	
Variant	vnt	

Variable scope prefixes

Scope	Prefix	Example	Note
Global	G	gstrUserName	This variable is global and string type
Module-level	M	mblnCalcInProgress	This variable is module and boolean type
Local to procedure	None	dblVelocity	This variable is local and double type

Constants

The constant names will be UPPER_CASE with underscores (_) between words. For example:

Example	Note
USER_LIST_MAX	
NEW_LINE	

Prefixes for ActiveX Data Objects (ADO)

Use the following prefixes to indicate ActiveX Data Objects.

ADO object	Prefix	Example
Command	Cm	cmTitles
Connection	Cn	cnTitles
Field	Fld	fldName
Field Collection	flds	fldsTitles
Parameter	prm	prmTitleName
Parameter Collection	prms	prmsNames
Recordset	Rs	rsTitles

Structured Coding Conventions

In addition to naming conventions, structured coding conventions, such as code commenting and consistent indenting, can greatly improve code readability.

Code Commenting Conventions

All procedures and functions should begin with a brief comment describing the functional characteristics of the procedure (what it does). Input, output patameters passed to a procedure should be described. Function return values and global variables that are changed by the procedure must also be described at the beginning of each procedure.

Section heading	Comment description
Purpose	What the procedure does (not how).
Inputs	Describe roles of input parameters
Outputs	Describe roles of output parameters
Returns	Explanation of the values returned by functions.
Author	Author of module

Remember the following points:

Every important variable declaration should include an inline comment describing the use of the variable being declared.

Variables, controls, and procedures should be named clearly enough that inline commenting is only needed for complex implementation details.

At the start of the.bas module that contains the project's Visual Basic generic constant declarations, we should include an overview that describes the application, enumerating primary data objects, procedures, algorithms, dialogs, databases, and system dependencies. Sometimes a piece of pseudocode describing the algorithm can be helpful.

Formatting Your Code

Here are a few pointers:

Standard, tab-based, nested blocks should be indented four spaces (as the Visual Basic default).

The functional overview comment of a procedure should be indented one space. The highest level statements that follow the overview comment should be indented one tab, with each nested block indented an additional tab. For example:

```
'*****************************************************
' Purpose:   Locates the first occurrence of a
'            specified user in the UserList array.
' Inputs:
'   strUserList():   the list of users to be searched.
'   strTargetUser:   the name of the user to search for.
' Returns:   The index of the first occurrence of the
'            rsTargetUser in the rasUserList array.
'            If target user is not found, return -1.
'*****************************************************

Function FindUser (strUserList() As String, strTargetUser As _
   String)As Integer
   Dim inti As Integer            ' Loop counter.
   Dim blnFound As Integer      ' Target found flag.
   intFindUser = -1
   inti = 0
   While inti <= Ubound(strUserList) and Not blnFound
      If strUserList(inti) = strTargetUser Then
         blnFound = True
         intFindUser = inti
      End If
   Wend
End Function
```

Grouping Constants

Visual Basic generic constants will be grouped in a single module to separate them from application-specific declarations.

& and + Operators

Always use the **&** operator when linking strings and the **+** operator when working with numerical values. Using the **+** operator to concatenate may cause problems when operating on two variants. For example:

```
vntVar1 = "10.01"
vntVar2 = 11
vntResult = vntVar1 + vntVar2    'vntResult = 21.01
vntResult = vntVar1 & vntVar2   'vntResult = 10.0111
```

Creating Strings for MsgBox, InputBox, and SQL Queries

When creating a long string, use the underscore line-continuation character to create multiple lines of code so that you can read or debug the string easily. This technique is particularly useful when displaying a message box (MsgBox) or input box (InputBox) or when creating an SQL string. For example:

```
Dim Msg As String
Msg = "This is a paragraph that will be " _
& "in a message box. The text is" _
& " broken into several lines of code" _
& " in the source code, making it easier" _
& " for the programmer to read and debug."
MsgBox Msg

Dim QRY As String
QRY = "SELECT *" _
& " FROM Titles" _
& " WHERE [Year Published] > 1988"
TitlesQry.SQL = QRY
```

Other conventions

Error trapping in development progress must follow these rules:

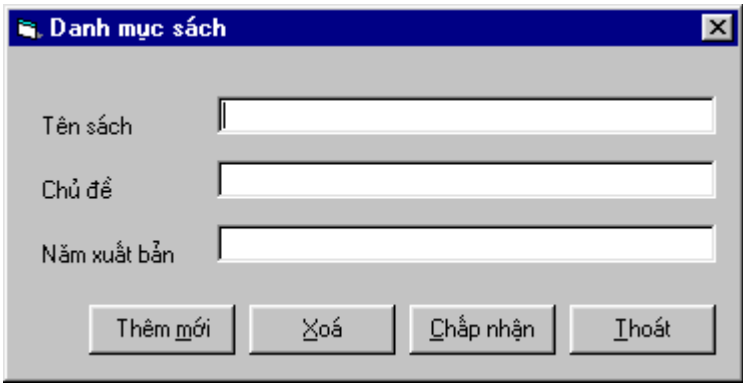
- Cascading error trapping. That mean all called functions will return system error codes, and showing message box will be implemented at the most exterior function/procedure.
- All system message will be located in resource file.
- Showing message boxes are implemented by pass parameters to a showing message global function.
- Error code contains 3 number.
- Versioning all modules: form module, code module...
- Display solution: 800x600 pixels
- Font size setting: Large font
- All of file name (*.vbp, *.frm, *.bas...) must less than 3 characters.
- Error code = 0 is OK.
- Error code < 0 is error.
- Error code >0 is warning.
- Function names should begin with a verb, such as InitNameArray or CloseDialog.

5.1.2 Form design standard

Common conventions in form design

Items	Conventions	Note
Interface	In Vietnamese	
Font name	MS Sans serif	Default
Font size	Default	
Font color	Black	Default
Font style	Normal	Default
Distance between command buttons	100 Points	
Command buttons alignment	Right	
Label alignment	Left	
All labels in a form must be collected in an array		
All command buttons in a form must have same width		
Caption of OK button	Chấp nhận	
Caption of Cancel button	Thoát	
Caption of Help button	Hướng dẫn	
Caption of Add button	Thêm mới	
Caption of Delete button	Xoá	
Caption of Edit button	Sửa	
Caption of Close button	Đóng	
Caption of Save button	Ghi	
Order of buttons in from (Left to right): Thªm míi-Ghi-Söa-Xo¸-ChÊp nhËn-Tho¸t-H-íng dÉn		
Default button	Chấp nhận	
Cancel button	Đóng	

Sample:



Form controls appearance conventions

Control	Property	Value	Note
Check box			
	All colors	Default	
Combo box, drop-			

down list box			
	Height	315	
	All colors	Default	
Command button			
	Height	375 Points	
	Back color	Button face	VB Default
Form			
	Border style	Fixed length	
	Back color	Button face	VB Default
	Startup position	CenterScreen	
Label			
	Back color	Button face	VB Default
Vervical scroll bar			
	Width	260 Points	
Horizontal scroll bar			
	Height	260 Points	
Option button			
	Height	255 Points	
Text box			
	Height	285 Points	
	All color	Default	

5.1.3 Report design standard (for Crystal Report)

Common conventions in report design

Items	Conventions	Note
Above of each total row must be a line, called ***Grouping line***		
Report boder	Only title row is bordered	
Colour of all objects in report (line, character...)	Black	

Report objects appearance conventions

Object	Property	Setting	Note
Report title			
	Font name	.VnArialH	
	Font style	Bold	
	Justify	Paper center	
	Space between report title and collumn heading	50 Points	Should be reviewed
Colunm heading			
	Font name	.VnArialNarrow	
	Font style	Bold	
	Font size	10	
	Justify	Column left	
	Before row spacing	6 Points	
	After row spacing	6 Points	
Report body (data)			
	Font name	.VnArialNarrow	
	Font style	Bold	
	Font size	9	
	Justify	Depend on column data type (Number: right, string: right, date: center)	
	Row spacing	0	
Total row			
	Font name	.VnArialNarrow	
	Font style	Bold	
	Font size	9	

	Justify	Right	
	Position	Under data block that summarized	
Grouping line			
	Width	1 Point	
Report boder			
	Width	1 Point	
	Boder column title only		
Page number			
	Font name	.VnArial	
	Font size	9	
	Font style	Normal	
	Position	Right, bottom of page (Report footer)	
	Style	Page/Total page	
Left sub title			
	Font name	.VnArial	
	Font size	10	
	Font style	Normal	
	Justify	Margin left	
Right sub title			
	Font name	.VnArial	
	Font size	10	
	Font style	Normal	

Sample:

Left subtitle Report title Right subtitle

Column heading 1	Column heading 2	**Column heading 3**	**Column heading 4**
Report body 1	Report body 2	Report body 3	Report body 4
Report body 1	Report body 2	Report body 3	Report body 4
			Total row 4
Report body 1	Report body 2	Report body 3	Report body 4
Report body 1	Report body 2	Report body 3	Report body 4
			Total row 4

(Page number) 1/5

5.1.4 Database design standards

All of object names (include: table names, view names, field names...) in database must follow these conventions:

- In English.
- Can containts one or more words and ~~no~~ underscore between these words.
- No acronym except listed in table Acronyms bellow.

- The first letter of each word must be capitalized.

5.2 Thiết kế trước khi viết chương trình

Có lẽ khâu quan trọng nhất trong lập trình là thiết kế. Sau khi thiết kế giao diện, bạn cần thiết kế cấu trúc chương trình. Cách thiết kế khác nhau sẽ dẫn đến cách hoạt động khác nhau và bảo trì, theo đó cũng khác nhau.

Code trong VB được tổ chức theo dạng cây phân nhánh. Một ứng dụng thông thường chứa một hoặc nhiều mô-đun. Mỗi biểu mẫu có một mô-đun, có thể thêm những mô-đun chuẩn chứa những đoạn chương trình dùng chung, và cũng có thể có thêm mô-đun lớp.

5.3 Các thao tác thông dụng trong cửa sổ Code

5.3.1 Soạn thảo Code

Ngoài khả năng soạn thảo văn bản để viết chương trình, cửa sổ Code còn hỗ trợ một số chức năng khác như:

Đánh dấu (bookmarks)

Dùng đánh dấu các dòng chương trình trong cửa sổ Code để dễ dàng xem lại về sau. Để bật tắt khả năng này, cũng như tìm kiếm dấu hiện hành, chọn Bookmarks từ menu Edit, hoặc chọn từ thanh công cụ Edit.

Dùng phím trong cửa sổ Code

Chức năng	**Phím tắt**
Xem cửa sổ Code	F7
Xem cửa sổ Object Browser	F2
Tìm kiếm	CTRL + F
Thay thế	CTRL + H
Tìm tiếp	SHIFT + F4
Tìm ngược	SHIFT + F3
Chuyển đến thủ tục kế tiếp	CTRL + DOWN ARROW
Chuyển đến thủ tục trước đó	CTRL + UP ARROW
Xem định nghĩa	SHIFT + F2
Cuộn xuống 1 màn hình	CTRL + PAGE DOWN
Cuộn lên một màn hình	CTRL + PAGE UP
Nhảy về vị trí trước đó	CTRL + SHIFT + F2
Trở về đầu của mô-đun	CTRL + HOME
Đến cuối mô-đun	CTRL + END
Chức năng	**Phím tắt**
Dời con trỏ sang phải 1 từ	CTRL + RIGHT ARROW
Dời con trỏ sang trái 1 từ	CTRL + LEFT ARROW
Dời con trỏ về cuối dòng	END
Dời con trỏ về đầu dòng	HOME
Lấy lại hành động trước đó	CTRL + Z
Xoá dòng hiện hành	CTRL + Y
Xoá 1 từ	CTRL + DELETE

Canh trái	TAB
Bỏ hành động canh trái trước đó	SHIFT + TAB
Xoá tất cả các điểm dừng (break-points)	SHIFT + SHIFT + F9
Xem menu cảm ngữ cảnh	SHIFT + F10

5.3.2 Một số chức năng tự động

5.3.2.1 Auto Syntax Check

Từ menu Tools, chọn Option… Hộp thoại xuất hiện

Khi Auto Syntax Check không bật lên, nêu ta viết 1 dòng chương trình như sau:

Form1.left =

rồi nhấn phím Enter. VB sẽ hiển thị dòng chương trình sai với mầu đỏ. Tuy nhiên, nó không giải thích thêm và ta có thể tiếp tục gõ chương trình. Nếu Auto Syntax Check được bật lên, khi ta vừa nhấn phím Enter, VB lập tức cho ta biết một số thông tin về lỗi và hiển thị con trỏ ngay dòng chương trình sai để chờ ta sửa. Trong trường hợp này, VB cần một giá trị bên phải dấu bằng.

5.4 Biến hằng và các kiểu dữ liệu

Dùng để chứa dữ liệu tạm thời cho tính toán, so sánh các hoạt động khác

Ta dùng toán tử (=) để tính toán và chứa giá trị vào biến

5.4.1 Khai báo biến

Để khai báo biến ta dùng lệnh Dim:

Dim <Tên biến > [As<kiểu dữ liệu>]

Biến khai báo trong thủ tục chỉ tồn tại khi thủ tục thi hành. Nó sẽ biến mất khi thủ tục chấm dứt. Giá trị của biến trong thủ tục là cục bộ đối với thủ tục đó, nghĩa là ta không thể truy nhập biến từ bên ngoài thủ tục. Nhờ đó, ta có thể dùng trùng tên biến cục bộ trong những thủ tục khác nha.

Kiểu dữ liệu trong khai báo Dim có thể là những kiểu cơ bản như Integer, String hoặc Currency. Ta cũng có thể dùng đối tượng của VB (như Object, Form1, TextBox) hoặc của các ứng dụng khác.

Khai báo biến trong phần Declarations của một mô-đun nghĩa là biến đó tông tại và có tầm hoạt động trong mô-đun đó.

Khai báo biến với từ khoá Public nghĩa là biến đó tồn tại và có tầm hoạt động của toàn ứng dụng .

Khai báo biến cục bộ với từ khoá Static nghĩa là mặc dầu biến đó biến mất khi thủ tục chấm dứt, nhưng giá trị của nó vẫn được giữ lại để tiếp tục hoạt động khi thủ tục được gọi trong lần sau.

5.4.2 Khai báo ngầm

Nghĩa là ta không cần khai báo tường minh trước khi sử dụng biến.

```
Function SafeSqr(num)
    TempVal = Abs(num)
```

```
        SafeSqr = Sqr(TempVal)
    End Function
```

Mặc dù cách này có vẻ thuận tiện nhưng có thể gây lỗi nếu ta gõ nhầm tên biến.

```
    Function SafeSqr(num)
        TempVal = Abs(num)
        SafeSqr = Sqr(TemVal)
    End Function
```

Hàm này trả về zero. Khi VB gặp tên mới, nó tạo ra một biến khác với tên đó.

5.4.3 Khai báo tường minh

Để tránh những rắc rối trên, ta nên quy định VB phải báo lỗi khi gặp một tên biến không khai báo. Ta đặt dòng lệnh :

```
Option Explicit
```

Trong phần Declarations của mô-đun. Một cách khác, từ menu **Tools**, chọn **Options**, chọn tab Editor và đánh dấu vào tuỳ chọn Require Variable Declaration. VB tự động chèn dòng lệnh Option Explicit vào một mô-đun mới, nhưng không phải là những mô-đun đã được tạo. Do đó, đối với các mô-đun này, ta phải thêm dòng lệnh bằng tay.

Option Explicit chỉ hoạt động trên từng mô-đun. Vì vậy, ta phải thêm dòng này vào mỗi mô-đun của biểu mẫu, mô-đun chuẩn, hay mô-đun lớp.

5.4.3.1 Tầm hoạt động của biến

Tầm hoạt động	**Private**	**Public**
Thủ tục	Biến chỉ tồn tại và hoạt động trong thủ tục	Không có
Mô-đun	Biến chỉ tồn tại và hoạt động trong mô-đun	Biến tồn tại và hoạt động trên mọi mô-đun

5.4.4 Khai báo biến Static

Để khai báo tát cả các biến cục bộ trong một thủ tục là Static, ta đặt từ khoá Static vào tên thủ tục:

```
Static Function RunningTotal(num)
```

VB sẽ hiểu rằng tất cả các biến khai báo trong thủ tục này đều là Static, dù cho chúng được khai báo là Private, là Dim hoặc thậm chí khai báo ngầm.

Từ khoá Static có thể đặt ở đầu thủ tục Sub hoặc Function, kể cả thủ tục xử lý sự kiện hoặc những hàm Private.

5.4.5 Hằng

Dùng để chứa những dữ liệu tạm thời nhưng không thay đổi trong suốt thời gian chương trình hoạt động. Sử dụng hằng số làm chương trình sáng sủa và dễ đọc

nhờ những tên gợi nhớ thay vì các con số. VB cung cấp một số hằng định nghĩa sẵn, nhưng ta có thể tự tạo hằng.

Ta có thể dùng cửa sổ Object Browser để xem danh sách các ứng dụng hằng có sẵn của VB và VBA(Visual basic for Application). Các ứng dụng khác cung cấp những thư viện đối tượng, như Microsoft Exel, Microsoft Project, hoặc các thư viện của điều khiển ActiveX cũng có hằng định nghĩa sẵn.

Trong trương hợp trùng tên hằng trong những thư viện khác nhau, ta có thể dung cách chỉ rõ tham chiếu hằng:

[<Libname>][<tên mô-đun>]<tên hằng>

Libname là tên lớp, tên điều khiển hoặc tên thư viện.

5.4.5.1 Khai báo hằng

|Public|private|Const<tên hằng>[**As**<kiểu dữ liệu>]= <biểu thức>

Tầm hoạt động

Hằng cũng có tầm hoạt động tương tự biến:

Hằng khai báo trong thủ tục chỉ hoạt động trong thủ tục

Hằng khai báo trong mô-đun chỉ hoạt động trong mô-đun

Hằng khai báo Public trong phần Declarations của mô-đun chuẩn có tầm hoạt động trên toàn ứng dụng.Khai báo Public không thể dùng trong mô-đun của biểu mẫu hoặc mô-đun lớp.

5.4.5.2 Kiểu dữ liệu

Kiểm soát nội dung của dữ liệu. VB dùng kiểu Variant như là kiểu mặc định. Ngoài ra, một số kiểu dữ liệu khác cho phép tối ưu hoá về tốc độ và kích cỡ chương trình. Khi dùng Variant, ta không phải chuyển đổi giữa các kiểu dữ liệu. VB tự động làm việc đó.

Một dòng lệnh có thể kết hợp nhiều kiểu khai báo :

Private I as Interger, Amt as double

Private YourName as String, BillsPaid as Currency

Private Test,Amount, J as integer

5.4.5.3 Kiểu số

Integer, Long, Double và Currency. Kiểu số tốn ít vùng chứa hơn kiểu Variant.

Tất cả biến kiểu số có thể được gán cho nhau và cho biến Variant. VB làm tròn thay vì chặt bỏ phần thập phân trước khi gắn nó cho số Integer.

Kiểu Integer tốn ít vùng nhớ hơn các kiểu khác, nó thường dùng làm biến đếm trong các vòng lặp For....Next.

Kiểu Single, Double, Currency dùng cho các số có phần thập phân. Currency hỗ trợ đến 4 chữ số phần thập phân và 15 chữ số cho phần nguyên, đùn cho ácc tính toán tiền tệ.

Các giá trị dấu chấm động được thể hiện là :$A*10^B$.Ví dụ:

$1.2341E12=1.2341*10^{12}$

3.402823E+38 cho số Single hoặc 1.7976931486232D+308 cho số Double

Ta dùng các phép cộng (+), trừ(-) nhân(*), chia(/ hoặc\). Dấu / là số chia thập phân. Ví dụ: 5/3 cho kết quả là 1.666666666667. Trong khi 5/3 cho kết quả là 1,

phần thập phân bị chặt bỏ. Phép tính này đặc biệt nhanh khi sử dụng trong vòng lặp.

5.4.5.4 Kiểu Byte

Thường dùng để chứa dữ liệu nhị phân. Tất cả các thao tác trên kiểu Integer có thể thực hiện trên kiểu Byte, ngoại trừ dấu. Vì Byte là kiểu không dấu (trong khoản từ 0-255), nó không thể nhận ra số âm.

5.4.5.5 Kiểu String

Mặc định, biến hay tham số kiểu chuỗi có chiều dài thay đổi, nó có thể tăng hoặc giảm tuỳ theo ta gán dữ liệu. Ta có thể khai báo chuỗi có chiều dài cố định:

Dim EmpName As String *50

Nếu ta gán một chuỗi ngắn hơn 50 ký tự, EmpName sẽ được thêm vào phần đuôi các kỹ tự khoảng trắng cho đầy 50 ký tự, nếu chuỗi gán vào dài hơn 50 ký tự, VB tự động chặt bỏ.

Khi làm việc với chuỗi, ta cần dùng các ham Trim và RTrim để cắt bỏ các ký tự trắng không cần thiết. Ngoài ra một số hàm thông dụng để thao tác trên chuỗi như:

a. Len: Lấy chiều dài chuỗi

b. Mid$: Trích chuỗi con từ chuỗi gốc

c. Left$: Trích chuỗi con từ phần đầu chuỗi gốc.

d. Right$:Trích chuỗi con từ phần đuôi của chuỗi gốc.

e. InStr: Tìm chuỗi con trong chuỗi gốc. Nếu hàm InStr trả về 0, nghĩa là không tìm thấy.

Tìm kiếm không phân biệt cõ chữ. Nhưng nếu tham số thứ 3 là vbBinaryCompare thì đây là tìm kiếm chuỗi có phân biệt chữ in hoa và chữ in thường.

f. Replace: Tìm và thay thể chuỗi. Replace(“Peter PeterWright”,” Peter ”,John,6)

Chuỗi kết quả là “John Wright”, bắt đầu từ vị trí thứ 6. Nếu muốn giữ lại phần đầu ta làm như sau:

Left$(“Peter Peter Wright”,5)&Replace (“Peter Peter Wright”,”Peter”,”John”,6)

Một tham số khác là số lần thay thế:

Replace(“Peter Peter Peter Wright”,”Peter”,”Hooray”,1,2)

Kết quả là “Hooray Hooray Peter Wright”, nghĩa là hai lần thay thế. THam số này mang giá trị mặc định là -1, nghĩa là thay thế toàn bộ.

Tham số cuối cùng tương tự hàm Instr(), cho biết nó có phân biệt chữ in hoa và chữ

thường hay không

Replace(“Peter Wright, “Peter”, “P.”,1,-1,vbTextCompare)

Kết quả là “P.Wright”.

Chuỗi có chiều dài cố định được khai báo Public hay Private trong mô-đun chuẩn. Trong mô-đun của biểu mẫu hoặc mô-đun lớp, nó phải được khai báo Private.

VB cho phép chuyển đổi một chuỗi thành thành một số nếu chuỗi đang thể hiện một con số. Ngược lại, ta cũng có thể chuyển một số thành chuỗi. Tuy nhiên nên cẩn thận, vì chuyển đổi một chuỗi có giá trị không phải số sẽ gây lỗi chương trình thi hành.

Một số lập trình viên Visual Basic thích dùng dấu + để nối chuỗi thay vì dùng dấu &. Mặc dù không khác nhau lắm, nhưng thực ra dùng dấu + có điểm bất

tiện. Vì là phép toán, nó có kiểm tra kiểu. Nếu ta có một số và một chuỗi nối với nhau, nó sẽ chuyển đổi từ số sang chuỗi trước khi thực sự kết nối. Hơn nữa, việc chuyển đổi này được làm tự động, không hề báo lỗi khi biên dịch.

5.4.5.6 Kiểu Boolean

Nếu ta có một biến có hai giá trị True/False, Yes/No,On/Off, ta nên dùng kiểu Boolean. Giá trị mặc định của Boolean là False.

```
 Dim blnRunning as Boolean
‘ Check to see ì the tape is running.
If recorder.Direction = 1 Then
blnRunning = True
End if
```

5.4.5.7 Kiểu Date

Khi các kiểu dữ liệu khác được chuyển sang Date, giá trị đứng trước dấu chấm là ngày, giá trị đứng sau dấu chấm là giờ. Nửa đêm là 0, giữa ngày là 0,5. Dấu âm thể hiện ngày trước 30/12/1999. Kiểu Date đã giải quyết vấn đề Y2K
Nhấn Ctrl-G để hiển thị cửa sổ **Immediate.**
Gõ vào:
“01/02/98” và nhấn Enter.
Nếu hiểu theo người Mỹ, “01/02/98” có nghĩa là ngày 2 tháng Giêng năm 1998, nếu hiểu theo người Anh thì đây là ngày 1 tháng 2 năm 1998. Nếu dùng ngày như trong hình trên thì VB hiểu rằng lấy 1 chia cho 2 rồi lấy kết quả chia cho 98!
Trở lại cửa sổ Immediate gõ vào: ?#01/02/98#
Dấu # cho biết là dữ liệu kiểu Date, không phải một biểu thức toán học. Tuy nhiên, định dạng ngày tháng hiển thị phụ thuộc vào quy định của Windows.
Hộp thoại này hiển thị khi người sử dụng nhấp đúp chuột vào biểu tượng Regional Setting trong cửa sổ Control Panel của Windows. Nó cho phép quy định kiểu ngày tháng tuỳ thuộc quốc gia. Bên trong chương trình VB xử lý ngày tháng theo kiểu Mỹ #01/02/98# là ngày 2 tháng Giêng năm 1998, nhưng nếu máy đang dùng theo hệ Anh thì nó sẽ hiển thị trên cửa sổ Immediate là 2/1/98

5.4.5.8 Kiểu Object

Biến kiểu Object chứa một địa chỉ 4 byte (32bit) trỏ đến đối tượng trong ứng dụng hiện hành hoặc các ứng dụng khác. Dùng lệnh Set để chỉ ra đối tượng thực sự:

```
Dim objDb As Object
Set objDb=OpenDatabase(“c:\vb5\Biblio.mdb”)
```

Khi khai báo biến đối tượng, nên chỉ ra tên lớp tường minh, như TextBox thay vì Control, Database thay vì Object). Ứng dụng sẽ chạy nhanh hơn, ta có thể xem danh sách các lớp trong cửa sổ Object Browser.

5.4.5.9 Kiểu Variant

Có thể chứa mọi loại dữ liệu, số, thậm chí mảng. Ta không cần chuyển đổi kiểu dữ liệu, VB làm việc đó một cách tự động.

```
Dim Somevalue 'Variant by default
Somevalue = "17"                  'SomeValue contains "17"(a two character string).
Somevalue = Somevalue – 15        'somevalue now cotains the numeric value 2.
Somevalue = "U" & Somevalue       'somevalue now cotains.
```

Variant cũng thuận tiện khi ta không biết trước kiểu dữ liệu

```
Private Sub cmdExplore_click()
Dim VarVariant As Variant
   VarVariant = 12
   Form1.Print VarType(VarVariant)
   VarVariant = "Peter"
   Form1.Print VarType(VarVariant)
   VarVariant = True
   Form1.Print VarType(VarVariant)
   VarVariant = #1/1/2001#
   Form1.Print VarType(VarVariant)
End Sub
```

Hàm VarType kiểm tra kiểu dữ liệu

Giá trị VarType	Giải thích
0 – vbEmpty	Không chứa gì cả
1 – vbNull	Không có dữ liệu hợp lệ
2 – vbInteger	Dữ liệu Integer dạng chuẩn
3 – vbLong	Dữ liệu kiểu Long Integer
4 - vbsingle	Dữ liệu kiểu chấm động single
5 – vbDouble	Dữ liệu kiểu chấm động Double
6 – vbCurrency	Kiểu Currency
7 – vbDate	Kiểu ngày giờ
8 – vbString	Kiểu chuỗi đơn giản
9 – vbObject	Kiểu đối tượng
10 – vbError	Có một đối tượng Error
11 – vbBoolean	Kiểu giá trị Boolean chuẩn
12 – vbVariant	Kiểu Variant
13 – vbDataObject	Kiểu DAO chuẩn
14 – vbDecimal	Giá trị thuộc hệ thập phân Decimal
17 – vbByte	Kiểu Byte
36 – UserDefinedType	Kiểu do người dùng định nghĩa
8192 - vbArray	Kiểu mảng

Tuy nhiên cần chú ý khi dùng biến Variant:

Nếu muốn thi hành các hàm số học, Variant phải chứa giá trị số.

Nếu muốn nối chuỗi, dùng toán tử & thay vì toán tử +.

Giá trị Empty

Đôi khi ta cần kiểm tra một giá trị có được gán cho biến hay chưa. Biến Variant có giá trị Empty trước khi nó được gán giá trị. Giá trị Empty là một giá trị đặc biệt không phải zero, không phải chuỗi rỗng(“”), không phải giá trị Null. Ta dùng ham IsEmpty để kiểm tra giá trị Empty:

If IsEmpty(z) then z =0
Khi một biến Variant chứa giá trị Empty, ta có thể dùng nó trong biểu thức. Nó có thể được xem là 0 hoặc chuỗi rỗng tuỳ theo biểu thức.
Giá trị Empty biến mất khi có một giá trị bất kỳ được gán cho Variant. Muốn trở về giá trị Empty, ta gán từ khoá Empty cho Variant.

Giá trị Null

Biến Variant chứa giá trị Null dùng trong những ứng dụng cơ sở dữ liệu thể hiện không có dữ liệu hoặc dữ liệu không xác định.

Dùng hàm IsNull để kiểm tra biến Variant có chứa Null hay không. Biến không bao giờ mang giá trị Null nếu ta không gán trực tiếp cho nó. Vì vậy, không cần phải dùng hàm IsNull.
Nếu gán Null cho một biến khác kiểu Variant, VB sẽ báo lỗi.

Giá trị Error

Trong một biến Variant, Error là một giá trị đặc biệt thể hiện một điều kiện lỗi vừa xảy ra trong thủ tục. Tuy nhiên, không như các lỗi khác, các xử lý lỗi thông thường của ứng dụng không xảy ra. Do đó, ta có thể xử lý dựa trên các giá trị lỗi. Giá trị Error được sinh ra bằng cách chuyển đổi giá trị lỗi dùng cho hàm CVErr.

5.4.5.10 Chuyển đổi giữa các kiểu dữ liệu

Hàm chuyển đổi	Đổi sang kiểu
Cbool	Boolean
Cbyte	Byte
Ccur	Currency
CDate	Date
CDbl	Double
Cint	Integer
CLng	Long
CSng	Single
Cstr	String
Cvar	Variant
CVErr	Error

Lưu ý rằng giá trị truyền cho hàm phải hợp lệ, nghĩa là phải thuộc khoảng của kiểu kết quả. Nếu không VB sẽ báo lỗi.

5.4.5.11 Kiểu mảng(Array)

Mảng là một xâu các biến có cùng tên và cùng kiểu dữ liệu. Dung Array làm chương trình đơn giản và rút gọn, vì ta có thể dùng vòng lặp. Mảng có biên trên và biên dưới, và các thành phần trong mảng là liên tục giữa 2 biên.
Khái niệm mảng ở đây khác với mảng các điều khiển (Control Array). Control Array không cho phép nạp hay thoát khỏi một thành phần ở giữa Array.
Có 2 loại biến mảng mảng có chiều dài cố định và mảng động, có chiều dài thay đổi lúc thi hành. Mảng có chiều dài cố định có thể được khai báo Public trong ứng dụng. Private trong mô-đun hoặc Private trong một thủ tục.

5.4.5.11.1 Mảng có chiều dài cố định

Biên trên và biên dưới

Biên trên được xác định ngay lúc khai báo .

```
Dim counters(14) As Integer
Public sums(20) As Double
```

Mặc định, biên dưới là 0. Ta có thể khai báo tường minh biên dưới:

```
Dim counter(1 To 15) As Integer
Dim sums(100 To 120) As String
```

a. Hàm **UBound** trả về phần tử cuối của mảng(Upper Bound).

b. Hàm **LBound** trả về phần tử đầu tiên của mảng (Lower Bound).

Mảng trong mảng

```
Private Sub command1_click()
   Dim intX As Integer
'Declare and populate an integer array
   Dim countersA(5) As Integer
For intX = 0 To 4
   countersA(intX) = 5
Next intX
'Declare and populate a string array
   Dim countersB(5) As String
For intX = 0 To 4
   countersB(intX) = "Hello"
Next intX
   Dim arrX(2) As Variant 'Declare a new two-member
   arrX(1) = countersA()
   arrX(2) = countersB()
   MsgBox arrX(1)(2) ' display a member of each array
   MsgBox arrX(2)(3)
End Sub
```

Mảng nhiều chiều

Ta khai báo một mảng 2 chiều có 10 phần tử

```
Static MatrixA(9, 9) As Double
Static MatrixA(1 To 10, 1 To 10) As Double
Dim MultiD(3, 1 To 10, 1 To 15)
```

Khai báo này tạo ra một mảng 3 chiều có kích cỡ 4×10×15, là số phần tử của ma trân,600

Nên thận trọng trong khi sử dung các mảng nhiều chiều, nhất là các mảng các Variant vì nó lớn hơn các kiểu dữ liệu khác.

5.4.5.11.2 Mảng động(dynamic Array)

Mảng này có thể thay đổi kích cỡ. là một trong những ưu điểm của Visual Basic, mảng động giúp quản lý bộ nhớ một cách hiệu quả. Ta có thể dùng một mảng lớn trong thời gian ngắn, sau đó xoá bỏ để trả vùng nhớ cho hệ thống

Khai báo

Khai báo Public hoặc Dim trong mô-đun, hoặc khai báo Static hay Dim trong thủ tục. Khai báo một mảng động bằng cách cho nó một danh sách không theo chiều nào cả.

Dim DynArray(0

Cấp phát số phần tử thực sự bằng dòng lệnh ReDim.

ReDim DynArray(x+1)

Sử dụng ReDim

Dòng lệnh ReDim chỉ có thể xuất hiện trong thủ tục. Khác với Dim hay Static, ReDim là một dòng lệnh thi hành, nó làm ứng dụng phải thực hiện một hành động lúc chạy chương trình. Sử dụng ReDim tương tự trong mảng có chiều dài cố định, dùng thay đổi số phần tử cũng như biên trên hoặc biên dưới. tuy nhiên, số chiều không thay đổi.

```
ReDim DynArray(4 to 12)
Dim Matrix1() as integer
Sub CalcValuesNow()
        -
        -
        -
        ReDim Matrix1(19,29)
End sub
```

Mỗi lần gọi ReDim, tất cả các giá trị chứa trong mảng hiện hành bị mất. Vb khởi tạo lại giá trị cho chúng (Empty đối với mảng Variant, 0 cho mảng số, chuỗi rỗng cho mảng chuỗi, hoặc nothing cho mảng các đối tượng). Cách này tiện lợi khi ta muốn thêm dữ liệu mới hoặc muốn xoá bớt vùng nhớ. Đôi khi, ta muốn thay đổi kích cỡ của mảng mà không mất dữ liệu. Ta dùng ReDim với từ khoá Preserve. Ví dụ, mở rộng mảng thêm một phần tử và không mất dữ liệu:

ReDim Preserve DynArray(UBound(DynArray)+1)

Tuy nhiên chỉ có biên trên của chiều cuối cùng trong mảng được thay đổi khi ta dùng Preserve. Nếu thay đổi chiều khác, hoặc biên dưới của chiều cuối cùng VB sẽ báo lỗi.

5.4.5.11.3 Một số tính năng mở rộng của mảng

Không những gán mảng cho một mảng, ta còn tạo các hàm trả về mảng và các thuộc tính trả về mản. Trong nhiều trường hợp, những kỹ thuật này sẽ cải tiến đáng kể tốc độ xử lý chương trình

Sao chép mảng

Trong Visual Basic 5, để sao chép từ một mảng sang một mảng khác, ta phải dung vòng lặp For Each quét qua mảng nguồn, rồi tuần tự ReDim lại mảng đích để copy từng phần tử.

Tuy nhiên cách này chỉ áp ụng cho mảng Dynamic mà thôi. Khi gán biến, có một số quy luật mà ta cần nhớ. Ví dụ: Ta có thể gán một giá trị kiểu Integer vào biến long, không vấn đề nhưng gán Long cho Integer sẽ gây lỗi tràn. Ngoài quy luật về kiểu dữ liệu, việc gán mảng cũng có những quy luật liên quan đên số chiều, kích thước của chiều và laọi mảng gì (mảng có chiều dài cố định hay mảng động)

Gán mảng với chiều và kiểu dữ liệu khác nhau có thể không thành công, do những nguyên nhân sau:

Mảng bên trái dấu gán(=) là mảng chiều dài cố định hay mảng động

Số chiều của mảng bên trái có đồng nhất với số chiều của mảng bên phải không

Số phần tử trên mỗi chiều của mỗi bên có tương thích không. Chiều có thể tương thích thậm chí khi khai báo khác nhau, ví dụ như một mảng bắt đầu từ số 0 trong khi mảng kia bắt đầu từ số 1 miên là chúng có cùng số phần tử

Kiểu dữ liệu cảu các phần tử mỗi bên phải tương thích.

5.5 Hàm và thủ tục

- Chia nhỏ chương trình thành nhiều phần logic, giúp gỡ rối dễ dàng.
- Thủ tục có thể được sử dụng lại trong một ứng dụng khác.

Các loại thủ tục

a. Thủ tục không trả về giá trị

[Private | Public | Static] Sub <Tên thủ tục> (Tham số)

Các dòng lệnh

End sub

b. Hàm luôn trả về giá trị:

[Private | Public | Static] Function <Tên hàm> (Tham số) [As <Kiểu dữ liệu>]

Các dòng lệnh

End Function

Trong trường hợp không khai báo As <type>, mặc định, VB hiểu là kiểu variant

c. Thủ tục thuộc tính

Có thể trả về và gán giá trị, hay đặt tham chiếu đến đối tượng.

Xem thủ tục trong modul hiện hành

Trong cửa sổ code, chọn Genaral trong hộp Object, và chọn tên thủ tục trong hộp Procedure.

Để xem thủ tục xử lý sự kiện chọn tên đối tượng từ hộp Object trong cửa sổ code, sau đó chọn tên sự kiện trong hộp procedure

Thoát khỏi thủ tục / hàm

Exit sub dùng để thoát khỏi thủ tục, **Exit Function** dùng để thoát khỏi hàm.

5.6 Cấu trúc điều khiển

5.6.1 Cấu trúc chọn

So sánh mặc định trong Visual basic mặc định là so sánh có phân biệt cỡ chữ. Nếu muốn tắt chế độ này, ta thêm dòng khai báo sau vào chương tình

Option Copare Text

Nếu muốn trả về trạng thái ban đầu, có 2 cách:

Đưa dòng khai báo:

Option Compare Binary

Chỉ cần xoá dòng khai báo “Option Compare Text”

Các biểu thức so sánh

Ký hiệu	Ý nghĩa
=	Bằng
<>	Khác
>	Lớn hơn
<	Nhỏ hơn
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng

5.6.1.1 If.. Then

Một dòng lệnh
If <điều kiện> Then <dòng lệnh>
Nhiều dòng lệnh
If <điều kiện> Then
    <dòng lệnh>
End if

Điều kiện là một so sánh hay một biểu thức mang giá trị số. Visual basic thông dịch giá trị này thành True / False. Nếu True thì Visual basic thi hành dòng lệnh sau từ khoá Then.

5.6.1.2 If.. Then...Else

If <điều kiện 1> Then
    [Khối lệnh - 1]
ElseIf <điều kiện 2> Then
    [Khối lệnh - 2]
.....
Else
    [Khối lệnh - n]
End if

5.6.1.3 Select Case

Giải quyết trường hợp có quá nhiều ElseIf được dùng, giúp chương trình sáng sủa dễ đọc. Biểu thức để so sánh được tính toán một lần vào đầu cấu trúc. Sau đó Visual basic so sánh kết quả biểu thức với từng Case. Nếu bằng nó thi hành khối lệnh trong Case đó.

Select Case <biểu thức kiểm tra>
    Case <danh sách biểu thức 1>
        Khối lệnh 1
    Case <danh sách biểu thức 2>
        Khối lệnh 2
    .....
    Case else
        Khối lệnh n
End Select

Mỗi danh sách biểu thức chứa một hoặc nhiều giá trị, các giá trị cách nhau bằng dấu phảy. Mỗi khối lệnh có thể chứa từ 0 đến nhiều dòng lệnh. Nếu có hơn một Case thoả mãn điều kiện thì Case đầu tiên được thực hiện. Case else không nhất thiết phải có, dùng trong trường hợp còn lại của các Case trước.

5.6.2 Cấu trúc lặp

5.6.2.1 Do..loop

Thi hành một khối lệnh với số lần lặp không định trước, tổng đó, một biểu thức điều kiện dùng so sánh để quyết định vòng lặp có tiếp tục hay không. điều kiện phải quy về False hoặc True.

Kiểu 1:

Do While <điều kiện>

<khối lệnh>

Loop

Kiểu 2:Vòng lặp luôn có ít nhất một lần thi hành khối lệnh

Do

<Khối lệnh>

Loop While <điều kiện>

Kiểu 3: Lặp trong khi điều kiện là False

Do until <điều kiện>

<khối lệnh>

Loop

Kiểu 4: Lặp trong khi điều kiện là False và có ít nhất một lần thi hành khối lệnh

Do

<khối lệnh>

Loop Until

5.6.2.2 For...Next

Biết trước số lần lặp. Ta dùng biến đếm tăng dần hoặc giảm dần trong vòng lặp.

For <biến đếm> = <điểm đầu> To <điểm cuối> [Step <bước nhảy>]

  <Khối lệnh>

Next [<Biến đếm>]

Biến đếm, điểm đầu, điểm cuối và bước nhảy là những giá trị số.

Bước nhảy có thể là âm hoặc dương. Nếu bước nhảy là dương, điểm đầu phải nhỏ hơn hoặc bằng điểm cuối, nếu bước nhảy là âm thì ngược lại.

5.6.2.3 For Each...Next

Tương tự vòng lặp For... Next, nhưng nó lặp khối lệnh theo số phần tử của một tập các đối tượng hay một mảng thay vì theo số lần lặp xác định. Vòng lặp này tiện lợi khi ta không biết chính xác bao nhiêu phần tử trong tập hợp.

For Each <phần tử> In <Nhóm>

  <khối lệnh>

Next <phần tử>

5.6.2.4 Vòng lặp While...Wend

Tương tự vòng lặp Do...While, nhưng ta khôing thể thoát vòng lặp bằng lệnh Exit. Vì vậy, vòng lặp kiểu này chỉ thoát khi biểu thức điều kiện sai.

While <điều kiện>
    <khối lệnh>
Wend

5.6.2.5 Câu lệnh GoTo

Được dùng cho bẫy lỗi.
On Error Goto ErrorHandler
Khi có lỗi, chương trình sẽ nhảy đến nhãn ErrorHandler và thi hành lệnh ở đó

5.6.3 Làm việc với cấu trúc

Ta có thể lồng các cấu trúc với nhau ví dụ có thể lồng nhiều vòng For Nẽex với nhau. Để thoát khỏi cấu trúc ta dùng lệnh Exit, Exit for cho phép thoát khỏi vòng For, Exit do cho phép thoát khỏi Do loop.

5.7 Gỡ rối chương trình

Không một chương tình nào là không có lỗi. Tuy nhiên, giảm khả năng lỗi đến mức tối thiểu là có thể làm được. Để có chương trình tốt, ta cần có thiết kế chặt chẽ. Sau đó, chương trình phải được viết sao cho có ít sinh lỗi và nếu có thì dễ tìm.

5.7.1 Một số giải pháp giảm lỗi

Thiết kế cẩn thận, ghi ra các vấn đề quan trọng và cách giải quyết cho từng phần. Ghi ra từng thủ tục và mục đích của nó.

Chú thích rõ ràng trong chương trình

Đối tượng có tham chiếu tường minh thay vì kiểu chung chung như Object, Control

Tuân thủ Coding convention

Một tổng những nguyên nhân gây lỗi là gõ sai tên biến hoặc nhầm lẫn điều khiển. Dùng Option Explicit để tránh trường hợp này.

Truyền giá trị khi gọi thủ tục

Là một trong những cách giảm thiểu khả năng lỗi. Tuy nhiên, trở ngại duy nhất của nó là tiêu tốn nhiêuvùng nhớ hơn truyền địa chỉ, làm chương tình chạy chậm hơn.

Đối tượng Err

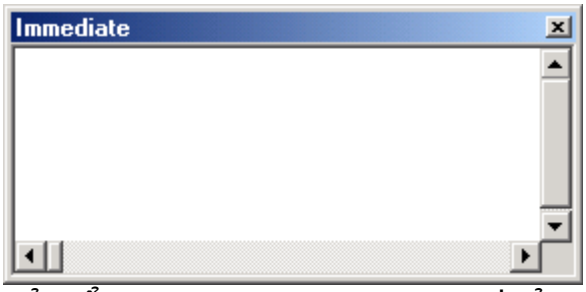
Là đối tượng do Visual basic cung cấp sẵn. Nó có vô số thuộc tính. Sau đây là những thuộc tính thông dụng:

Thuộc tính	Giải thích
Number	Giá trị mặc định, số hiệu lỗi
Description	Mô tả lỗi
Source	Tên đối tượng gây ra lỗi

5.7.2 Gỡ rối

Có thể tạm dừng chương trình bằng cách chọn Break từ menu Run hoặc nhấn trên thanh công cụ, hoặc nhấn trên tổ hợp phím Ctrl-Break. Ta cũng có thể đặt dòng lệnh Stop trong chương trình nhưng sẽ có cách khác tốt hơn.

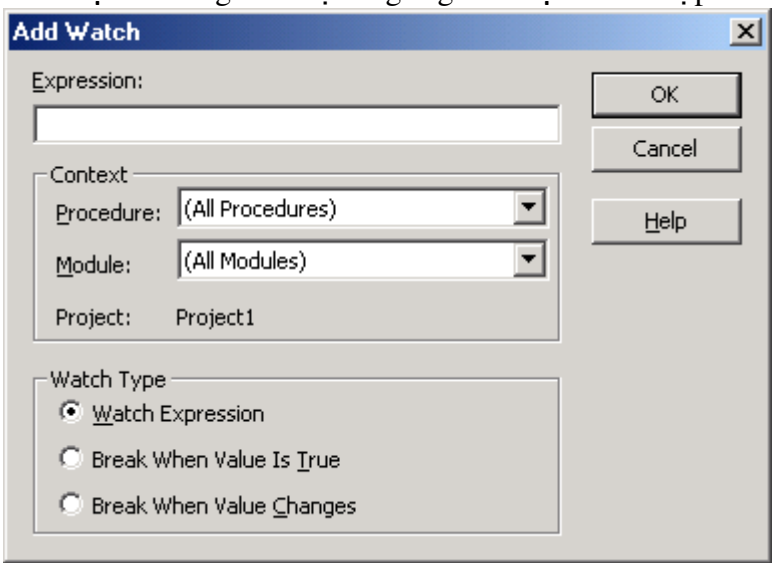
Cửa sổ Immediate



Cửa sổ này cho phép ta xem các giá trị của các biến trong form khi ta chạy gỡ rối

Cửa sổ Watch

Hiển thị các giá trị của một biến, thuộc tính hay biểu thức bất kỳ. Thậm chí có thể buộc chương trình tạm ngưng sau một số lần lặp.



Đi qua từng dòng chương trình

Thanh Debug

Thứ tự nút bấm từ trái sang phải như sau:

- Start: thi hành chương trình
- Break: tạm dừng chương trình
- End: Kết thúc chương trình
- BreakPoint: Điểm đánh dấu dòng lệnh để tạm dừng chương trình. Nút này được sử dụng để bật tắt chế độ breakpoint. Khi có lỗi xảy ra và ta chưa khoanh được khu vực nghi ngờ, thì Breakpoint là giải pháp tốt nhất để cô lập vùng chương trình bị lỗi.
- Step Into: Nếu dòng lệnh hiện hành đang gọi một thủ tục, nhấn F8 sẽ nhảy vào bên trong thủ tục.

- Step Over: Nếu dòng lệnh hiện hành đang gọi một thủ tục, nhấn Shipt-F8 sẽ chạy qua thủ tục.
- Step Out: Nếu điểm dừng đang ở trong một thủ tục, nhấn Ctrl-Shift-F8 sẽ chạy hết thủ tục và dừng ở dòng kế tiếp sau lệnh gọi thủ tục
- .....

5.8 Bẫy lỗi

5.8.1 Lệnh On Error

Lệnh On Error dùng trong hàm hay thủ tục báo cho Visual basic biết cách xử ký khi lỗi xảy ra.

On Error GoTo <Nhãn>

Dùng On error Goto 0 tắt xử lý lỗi.

5.8.2 Kết thúc bẫy lỗi

6 Lập trình xử lý giao diện

6.1 Menu

Có 2 loại menu:

- Menu thả xuống(Drop-down menu): là dạng menu thông dụng nhất.
- Menu bật ra (Popup menu): Thường hiển thị khi ta nhấn nút phải chuột

Menu cũng là một loại điều khiển, nhưng windows sẽ kiểm soát việc vẽ menu. Lập trình viên chỉ quản lý phần xử lý các sự kiện mà thôi.

6.1.1 Dùng trình soạn thảo menu để tạo menu

Menu không chứa trong hộp công cụ như những điều khiển khác, mà được thiết kế bằng trình soạn thảo menu. Từ menu tools, chọn Menu editor để mở rộng chương trình này hoặc dùng Ctrl + E hoặc nhấn biểu tượng  trong menu của Visual basic

Chú ý nếu chưa có biểu mẫu thì biểu tượng này không xuất hiện trên thanh công cụ.

6.1.1.1 Các thuộc tính của menu

Thuộc tính của menu không chứa trong cửa sổ thuộc tính như các điều khiển khác mà đặt trong trình soạn thảo menu.

Thuộc tính caption: Là chuỗi ký tự hiển thị trên menu.

Thuộc tính name: Phải được đặt duy nhất và dễ nhớ. Có 2 cách đặt tên:

- Nhóm các mục có cùng cha trên menu vào chung một dãy các điều khiển và dùng chung một tên. Cách này được Visual basic hết sức khuyến khích.
- Mỗi mục có một tên riêng, nhưng nên bắt đầu bằng mnu. ví dụ mnuFile

Thuộc tính index: Dùng với dãy các điều khiển menu. Trong đó, vì có nhiều mục cùng tên nên index được dùng cho phân biệt giữa chúng với nhau.

Thuộc tính shortcut: Người sử dụng có thể nhấn chuột để chọn menu theo cách bình thường, hoặc dùng phím tắt. VD: nhấn Ctrl+C thay vì chọn Copy.

Thuộc tính Windows list: dùng trong các ứng dụng MDI. Đây là những ứng dụng có một biểu mẫu chính và nhiều biểu mẫu con. Thuộc tính windowsList ra lệnh cho Visual basic hiển thị tiêu đề của các cửa sổ con trên menu.

Thuộc tính Checked: Nếu chọn thuộc tính này, trên menu sẽ hiển thị một dấu bên cạnh. Tuy nhiên, thuộc tính này không được gán cho những mục menu đang chứa menu con.

Thuộc tính enable: Nếu thuộc tính này không được chọn người sử dụng không thể chọn và đó được.

Thuộc tính Visible: Nếu thuộc tính này không được chọn mục này sẽ biến mất khỏi màn hình.

Thuộc tính NegotiatePosition: Quản lý vị trí gắn menu trong trường hợp sử dụng các đối tượng ActivateX.

Tách nhóm menu: Nếu menu có khá nhiều mục, tốt nhất ta nên chia chúng thành nhiều nhóm nhỏ.

6.1.2 Viết chương trình điều khiển menu

Để lập trình cho menu, ta mở cửa sổ thiết kế biểu mẫu và nhấn chuột lên mục mà ta muốn xử lý.

6.1.2.1 Pop-up menu

Ví dụ mẫu - tạo pop-up menu

Bạn tạo ra một menu file có Open, Save, Save as...

Mở cửa sổ code, trong sự kiện mouseUp của biểu mẫu, đưa vào dòng lệnh

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
   If Button = vbRightButton Then
      PopupMenu mnuFile, vbPopupMenuLeftAlign
   End If
End Sub
```

Tắt thuộc tính Visible của menu file.

Chạy chương trình, khi ta nhấn chuột một menu pop-up sẽ hiện thị.

6.1.2.2 Menu động

Mảng điều khiển

Là một mảng với các phần tử là những điều khiển. Lần đầu bạn gặp mảng điều khiển là khi bạn copy điều khiển trên biểu mẫu. Visual basic sẽ hỏi:

Nếu ta dán một điều khiển trùng tên với một điều khiển khác, lable1 chẳng hạn, đã có sẵn, Visual basic cho rằng ta muốn tạo mảng điều khiển. Nếu nhấn Yes, VB sẽ đặt tên cho mảng lable1, và thuộc tính index lần lượt là 0 và 1. Nếu thêm một điều khiển nữa, Visual basic tự đông tăng index. Nó không nhắc lại vì ta đã có mảng điều khiển. Mặc dù trùng tên nhưng ta có thể xác định các phần tử một cách dễ dàng nhờ thuộc tính index mà Visual basic tạo cho nó.

6.2 Hộp thoại

Hộp thoại(dialog) là một trong những cách thức để windows giao tiếp với người sử dụng, có 4 kiểu hộp thoại:

- Thông điệp
- Hộp nhập
- Các hộp thoại thông dụng
- Hộp thoại hiệu chỉnh.

6.2.1 Thông điệp(Message box)

Là dạng hộp thoại đơn giản nhất, gồm hai loại:

- Chỉ cung cấp thông tin
- Tương tác với người sử dụng.

6.2.1.1 Chiều dài thông điệp

Windows tự động cắt chuỗi khi nó quá dài, nhưng đôi khi việc này không đúng ý muốn của lập trình viên. Ta có thể làm bằng tay:

MsgBox "This is a multi-line" & chr$(10) & "message."

Hàm chr$() trả về ký tự có mã ASCII là tham số truyền vào chr$(10) trả về dòng ký tự mới.

6.2.1.2 Các loại thông điệp

Hằng số	**Thể hiện**
vbOKOnly	**OK**
vbOKCancel	**OK, Cancel**
vbAbortRetryIgnore	**Abort, Retry, Ignore**
vbYesNoCancel	**Yes, No, Cancel**
vbYesNo	**Yes, No**
vbRetryCancel	**Retry, Cancel**

6.2.2 Hộp nhập(Input box)

Input box ít được dùng. Lý do là:

- Không có cách nào để kiểm định dữ liệu mà người sử dụng đưa vào khi họ chưa nhấn Enter. Nếu dùng biểu mẫu do chính mình thiết kế, ta có thể đưa vào hộp văn bản và viết chương trình để xử lý sự kiện liên quan đến việc kiểm tra dữ liệu mà với Input box không thể làm được.
- InputBox chỉ cho người sử dụng nhập rất ít thông tin. Muốn lấy ra được nhiều thông tin, nên dùng biểu mẫu tự thiết kế.

6.2.3 Các hộp thoại thông dụng(Common dialog)

Bởi vì các hộp thoại này xuất hiện ở mọi nơi, nên thay vì phải viết chương trình nhiều lần, Windows chứa chúng trong một DLL, Comdlg32.Dll hay Comdlg32.ocx

Có 6 hộp thoại:

- Mở tập tin
- Lưu tập tin
- Chọn màu
- Chọn fonts
- In ấn
- Trợ giúp

Tuy nhiên khi thiết kế mẫu ta chỉ thấy duy nhất một điều khiển là CommonDialog.

Tên	**Phương thức**
Open file	Showopen
Save file	Showsave
Color	Showcolor
Font	ShowFont
Print	ShowPrint

Help	ShowHelp

6.2.4 Hộp thoại hiệu chỉnh

Ưu điểm của hộp thoại này là ta có thể thiết kế theo ý thích. Trở ngại của nó khi thi hành là từng biểu mẫu sử dụng tài nguyên hệ thống như bộ nhớ, thời gian CPU. Nếu dùng nhiều hộp thoại hiệu chỉnh trong ứng dụng có thể mất dần tài nguyên và khả năng chống lại treo máy là rất yếu.

6.3 Thanh công cụ(ToolBar)

6.3.1 Trong ứng dụng đơn giản

Là tính năng chuẩn của các ứng dụng chạy trên windows. Nó cho phép truy cập nhanh đến các chức năng của menu. Tạo toolbar được hỗ trợ cho phiên bản profestional và Enterprice.

6.3.2 Nhúng đối tượng

Khi nhúng một đối tượng vào nơi chứa và đối tượng đang được chọn, menu và toolbar của đối tượng sẽ được hiển thị trong nơi chứa.

Ta có thể cho phép hiển thị menu hay toolbar của đối tượng nhúng bằng thuộc tính NegotiateMenus của biểu mẫu. Khi thuộc tính này là True (mặc định), và nơi chứa có dùng menu hoặc thuộc tính này là False, menu của đối tượng không thể "trộn" được.

Lưu ý thuộc tính này không có trong biểu mẫu MDI.

Thuộc tính NegotiateToolbars của biểu mẫu MDI quyết định toolbar của đối tượng nhúng có được đặt trong biểu mẫu chứa hay không. Nhưng nó đòi hỏi nơi chứa phải có toolbar. Nếu thuộc tính này là True toolbar của đối tượng nhúng được hiển thị trong nơi chứa. Nếu là False Toolbar được hiển thị "di động" trong nơi chứa.

Lưu ý rằng thuộc tính này chỉ áp dụng cho biểu mẫu MDI.

Nếu biểu mẫu MDI có toolbar, nghĩa là nó có một hộp hình. Thuộc tính Negotiate của hộp hình quyết định toolbar của nơi chứa vẫn hiển thị hoặc sẽ bị thay thế bằng toolbar của đối tượng nhúng khi nó được chọn. Nếu Negotiate là True, toolbar nhngs trộn vào toolbar chính. Ngược lại toolbar nhúng thay thế toolbar chính.

6.4 Thanh trạng thái

Điều khiển thanh trạng thái (statusBar) cung cấp một cửa sổ, thường ở phần cuối cùng của cửa sổ chính, trên đó, ứng dụng có thể hiển thị các trạng thái dữ liệu khác nhau. StatusBar có thể được chia tối đa thành 16 panel dùng để chứa hình ảnh hay văn bản. Thuộc tính kiểm soát cách thể hiện của từng panel bao gồm Width, Alignment (của văn bản và hình ảnh) và Bevel. Ngoài ra ta có thể dùng một trong 7 giá trị của Style để tự động hiển thị dữ liệu thông dụng như ngày, giờvà trạng thái bàn phím.

Vào lúc thiết kế, ta có thể tạo các bảng báo và hiệu chỉnh cách thể hiện của chúng bằng cách đổi các giá trị trong tab panel của hộp thoại Property page. Hộp thoại này được mở thông qua cửa sổ thuộc tính của điều khiển Statusbar.

Vào lúc thi hành, các đối tượng Panel có thể được cấu hình lại để phản ánh các chức năng khác nhau, tuỳ theo trạng thái của ứng dụng.

Thanh công cụ và thanh trạng thái cung cấp những công cụ giúp tạo ra một giao diện tiết kiệm mà đầy đủ thông tin.

6.5 Xử lý chuột và bàn phím

6.5.1 sự kiện chuột

Sự kiện	Giải thích
MouseDown	Xảy ra khi người sử dụng nhấn một nút chuột bất kỳ.
MouseUp	Xảy ra khi người sử dụng thả một nút chuột bất kỳ.
MouseMove	Xảy ra khi con trỏ chuột di chuyển đến một điểm mới trên mà hình

Biểu mẫu hoặc điều khiển có thể bắt sự kiện chuột khi con trỏ chuột đi qua.

Tham số truyền

Tham số	Giải thích
Button	Cho biết nút chuột nào được nhấn
Shift	Cho biết phím SHIFT, CTRL hay ALT được nhấn
x.y	Vị trí con trỏ chuột, với hệ toạ độ của đối tượng bắt sự kiện.

6.5.2 Hiệu chỉnh con trỏ chuột

Ta có thể dùng thuộc tính MousePointer và MouseIcon để hiện thị một biểu tượng con trỏ màn hình hay con trỏ chuột hiệu chỉnh.

Thuộc tính MousePointer cho phép chọn một trong 16 kiểu con trỏ. Sau đây là một vài con trỏ thường dùng.

Con trỏ chuột	Hằng	Mô tả
	vbHourglass	Thể hiện một hoạt động đang tiền hành và yêu cầu người sử dụng chờ.
	vbSizePointer	Thông báo chức năng thay đổi, ví dụ nó cho người sử dụng biết rằng có thể hiệu chỉnh cửa sổ.
	vbNoDrop	Cảnh báo với NSD rằng hành động này không thể thi hành được.

Giá trị mặc định của thuộc tính MousePointer là 0-Default và hiển thị theo kiểu Windows quy định.

6.5.3 Sự kiện bàn phím

Sự kiện chuột và bàn phím có vai trò chủ yếu trong hoạt động tương tác giữa người sử dụng và chương trình.

Mặc dù hệ điều hành cung cấp một số chức năng cơ bản cho bàn phím nhưng ta có thể khai thác và phát triển các thế mạnh của chúng.

Ta có thể kiểm soát phím nhấn theo 2 mức: điều khiển hoặc biểu mẫu. Mức điều khiển cho phép lập trình với điều khiển, mức biểu mẫu cho phép ta lập trình với ứng dụng.

Sự kiện bàn phím	**Xảy ra**
KeyPress	Khi một phím có mã ASCII được nhấn
KeyDown	Khi một phím bất kỳ được nhấn
KeyUp	Khi một phím bất kỳ được thả.

Chỉ có sự kiện đang forcus mới bắt sự kiện bàn phím. Đối với biểu mẫu, nó chỉ bắt được khi nó được kích hoạt và mọi điều khiển trên biểu mẫu đều không có forcus. Điều này chỉ xảy ra với biểu mẫu trống hoặc biểu mẫu có điều khiển bị cấm. Tuy nhiên nếu quy định thuộc tính KeyPreview của biểu mẫu thành True, biểu mẫu sẽ nhận mọi sự kiện bàn phím của mọi điều khiển trên nó trước khi các điều khiển này nhận được. Cách này hữu dụng khi ta muốn thi hành cùng một hoạt động khi một phím bất kỳ được nhấn, bất kể điều khiển nào đang forcus.

KeyDown và KeyUp có thể phát hiện những tình huống mà KeyPress không thể phát hiện:

- Tổ hợp phím SHIFT, CTRL và ALT
- Phím định hướng (← → ↑ ↓) lưu ý rằng một số điều khiển (nút lệnh, nút tuỳ chọn, và hộp đánh dấu) không bắt sự kiện phím định hướng. Thay vào đó, các phím này gây ra sự dịch chuyển của một điều khiển khác.
- PAGEUP và PAGEDOWN
- Phân biệt được phím số ở bàn phím phải với phím số ở bàn phím trái
- Đáp ứng khi thả phím
- Phím chức năng không trùng với menu.

Sự kiện bàn phím không loại trừ nhau. Khi người sử dụng nhấn một phím, cả KeyDown và KeyPress cùng phát. Khi người sử dụng nhấn một trong những phím mà KeyPress không phát hiện được, chỉ có keydown và xảy ra, đó là KeyUp.

Trước khi dùng KeyUp, KeyDown phải đảm bảo rằng KeyPress không làm được. Sự kiện này phát hiện các phím có mã ASCII chuẩn: Chữ cái, chữ số, dấu ngắt câu, Enter, TAB và BACKSPAGE. Nói chung, viết chương trình cho sự kiện KeyPress thì dễ hơn.

7 Xử lý tập tin

7.1 Mô hình FSO(File System Object model)

Cung cấp cho ứng dụng khả năng tạo, thay đổi, di chuyển, xoá các thư mục, dò tìm xem chúng có tồn tại hay không, nếu có thì ở đâu. Nó cũng cho phép lấy các thông tin về thư mục như tên, ngày tạo, ngày sửa đổi gần nhất...

Mô hình FSO chưa trong thư viện kịch bản (Scripting type library- csrrun.dll), hỗ trợ tạo và thao tác với tập tin văn bản thông qua đối tượng TextStream. Nó chưa hỗ trợ tập tin nhị phân, ta phải dùng lệnh Open với cờ Binary.

Mô hình này chứa các đối tượng sau:

Đối tượng	Giải thích
Drive	Cho phép thu thập thông tin về ổ đĩa như dung lượng, tên chia sẻ... Lưu ý rằng không nhất thiết là ổ cứng. Nó có thể là CD-ROM, là ổ đĩa RAM, hoặc là ổ mạng.
Folder	Cho phép tạo, xóa, di chuyển thư mục hay thu thập các thông tin hệ thống như tên thư mục, đường dẫn...
Files	Cho phép tạo, xoá, di chuyển tập tin hay thu thập các thông tin hệ thống....
FileSystemObject	Các thuộc tính và phương thức cho phép tạo xóa, thu thập thông tin về ổ đĩa, thư mục, tập tin.
TextStream	Cho phép đọc và ghi văn bản

Nếu chưa có sẵn tham chiếu đến FSO bạn có thể reference đến nó từ menu Project\reference...

7.2 Xử lý các tập tin với các dòng lệnh và hàm I/O cổ điển

Mặc dù có FSO, các lệnh cổ điển vẫn được hỗ trợ đầy đủ trong Visual basic. Nếu ta thiết kế ứng dụng để làm việc với cơ sở dữ liệu, ta sẽ không cần cung cấp các truy cập đến tập tin. Điều khiển dữ liệu và ràng buộc dữ liệu cho phép đọc và ghi dữ liệu trực tiếp trên cơ sở dữ liệu, dễ dàng hơn truy cập đến tập tin.

7.2.1 Các kiểu truy cập tập tin

Tập tin là một dãy các byte lưu trên đĩa. Khi ứng dụng truy cập đến tập tin, nó xem các byte như là ký tự, bản ghi,số, chuỗi...Có 3 kiểu truy cập tập tin:

- **Tuần tự:** đọc và ghi các tập tin văn bản theo cá chuỗi liên tục.
- **Ngẫu nhiên:** đọc và ghi tập tin văn bản hoặc nhị phân có cấu trúc theo các bản ghi có chiều dài xác định
- **Nhị phân:** đọc và ghi các tập tin có cấu trúc không xác định.

Hàm truy cập tập tin

Hàm/ dòng lệnh	**Tuần tự**	**Ngẫu nhiên**	**Nhị phân**
Close	X	X	X
Get		X	X
Input()	X		X
Input #	X		

Line Input #	X		
Open	X	X	X
Print #	X		
Put		X	X
Type.. End type		X	
Write #	X		

7.2.1.1 Tập tin tuần tự

7.2.1.1.1 Mở tập tin tuần tự

Open *pathname* **For | Input | Output | Append | As** *filenumber* **| Len =** *buffersize*|

Khi mở một tập tin tuần tự với **Input,** tập tin phải có sẵn. Nếu không Visual basic sẽ báo lỗi. Khi ta mở một tập tin chưa có sẵn với **Output** hoặc **Append** lệnh Open sẽ tạo mới tập tin và mở nó.

Tham số tuỳ chọn Len chỉ ra số ký tự trong vùng đệm khi sao chép dữ liệu giữa tập tin và chương trình.

Sau khi mở tập tin với Input, Output hoặc Append ta phải đóng nó bằng lệnh Close.

7.2.1.1.2 Soạn thảo tập tin

Trước hết, đọc nội dung tập tin vào biến, sửa biến và ghi biến vào tập tin.

Đọc chuỗi từ tập tin

Mở tập tin với Input. Sau đó, dùng dòng lệnh line Input #, Input() hoặc Input # để sao chép nội dung tập tin vào biến. Visual basic cung cấp những hàm để đọc ghi 1 ký tự hoặc một dòng vào tập tin. Đoạn code sau đọc từng dòng:

```
Dim LinesFromFile, NextLine As String
Do Until EOF(FileNum)
    Line Input #FileNum, NextLine
    LinesFromFile = LinesFromFile + NextLine + Chr(13) + Chr(10)
Loop
```

Mặc dù Line Input# nhận ra cuối chuỗi khi nó đọc đến ký tự xuống dòng, nhưng nó không lấy ký tự này khi đưa và biến. Nếu muốn lấy ký tự xuống dòng ta phải tự xử lý code.

Ta cũng có thể dùng Input # để đọc một danh sách gồm các số hay chuỗi trong tập tin. Ví dụ, để đọc một dòng trong tập tin danh sách thư điện tử:

```
Input #filename, name, street, city,state,zip
```

Dùng Input để sao chép một số ký tự từ tập tin vào biến:

```
LinesFromFile = Input(n, Filename)
```

Để copy toàn bộ tập tin vào biến, dùng InputB để copy từng byte. Vì InputB trả về chuỗi ANSI, ta phải dùng StrConv để chuyển đổi từ ANSI sang UNICODE.

```
LinesFromFile=  StrConv(InputB(LOF(FileNum), FileNum), vbUnicode)
```

Ghi chuỗi và tập tin

Để ghi nội dung biến vào tập tin, mở tập tin với Output hay Append, sau đó dùng Print #. Ví dụ, tình soạn thảo văn bản copy nội dung của hộp văn bản vào tập tin:

Print #FileNum, Thebox.text

Visual basic cũng hỗ trợ lệnh Write #, dùng để ghi một danh sách các số, chuỗi vào tập tin. Dấu phẩy được dùng để tách biệt:

Dim strString as String, intNumber as Integer

strString = "AnyCharacters"

intNumber = 23456

Write #Filenum, strString, intNumber

Đoạn code này ghi một chuỗi và một số vào một tập tin được chỉ ra bởi FileNum.

7.2.1.2 Tập tin tuần tự

Mô hình FSO không hỗ trợ tập tin ngẫu nhiên. Các byte trong tập tin truy cập ngẫu nhiên có dạng bản ghi đồng nhất, mỗi bản ghi chứa một hoặc nhiều trường. Mỗi trường có kiểu và kích thước xác định. Mỗi bản ghi có nhiều trường tương đương kiểu người dùng định nghĩa. Ví dụ: Kiểu worker tạo mẫu tin 19 bytes gồm 3 trường:

```
Type worker
   LastName As String *10
   Tilte As String * 7
   Rank As String *2
End Type
```

Sau khi định nghĩa kiểu mẫu tin, khai báo các biến mà ứng dụng cần để xử lý 1 tập tin mở để truy cập ngẫu nhiên:

Public Employee as worker

7.2.1.2.1 Mở tập tin ngẫu nhiên

Open *pathname* [**For Random] As** *filenumber* **Len =** *reclength*

Vì Random là kiểu mặc định từ khoá **For random** là tuỳ chọn.

Biểu thức **Len =** *reclength* chỉ ra kích thước tính bằng byte của một bản ghi. Lưu ý rằng mỗi biến chuỗi trong Visual basic chứa một chuỗi Unicode và ta phải chỉ ra chiều dài tính bằng byte của chuỗi đó. Nếu *reclength* nhỏ hơn chiều dài thực sự của bản ghi ghi đến tập tin, lỗi sé sinh ra. Nếu *reclength* lớn hơn chiều dài thực sự của bản ghi, bản ghi sẽ được ghi, mặc dù một số khoảng trống đĩa bị lãng phí.

7.2.1.3 Tập tin truy cập nhị phân

Trong mô hình FSO không hỗ trợ tập tin này. Dùng tập tin nhị phân khi yêu cầu kích thước tập tin nhỏ. Khi thao tác nên dùng biến byte thay vì String, vì dữ liệu nhị phân không thể chứa đúng vào biến String.

7.2.1.3.1 Mở tập tin nhị phân

Cú pháp lệnh:

Open *pathname* **For Binary As** *filenumber*

7.2.1.3.2 Chứa thông tin trong các trường hợp có độ dài thay đổi

Ta có thể giảm thiểu việc sử dụng khoảng trống đĩa bằng tập tin nhị phân.

```
Type Person
   ID As Integer
   MonthlySalary As Currency
   LastReviewDate As Long
   FirstName As String
   LastName As String
   Tilte As String
End Type
Public Empl As Person
```

Mỗi bản ghi chỉ chứa một số byte cần thiết do các trường có chiều dài biến đổi. Trở ngại là ta không thể truy cập bản ghi một cách ngẫu nhiên, mà ta phải truy cập tuần tự theo độ dài của mỗi bản ghi. ta có thể tìm một byte trong tập tin, ngưng không thể biết byte đó thuộc bản ghi nào.

7.3 Các điều khiển trên hệ thống tập tin

- Hộp danh sách ổ đĩa
- Hộp danh sách thư mục
- Hộp danh sách tập tin

Điều khiển hệ thống tập tin lấy thông tin tự động từ hệ điều hành, ta có thể truy cập hoặc quyết định nên hiển thị những phần nào thông qua các thuộc tính của chúng.

7.3.1 Hộp danh sách ổ đĩa

Là hộp danh sách kiểu drop down ổ đĩa. Mặc định là ổ đĩa hiện hành được hiển thị trên hệ thống của người sử dụng. khi điều khiển có forcus, NSD có thể gx vào một tên ổ đĩa hợp lệ hoặc nhấn chuột để chọn một ổ đĩa trong danh sách các ổ đĩa. Nếu Người sử dụng chọn một ổ đĩa mới tên mới sẽ hiện thị nên đầu danh sách.

Ta có thể dùng code để quy định ổ đĩa được chọn.

Drive1.Drive = “C:\”

Chọn một ổ đĩa không làm thay đổi ổ điãc hiện hành, trừ phi ta lấy thuộc tính Drive gán cho hàm ChDrive

ChDrive = Drive1.Drive

7.3.2 Hộp danh sách thư mục

Hộp danh sách thư mục hiển thị cấu trúc thư mục của ổ đĩa hiện hành trên hệ thống, bắt đầu bằng thư mục gốc. Cấu trúc thể hiện hình cây phân nhánh. Thư mục hiện hành trở thành đậm và thụt vào so với thư mục cha và cứ thế thụt vào cho đến thư mục gốc. Khi người sử dụng di chuyển chuột lên xuống, mỗi thư mục trở thành đậm.

7.3.2.1 Xác định từng thư mục

Mỗi thư mục trong hộp có một số hiệu kèm theo. Khả năng này không được hỗ trợ trong hộp thoại thông dụng. Thư mục chỉ ra bởi thuộc tính Path (Dir1.Path) luôn có listIndex là -1. Thư mục ngay sát bên trên nó có listIndex là -2, kế tiếp là -3....đến gốc. Thư mục con đầu tiên của Dir1.Path có listIndex là 0. Nếu có nhiều thư mục con ở mức này thì kế tiếp nó có ListIndex là 1, tiếp theo là 2...

7.3.2.2 Định thư mục hiện hành

Dùng thuộc tính Path của hộp danh sách thư mục để xem hoặc quy đinh thư mục hiện hành (ListIndex =-1). Ví dụ gán "C:\Luong" vào Drive1.Path thì Luong là thư mục hiện hành.

Tương tự, ta có thể gán thuộc tính Path của hộp danh sách thư mục là thuộc tính Drive của hộp danh sách ổ đĩa.

Dir1.Path = Drive1.Drive

Khi đó, hộp danh sách thư mục hiện thị tất cả thư mục ở trên, và các thư mục ngay ngay mục hiện hành của ổ. Tuy nhiên nó không quy định thư mục hiện hành ở mức hệ điều hành, nó chỉ làm nổi bật và cho nó giá trị ListIndex = -1.

Để định thư mục hiện hành dùng lệnh ChDir.

ChDir Dir1.Path

Trong một ứng dụng dùng các điều khiển tập tin, ta có thể định thư mục hiện hành là thư mục chứa tập tin EXE thông qua đối tượng Application:

ChDrive App.Path

ChDir App.Path

Chú ý thuộc tính Path chỉ có thể được sử dụng trong lúc chạy chương tình không có sẵn vào lúc thiết kế.

7.3.2.3 Nhấn chuột vào một thành phần trong Hộp danh sách thư mục

Khi nhấn chuột vào một thành phần thì nó trở thành đậm. Khi người sử dụng nhấn đúp chuột lên đó, nó được gán xào thuộc tính Path, và ListIndex được đổi thành -1. Hộp danh sách thư mục được đổi lại để hiện thị các thư mục con.

7.3.2.4 Tìm vị trí tương đối của thư mục

Thuộc tính ListCount trả về số thư mục bên dưới thư mục hiện hành, không phải tổng số thành phần trong hộp danh sách thư mục.

7.3.3 Hộp danh sách tập tin

liệt kê các tập tin chứa trong thư mục chỉ ra bởi thuộc tính Path. Ta cũng có thể hiển thị các tập tin trong thư mục hiện hành trên ổ đĩa hiện hành:

File1.Path = Dir1.Path

Ta có thể hiển thị các tập tin thông qua điều kiện lọc dùng thuộc tính Pattern Ví dụ:

File1.Pattern = "*.frm; *.bas"

Visual basic hỗ trợ ký tự "?" VD: "???.txt" hiển thị tất cả các tậpt in có phần mở rộng là ".txt"

Làm việc ví thuộc tính của tập tin

Thuộc tính bao gồm: Archive, Normal, System, Hidden và ReadOnly. Ta có thể dùng thuộc tính để chỉ ra loại tập tin nào sẽ được hiển thị trong danh sách. Bởi mặc định, giá trị của các thuộc tính System và Hidden là False; Normal, Archive và ReadOnly là True.

Để hiển thị chỉ những tập tin ReadOnly chỉ cần đổi thuộc tính này là true và các thuộc tính khác là False

7.4 Điều khiển richtextbox

Cho phép Người sử dụng nhập văn bản và thực hiện một số thao tác định dạng giống như nạp, ghi dữ liệu trong một trình soạn thảo văn bản.

Ta cũng có thể đánh dấu đoạn văn bản, chuyển thành ký tự đậm, nghiêng...ta cũng có thể canh trái, canh phải các đoạn văn bản.

Điều khiển cho phép mở và ghi tập tin dước dạng rtf và ascci thông qua hàm loadfile và Savefile.

Để dùng điều khiển RichTextBox ta thêm tập tin Richtx32.ocx và đề án. khi chạy ứng dụng cài tập tin này trong thư mục system của windows.

7.4.1 Phương thức loadfile

Cú pháp: *Object.***LoadFile** *pathname, filetype*

Pathname là đường dẫn đến tập tin

fileType: Tuỳ chọn, là hằng số thể hiện kiểu tập tin:

Hằng	Giá trị	Giải thích
rtfRTF	0	(Mặc định) RTF, tập tin lưu phải là tập tin.rtf hợp lệ
rtfText	1	Văn bản, điều khiển đang lưu tệp văn bản.

Khi nạp một tập tin dùng LoadFile, toàn bộ nội dung của điều khiển được thay thế bằng nội dung tập tin. Điều này làm giá trị của thuộc tính Text và rftText thay đổi

Ta cũng có thể dùng hàm Input và thuộc tính TxtRTF và selRTF để đọc tập tin.rtf ví dụ, ta có thể đọc nội dung tập tin vào điều khiển như sau:

```
Open “Mytext.rtf” For Input As 1
RichTextBox2.TextRTF = strconv(InputB$(LOF(1),1), vbUniCode)
```

7.4.2 Phương thức savefile

Lưu nội dung của điều khiển RichTextbox vào tập tin

Cú pháp: *Object.***SaveFile(** *pathname, filetype)*

Pathname là đường dẫn đến tập tin

fileType: Tuỳ chọn, là hằng số thể hiện kiểu tập tin:

Hằng	Giá trị	Giải thích
rtfRTF	0	(Mặc định) RTF, tập tin lưu phải là tập tin.rtf hợp lệ
rtfText	1	Văn bản, điều khiển đang lưu tệp văn bản.

Ta cũng có thể dùng hàm Write và thuộc tính TextRTF và SelRTF để ghi vào tập tin.rtf. Ta cũng có thể lưu nội dung được đánh dấu trong điều khiển vào tập tin.rtf:

```
Open “Mytext.rtf” For Output As 1
```

```
Print #1,RichTextBox2.SelRTF
```

8 Sử dụng DLL và Windows API

8.1 DLL và cấu trúc của Windows

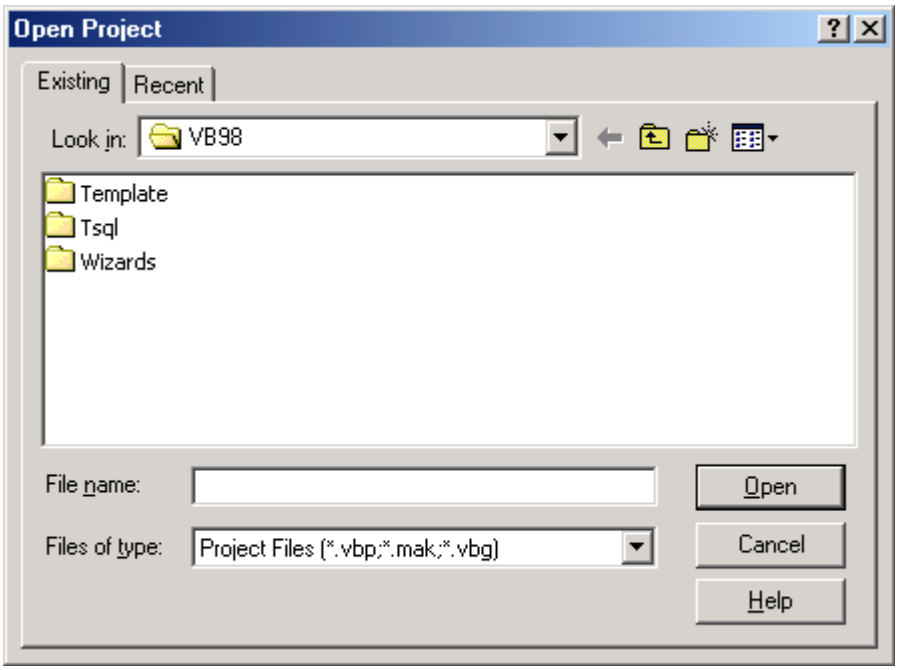
8.1.1 Các hộp thoại thông dụng

Dưới đây là một hộp thoại mở tập tin

Hộp thoại này không chỉ tiết kiệm thời gian cho chúng ta, mà còn tạo nên một giao diện người sử dụng gần gũi và thống nhất với môi trường Windows

Hộp thoại này do Windows cung cấp, nằm trong thư viện \Windows\System\comdlg32.Dll. Tập tin này chứa các đoạn chương trình tạo ra các hộp thoại thông dụng khác nhau. Do tập hợp lại trong một thư mục dùng chung là Windows\system, nó cho phép mọi ứng dụng Windows có quyền truy cập và thậm chí bản thân Windows cũng vậy.

Các tập tin DLL này được biên dịch với C/C++.



Hình 9.1 *Hộp thoại mở tập tin là một trong những hộp thoại thông dụng của Windows*

8.1.1 Thư viện liên kết động

Đối với các ngông ngữ lập trình cổ điển như C, khi biên dịch chương trình, ta có một tập tin EXE duy nhất có thể được thi hành mà không cần bất cứ tập tin nào khác (tập tin.vbp của Visual Basic không phải trường hợp này, vì nó chỉ chạy trong môi trường Visual Basic). Toàn bộ chương trình cần thiết được chứa hết trong một tập tin EXE. Tuy nhiên, cũng có rất nhiều thư viện C được dùng rộng rãi. Vấn đề là làm sao sử dụng lại các đoạn chương trình viết sẵn trong chương

trình của ta. Đó chính là ***liên kết*** (*link*)**.** Có hai loại liên kết : ***liên kết tĩnh*** (*static link*) và ***liên kết động*** (*dynalic link*).

Liên kết tĩnh

Cung cấp một kết nối nền vững giữa chương trình và module viết sẵn ngay lúc thiết kế; tương tự ta tạo module trong Visual Basic và gọi thủ tục trong đó, chỉ khác là liên kết tĩnh thì chứa bên ngoài Visual Basic. Tuy nhiên, để sử dụng liên kết tĩnh, ta c ần copy phần chương trình viết sẵn của thư viện vào tập tin chương trình khi biên dịch. Từ đó trở đi, chúng trở thành một phần của chương trình và bị khoá chặt với chương trình.

Liên kết động

Là giải pháp linh hoạt hơn liên kết tĩnh. Tập tin thư viện bên ngoài không bị ràng buộc với bên ngoài. Nó chứa ở một nơi sao cho tập tin EXE có thể tìm ra và gửi thông điệp cho nó. Khi thi hành, các thông điệp này là những cuộc gọi đến các hàm/thủ tục, yêu cầu phần chương trình nào đó của DLL được thi hành.

Các DLL của Visual Basic

Có lẽ Visual Basic là một minh hoạ cho việc sử dụng DLL. Vào thư mục *\Windows\system*, ta sẽ thấy một loạt các tập tin cấu tạo nên cơ chế thi hành của VB. Ví dụ, ***VB5DB.DLL*** chứa chương trình kiên kết với DAO (*Đối tượng truy cập dữ liệu*) lúc thi hành để ứng dụng có thể tìm kiếm các cơ sở dữ liệu cục bộ.

Khi ta xây dựng một ứng dụng cơ sở dữ liệu và biên dịch nó, tập tin EXE không biết gì về cơ sở dữ liệu. Thay vào đó, nó sử dụng một số đoạn chương trình của VB cho phép nạp thư viện ***VB5DB.DLL*** lúc thi hành và gọi các hàm trong đó.

Thế mạnh của DLL

Nhất quán : Người sử dụng ưa chuộng Windows vì nó không ít thì nhiều có một giao diện người sử dụng phổ biến cho mọi ứng dụng. Ví dụ các hộp thoại thông dụng, các menu, thanh công cụ của *Office97* … Nghĩa là có những đoạn chương trình chung để tạo ra chúng.

Dễ bảo trì : Những thay đổi hoặc bổ sung nếu có sẽ thể hiện trên mọi ứng dụng.

Tập tin EXE nhỏ hơn : Do một phần công việc chứa ở nơi khác, và không gắn kết "cứng nhắc" như liên kết tĩnh, kích cỡ tập tin EXE được giảm nhỏ. Chỉ có điều là DLL còn chứa nhiều phần khác, không chỉ là những gì chương trình của ta cần.

Cấu trúc Windows

DLL là nền tảng của thiết kế Windows. Windows thực chất là tập hợp các ***DLL*** để các ứng dụng khác nhau có thể dùng chung. Bên trong các ***DLL*** này là hàng trăm hàm/thủ tục. Ta gọi chúng là ***Windows API***.

8.2 WIN API

Giao diện lập trình ứng dụng (*Application Programmer's Interface*) là tập hợp các hàm/thủ tục có sẵn của Windows. Chúng gắn gũi với ngôn ngữ C/C++ hơn.

Visual Basic được thiết kế theo kiểu che bớt các công việc bên dưới hệ thống. Phần lớn các cuộc goi đến các hàm API được lồng trong các dạng lệnh Visual Basic, từ khoá, phương thức và thuộc tính. Chúng sẽ được thông dịch thành ***WinAPI*** bên trong của Visual Basic.

Tuy nhiên, vẫn có một số hàm API mà Visual Basic không có phần tương đương. Ví dụ, Visual Basic chuẩn không có cách nào cho người lập trình điều khiển hệ thống *multimedia* của Windows, nhưng với ***WinAPI***, ta có thể đạt được kết quả. Hiểu rõ ***WInAPI***, ta có thể khám phá những năng lực tiềm tàng của chúng.

Lớp bọc API và các điều khiển hiệu chỉnh

Điều khiển hiệu chỉnh (OCX hay ActiveX) bản thân chúng là những lớp bọc API, chúng chuyển giao các chức năng theo kiểu Visual Basic một cách thân thiện.

Điều khiển ***ActiveX*** và ***OLE Automation Servers*** đưa chương trình vào các đề án mà không cần phải có một DLL thực sự.

Ta cũng có thể gọi API trong các module lớp, nghĩa là đưa năng lực API vào đối tượng Visual Basic.

8.3 Sử dụng API

8.3.1 Tìm kiếm API

Ta có thể tìm các API thông qua tập tin ***Trợ giúp*** (*Help*) của Visual Basic, qua sách tra cứu

Trình duyệt API (Text API Viewer)

Được cung cấp sẵn khi cài Visual Basic. Khi ta cần tra cứu cú pháp chính xác của hàm API, ta dùng ***Text API Viewer***. Tuy nhiên, để có thông tin chi tiết hơn như hàm API làm gì, truyền tham số gì, trả về giá trị gì, ta cần có quyển sách tra cứu.

Ngoài ra, chương trình này còn cho phép copy nội dung API đến *clipboard* để dán vào chương trình.

8.3.2 Các DLL của Windows

Các API được tổ chức trong bốn DLL chính của Windows:

a. **KERNEL32**:

Là DLL chính, đảm nhiệm quản lý bộ nhớ, thực hiện chức năng đa nhiệm và những hàm ảnh hưởng trực tiếp đến hoạt động của Windows.

b. **USER32**:

Thư viện quản lý Windows. Thư viện này chứa các hàm xử lý menu, định giờ, truyền tin, tập tin và nhiều phần không được hiển thị khác của Windows.

c. **GDI32**:

Giao diện thiết bị đồ hoạ (Graphics Device Interface). Thư viện này cung cấp các hàm vẽ trên màn hình, cũng như kiểm tra phần biểu mẫu nào cần vẽ lại.

d. **WINNM**:

Cung cấp các hàm multimedia để xử lý âm thanh, nhạc, video thời gian thực, lấy mẫu, v.v… Nó là DLL 32 bit. (Thư viện 16 bit tên là ***MMSYSTEM***)

Ta có thể tìm các tập tin này trong thư mục *\Windows\system*. Ngoài ra, còn có các DLL nhỏ hơn, cũng được dùng phổ biến để cung cấp các dịch vụ đặc biệt cho ứng dụng.

Trên đây là các tên DLL 32 bit. Phiên bản VB4 là bản cuối cùng còn hỗ trợ 16 bit.

8.3.3 Gọi API

Gọi API không khác gì với gọi hàm/ thủ tục trong module của đề án. Ví dụ ta có thủ tục:

```
Public sub FindText(obiDataControl as Control, _
SFilename as String)
‘ Code to implement function here
End sub
```

Để gọi thủ tục ta dùng :

```
FindText datTitles, “Titles”
```

Chỉ có điều API là một thủ tục không chỉ nằm ngoài module mà còn nằm ngoài Visual Basic.

8.3.3.1 Khai báo một cuộc gọi API:

Trước khi dùng hàm của DLL, ta cần khai báo hàm đó. Visual Basic cần biết:

- Tên hàm / thủ tục.
- Tập tin DLL chứa nó.
- Tham số truyền.
- Kiểu dữ liệu truyền về nếu là hàm.

Khai báo API tương tự khai báo hằng/ thủ tục thông thường. Ta vẫn bắt đầu bằng từ khoá ***Sub/Function***, chỉ khác là trước đó phải có từ khoá ***Declare.***

- ***Ví dụ mẫu - Tạo cửa sổ nhấp nháy bằng cách gọi API***

1. Tạo đề án chuẩn mới
2. Vẽ điều khiển định giờ (timer) trên biểu mẫu và định thuộc tính Interval là 10. Nó sẽ gây ra một sự kiện timer mỗi 10 mi-li-giây.

 ***Hình 9.2** Biểu tượng điều khiển Timer trên hộp công cụ.*

3. Nhấn đúp lên cửa sổ này để mở ***Cửa sổ Code***

```
Private Sub Timer1_Timer()
      Dim nReturnValue As Integer
      nReturnValue = Flash(Form1.hWnd, True)
End Sub
```

4. Khai báo hàm ***Flash*** trong ***General Declarations***:

```
Private Declare Function Flash Lib "User32"  _
      Alias "FlashWindow"  _
      (ByVal hWnd As Long,  _
      ByVal bInvert As Long) As Long
```

5. Thi hành chương trình. Khi biểu mẫu xuất hiện, tiêu đề của nó nhấp nháy.

Mặc dù ta thấy chương trình này rất đơn giản, nhưng nếu viết bằng các hàm Visual Basic thông thường, nó rất phức tạp và tốn rất nhiều chương trình.

Từ khoá ***Declare*** báo VB biết đây là khai báo một hàm của DLL.

Sau ***Declare*** là từ khoá ***Sub*** hay ***Function***, cho biết đây là thủ tục hay hàm. Ta chỉ có một trong hai lựa chọn.

Từ khoá Lib cho biết tên DLL đang chứa hàm/ thủ tục đó. Ở đây là thư viện User32. Từ khoá Alias cho biết tên thực sự của thủ tục / hàm trong thư viện. Nó có thể khác với tên ta khai báo trước từ khoá Lib.

Cuối cùng là khai báo các tham số truyền, cùng với kiểu dữ liệu hàm trả về.

Ở đây tham số được truyền là :

```
(ByVal hWnd As Long, ByVal bInvert As Long) As Long
```

Tham số đầu, ***hWnd,*** là "handle", xác định cửa sổ cần nhấp nháy. Tham số thứ hai, ***bInvert*** là giá trị *Boolean*. Nếu ***bInvert*** được truyền vào có giá trị *True*, thanh tiêu đề sẽ nhấp nháy. Để trả về trạng thái đầu, ta phải gọi lại lần nữa, với ***bInvert*** mang giá trị *False*.

Với nhiều hàm API, tên Alias trùng với tên thực. Khi đó Visual Basic sẽ tự động loại bỏ phần Alias. Ví dụ:

```
Private Declare Function FlashWindow Lib "User32"  _
      Alias "FlashWindow"  _
      (ByVal hWnd As Long,  _
      ByVal bInvert As Long) As Long
```

Visual Basic sẽ đổi thành:

```
Private Declare Function FlashWindow Lib "User32"  _
      (ByVal hWnd As Long,  _
      ByVal bInvert As Long) As Long
```

Tuy nhiên một số có tên không hợp lệ đối với Visual Basic, như ***_lopen***, một số khác có nhiều phiên bản, ví dụ có ký tự **A** và **W** ở cuối tên. Nói chung, tốt nhất nên dùng tên thực của API. Một số lập trình viên dùng ***Alias*** để thay thế tên hàm, hoặc thậm chí khai báo hai tên cho hai phiên bản hàm để nhận các tham số truyền khác nhau.

```
nReturnValue = Flash(Form1.hWnd, True)
```

Sau khi khai báo ***hàm API,*** ta có thể gọi API như một hàm hoặc thủ tục Visual Basic thông thường. Gọi *Flash* là gọi đến API trong DLL, và ta lưu giá trị trả về trong biến ***nReturnValue***.

Đối với các hàm thông thường, ta có thể không cần sử dụng giá trị trả về của hàm. Tuy nhiên, ta vẫn cần chứa giá trị trả về vào một biến dù ta không có ý định sử dụng nó. Phần lớn API trả về mã lỗi kiểu số, và ta có thể dùng nó để kiểm tra mọi việc có hoạt động chính xác hay không.

Trong thực tế, bỏ qua giá trị trả về không chỉ là lười biếng mà còn thực sự nguy hiểm nếu ta đang gọi nhiều API.

Sử dụng API sai có thể dẫn đến treo Windows, nếu không nói là treo máy. Khi làm việc với các API phức tạp, như những hàm cần cấp phát nhiều vùng nhớ và tài nguyên hệ thống. Không nên bắt chiếc các lập trình viên cẩu thả bỏ qua các giá trị trả về. Vì hàm DLL nằm ngoài ứng dụng, chúng tự kiểm tra lỗi – ta chỉ biết có sai sót thông qua giá trị trả về.

8.3.3.2 Handle

Lấy biểu mẫu làm ví dụ. Windows dùng một cấu trúc để lưu giữ thông tin của biểu mẫu. Thông tin này đồng nhất với thông tin chứa trong cửa sổ ***Properties***. Windows chứa cấu trúc của từng cửa sổ trong một danh sách dài gồm các cấu trúc dữ liệu liên quan đến mọi cửa sổ của mọi chương trình đang chạy. Để xác định cấu trúc nào thuộc cửa sổ nào, nó dùng ***handle***. Nó không dùng tên biểu mẫu vì tên cũng là một thuộc tính của biểu mẫu. *Handle* chính là số ID của một đối tượng trong Windows.

Khi ta bắt đầu dùng API, nhất là những API có xử lý với biểu mẫu, ta sẽ thường xuyên làm việc với handle. Visual Basic chứa handle như một thuộc tính chỉ được đọc, có thể dùng làm tham số truyền cho những hàm của Windows khi cần.

Thuộc tính này gọi là ***hWnd*** (*handle đến một cửa sổ*), chỉ có thể truy cập lúc thi hành. Mặc dù nó không mang ý nghĩa trong chương trình, nhưng nó có thể được đọc, và truyền như một tham số đến API. Các API có liên quan hiển thị cửa sổ sẽ cần tham số ***hWnd*** để biết chính xác cửa sổ mà nó cần xử lý.

8.3.3.3 Khai báo tham số truyền

Điểm quan trọng trong khai báo tham số truyền cho API là từ khoá ***Byval***.

Với chương trình thông thường, nếu truyền giá trị cho hàm, Visual Basic biết rằng nó chỉ xử lý với bản sao của tham số.

```
Function Square(Byval Number as Double) as Double
```

Một cách khác để truyền tham số là truyền tham chiếu. tham số truyền là biến chứ không phải là bản sao của nó. Do đó nếu hàm thay đổi tham số, các thay đổi này sẽ ảnh hưởng lên biến truyền vào. Nếu không chỉ rõ ***Byval***, VB sẽ tự động xem đó là truyền tham chiếu.

Nếu là hàm hoặc thủ tục do ta viết, nếu có sai sót dothiếu ***Byval***, hậu quả không nghiêm trọng, Windows không bị treo.

Tuy nhiên, với các DLL, tình hình nguy hiểm hơn nhiều. Nếu ta quên ***Byval***, VB tự động truyền một con trỏ đến biến. Nó cho biết địa chỉ của biến trên vùng nhớ. Sau đó hàm này đến địa chỉ đó và lấy giá trị về.

Nếu một hàm của DLL chờ một kết quả trong khoảng từ 0 đến 3, và ta truyền một biến tham chiếu, giá trị thực sự truyền vào có thể là 1002342, là địa chỉ vùng nhớ của biến. Hàm này sẽ xử lý số 1002342 thay vì số thuộc khoảng (0-3), kết quả là hệ thống treo.

Không hề có thông báo lỗi ở đây; ta chỉ biết được API bị lỗi khi hệ thống rối loạn và treo cứng. Một trong những kinh nghiệm khi làm việc với API là lưu lại. Vì chúng ta đang mạo hiểm ra ngoài vùng an toàn của Visual Basic, khi bị lỗi, hệ thống treo và ta mất hết dữ liệu. Luôn luôn lưu đề án trước khi chạy đoạn chương trình goin API. Từ menu ***Tools***, chọn ***Options*** để mở hộp thoại ***Options***. Chọn tab ***Environment***, đánh dấu vào tuỳ chọn ***Save Changes***.

8.3.3.4 Sử dụng lớp với API

Sử dụng riêng lẽ từng hàm API sẽ gây khó khăn cho những người đọc chương trình nếu họ không phải là người lập trình ban đầu, nhất là đối với các ứng dụng lớn.

Giải pháp của Visual Basic 6 là chuyển các API thành các lớp (các điều khiển ActiveX). Từng API có thể xếp vào những nhóm tuỳ thuộc lĩnh vực nó xử lý. Các nhóm này có thể chuyển thành các lớp của Visual Basic. Ví dụ, tạo một lớp có các chức năng về multimedia của các API về lĩnh vực này.

8.4 Dùng API khai thác khả năng Multimedia

8.4.1 Lớp multimedia

Lớp này chứa một bộ các lệnh multimedia thông dụng. Khi một đối tượng được tạo từ lớp, nó mang những chức năng tương tự một điều khiển – có thể xem hay quy định thuộc tính, các phương thức. Nó che đi các lệnh gọi API.

Các phương thức mà lớp này hỗ trợ:

Phương thức	**Mô tả**
MmOpen	Mở tập tin (video, âm thanh, nhạc, v.v...) chuẩn bị *Play*
MmClose	Đóng tập tin đang mở, ngăn cấm hoạt động *Play*
MmPause	Dừng *Play* trên tập tin hiện hành
MmStop	Dừng hẳn *Play*
MmSeek	Tìm một vị trí trong tập tin

MmPlay	*Play* tập tin đang mở, phát ra âm thanh trong loa

Các phương thức này là những hàm riêng rẽ trong lớp MMedia.cls và cho phép sử dụng các API theo nhiều cách.

Sau đây là các thủ tục thuộc tính trong tập tin nguồn:

Thuộc tính	**Mô tả**
Filename	Tên của tập tin đang mở
Length	Chiều dài của tập tin đang mở
Position	Vị trí hiện hành trong tập tin – ta có thể kết hợp với thuộc tính *Length* để hiển thị trạng thái Play
Satus	Một từ cho biết trạng thái tập tin (*Play, dừng tạm, dừng hẳn, v.v...*)
Wait	Nếu là *True*, chương trình sẽ chờ đến khi *Play* xong mới làm tiếp. Nếu là *False*, nó thi hành theo kiểu *đa nhiệm*

- ***Ví dụ mẫu - Sử dụng lớp Multimedia***

1) Mở tập tin TestMM.vbp
2) Điều chỉnh kích cỡ biểu mẫu chính và vẽ một nút lệnh và một điều khiển hộp thoại thông dụng:

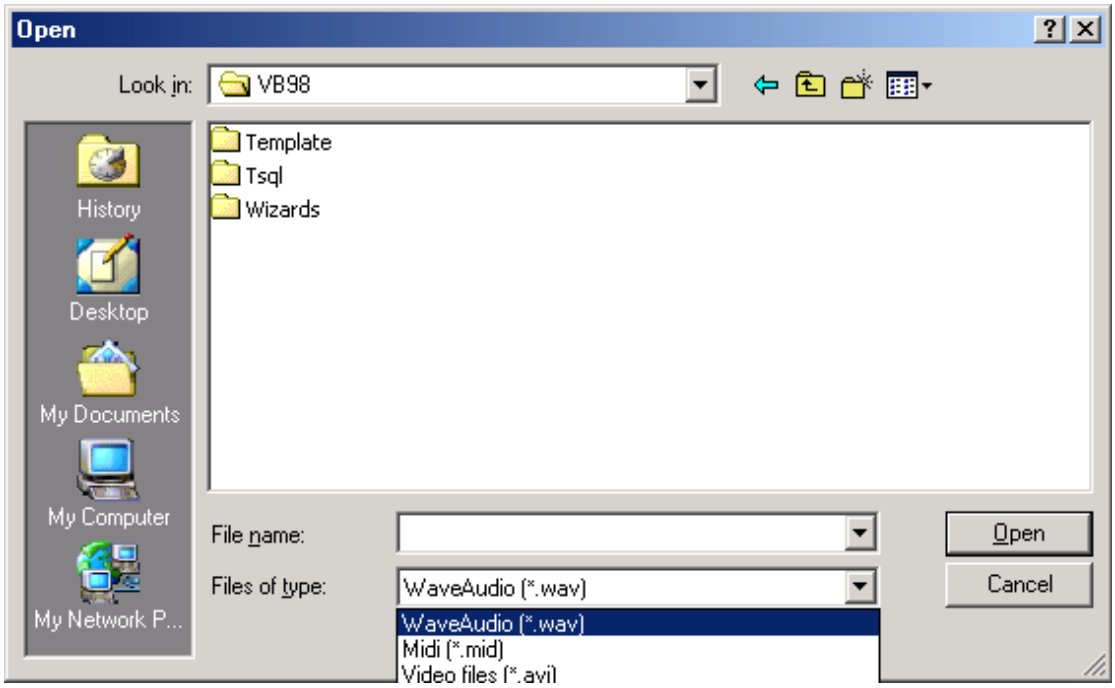
Hình 9.3 Thiết kế biểu mẫu

Nếu không thấy điều khiển hộp thoại thông dụng trên hộp công cụ, từ menu Project, chọn Components, và chọn vào hộp đánh dấu "Microsoft Common Dialog Control 6.0".

3) Để hộp thoại (thông dụng) bật ra khi nhấn vào nút lệnh, ta xử lý sự kiện Click trên nút lệnh bằng cách gõ vào:

```
Private Sub Command1_Click()
     With CommonDialog1
            .Filter = "WaveAudio (*.wav)|*.wav|Midi
          (*.mid)|*.mid|Video files (*.avi)|*.avi"
          .FilterIndex = 0
          .ShowOpen
       End With
End Sub
```

4) Chạy chương trình và nhấn nút lệnh, ta sẽ thấy hộp thoại mở tập tin quen thuộc xuất hiện cho ta chọn tập tin multimedia:



Hình 9.4 Chọn mở tập tin multimedia

5) Kế tiếp ta chuyển lớp multimedia thành một đối tượng.

```
Private Sub Command1_Click()
      Dim Multimedia As New MMedia
      With CommonDialog1
            .Filter  =  "WaveAudio  (*.wav)|*.wav|Midi
            (*.mid)|*.mid|Video files (*.avi)|*.avi"
            .FilterIndex = 0
            .ShowOpen
      End With
      If CommonDialog1.Filename <> "" Then
            Multimedia.mmOpen CommonDialog1.Filename
            Multimedia.mmPlay
      End If
End Sub
```

Thi hành chương trình. Tìm một tập tin multimedia trên đĩa cứng (thường chứa trong thư mục *\Windows\Media*) và play.

Lưu ý rằng để play các tập tin âm thanh như ***WAV*** và ***MID***, ta cần có *card* âm thanh trên máy .

Trong dòng đầu của sự kiện click, ta tạo một đối tượng multimedia dẫn xuất từ lớp ***MMedia***. Đây là bước chuyển từ một lớp sang một đối tượng.

```
Private Sub Command1_Click()
      Dim Multimedia As New MMedia
```

Bốn dòng kế sử dụng đối tượng ***multimedia*** để mở tập tin dùng phương thức ***mmOpen*** và ***Play*** bằng phương thức ***mmPlay***.

```
If CommonDialog1.Filename <> "" Then
      Multimedia.mmOpen CommonDialog1.Filename
      Multimedia.mmPlay
End If
```

Tạo lớp bao bọc các API làm vấn đề đơn giản hơn. Nếu lớp này được đem thương mại hoá, người sử dụng nó sẽ không cần phải hiểu về API, họ chỉ cần biết cách thức hoạt động của lớp mà thôi.

8.4.1.1 Tìm hiểu lớp Multimedia

Windows có nhiều phân hệ, mỗi phân hệ đảm nhiệm một chức năng nhất định. Một trong những phần này là MCI. MCI là tên gọi tắt của *Multimedia Control Interface*, cung cấp một giải pháp độc lập với thiết bị để sử dụng các tính năng của Windows thông qua chương trình.

Khi viết chương trình trò chơi trên DOS, ta phải xử lý với nhiều chuẩn card âm thanh và hình ảnh khác nhau. Tính năng độc lập với thiết bị, và các chương trình điều khiển thiết bị cung cấp bởi Windows cho phép ta làm việc với bất kỳ *card* âm thanh, hình ảnh nào với cùng chương trình, miễn là chúng được hỗ trợ bởi Windows.

MCI cung cấp lớp đệm giữa lập trình viên và các thiết bị dùng xử lý dữ liệu *multimedia* như các *card* âm thanh, hình ảnh.

MCI sẽ làm việc với các chương trình điều khiển thiết bị của Windows, và cuối cùng là phần cứng *multimedia*. Lập trình viên, yêu cầu MCI dùng hàm API ***mciSendString***. Lệnh này sau đó được gọi xuống chương trình điều khiển thiết bị, ta không cần quan tâm.

MCI là một đối tượng độc lập. Nó có thể được lập trình và có ngôn ngữ lập trình riêng. Khi ta dùng ***mciSendString***, ta đang lập trình MCI.

8.4.1.2 Sử dụng mciSendString

Cú pháp của **mciSendString**:

<ResultCode> = mciSendString("<*Command*>", _
<*ReturnString*>, <*ReturnLength*>, <*CallbackHadle*>)

<ResultCode> là một số long integer, và thay đổi tuỳ theo dòng lệnh.

<Command> đặt trong đấu trích dẫn, phải là một từ dưới dạng chuỗi ký tự và là lệnh gửi đến MCI; như là ***Play*** để play một tập tin, ***Open*** để mơe tập tin, v.v...

Một số lệnh MCI trả về một chuỗi ký tự. Lệnh ***Status*** trả về một chuỗi cho biết tập tin dừng hẳn (*Stopped*), hay đang chơi (*Playing*), hay dừng tạm (*Pause*), v.v...

API cần biết bao nhiêu dữ liệu được chứa trong biến chuỗi, tham số kế tiếp là chiều dài chuỗi. Do đó, nếu ta phát lệnh đến MCI trả về một chuỗi, ta phải truyền một biến chuỗi có chiều dài nhất định và cho biết chiều dài của nó:

```
Dim sReturnString As String * 255
Dim nReturn As Long
```

```
nReturn = mciSendString("status waveaudio mode",  _
     sReturnString, 255, 0)
```

Thêm * **255** vào khai báo ***sReturnString*** cho biết chiều dài của nó là 255.

8.4.1.3 Sử dụng hàm Callback trong Visual Basic

Hàm ***Callback*** thực ra chỉ áp dụng cho C/C++, Delphi, hay một số ngôn ngữ biên dịch cấp thấp, không dùng với Visual Basic. Tuy nhiên, VB6 cho phép ta sử dụng hàm ***Callback*** mà không cần thêm các chương trình phụ đặc biệt như trong các phiên bản trước.

Khi ta dùng API, chương trình của ta không thể nào biết được điều gì đang xảy ra khi hàm đang chạy. Ta phải chờ đến khi nó kết thúc, và kiểm tra giá trị trả về. Ý tưởng của hàm ***Callback*** là một API mà khi chạy, nó có thể gọi đến một hàm hoặc thủ tục của chương trình ta đang viết.

Ta phải tạo một hàm Public ở trong một module chương trình của Visual Basic, với các tham số truyền cần thiết của API. Sau đó, khi gọi API, ta gửi một con trỏ (pointer) - địa chỉ vùng nhớ của hàm Callback. Ta phải dùng toán tử mới AddressOf:

```
nResult = someAPIFunction(ParamOne, ParamTwo, _
     AddressOf  MyCallback)
```

Khi API chạy, nó gọi mmọt hàm trong chương trình của chúng ta và gửi các tham số cần thiết. Thường nó được được dùng để cập nhật thanh trạng thái, lấy danh sách font hệ thống, và các công việc khác.

Như đã nói, chúng ta sẽ không nói thêm về các hàm ***Callback***. Các hàm này làm phức tạp hơn cho chương trình và nhiều khi làm treo hệ thống. Tuy nhiên, trợ giúp của Visual Basic sẽ cung cấp một số ví dụ nếu bạn uốn tìm hiểu kỹ hơn.

8.4.1.4 Mở tập tin Media

Ta gửi tên tập tin cho lệnh **Open** để mở tập tin. Đây là tên chuẩn như: **C:\Video.avi**.

```
Open <filename> Type <typestring> Alias <aname>
...
...
'Issue command  to do something to the file
...
...
Close <aname>
```

Sau từ khoá ***Type*** là kiểu tập tin. Kiểu chuẩn của Windows là ***WaveAudio*** đối với tâp tin ***WAV***, ***AVIVideo*** đối với ***AVI***, và ***Sequencer*** đối với ***MID***.

Alias dùng để thay thế tên tạp tin mở:

```
Open c:\video.avi Type AVIVideo Alias Peter
```

Nếu ta gửi dòng lệnh này đến MCI bằng ***MCISenString***, nó yêu cầu MCI mở tập tin ***C:\video.avi*** như một tập tin video của Microsoft, và nó sẽ dùng tên ***Peter*** để chỉ ra tập tin này.

Mỗi lần mở tập tin, lệnh MCI có thể dùng bí danh để chơi tập tin, dừng hẳn hay tạm dừng, hoặc hiển thị trạng thái, v.v... Ví dụ:

```
Play Peter
Pause Peter
Stop Peter
```

Sau đó, ta cần đóng tập tin bằng cách gửi lệnh Close, theo sau là bí danh của tập tin.

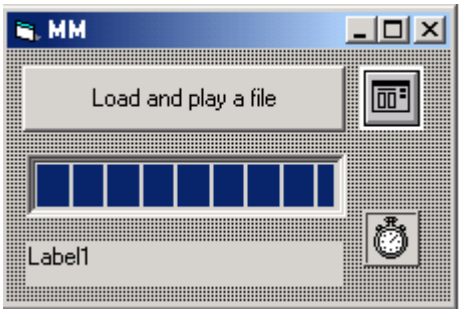
```
nReturn = mciSendString("Close Peter", "",0,0)
```

- ***Ví dụ mẫu - Hiển thị trạng thái và vị trí của tập tin Multimedia***

1. Mở đề án TestMM.vbp
2. Chúng ta sẽ thêm một số điều khiển để xem thuộc tính ***Status*** và ***Position*** của lớp ***MMedia***. Thêm một điều khiển ***thanh diễn tiến*** (*ProgressBar*), một nhãn, một điều khiển định giờ (*timer*):

Hình 9.5 *Thiết kế biểu mẫu*

Nếu không thấy điều khiển ***ProgressBar*** trên hộp công cụ, từ menu



Project, chọn ***Components***, chọn vào hộp đánh dấu "*Microsoft Windows Common Controls 6.0*".

3. Mở cửa sổ ***Properties*** của điều khiển ***Timer***, đổi thuộc tính ***Enabled*** thành *False*, và ***Interval*** là 500. Xoá ***Caption*** của điều khiển nhãn.
4. Nhấn đúp chuột lên nút lệnh để mở sự kiện ***Click***:

```
Private Sub Command1_Click()
      …
      …
      If CommonDialog1.Filename <> "" Then
            Multimedia.Wait = False
            Multimedia.mmOpen
            CommonDialog1.Filename
            ProgressBar1.Value = 0
            ProgressBar1.Max                    =
            Multimedia.Length
            Timer1.Enabled = True
            Multimedia.mmPlay
      End If

End Sub
```

5. Trở về biểu mẫu, nhấn đúp chuột lên điều khiển ***Timer1*** để mở sự kiện ***Timer***.

```
Private Sub Timer1_Timer()
      ProgressBar1.Value = Multimedia.Position
      Label1 = "Status: " & Multimedia.Status
      If ProgressBar1.Value = ProgressBar1.Max Then
            Multimedia.mmClose
            Timer1.Enabled = False
      End If
End Sub
```

Có một vấn đề nhỏ. Ta đã định nghĩa biến chỉ đến *instance* của lớp ***MMedia*** trong hàm sự kiện ***command1_Click().*** Bây giờ chúng ta lại muốn chỉ đến nó từ trong Timer1_Timer().

GHI CHÚ *Bạn sẽ được giải thích khái niêm instance trong chương 13 - Lập trình hướng đối tượng*

6. Trong sự kiện ***Click*** của nút lệnh, chọn dòng khao báo biến ***Mutilmedia***, nhấn phím *Ctrl-X* để cắt nó vào *Clipboard* và xoá nó khỏi sự kiện ***Command1_Click***. Sau đó, chọn vào danh sách (***General***) trong cửa sổ Code, nhấn phím *Ctrl –V* để dán nó vào vùng ***General Declarations***. Biến khai báo đặt trong vùng này sẽ là biến toàn cục đối với biểu mẫu này.
7. Thi hành chương trình. Nhấn nút "*Load and Play a file*", và chọn một tậptin AVI, ví dụ tập tin video "*Welcome to windows 95*".
8. Ta sẽ thấy thanh diễn tiến cho thấy bao nhiêu phần trăm của tập tin đang chơi. Khi video kết thúc ta thấy kết quả hiển thị: ***Stopped***.

Khi ta mới nhấn nút lệnh, chương trình thiết lập các khởi tạo cho thuộc tính trước khi chơi tập tin:

```
If CommonDialog1.Filename <> "" Then
      Multimedia.Wait = False
            Multimedia.mmOpen CommonDialog1.Filename
            ProgressBar1.Value = 0
            ProgressBar1.Max = Multimedia.Length
            Timer1.Enabled = True
            Multimedia.mmPlay
      End If
```

Đối tượng *multimedia* có thuộc tính tên là ***Wait***. Thuộc tính này quyết định chương trình có tiếp tục thi hành (đa nhiệm) trong khi chơi tập tin, hay phải dừng và chờ đến đến khi nó hoàn tất. Phương thức ***mmPlay*** theo dõi giá trị của biến ***bWait***. Nếu biến này có giá trị *True*, nó thêm ***Wait*** vào lệnh gọi *mciSendString:*

```
      Public Sub mmPlay()
            Dim nReturn As Long
            If sAlias = "" Then Exit Sub
            If bWait Then
                  nReturn = mciSendString("Play " &  _
                  sAlias & " wait", "", 0, 0)
            Else
                  nReturn = mciSendString("Play " & sAlias,
      "", 0, 0)
            End If
      End Sub
```

Làm sao biết giá trị ***bWait*** ? Nhắc lại rằng ta có thể cung cấp các hàm thuộc tính cho phép đọc hoặc quy định giá trị của biến nội bộ:

```
      Property Get Wait() As Boolean
            ' Routine to return the value of the object's
      wait property.
             Wait = bWait
      End Property

      Property Let Wait(bWaitValue As Boolean)
            ' Routine to set the value of the object's
      wait property
              bWait = bWaitValue
      End Property
```

Bước kế là mở tập tin ta muốn chơi. Ta dùng phương thức ***mmOpen*** để mở tập tin.

8.4.1.4.1 Mở tập tin

Trước hết, ta khai báo một cặp biến cục bộ để giữ giá trị tạm thời.

```
      Public Sub mmOpen(ByVal sTheFile As String)
            Dim nReturn As Long
            Dim sType As String
```

```
        If sAlias <> "" Then
              mmClose
        End If

        Select Case UCase$(Right$(sTheFile, 3))
              Case "WAV"
                    sType = "Waveaudio"
              Case "AVI"
                    sType = "AviVideo"
              Case "MID"
                    sType = "Sequencer"
              Case Else
                    Exit Sub
        End Select
        sAlias = Right$(sTheFile, 3) & Minute(Now)
        If  InStr(sTheFile, " ") Then
              sTheFile = Chr(34) & sTheFile & Chr(34)
        End if
        nReturn =mciSendString("Open " & sTheFile  _
        & "  ALIAS " &  sAlias  & " TYPE " & sType _
        & " wait", "", 0, 0)
End Sub
```

Trước hết, hàm ***mmOpen*** kiểm tra biến ở mức module gọi là sAlias.

```
If sAlias <> “” then
      mmClose
End if
```

Làm việc với MCI, ta nên dùng bí danh cho từng tập tin mở. Ở đây lớp ***MMedia*** thiết lập một tên cho bí danh và chứa bí danh vào biến ***sAlias***. Khi ta tiếp tục mở một tập tin kế tiếp bằng ***mmOpen***, hoặc chỉ ra thuộc tính tên tập tin, chương trình kiểm tra điều này và gọi một thủ tục khác để đóng tập tin thứ nhất. Đóng tập tin khi ta cần giải phóng vùng nhớ và tăng tốc độ chơi tập tin.

Cấu trúc ***Select Case*** kiểm tra từng loại tập tin.

Lệnh ***wait*** cho phép chương trình tiếp tuch chạy cho đến khi nạp thành công. Nếu không có ***wait***, trên một máy nhanh với đĩa cứng chậm, có thể có vấn đề. Ta có thể cố chơi tập tin trước khi nó được nạp xong, đơn giản bởi vì chương trình chạy nhanh hơn đĩa cứng. Lưu ý rằng nó không giống thuộc tính ***Wait*** trước đây để điều khiển chương trình tiếp tục chạy khi tập tin đang chơi chứ không phải đang nạp.

8.4.1.4.2 Lấy chiều dài tập tin

Dùng ***mciSendString*** để lấy hoặc quy định chiều dài. Thuộc tính ***Length*** của lớp ***MMedia*** chỉ có tính chất được phép đọc mà thôi, và ta không cung cấp hàm ***Property Let***.

```
        Property Get Length() As Single
              Dim nReturn As Long, nLength As Integer
              Dim sLength As String * 255
```

```
        If sAlias = "" Then
              Length = 0
              Exit Property
        End If

        nReturn = mciSendString("Status " & sAlias _
              & length",  Length, 255, 0)
        nLength = InStr(sLength, Chr$(0))
        Length = Val(Left$(sLength, nLength - 1))
  End Property
```

Trước hết ***sAlias*** được kiểm tra xem tập tin có đang mở hay không ? Nếu chưa mở, giá trị trả về từ thủ tục thuộc tính là 0. Nếu tập tin mở rồi, lệnh ***Status Length*** của MCI được dùng.

Ta không lo về cách tính chiều dài tập tin, vì đơn vị đo phù hợp với thanh diễn tiến.

Lệnh *Status* là lệnh MCI đặc biệt, có thể kết nối với các từ khoá như ***Length, Position, Mode*** để xác định các thông tin về tập tin hiện hành. Nó trả về các thông tin này trong một chuỗi ký tự có chiều dài nhất định được truyền vào mciSendString. Trong ví dụ này chuỗi trả về là ***sLength*** và dài 255 ký tự.

Dĩ nhiên, nó không luôn chứa 255 ký tự trả về từ lệnh ***Status***. Dùng hàm ***InStr*** để cắt bỏ các ký tự 0 lấp đầy khoảng trống.

Chiều dài chứa trong chuỗi được trích chuỗi và chuyển đổi sang kiểu số trước khi gán cho ***Length.***

8.4.1.4.3 Lấy vị trí hiện hành

Lệnh ***Status Position*** có thể được gọi nhiều lần để xác định vị trí hiện hành của tập tin đang chơi:

```
  Property Get Position() As Single
        Dim nReturn As Integer, nLength As Integer
        Dim sPosition As String * 255

        If sAlias = "" Then Exit Property
        nReturn = mciSendString("Status " & sAlias _
              & " position", sPosition, 255, 0)
        nLength = InStr(sPosition, Chr$(0))
        Position = Val(Left$(sPosition, nLength - 1))
  End Property
```

Thay vì gửi ***Status Length***, ta gửi ***Status Position***.

8.4.1.4.4 Lấy trạng thái hiện hành

Để lấy chuỗi ký tự trạng thái còn gọi là ***mode***, ta truy vấn thuộc tính ***Status*** của lớp. Ta cũng sử dụng hàm thuộc tính ***Property Get*** hầu đồng nhất với thuộc tính ***Position*** trên đây.

Chỉ khác là ta gửi ***Status Mode*** thay vì ***Status Length*** hay ***Status Position*** cho ***mciSendString***. Dĩ nhiên, không cần chuyển đổi sang kiểu số:

```
  ...
  nReturn = mciSendString("Status " & sAlias &  _
        " mode", sStatus, 255, 0)
```

```
nLength = InStr(sStatus, Chr$(0))
Status = Left$(sStatus, nLength - 1)
...
```

Trở lại sự kiện ***Command1_Click***. Cho tới giờ, ta đã định nghĩa thuộc tính ***Wait***., mở tập tin, thiết lập thanh diễn tiến. Trước khi chơi tập tin, ta quy định ***Timer***. Sau đó, ta chơi tập tin bằng cách gọi phương thức ***mmPlay*** của đối tượng ***Multimedia.***

```
If CommonDialog1.Filename <> "" Then
      Multimedia.Wait = False
      Multimedia.mmOpen CommonDialog1.Filename
      ProgressBar1.Value = 0
      ProgressBar1.Max = Multimedia.Length
      Timer1.Enabled = True
      Multimedia.mmPlay
End If
```

8.4.1.4.5 Chơi trên tập tin

Trước hết, kiểm tra tập tin mở thông qua biến ***sAlias***; sau đó nếu thoả điều kiện, nó thi hành lệnh ***Play*** của MCI.

```
Public Sub mmPlay()
      Dim nReturn As Long
      If sAlias = "" Then Exit Sub

      If bWait Then
            nReturn  =  mciSendString("Play  "  &    _
            sAlias & " wait", "", 0, 0)
      Else
            nReturn = mciSendString("Play " & _
            sAlias, "", 0, 0)
      End If
End Sub
```

8.4.1.4.6 Cập nhật thanh diễn tiến và điều khiển nhãn.

Công việc sau cùng là cập nhật thanh trạng thái và nhãn trên biểu mẫu, khi tập tin đang chơi. Trước khichơi tập tin, đặt điều khiển ***Timer*** với Interval là 500. Vậy nó sẽ kích hoạt đếm mỗi nửa giây. Khi đó, đoạn chương trình sau được thi hành:

```
Private Sub Timer1_Timer()
      ProgressBar1.Value = Multimedia.Position
      Label1 = "Status: " & Multimedia.Status
      If ProgressBar1.Value = ProgressBar1.Max Then
            Multimedia.mmClose
            Timer1.Enabled = False
      End If
End Sub
```

Cuối cùng cần phải ngưng lại khi đạt đến cuối tập tin. Có thể thực hiện điều này bằng cách so sánh giá trị hiện hành và giá trị ***Max*** của thanh diễn tiến. Khi chúng bằng nhau, tập tin được đóng bằng phương thức ***mmClose***. Sau đó, cấm Timer để ngăn hàm này chạy cho đến khi mở tập tin khác.

8.4.1.4.7 Tóm tắt các lệnh của MCI

Lệnh	**Mô tả**
Play	Chơi một tập tin
Pause	Tạm dừng chơi, sẵn sàng bắt đầu mọi lúc
Stop	Dừng hẳn - cần chuyển đến một vị trí nào đó để tiếp tục chơi
Seek	Theo sau là một con số, chuyển đến vị trí trong tập tin
Status Mode	Trả về một chuỗi ký tự thể hiện trạng thái tập tin(đang choi, đang mở, tạm dừng, dừng hẳn....)
Status Position	Trả về vị trí tập tin mà *playback* đã đạt đến
Status Length	Trả về chiều dài tập tin và hỗ trợ để dưa con số trả về từ Status Position vào một ngữ cảnh có ý nghĩa nào đó.
Close	Đóng tập tin và giải phóng vùng nhớ nó chiếm trước đó

Ngoài ra MCI còn hỗ trợ một số lệnh khác và một số lệnh đặc biệt cho mỗi định dạng tập tin.

9 Thêm trợ giúp vào ứng dụng

9.1 Thêm hỗ trợ cho Help

Thêm hỗ trợ cho Help vào ứng dụng VB gần như khá đơn giản. Tất cả những gì ta cần làm là chỉ ra một thuộc tính, ***HelpFile*** (và dĩ nhiên, viết và biên dịch *tập tin Help*) để hiển thị *Help* khi người sử dụng nhấn phím ***F1*** hay yêu cầu *Help* từ Menu. Một thuộc tính khác là ***HelpContextID***, dùng để cung cấp chủ đề *Help* theo ngữ cảnh cho bất kỳ giao diện người sử dụng nào trong chương trình. Quá trình gắn thêm ***Help*** là như nhau trong cả ***WinHelp*** và ***HTML Help***.

9.1.1 Thuộc tính HelpFile

Thuộc tính *HelpFile* của đối tượng ***App*** được dùng để chỉ ra tên của tập tin *Help* cho ứng dụng. Nó đòi hỏi một tập tin hợp lệ của ***WinHelp*** (.hlp) hoặc là ***HTML Help*** (.chm). Nếu tập tin không tồn tại, lỗi sẽ xảy ra.

9.1.1.1 Chỉ ra thuộc tính HelpFile

1. Chọn *Project Properties* từ menu *Project* để mở hộp thoại *Project Properties*.
2. Trong trường *Help File Name* của tab *General*, gõ vào đường dẫn và tên của tập tin *Help* của ứng dụng (*.hlp* hay.*chm*).

Ta còn có thể chỉ ra ***HelpFile*** bằng cách lập trình. Đoạn chương trình sau đây chỉ ra một tập tin ***HTML Help*** chứa trong cùng thư mục với tập tin thi hành của ứng dụng:

```
Private Sub Form_Load()
 App.HelpFile = App.Path & "\foo.chm"
End Sub
```

Đối tượng ***ErrObject*** cũng có thuộc tính ***HelpFile***, cho phép ta chỉ ra một tập tin *Help* khác cho các thông báo lỗi. Ví dụ, nếu ta có một vài ứng dụng sử dụng chung các thông báo lỗi, ta có thể đặt *Help* cho các thông báo lỗi trong một tập tin *Help* duy nhất và gọi nó bằng thuộc tính ***Err.HelpFile*** trong mỗi ứng dụng.

9.1.2 Thuộc tính HelpContextID

Thuộc tính ***HelpContextID*** được dùng để liên kết một phần giao diện người sử dụng (như là điều khiển, biểu mẫu hay *menu*) với một chủ đề liên quan trong tập tin ***Help***. Thuộc tính ***HelpContextID*** phải có kiểu là một số Long tương ứng với ***Context ID*** của một chủ đề trong tập tin ***WinHelp*** (.hlp) hay ***HTML Help*** (.chm).

Ví dụ, ta có thể nhập 10000 vào thuộc tính ***HelpContextID*** của hộp văn bản. Khi người sử dụng chọn hộp văn bản và nhấn ***F1***, VB tìm kiếm chủ đề có *Context ID* là 10000 trong tập tin *Help* được chỉ ra thuộc tính ***HelpFile*** của ứng dụng. Nếu nó tìm thấy, một cửa sổ ***Help*** sẽ được mở và hiển thị chủ đề; nếu không, lỗi sẽ xuất hiện và chủ đề mặc định của tập tin ***Help*** sẽ được hiển thị.

Ta nên sử dụng ***HelpContextID*** duy nhất cho mỗi chủ đề *Help* trong tập tin *Help*. Trong một số trường hợp, nếu muốn, ta có thể gán cùng ***HelpContextID*** cho một đối tượng nếu như chúng sử dụng chung một chủ đề *Help*.

Ta không nhất thiết phải nhập một ***HelpContextID*** cho mỗi điều khiển trên biểu mẫu. Nếu người sử dụng nhấn ***F1*** trên điều khiển với ***HelpContextID*** 0 (giá trị mặc định), VB sẽ tìm kiếm một ***HelpContextID*** hợp lệ cho nơi chứa của điều khiển.

9.1.2.1 Gán HelpContextID cho một điều khiển hay biểu mẫu.

1. Chọn một điều khiển hay biểu mẫu mà ta muốn nhập vào ***HelpContextID***.
2. Nhấn đúp ***HelpContextID*** trong cửa sổ ***Properties*** và gõ vào giá trị Long hợp lệ.

Theo dõi giá trị mà ta nhập vào sao cho ta có thể dùng cùng giá trị đó cho *Context ID* của chủ đề ***Help*** tương ứng.

Chú ý: Đối với điều khiển **CommonDialog** và có lẽ đối với một số điều khiển khác, tên của thuộc tính này là **HelpContext** thay vì **HelpContextID**.

9.1.2.2 Gán HelpContextID cho menu

1. Chọn ***Menu Editor*** từ menu ***Tools***.
2. Chọn mục menu mà ta muốn nhập vào một ***HelpContextID***.
3. Nhập vào một giá trị Long hợp lệ trong hộp ***Select the HelpContextID***.

Theo dõi giá trị mà ta nhập vào sao cho ta có thể dùng cùng giá trị đó cho *Context ID* của chủ đề *Help* tương hợp.

HelpContextID còn có thể được nhập vào bằng cách lập trình:

```
Private Sub Form_Load()
 Command1.HelpContextID = 12345
 MenuHelp.HelpContextID = 23456
 Err.HelpContext = 34567
End Sub
```

9.2 Thêm hỗ trợ cho WHAT'S THIS HELP

VB cho phép ta thêm ***What's This Help*** vào các ứng dụng một cách dễ dàng. *What's This Help* cung cấp phần truy cập nhanh đến văn bản Help trong một cửa sổ bật ra mà không cần phải mở *Help Viewer*. *What's This Help* chủ yếu được sử dụng để cung cấp trợ giúp đơn giản cho các phần giao diện người sử dụng như là các trường dữ liệu nhập. VB hỗ trợ các chủ đề *What's This Help* trong các tập tin ***WinHelp*** (.hlp) và ***HTML Help***.

Việc gán giá trị *True* cho thuộc tính ***WhatsThisHelp*** của biểu mẫu làm cho ***What's This Help*** hoạt động được. Khi đó, phần *Help* theo ngữ cảnh cho biểu mẫu sẽ bị vô hiệu hoá.

9.2.1 Kích hoạt What's This Help cho biểu mẫu

1. Với biểu mẫu đã được chọn, nhấn đúp lên thuộc tính ***WhatsThisHelp*** trong cửa sổ ***Properties*** để định giá trị cho nó là *True*.

2. Cài đặt các thuộc tính sau đây để thêm một nút ***What's This*** vào thanh tiêu đề của biểu mẫu:

Thuộc tính	Cài đặt
BorderStyle	*1-Fixed Single* hay *2-Sizable*
MaxButton	False
MinButton	False
WhatsThisButton	True

3. Chọn một điều khiển mà ta muốn cung cấp ***What's This Help***và gán một giá trị duy nhất cho thuộc tính ***WhatsThisHelpID*** của điều khiển.

Theo dõi giá tị mà ta nhập vào sao cho ta có thể dùng cùng giá trị đó cho *Context ID* của chủ đề *Help* tương ứng.

Ta cũng có thể cho phép ***What's This Help*** mà không dùng nút ***What's This*** bằng cách chỉ ra thuộc tính ***WhatsThisHelp*** của biểu mẫu là *True* và gọi phương thức ***WhatThisMode*** của biểu mẫu hay phương thức ***ShowWhatThis*** của điều khiển.

9.3 Cung cấp help cùng với ứng dụng

Bước cuối cùng trong việc thêm *Help* vào ứng dụng là chắc chắn rằng nó sẽ đến tay người sử dụng. Các yêu cầu cho việc cung cấp *Help* cùng với ứng dụng có hơi khác giữa WinHelp và *HTML Help*.

9.3.1 Cung cấp WinHelp

Bởi vì mọi hệ thống Windows đều có cài đặt sẵn ***Trình xem Trợ giúp của Windows*** (*Windows Help Viewer*), ta chỉ còn phải cung cấp tập tin Help (.hlp). ***Trình đóng gói và Triển khai*** (*Package and Deployment Wizard*) tự động thêm các phần phụ thuộc cho tập tin *Help* được tham chiếu bởi ứng dụng. Nếu ta tạo ra phần cài đặt bằng các công cụ khác, ta phải bảo đảm rằng tập tin.*hlp* được đưa vào và được cài đặt vào đúng vị trí (thường là trong cùng thu mục với ứng dụng hoặc là thư mục *Windows\Help*).

9.3.2 Cung cấp HTML Help

HTML Help là một kỹ thuật tương đối mới, do đó, ta không thể giả định rằng mọi người sử dụng đề có những tập tin cần thiết để xem HTML Help. ***Trình đóng gói và Triển khai*** (*Package and Deployment Wizard*) sẽ thêm các phần liên quan đối với tập tin *HTML Help (.chm)* được tham chiếu bởi ứng dụng; nhưng nó không thêm tất cả các phần liên quan đến tập tin ***HTML Help Viewer***. Ta cần phải sửa lại phần cài đặt để đưa các tập tin này vào. Tra cứu các tài liệu cho công cụ *HTML Help* để hiểu thêm về các tập tin nào được yêu cầu trong một tình huống cho trước.

10Lập trình hướng đối tượng

10.1 Giới thiệu về đối tượng

Từ đầu quyển đến giờ, chúng ta chỉ sử dụng biến để chứa những dữ liệu tạm thời trong ứng dụng, chẳng hạn như những giá trị do người sử dụng nhập vào qua giao diện. Tuy nhiên, đây chỉ là một phần nhỏ của VB. VB 6 thực chất là một công cụ lập trình hướng đối tượng rất mạnh.

Bạn có thể cho rằng kỹ thuật này vượt quá khả năng một người mới học lập trình VB. Tuy nhiên, không hẳn như vậy. ***Lập trình hướng đối tượng*** (*Object Oriented Programming* – OOP) giúp lập trình dễ dàng hơn.

Các ví dụ dùng trước đây được lập trình theo kiểu lập trình cổ điển. Điều này không có gì sai bởi vì đây là những chương trình nhỏ và việc sử dụng OOP cho chúng cũng không phù hợp. Với kiểu lập tình cổ điển, còn gọi là ***Phát triển phần mềm theo cấu trúc*** (*Structured Software Development*), ta phải xác định dữ liệu cũng như cách thức để xử lý dữ liệu trong ứng dụng. Một giao diện người sử dụng được cung cấp để hiển thị và nhận dữ liệu từ người sử dụng, sau đó, các hàm và thủ tục con được xây dựng để thực sự xử lý dữ liệu. Điều này có vẻ đơn giản. Để giải quyết một vấn đề lớn, ứng dụng chia thành nhiều vấn đề nhỏ để giải quyết một vấn đề lớn, ứng dụng chia thành nhiều vấn đề nhỏ để giải quyết trong các hàm / thủ tục.

OOP hơi khác một chút. Với lập trình có cấu trúc, cách thức xây dựng ứng dụng, cách chúng kết hợp ở mức chương trình rất khác biệt với thực tế cuộc sống. Lấy một ứng dụng tính lương làm ví dụ. Khi nhân viên được nhận vào làm việc, các thông tin về nhân viên đó sẽ được nhập vào hệ thống tính lương. Sử dụng kỹ thuật lập trình có cấu trúc, ta sẽ dùng một biểu mẫu để chứa các thông tin của nhân viên và viết chương trình để copy tất cả thông tin đã nhập vào biểu mẫu đó vào CSDL chứa ở đâu đó trên mạng công ty. Để tạo ra phiếu trả lương, ta cần có một biểu mẫu in phiếu trả lương cho phép NSD chương trình chọn một nhân viên sẽ trả lương, rồi viết chương trình để thu thập tất cả thông tin từ CSDL và định dạng nó rồi đưa ra máy in.

Ta có thể thấy rằng, giải pháp này nặng về kỹ thuật và nghiêng về xử lý máy tính hơn là cách thực hiện trong thực tế cuộc sống. Lập trình hướng đối tượng sẽ làm cho mọi chuyện trở nên đơn giản hơn nhiều.

Với OOP, ta viết một chương trình dựa trên các *đối tượng* của thực tế cuộc sống. Ví dụ, nếu ta đang viết một ứng dụng tính lương, đối tượng mà ta cần làm việc sẽ là ***phòng ban*** và ***nhân viên***. Mỗi đối tượng này có các ***thuộc tính***: ví dụ, một nhân viên có tên và số; một phòng ban có *vị trí* và *trưởng phòng*. Thêm vào đó, có một số ***phương thức*** để phòng phát lương áp dụng cho các đối tượng trên - mỗi tháng một lần, nó quyết định áp dụng phương thức *phát lương* cho các đối tượng nhân viên. Lập trình OOP cũng tương tự như thế: Ta quyết định đối tượng nào là cần thiết, đối tượng có những thuộc tính nào, và ta sẽ áp dụng những phương thức nào cho đối tượng.

Ta có thể thấy rằng, đây là giải pháp hết sức gần gũi với những vấn đề của thực tế cuộc sống mà ta thường xuyên gặp phải. Nhân viên được xem là đối tượng trong một ứng dụng, và phòng ban là đối tượng có liên quan với nhân viên.

Với lập trình có cấu trúc, ta có xu hướng xem dữ liệu và cách thức xử lý dữ liệu là hai phần tách biệt nhau, hoàn toàn khác với các đối tượng và cách xử lý trong thực tế cuộc sống mà ta vẫn thường làm. Với OOP, ta đóng gói dữ liệu và các chức năng xử lý dữ liệu trong một ***đối tượng*** (*Object*) giống hệt với đối tượng trong thực tế cuộc sống. Nhân viên có *tên* và *địa chỉ*, vì vậy, đây sẽ là các thuộc tính của đối tượng *Nhân viên* - dữ liệu. Nhân viên có thể *được nhận việc* và *bị đuổi việc*, vì vậy, đây sẽ là các phương thức của đối tượng *Nhân viên* – chức năng.

Bằng cách chia ứng dụng thành nhiều đối tượng và phát triển trên các đối tượng, kỹ thuật này gần gũi hơn với đời sống. Nó giúp tạo ra những chương trình dễ đọc dễ bảo trì. Kỹ thuật này cũng là chọn lựa của nhiều công ty lớn phát triển trong phần mềm.

10.1.1 Đối tượng trong VB

Như vậy các lý thuyết trình bày trên đây thể hiện như thế nào trong VB? Khi ta quyết định đặt một hộp văn bản vào biểu mẫu, ta có phải gọi thủ tục con để tạo hộp văn bản, một thủ tục con khác để đặt hộp văn bản vào vị trí, rồi gọi một thủ tục con khác nữa để định giá trị khởi động? Ta có phải luôn gọi một hàm mỗi khi người sử dụng nhập một giá trị bất kỳ vào hộp văn bản? Dĩ nhiên không phải như vậy.

Những gì ta cần làm là kéo và thả một đối tượng (hay một điều khiển), chẳng hạn như hộp văn bản vào biểu mẫu, rồi dùng các thuộc tính để sửa đổi cách thể hiện chúng. Khi người sử dụng nhập dữ liệu vào hộp văn bản, hộp văn bản sẽ thông báo cho ta biết thông qua các sự kiên ***Change*** và sự kiện ***KeyPress***. Mặc dù trước đây, bạn không nhận ra điều này, nhưng ở một mức độ nào đó, bạn đã thực hiện lập trình hướng đối tượng rồi đó.

Ngoài các đối tuợng hay điều khiển được cung cấp sẵn, VB còn cho phép lập trình viên tạo ra các đối tượng thông qua cơ chế ***modul lớp*** (*Class module*). Trong lớp Modul, ta định nghĩa các thuộc tính và phương thức của một đối tượng. Sau khi hoàn tất, để sử dụng đối tượng, trước hết, ta tạo ra đối tượng và gọi các hàm / thủ tục trong modul lớp.

Các đối tượng này có một số đặc tính chung:

- Từng đối tượng phải có chức năng tổng quát, được định nghĩa vừa đủ để hiểu nhưng khá mềm dẻo để có thể sử dụng được; nhưng cho phép phát triển thêm tuỳ theo yêu cầu. Ví dụ, một nút lệnh phải có chức năng chung là nhấn vào để thi hành một công việc gì đó. Tuy nhiên, cách thể hiện và hoạt động của nó trong từng trường hợp có thể thay đổi chút ít tuỳ theo cách ta cài thuộc tính và viết code cho phương thức để phản ánh với sự kiện.
- Đối tượng giao tiếp bên ngoài thông qua thuộc tính, phương thức, và sự kiện được định nghĩa trước cho nó. Tổ hợp của 3 khái niệm này gọi là giao diện (Interface). Đó là những yếu tố cần biết về một đối tượng để sử dụng chúng.
- Có thể sử dụng nhiều đối tượng trong một đề án, ta cũng có nhiều thể hiện khác nhau của một kiểu đối tượng.
- Người sử dụng đối tượng không cần quan tâm đến cách lập trình bên trong đối tượng.

- Bởi vì người sử dụng chỉ thấy đối tượng điều khiển, ta có thể thay đổi hoạt động bên trong của đối tượng sao cho những thay đổi này không ảnh hưởng đến ứng dụng đang dùng, nghĩa là không thay đổi Interface.

10.1.2 Modul Lớp

Khuôn mẫu để tạo đối tượng là ***modul lớp***. Sau này, *modul lớp* còn được dùng để tạo ***điều khiển ActiveX***, một kỹ thuật cao hơn của lập trình hướng đối tượng.

Trong bước lập trình căn bản với VB, ta dung modul để chứa các hàm hay thủ tục. Tuỳ theo tầm hoạt động của hàm / thủ tục này, ta có thể gọi chúng trực tiếp từ modul.

Nhưng modul lớp thì không bao giờ được gọi trực tiếp. Để sử dụng một lớp, ta phải tạo đối tượng từ lớp thông qua lệnh ***New***.

Ở đây đối tượng được tạo từ lớp ***MyClass***, còn biến đối tượng MyObject cung cấp một tham chiếu đến đối tượng.

```
Dim MyObject As New myClass
```

Dòng lệnh trên tạo một đối tượng gọi là ***MyObject*** theo mô tả của lớp ***MyClass***. Hành động này gọi là tạo một *Instance* từ lớp.Trong cửa sổ ***Properties***, ta có thể phân biệt tên lớp và tên đối tượng. *Combo1* là tên đối tượng, trong khi *ComboBox* là tên lớp.

Ta có thể tạo ra vô số *Instance* từ một lớp. Mỗi *Instance* có thể khác nhau một chút tuỳ theo cách ta quy định thuộc tính và sử dụng phương thức.

10.1.2.1 Thuộc tính và phương thức của lớp

Bên trong một lớp, ta có ***thủ tục phương thức*** (*Method Procedures*) và ***thủ tục thuộc tính*** (*Property Procedures*). Quy định một thuộc tính, nghĩa là ta đang gọi *hàm xử lý sự kiện* ***Property Let***.

Trong VB4, *modul lớp* còn rất thô sơ. Đến VB5, nó bắt đầu hỗ trợ gần gũi hơn cho lập trình hướng đối tượng. Ví dụ: ta có thể tạo ra những điều khiển có thể kết hợp trong môi trường phát triển VB (và những ngôn ngữ khác). Thuộc tính của chúng hiển thị trong cửa sổ ***Properties***, những sự kiện của chúng xuất hiện trong danh sách thả xuống chứa trong cử sổ ***Code***.

Trong VB6, phiên bản *Professional* và *Enterprise* hỗ trợ ***Trình xây dựng Lớp*** (*Class Builder*) giúp lập trình viên làm việc rất dễ dàng với lớp. Nó cung cấp một loạt các hộp thoại, hướng dẫn ta từng bước để tạo lớp. Tuy nhiên, trong ví dụ này, để hiểu tổ chức một lớp, ta sẽ tạo bằng tay.

- ***Ví dụ mẫu - Thiết kế lớp có chức ăng di chuyển hộp trên màn hình:***

1. Tạo đề án mới, kiểu ***Standard EXE***.
2. Từ menu ***Project***, chọn ***Add Class Module***.
3. Chọn ***Class Module*** và nhấn ***Open***. Cửa sổ ***Code*** sẽ hiển thị. Nếu nhìn vào cửa sổ *Project Explorer*, ta sẽ thấy một lớp mới xuất hiện.
4. Vì ta muốn tạo một lớp ***Box***, nên ta đổi tên lớp *Class1* sao cho gợi nhớ: ***clsBox***. Cụm từ "cls" thể hiện đây là lớp, nhờ đó, chương trình trở nên dễ đọc

hơn. Để thực hiện điều này, tìm lớp *Class1* trong cửa sổ *Properties*, đổi thuộc tính ***Name*** của nó thành *clsBox*.

10.1.2.1.1 Thuộc tính của Lớp – Public và Private

Lớp ***Box*** có 4 thuộc tính là toạ độ góc trái trên (X,Y), chiều cao (***Height***) và chiều rộng (***Width***). Bây giờ ta cần khai báo các thuộc tính trên là *Public* hay *Private*.

Khi một thuộc tính được khai báo là ***Public*** trong một lớp, nó sẽ được sử dụng bởi bất kỳ đoạn chương trình nào có sử dụng lớp này. Trái lại, nếu thuộc tính là ***Private*** trong một lớp thì nó sẽ không được truy cập bởi bất cứ đoạn chương trình nào khác.

Ta thử tìm hiểu về thuộc tính *Public*. So sánh cách khai báo của một thuộc tính *Public* với một biến *Public*. Ở đây, biến Public trong một lớp giống như một biến Public bất kỳ nào khác, chỉ có điều là khi ta xử lý nó trong chương trình thì giống như ta đang xử lý với một thuộc tính.

Ví dụ, nếu ta khai báo thuộc tính X là một biến Public, sau đó, khai báo một đối tượng gọi là ***MyBox*** dựa trên lớp này, ta có dòng lệnh sau:

```
MyBox.X = 1000
```

Tham chiếu đến X tương tự như khi ta xử lý với một thuộc tính thông thường trên các đối tượng hoặc điều khiển bất kỳ khác. Nhưng những gì chúng ta làm là cho phép người sử dụng đối tượng của chúng ta đổi X thành giá trị mà họ mong muốn.

Bây giờ ta sẽ khai báo X là thuộc tính *Public*, nó cũng tương tự. Nhưng nó cũng không giống hẳn. Đối với thuộc tính Public, mỗi khi nó bị đổi giá trị, một đoạn chương trình bên trong lớp sẽ thi hành. Trong đoạn chương trình này, ta có thể quyết định ta muốn giá trị nào đó mà người sử dụng chỉ ra, và nếu không thì làm một tác vụ gì đó. Do đó, sự khác nhau giữa biến và thuộc tính là: thuộc tính luôn có một đoạn chương tình chạy bên trong mỗi khi nó được truy cập.

Dùng thuộc tính thay cho biến cũng hạn chế khả năng sai sót vì giá trị truyền vào lớp được kiểm nghiệm nhờ đoạn chương trình kiểm tra bên trong lớp.

Trong thực tế, thuộc tính hữu dụng hơn biến vì đôi khi ta cần một xử lý hơn là chỉ gán giá trị. Ví dụ, đổi thuộc tính ***Color*** của một hộp văn bản hiệu chỉnh làm nó đổi màu trên màn hình. Đây là một tác vụ không thể thực hiện được với biến. Đoạn chương trình bên trong sẽ gọi một phương thức để thi hành tác vụ này.

Ví dụ mẫu – Thêm thuộc tính vào lớp.

1. Ta khai báo biến để chứa giá trị thuộc tính

```
Option Explicit
Private mvarX As Integer
```

Biến này có tầm hoạt động bên trong ***modul lớp***.

2. Thêm chương trình vào thuộc tính X:

```
Public Property Let X(ByVal vData As Integer)
   mvarX = vData
End Property
Public Property Get X() As Integer
   X = mvarX
End Property
```

Đoạn chương trình này không thi hành trực tiếp trừ phi nó được gọi thông qua thuộc tính đối tượng.

```
Dim MyBox As New clsBox
MyBox.X = 100
```

Khi ta gán giá trị 100 cho thuộc tính X, thực chất, ta đang gọi thủ tục ***Property Let X***:

```
Public Property Let X(ByVal vData As Integer)
   mvarX = vData
End Property
```

Giá trị 100 truyền cho tham số ***vData***. Sau đó, nó được gán cho biến cục bộ *mvarX*, nghĩa là thuộc tính đã được thay đổi và chứa vào đối tượng. Để xem giá trị thuộc tính, ta gọi:

```
New_Position = MyBox.X
```

Nghĩa là thủ tục ***Property Get X*** thi hành:

```
Public Property Get X() As Integer
   X = mvarX
End Property
```

- Thủ tục Property Let được gọi khi đổi giá trị thuộc tính. Giá trị đổi sẽ được chứa vào một biến cục bộ bên trong lớp.
- Thủ tục Property Get được gọi khi cần đọc giá trị thuộc tính. Giá trị chứa trong biến cục bộ được trả về Property Get.

Tuy nhiên, hai thủ tục thuộc tính này chỉ làm việc với các kiểu dữ liệu cơ bản như ***Variant, String, Integer***...Đối với thuộc tính chứa đối tượng, thay vì dùng ***Property Let***, ta dùng ***Property Set***. Ví dụ:

```
Public Property Set Font (Byval New_Font As stdFont)
 Set mvarFont = New_Font
End Property
```

Để định thuộc tính ***Font*** của đối tượng ***MyObject*** từ ứng dụng, ta gửi cho nó đối tượng Font ***myFont***. Tuy nhiên, để đảm bảo VB dùng thủ tục ***Property Set***, ta đặt từ khoá ***Set*** trước thuộc tính:

```
Dim myFont As New StdFont
myFont.Name=”Courier”
myFont.Bold=True
Set MyObject.Font = myFont
```

Tương tự, ta hoàn tất các thuộc tính còn lại của lớp ***clsBox***.

3. Trong phần *General Declarations*, thêm các biến cục bộ.

```
Option Explicit
Private mvarY As Integer
Private mvarWidth As Integer
Private mvarHeight As Integer
```

4. Thêm các thủ tục tiếp theo:

```
Public Property Let Y(ByVal vData As Integer)
   mvarY = vData
End Property

Public Property Get Y() As Integer
   Y = mvarY
```

```
End Property

Public Property Let Width(ByVal vData As Integer)
   mvarWidth = vData
End Property

Public Property Get Width() As Integer
   Width = mvarWidth
End Property

Public Property Let Height(ByVal vData As Integer)
   mvarHeight = vData
End Property

Public Property Get Height() As Integer
   Height = mvarHeight
End Property
```

5. Lưu modul thành tập tin *clsBox.cls*
6. Đến đây, ta cần 2 phương thức nữa là vẽ hộp (***DrawBox***) và xoá hộp (***ClearBox***). Cả 2 phương thức có một tham số truyền là đối tượng để vẽ hộp lên. Nó có thể là *biểu mẫu, hộp hình...*

10.1.2.1.2 Phương thức của Lớp

- ***Ví dụ mẫu – Thêm phương thức cho lớp***

7. Thêm đoạn chương trình sau vào modul lớp:

```
Public Sub DrawBox(Canvas As Object)
Canvas.Line (mvarX, mvarY)-(mvarX + mvarWidth, mvarY + mvarHeight),,
 B
End Sub
```

Đoạn chương trình này sử dụng phương thức ***Line*** của đối tượng ***Canvas***. Phương thức ***Line*** sẽ vẽ một hộp trên biểu mẫu nếu ta đưa vào tham số cuối cùng B (B có nghĩa là hộp – box).

8. Kế đến, ta thêm ***ClearBox*** vào ***Lớp***:

```
Public Sub ClearBox(Canvas As Object)
Canvas.Line (mvarX, mvarY)-(mvarX + mvarWidth, mvarY + _
    mvarHeight), Canvas.BackColor, B
End Sub
```

9. Lưu modul với tên *clsBox.cls*

Vì 2 thủ tục này sẽ được dùng làm phương thức của đối tượng, nên chúng được khai báo ***Public***, nghĩa là chúng có thể được gọi từ bên ngoài modul.

10.1.2.2 Tạo Instance cho lớp

- ***Ví dụ mẫu - Tạo hoạt hình với đối tượng hộp***

1. Nếu bạn đang mở đề án trong ví dụ trước, chỉ cần nhấn đúp chuột lên biểu mẫu để mở cửa sổ ***Code***. Nếu không, tạo đề án mới kiểu ***Standard EXE***. Từ

menu *Project*, chọn *Add Class Module*; sau đó, chọn tab *Existing* trong hộp thoại và chọn *clsBox.cls*.

2. Tìm sự kiện ***Click*** trong hộp danh sách và đưa đoạn chương trình sau vào:

```
Dim A_Box As New clsBox
```

Biến đối tượng A_Box sẽ giữ một *Instance* của lớp. Từ khóa ***New*** rất quan trọng, nếu thiếu nó, VB sẽ cho rằng ta muốn tạo một bản sao của đối tượng ***clsBox*** hiện hành. Khi tham chiếu đến nó, ta sẽ gặp lỗi.

3. Đưa đoạn chương trình sử dụng đối tượng:

```
Private Sub Form_Click()
   Dim A_Box As New clsBox
   Dim nIndex As Integer
   With A_Box
     .Y = 0
     .Width = 1000
     .Height = 1000
     For nIndex = 0 To 1000
       .ClearBox Me
       .X = nIndex
       .DrawBox Me
     Next
   End With
End Sub
```

4. Thi hành chương trình. Nhấn chuột vào biểu mẫu, ta sẽ thấy hộp trượt dọc theo biểu mẫu.

10.1.2.2.1 Kiểm tra giá trị thuộc tính

Trong thủ tục thuộc tính ***Property Let X***, nếu ta truyền vào chuỗi ký tự "***Hello World***", trình biên dịch sẽ báo lỗi.

```
Public Property Let X(ByVal vData As Integer)

mvarX = vData
End Property
```

Tuy nhiên, nếu ta truyền vào số -7983, hộp chắc chắn sẽ không hiển thị trên biểu mẫu. Ta có thể cấm điều nầy bằng cách:

```
Public Property Let X(ByVal vData As Integer)
if vData >0 Then mvarX = vData
End Property
```

Đối tượng sẽ bỏ qua giá trị âm truyền vào.

10.1.2.2.2 Thuộc tính chỉ được đọc (Read – Only)

Đối với thuộc tính chỉ được đọc, ta không thể thay đổi giá trị thuộc tính. Muốn vậy, ta chỉ cần loại bỏ thủ tục ***Property Let*** trong modul lớp.

10.1.3 Tham số tuỳ chọn

Ta có thể sử dụng tham số tuỳ chọn trong các *phương thức*, thậm chí các *thủ tục của thuộc tính*. Ví dụ, ta có thể thêm tham số màu cho phương thức ***DrawBox***. Khi

đó, chỉ với phương thức ***DrawBox***, ta có thể vẽ hoặc xoá hộp mà không cần gọi ***ClearBox***.

Ví dụ mẫu – Dùng tham số tuỳ chọn

1. Dừng chương trình. Trong cửa sổ *Project Explorer*, nhấn đúp chuột lên ***clsBox*** để mở cửa sổ Code.
2. Tìm phương thức ***ClearBox***, đánh dấu khối thủ tục và nhấn phím ***Delete*** để xoá nó đi.
3. Sửa phương thức ***DrawBox*** để thêm vào tham số tuỳ chọn màu:

```
Public Sub DrawBox(Canvas As Object, Optional lColor As Long)
  If IsMissing(lColor) Then
    Canvas.Line (mvarX, mvarY)-(mvarX + mvarWidth, mvarY + _
     mvarHeight),, B
  Else
    Canvas.Line (mvarX, mvarY)-(mvarX + mvarWidth, mvarY + _
     mvarHeight), lColor, B
  End If
End Sub
```

4. Đến đây, chương trình chưa thể biên dịch, vì vẫn còn một dòng lệnh tham chiếu đến phương thức ***ClearBox***.

.DrawBox Me

Xoá dòng này và thay thế bằng dòng lệnh

.DrawBox Me, Me.BackColor

5. Thi hành chương trình. Không có thay đổi trong kết quả.

Từ khoá ***Optional*** cho biết tham số phía sau nó không nhất thiết phải truyền khi gọi phương thức. Để biết được khi nào có tham số được truyền, ta dùng hàm ***IsMissing***. Hàm này trả về giá trị True/False. Nếu không có tham số truyền, nó trả về True; nếu có, nó trả về False. Tuỳ theo trường hợp mà ta có xử lý tương ứng. Ở đây, khi có tham số truyền, ta gọi hàm ***Line*** có chỉ định màu.

Lưu ý rằng tham số tuỳ chọn phải là tham số cuối cùng trong danh sách được truyền. Ví dụ, dòng lệnh sau đây không được chấp nhận:

Public Sub MyRoutine (Optional sName As String, nAge As Integer)

Ta phải sửa thành:

Public Sub MyRoutine (nAge As Integer, Optional sName As String)

Thận trọng:

Mặc dù tham số tuỳ chọn có tính linh hoạt, giúp ta giảm số dòng chương trình, nhưng nó cũng cho ta những rắc rối kèm theo.

Một trong những lý do để chuyển từ lập trình theo cấu trúc sang lập tình hướng đối tượng là làm cho chương tình dễ đọc, dễ hiểu, dễ xây dựng và dễ bảo trì. Tham số tuỳ chọn làm giảm tính an toàn và tăng độ phức tạp của chương trình khi ta cần gỡ rối.

Tham số tuỳ chọn cung cấp tính năng tái sử dụng chương trình, giúp tạo ra các đối tượng hay thành phần đa mục đích nhưng ta phải cẩn thận và tự hỏi: thêm vào thì có lợi ích gì cho thủ tục của ta, hay là chương trình sẽ tốt hơn nếu ta không dùng nó?

10.1.4 Sự kiện của lớp

Định nghĩa sự kiện cho lớp đã có trong VB5. Nó vẫn hữu dụng với VB6. Chẳng hạn ta muốn mỗi lần hộp được vẽ trên màn hình, sự kiện ***Draw*** gây ra hoạt động cập nhật trên màn hình.

Ví dụ - Định nghĩa và kích hoạt sự kiện

1. Định nghĩa sự kiện ***Draw***. Một trong những thông tin cần cập nhật là toạ độ (x,y) của hộp. Mở cửa sổ ***Code*** và thêm dòng lệnh sau vào phần ***General Declarations***:

```
Public Event Draw(X As Integer, Y As Integer)
```

Tuy nhiên, dòng lệnh này chưa thể hiện lúc nào thì sự kiện được kích hoạt.

2. Ta muốn sự kiện Draw được sinh ra mỗi khi hộp được vẽ trên biểu mẫu. Tìm phương thức ***DrawBox*** và thêm dòng lệnh in đậm vào cuối của phương thức này:

```
Public Sub DrawBox(Canvas As Object, Optional lColor As Long)

    If IsMissing(lColor) Then
       Canvas.Line (mvarX, mvarY)-(mvarX + mvarWidth, mvarY + _
        mvarHeight),, B
    Else
       Canvas.Line (mvarX, mvarY)-(mvarX + mvarWidth, mvarY + _
        mvarHeight), lColor, B
    End If
    RaiseEvent Draw(mvarX, mvarY)
End Sub
```

3. Tìm thủ tục xử lý sự kiện ***Click*** của biểu mẫu. Tìm và xoá dòng lệnh tạo đối tượng ***A_Box*** và thêm một dòng vào phần *General Declarations*:

```
Private WithEvents A_Box As clsBox
Private Sub Form_Click()
   Dim nIndex As Integer
   With A_Box
     .Y = 0
     .Width = 1000
     .Height = 1000
      For nIndex = 0 To 1000
        .DrawBox Me, Me.BackColor
        .X = nIndex
        .DrawBox Me
      Next
   End With
End Sub
```

4. Thêm một dòng vào sự kiện ***Form_Load***:

```
Private Sub Form_Load()
   Set A_Box = New clsBox
End Sub
```

5. Chọn ***A_Box*** từ danh sách trong cửa sổ ***Code***. Chọn sự kiện ***Draw*** từ danh sách các sự kiện.

6. Trong sự kiện này, ta dùng lệnh ***Print*** để in ra toạ độ của hộp trong cửa sổ gỡ rối (hay còn gọi là cửa sổ Immediate)

```
Private Sub A_Box_Draw(X As Integer, Y As Integer)
   Debug.Print "The box just got drawn at " & X & ", " & Y
End Sub
```

7. Thi hành chương trình. Nhấn chuột trên biểu mẫu, ta thấy hộp trượt qua màn hình. Đồng thời trong cửa sổ *Immediate*, ta thấy các dòng văn bản hiển thị toạ độ hiện hành của hộp.

Ở đây, ta dùng phương thức ***RaiseEvent*** để yêu cầu VB phát ra sự kiện ***Draw***, và truyền 2 giá trị của 2 biến thuộc tính ***mvarX*** và ***mvarY*** chứa toạ độ (x, y) của hộp cho sự kiện ***Draw*** mới.

Để có thể xử lý các sự kiện của một đối tượng tự tạo, ta cần khai báo đối tượng hơi khác một chút. Trước hết, nó phải được khai báo là ***Private*** trong biểu mẫu (hoặc modul), thay vì là Private trong một thủ tục. Sau đó, ta phải dùng từ ***WithEvents*** thay vì ***Dim***:

```
Private WithEvents A_Box As clsBox
```

Từ khóa ***WithEvents*** báo cho VB biết ta đang khai báo một đối tượng có sự kiện, và ta dự định viết chương trình để xử lý những sự kiện này.

Chú ý từ khoá ***New*** bị loại bỏ trong dòng lệnh ***WithEvents***. Khi ta viết:

```
Dim A_Box As New clsBox
```

Nghĩa là không chỉ thông báo cho VB rằng ta sắp sử dụng một đối tượng dựa theo lớp ***clsBox***, ta còn yêu cầu VB cấp phát vùng nhớ cho đối tượng và tạo nó. Tuy nhiên, do hạn chế của VB, điều này không được thực hiện với từ khoá ***WithEvents***. Thay vào đó, ta phải tạo đối tượng riêng, bằng cách thêm dòng lệnh vào sự kiện ***Form_Load***.

Tóm lại, đối với sự kiện, ta cần nhớ:

- Khai báo sự kiện dùng Public Event.
- Phát sự kiện dùng RaiseEvent
- Tạo đối tượng với Dim WithEvents, không dùng New.
- Tạo đối tượng như sau:

  Set <đối tượng> = New <lớp>
- Viết chương trình để bắt sự kiện tương tự xử lý sự kiện của điều khiển.

10.1.5 Huỷ đối tượng

Sau khi sử dụng một đối tượng và không cần dùng nữa, ta cần huỷ nó đi. Điều này đặc biệt quan trọng khi ta sử dụng nhiều đối tượng trong ứng dụng. Nếu không huỷ đối tượng, sự hao hụt vùng nhớ sẽ làm giảm khả năng hoạt động của ứng dụng. Ta dùng dòng lệnh sau:

```
Set <đối tượng> = Nothing
```

Nơi lý tưởng để huỷ một đối tượng là trong sự kiện ***Unload*** của biểu mẫu.

Ví dụ:

```
Private Sub Form_Unload (Cancel As Integer)
 Set A_Box = Nothing
End Sub
```

10.2 Biến đối tượng

Cho đến bây giờ, ta chỉ tham chiếu đến điều khiển hay biểu mẫu thông qua tên ta đặt cho chúng lúc thiết kế. Cách làm này chỉ phù hợp đối với các chương tình đơn giản. Đặt đối tượng vào biến và tham chiếu đến nó bằng tên biến cho phép ta sử dụng cùng đoạn chương trình cho vô số các instance khác nhau của một kiểu đối tượng.

Với biến đối tượng ta có thể:

- Tạo điều khiển mới trong lúc thi hành.
- Copy điều khiển để sinh ra một instance mới của điêu khiển hiện hành.
- Tạo bản sao biểu mẫu cùng tên, cùng điều khiển và chương trình; nhưng từng biểu mẫu chứa và xử lý những dữ liệu khác nhau – tương tự nhiều tài liệu trong ứng dụng của Word hay nhiều ***bảng tính*** trong Excel.

Biến đối tượng cung cấp khả năng xây dựng các thủ tục tổng quát để xử lý với những điều khiển nhất định. Ví dụ, một thủ tục kiểm tra dữ liệu của hộp văn bản chỉ dùng trong trường hợp tên điều khiển được chỉ ra trong chương tình. Tuy nhiên, để thủ tục trở thành độc lập với điều khiển bất kỳ, ta xem điều khiển như một biến đối tượng.

Dim NewEmployee As New cEmployee

10.2.1 Tạo điều khiển lúc thi hành

Cách đơn giản nhất là tạo một mảng điều khiển vào lúc thiết kế, sau đó, mở rộng mảng bằng chương trình lúc thi hành. Nếu ta định thuộc tính ***Index*** của điều khiển đầu tiên là 0 lúc thiết kế, ta có thêm điều khiển lúc thi hành. Điều khiển tạo lúc thi hành có cùng tên, kiểu, và thủ tục xử lý sự kiện như điều khiển ban đầu.

Tương tự biến mảng, ta có thể mở rộng hoặc rút gọn mảng điều khiển. Điểm khác nhau là ta không ***Redim*** mảng điều khiển như với biến mảng. Thay vào đó, ta phải ***Load*** bản *instance* mới của điều khiển vào mảng. Khi muốn xoá điều khiển, ta ***Unload*** chúng.

Ví dụ mẫu - Tạo điều khiển lúc thi hành

Thử tạo một dãy các nút lệnh trên biểu mẫu. Vẽ một nút lệnh lúc thiết kế và dùng chương trình để tạo phần còn lại.

1. Tạo đề án mới và vẽ một nút lệnh trên biểu mẫu.
2. Đổi thuộc tính ***Index*** thành 0. Khi ấy một mảng điều khiển có một phần tử được tạo ra.
3. Đưa đoạn chương tình sau vào thủ tục ***Click*** của nút lệnh:

```
Private Sub Command1_Click(Index As Integer)

  Static sNextOperation As String
  Dim nIndex As Integer
  For nIndex = 1 To 5
    If sNextOperation = "UNLOAD" Then
      Unload Command1(nIndex)
    Else
      Load Command1(nIndex)
```

```
        With Command1(nIndex)
          .Top = Command1(nIndex - 1).Top + Command1(nIndex - 1).Height
          .Caption = nIndex
          .Visible = True
        End With
      End If
    Next
    If sNextOperation = "UNLOAD" Then
      sNextOperation = "LOAD"
    Else
      sNextOperation = "UNLOAD"
    End If
  End Sub
```

4. Thi hành đoạn chương trình và nhấn trên nút lệnh vài lần. Mỗi lần nhấn, 5 nút lệnh được tạo hoặc xoá.
5. Lưu vào đĩa với tên ***NewCtrl.vbp***

Vòng lặp *For...Next* tạo hoặc xoá nút lệnh này tuỳ theo nội dung biến *sNextOperation*. Trước hết, nội dung của *sNextOperation* được kiểm tra để xem cần ***Load*** hay ***Unload*** các phần tử. Lần đầu, *sNextOperation* chưa được gán, nó rơi vào phần False, nghĩa là ***Load***.

Bởi mặc định, điều khiển mới tạo lúc thi hành xuất hiện tại cùng vị trí với điều khiển gốc, và không hiển thị. Do đó, ta có thể đổi vị trí và điều chỉnh kích cỡ mà không để người sử dụng thấy. Nó cũng cấm Windows vẽ lại mỗi lần nạp các điều khiển, không những làm chậm chương trình mà còn hiển thị không có trật tự trong khi ta đang di chuyển chúng. Ta chỉ cho chúng hiển thị sau khi đã có vị trí mới.

10.2.2 Sự kiện của mảng điều khiển

Mặc dù hiển thị khác nhau, 6 phần tử vẫn chia sẻ một thủ tục xử lý sự kiện, vì nhấn vào một nút bất kỳ, chúng đề đáp ứng như nau. Ta có thể xử lý các sự kiện theo từng nút lệnh phân biệt, dựa trên ***Index*** của mảng điều khiển.

Ví dụ mẫu - Xử lý sự kiện với mảng điều khiển

1. Mở đề án ***NewCtrl.vbp*** và chọn sự kiện ***Click*** trên nút lệnh.
2. Thêm đoạn chương trình sau vào:

```
Private Sub Command1_Click(Index As Integer)
  Static sNextOperation As String
  Dim nIndex As Integer
    Select Case Index
    Case 0
      For nIndex = 1 To 5
      If sNextOperation = "UNLOAD" Then
       Unload Command1(nIndex)
      Else
      Load Command1(nIndex)
        With Command1(nIndex)
       .Top = Command1(nIndex - 1).Top + Command1(nIndex - 1).Height
```

```
        .Caption = nIndex
         .Visible = True
        End With
      End If
      Next
      If sNextOperation = "UNLOAD" Then
         sNextOperation = "LOAD"
      Else
         sNextOperation = "UNLOAD"
      End If
    Case 1, 2, 3, 4, 5
      MsgBox "You pressed Button " & Index
    End Select
  End Sub
```

3. Thi hành chương trình. Nhấn chuột trên từng nút lệnh, một thông điệp xuất hiện cho biết thứ tự nút nhấn.

10.2.3 Quản lý điều khiển như biến đối tượng

Không chỉ dùng biến đối tượng như mảng điều khiển, ta còn có thể truyền biến đối tượng và mảng đối tượng vào thủ tục hay hàm.

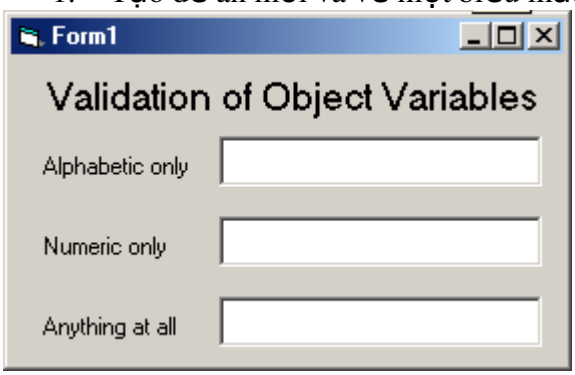
Ví dụ, ta có khoảng 30 hợpp văn bản trên biểu mẫu, từng cái nhận dữ liệu khác nhau. Một số chỉ nhận kiểu số, một số chỉ nhận chữ cái, số khác chấp nhận cả hai, trong khi số còn lại kiểm tra số ký tự nhập vào xem có vượt quá số lượng quy định không? Nếu xử lý riêng rẽ từng điều khiển, ta sẽ tốn rất nhiều đoạn chương tình. Giải pháp là xem hộp văn bản như một đối tượng, và truyền nó đến một thủ tục tổng quát.

10.2.3.1 Hàm kiểm tra hộp văn bản

Hàm này sẽ tự động biết kiểu dữ liệu mà mỗi hộp văn bản cũng như chiều dài tối đa của dữ liệu.

Ví dụ mấu - Kiểm tra hộp văn bản

1. Tạo đề án mới và vẽ một biểu mẫu gồm các điều khiển như sau:



2. Thiết lập thuộc tính cho hộp văn bản như sau:

Mô tả	*Thuộc tính*	*Giá trị*
Hộp "alphabetic only" (chỉ	Name	txtValidate

nhận chữ)	Index Tag	0 A12
Hộp "Numbers" (chỉ nhận số)	Name Index Tag	txtValidate 1 N5
Hộp "Anything" (nhận mọi thứ)	Name Index Tag	txtValidate 0 *4

Lưu ý rằng chữ cái trong thuộc tính ***Tag*** trong cửa sổ ***Properties*** phải là chữ in hoa, nếu không ví dụ không hoạt động.

3. Phần còn lại là viết chương trình. Mở cửa sổ ***Code***, đưa đoạn chương trình sau vào:

```
Option Explicit
Private Sub ValidateKeyPress(txtControl As TextBox, nKeyAscii As Integer)
  Dim sMaxLength As String
  Dim sKey As String * 1
  If nKeyAscii < 32 Or nKeyAscii > 126 Then Exit Sub
  sMaxLength = Right$(txtControl.Tag, Len(txtControl.Tag) - 1)
  If Len(txtControl.Text) >= Val(sMaxLength) Then
    Beep
    nKeyAscii = 0
    Exit Sub
  End If
  Select Case Left$(txtControl.Tag, 1)
    Case "A"
      sKey = UCase(Chr$(nKeyAscii))
      If Asc(sKey) < 65 Or Asc(sKey) > 90 Then
        Beep
        nKeyAscii = 0
        Exit Sub
      End If
    Case "N"
      If nKeyAscii < 48 Or nKeyAscii > 57 Then
        Beep
        nKeyAscii = 0
        Exit Sub
```

```
        End If
    End Select
End Sub
```

4. Thi hành chương trình.

ValidateKeyPress là thủ tục ở mức biểu mẫu và được khai báo trong phần ***General***. *nKeyAscii* là mã của phím nhấn. Vì từ khoá ***ByVal*** không được nêu ra, nên tham số được truyền bằng tham chiếu. Đổi ***KeyAscii*** về 0 trong sự kiện *KeyPress* nghĩa là phím nhấn.

Thuộc tính ***Tag*** được dùng như một nhãn riêng đa năng cho các điều khiển. Nó cho biết kiểu dữ liệu được cho phép trong mỗi hộp văn bản. Ký tự đầu tiên định nghĩa kiểu dữ liệu cho phép, 'A' nghĩa là chỉ có chữ cái, 'N' nghĩa là số, còn lại là các kiểu khác. Con số kế tiếp quy định số ký tự tối đa trong mỗi hộp văn bản.

Ta có thể đổi kiểu dữ liệu của mỗi hộp văn bản bằng cách đổi thuộc tính ***Tag*** trong cửa sổ ***Properties***.

Giai đoạn đầu tiên, chương tình kiểm tra ***nKeyAscii*** cho các ký tự đặc biệt. Nếu chúng được nhấn, phím đó sẽ được bỏ qua.

```
If nKeyAscii < 32 Or nKeyAscii > 126 Then Exit Sub
```

Dòng lệnh kế lấy giá trị từ thuộc tính ***Tag*** đưa vào biến *sMaxLength*.

Sau đó, kiểm tra chiều dài tối đa:

```
    If Len(txtControl.Text) >= Val(sMaxLength) Then
        Beep
        nKeyAscii = 0
        Exit Sub
    End If
```

Select Case so sánh phím ký tự với kiểu dữ liệu quy định trong thuộc tính ***Tag***. Hàm ***Chr$*** chuyển mã ký tự ***nKeyAscii*** thành chuỗi ký tự tương ứng. Hàm ***Asc*** làm ngược lại và trả về mã ký tự ASCII của một ký tự.

10.2.4 Khai báo biến đối tượng

Ta có thể khai báo một biến đối tượng một cách tường minh như khai báo biến thông thường bằng cách cung cấp một kiểu dữ liệu mà VB nhận ra.

```
Dim txtControl As TextBox
```

Chương trình sẽ hiệu quả, dễ gỡ rối và chạy nhanh hơn khi khai báo biến đối tượng tường minh. Tuy nhiên, VB cũng cho phép khai báo một biến đối tượng "ẩn":

```
Dim cltControl As Control
```

Tham số hàm và thủ tục có thể khai báo kiểu này. Nó cho phép ta truyền một điều khiển bất kỳ. Sau đó, dùng dòng lệnh TypeOf để kiểm tra kiểu điều khiển liên quan đến một đối tượng.

Nếu khai báo tường minh, VB kiểm tra thuộc tính của đối tượng ngay lúc biên dịch. Nếu khai báo ẩn, VB chỉ kiểm tra thuộc tính lúc thi hành.

10.2.4.1 Kiểu của biến đối tượng

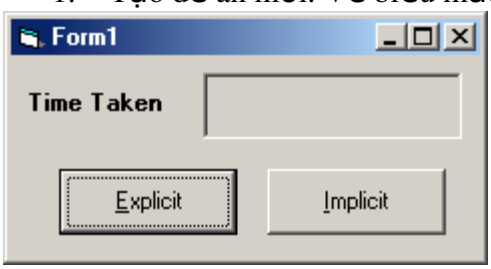
Sau đây là danh sách các kiểu đối tượng tường minh mà VB có thể nhận ra:

CheckBox	ComboBox	CommandButton	MDIForm
Data	DirListBox	DriveListBox	FileListBox
Grid	Frame	HscrollBar	Image
Label	Line	ListBox	menu
OptionButton	OLE	PictureBox	Shape
TextBox	Timer	VscrollBar	Form

Đây là những đối tượng chuẩn. Chúng là những tên lớp và được đặt kế bên tên điều khiển trong cửa sổ Properties.

Ví dụ mẫu – So sánh khai báo tường minh và ẩn

1. Tạo đề án mới. Vẽ biểu mẫu như sau:



2. Đặt tên điều khiển nhãn trống là ***lblTime***, nút lệnh là ***cmdExplicit*** và ***cmdImplicit***.
3. Mở cửa sổ ***Code***, đưa đoạn chương trình sau vào:

```
Private Sub cmdExplicit_Click()
   Dim varTime As Variant
   Dim nIndex As Integer
   varTime = Now
   For nIndex = 1 To 15000
      Time_Explicit cmdExplicit, nIndex
   Next
   cmdExplicit.Caption = "&Explicit"
   lblTime.Caption = Minute(Now - varTime) & " Mins, " & _
       Second(Now - varTime) & " Secs"
End Sub

Private Sub cmdImplicit_Click()
   Dim varTime As Variant
   Dim nIndex As Integer
   varTime = Now
   For nIndex = 1 To 15000
      Time_Implicit cmdImplicit, nIndex
   Next
```

```
    cmdImplicit.Caption = "&Implicit"
    lblTime.Caption = Minute(Now - varTime) & " Mins, " & _
        Second(Now - varTime) & " Secs"
End Sub
```

Đưa 2 thủ tục sau vào phần (***General***)

```
Private Sub Time_Explicit(cmdCommand As CommandButton, nNumber As Integer)
    cmdCommand.Caption = nNumber
End Sub

Private Sub Time_Implicit(cmdCommand As Control, nNumber As Integer)
    cmdCommand.Caption = nNumber
End Sub
```

4. Thi hành ứng dụng bằng cách nhấn ***F5***.
5. Khi biểu mẫu xuất hiện, nhấn nút ***Explicit***. Chương trình báo thời gian hiển thị liên tục 15000 tiêu đề khác nhau trên nút lệnh. Nút lệnh là một biến đối tượng được khai báo tường minh.
6. Nhấn chuột trên nút ***Implicit***. Mọi việc xảy ra tương tự, nhưng lần này nút lệnh là một biến đối tượng được khai báo ẩn.

Để ý bạn sẽ thấy nhấn trên ***Implicit*** chậm hơn nhấn trên ***Explicit*** khoảng 10%.

10.3 Tập hợp

Mỗi biểu mẫu trong ứng dụng có một tập hợp các điều khiển nội tại. Trong lúc thi hành, ta có thể dùng tập hợp để truy cập đến điều khiển trên biểu mẫu. Ta không cần biết tên của mỗi điều khiển, thậm chí kiểu điều khiển. Khác với mảng điều khiển, ta không cần khai báo, và tất cả điều khiển trên biểu mẫu được tự động xem như một phần của biểu mẫu. Khi thêm hay xoá một điều khiển trên biểu mẫu, VB tự động quản lý việc thêm hay xoá điều khiển trong tập hợp.

Truy cập phần tử trong tập hợp tương tự truy cập phần tử trong mảng thông thường.

Tập hợp rất tiện lợi cho các biểu mẫu nhập liệu. Ví dụ, ta có thể viết một thủ tục chung để duyệt qua tập hợp này và tìm kiếm chỉ những điều khiển dữ liệu và sau đó gán cho thuộc tính CSDL của chúng một đường dẫn đến tập tin CSDL của khách hàng.

Khác với mảng điều khiển, tập hợp điều khiển không hỗ trợ các sự kiện. Tuy nhiên, từng điều khiển trong tập hợp đều có những thuộc tính, phương thức và sự kiện.

10.3.1 Thuộc tính Controls

Thuộc tính ***Controls*** của biểu mẫu chỉ được truy cập thông qua chương tình. Nó thực chất là một mảng các biến đối tượng, trong đó, mỗi phần tử của mảng là một điều khiển đơn: phần tử số 0 là điều khiển đầu tiên trên biểu mẫu, phần tử số 1 là điều khiển thứ 2...

Số thứ tự trong mảng được gán tự động khi ta vẽ điều khiển lên biểu mẫu trong lúc thiết kế. Nếu có 2 hộp văn bản trên biểu mẫu, ta có thể đổi thuộc tính Text của từng điều khiển như sau:

```
Form1.Controls(0).Text=”Control 0”
Form1.Controls(1).Text=”Control 1”
```

Cú pháp:

<tên biểu mẫu>.Controls(<số thứ tự>).<thuộc tính>=<giá trị>

10.3.2 Xác định điều khiển trên biểu mẫu

Mảng ***Controls*** có một thuộc tính gọi là ***Count***, xác định số điều khiển trên biểu mẫu. Lưu ý rằng mảng được đánh số từ 0. Nếu ***Count*** là 3, nghĩa là các phần tử sẽ đánh số lần lượt là 0,1,2.

Ta có thể dùng ***TypeOf*** để xử lý với nhóm các điều khiển tương tự. Mặc dù không chỉ ra chính xác một phần tử, nhưng ta có thể xác định các điều khiển cùng kiểu.

```
For cControlNo = 0 To Form1.Controls.Count –1
 If TypeOf  Form1.Controls(nControlNo) Is TextBox Then
:
:
End if
Next
```

Ta duyệt qua từng phần tử của tập hợp trên Form1 từ 0 đến phần tử cuối cùng, ***Count –1***. Với từng điều khiển, ta dùng ***TypeOf*** để kiểm tra xem nó có phải là hộp văn bản hay không.

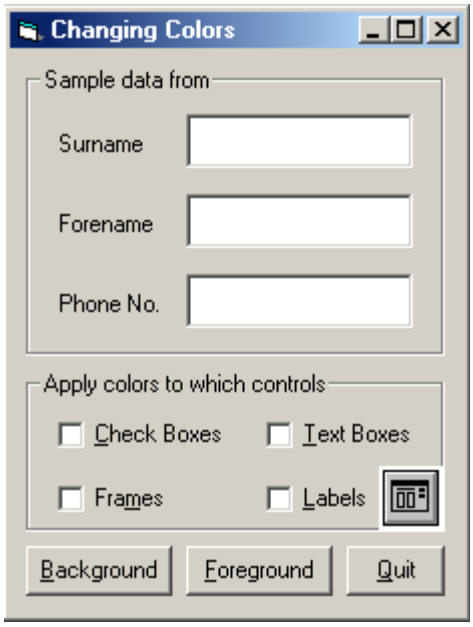
Tuy nhiên, làm việc trên chỉ mục hơi thô thiển, ta dùng cách khác trực quan hơn:

```
For Each objControl In Form1.Controls
 If TypeOf  objControl Is TextBox Then
:
:
End if
Next
```

Ví dụ mẫu -Đổi màu

Dùng mảng điều khiển để đổi màu các điều khiển trên biểu mẫu.

1. Tạo đề án mới và vẽ các điều khiển lên biểu mẫu như sau:



Lưu ý rằng ta cần đặt 2 ***điều khiển khung*** lên biểu mẫu, sau đó thêm các điều khiển khác lên trên. Khi đó, mỗi lần di chuyển khung, điều khiển cũng sẽ di chuyển theo.

2. Đặt tên biểu mẫu là *frmColors* và tên nút lệnh lần lượt là *cmdBackground*, *cmdForeground, cmdQuit*. Tên hộp đánh dấu lần lượt là *chkCheckBoxes, chkFrames, chkTextBoxes* và *chkLabels*.
3. Thêm một điều khiển hộp thoại thông dụng vào biểu mẫu và đặt tên là *dlgColors*.

Lưu ý rằng nếu điều khiển chưa có trong hộp công cụ, chọn ***Components***... từ menu ***Project***, sau đó, đánh dấu chọn *Microsoft Common Dialog Control 6.0*.

4. Viết chương trình cho nút Background:

```
Private Sub cmdBackground_Click()
   Dim nColor As Long
   Dim FormControl As Control
   On Error GoTo BackcolorError
   dlgColors.CancelError = True
   dlgColors.Flags = &H1&
   dlgColors.ShowColor
   nColor = dlgColors.Color
   For Each FormControl In frmColors.Controls
            If TypeOf FormControl Is TextBox And chkTextBoxes.Value = 1 Then
FormControl.BackColor = nColor
              If TypeOf FormControl Is Frame And chkFrames.Value = 1 Then
FormControl.BackColor = nColor
               If TypeOf FormControl Is Label And chkLabels.Value = 1 Then
FormControl.BackColor = nColor
          If TypeOf FormControl Is CheckBox And chkCheckBoxes.Value = 1 Then
FormControl.BackColor = nColor
   Next
Exit Sub
BackcolorError:
```

```
   MsgBox ("You pressed the cancel button.")
End Sub
```

5. Chương trình cho nút Foreground:

```
Private Sub cmdForeground_Click()
   Dim nColor As Long
   Dim FormControl As Control
   On Error GoTo ForecolorError
   dlgColors.CancelError = True
   dlgColors.Flags = &H1&
   dlgColors.ShowColor
   nColor = dlgColors.Color
   For Each FormControl In frmColors.Controls
          If TypeOf FormControl Is TextBox And chkTextBoxes.Value = 1 Then
FormControl.ForeColor = nColor
            If TypeOf FormControl Is Frame And chkFrames.Value = 1 Then
FormControl.ForeColor = nColor
             If TypeOf FormControl Is Label And chkLabels.Value = 1 Then
FormControl.ForeColor = nColor
        If TypeOf FormControl Is CheckBox And chkCheckBoxes.Value = 1 Then
FormControl.ForeColor = nColor
   Next
Exit Sub
ForecolorError:
   MsgBox ("You pressed the cancel button.")
End Sub
```

6. Viết chương trình cho nút Quit:

```
Private Sub cmdQuit_Click()
   ' Quit the application by unloading the form
   Unload frmColors
End Sub
```

7. Thi hành chương trình. Chọn vào hộp đánh dấu để chọn kiểu điều khiển ta muốn đổi màu, nhấn nút ***Background*** hay ***Foreground*** để thi hành việc đổi. Một hộp thoại màu xuất hiện cho phép ta chọn màu.

Thuộc tính ***CancelError*** của điều khiển hộp thoại thông dụng được quy định là True, nghĩa là lỗi số 32755 được phát sẽ gửi chương trình trực tiếp bẫy lỗi.

Vòng lặp For Each...Next duyệt qua từng điều khiển trong tập hợp, đặt từng phần tử vào biến đối tượng ***FormControl*** Dùng ***TypeOf*** để kiểm tra kiểu điều khiển. Lưu ý dùng từ khoá ***Is*** với ***TypeOf*** thay vì dùng dấu kiểm tra bằng.

10.4 Biểu mẫu MDI

Biểu mẫu MDI cho phép nhóm các biểu mẫu và chức năng trong một cửa sổ lớn. Tuy nhiên, biểu mẫu MDI có một số nhược điểm: chỉ có một vài điều khiển được vẽ trên biểu mẫu MDI. Đó là điều khiển định giờ và hộp hình. Trong phiên bản Professional và Enterprise ta có thể vẽ thêm thanh trạng thái và thanh công cụ. Hộp hình vẽ trong biểu mẫu MDI luôn có cùng bề rộng với biểu mẫu và tự động được đặt ở phần trên cùng hoặc dưới cùng của biểu mẫu. Ta không thể điều chỉnh bằng tay. Nếu ta cố canh trái hoặc canh phải, hộp hình sẽ chiếm toàn bộ biểu mẫu MDI.

10.4.1 Biểu mẫu con (Child Form)

Thuộc tính MDIChild của một biểu mẫu là một giá trị True/False cho biết biểu mẫu có phải là biểu mẫu con trong một biểu mẫu MDI hay không. Bởi vì VB chỉ cho phép tồn tại một biểu mẫu MDI trong ứng dụng, biểu mẫu con tự động nhận biết cửa sổ cha và khi thi hành, nó chỉ hoạt động bên trong cửa sổ cha.

Vào lúc thiết kế, không thể phân biệt cửa sổ độc lập với cửa sổ con, chỉ khác nhau ở chỗ thuộc tính ***MDIChild*** mà thôi. Thuộc tính này không gán được vào lúc thi hành, nếu không, ta sẽ nhận thông báo lỗi trước khi chương trình treo.

Ví dụ mẫu - Cửa sổ con

1. Tạo đề án mới và đặt tên biểu mẫu mặc định là frmChild. Đổi thuộc tính MDIChild thành True.
2. Từ menu Project, chọn Add MDI Form để tạo một cửa sổ MDI và đặt tên cho nó là frmParent.
3. Thêm menu cho biểu mẫu MDI gồm 2 mục: New và Exit. Đặt tên cho chúng là mnuFNew và mnuFExit.
4. Thêm menu cho cửa sổ con bao gồm: File, Edit, View, Options.
5. Viết chương trình cho menu New

```
Private Sub mnuFNew_Click()
 Load frmChild
End Sub
```

6. Từ menu Project, chọn Project1 Properties và chọn biểu mẫu khởi động là biểu mẫu MDI.
7. Thi hành ứng dụng. Khi mới xuất hiện, cửa sổ MDI chưa có cửa sổ con và hiển thị menu của chính nó. Nếu ta chọn New từ menu File, cửa sổ con hiển thị. menu của biểu mẫu MDI được thay thế bằng menu của cửa sổ con. Trạng thái đầu của cửa sổ MDI sẽ được phục hồi toàn bộ của sổ con bị tắt.
8. Lưu đề án với tên ***MDIChild.vbp***.

10.4.2 Tạo Instance của biểu mẫu

Sử dụng biến đối tượng để tạo ra những bản sao của một biểu mẫu. Từng bản sao có các điều khiển và menu như nhau, nhưng có những dữ liệu khác nhau. Mặc dù chương trình cũng như tên biến và tên điều khiển như nhau, nhưng dữ liệu được chứa ở những nơi khác nhau trong bộ nhớ.

Ví dụ mẫu - Tạo Instance của biểu mẫu

1. Mở lại đề án MDIChild.vbp. Chon biểu mẫu frmParant.
2. Chọn New từ menu File của biểu mẫu MDI. Mở cửa sổ Code và đưa đoạn chương trình sau vào:

```
Private Sub mnuFNew_Click()
   Dim OurNewForm As New frmChild
   OurNewForm.Show
End Sub
```

3. Đoạn chương trình trên dùng biến đối tượng để tạo một instance cho cửa sổ frmChild.
4. Xoá toàn bộ menu của cửa sổ con.

5. Thi hành chương trình. Mỗi lần nhấn New, một cửa sổ mới được tạo.
6. Lưu đề án với tên mới bằng cách chọn Save File Form As... và Save Project As... từ menu File. Đặt tên là MDIChild1.vbp

10.4.3 Xác định biểu mẫu

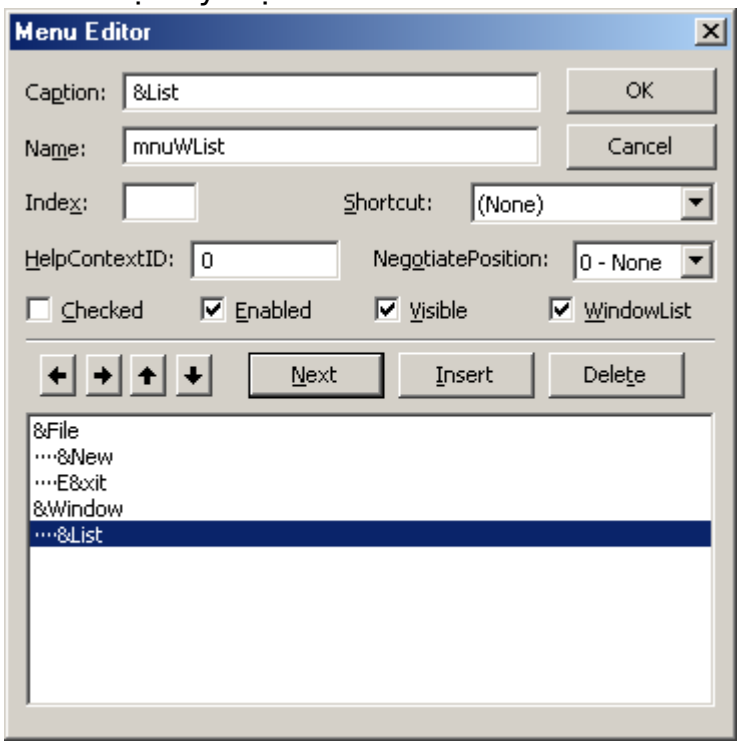
Vì ta có thể tạo ra 10 biểu mẫu đồng nhất có cùng tên, nên việc xác định cửa sổ là cần thiết. Từ khoá Me cho phép ta tham chiếu đến cửa sổ hiện hành, là cửa sổ đang có focus, hay nói cách khác, là cửa sổ nhận được mọi phím nhấn hay click chuột bất kỳ.

Ta có thể dùng:
activeform.txtEmployee.text = "Peter"
nhưng dùng me là cách thông dụng nhất.

10.4.4 Tạo danh sách cửa sổ

Ví dụ mẫu - Tạo danh sách cửa sổ

1. Mở đề án MDIChild1.vbp. Chọn hiển thị biểu mẫu frmParent
2. Tạo tuỳ chọn Window trên menu



Đưa đoạn chương trình sau vào:

```
Private Sub mnuWArrange_Click()
   frmParent.Arrange vbArrangeIcons
End Sub

Private Sub mnuWCascade_Click()
   frmParent.Arrange vbCascade
End Sub
```

```
Private Sub mnuWTile_Click()
   frmParent.Arrange vbTileHorizontal
End Sub
```

3. Thi hành ứng dụng với các menu được tạo.

Sắp xếp cửa sổ

Dùng phương thức ***Arrange*** với biểu mẫu MDI để sắp xếp các cửa sổ con. Các hằng nội tại sau đây là các kiểu sắp xếp của sổ do VB cung cấp:

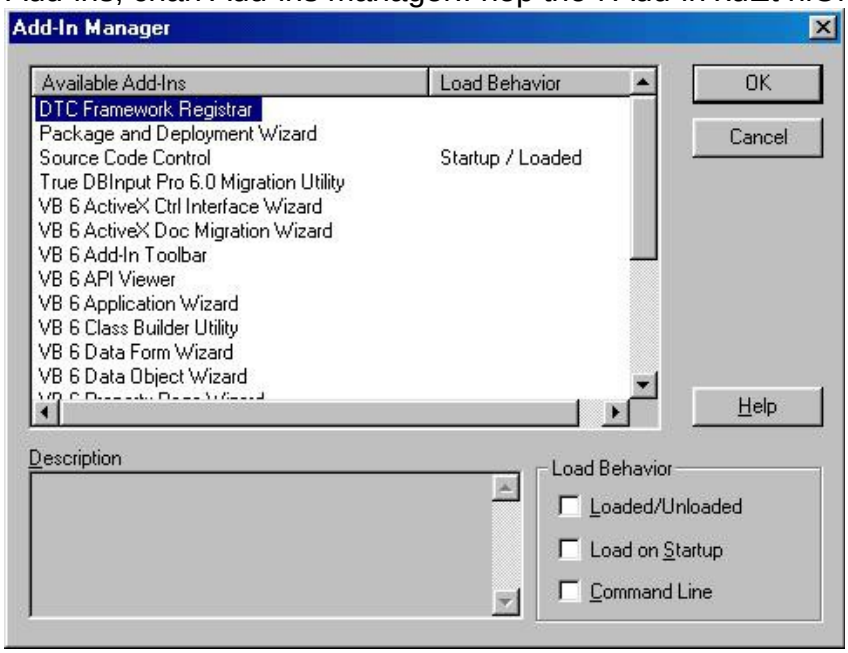
Giá trị	Hằng	Mô tả
0	vbCascade	Xếp các cửa sổ con theo kiểu thác nước trải từ góc trái trên qua góc bên phải dưới.
1	vbTileHorizontal	Dàn đều các cửa sổ con sao cho chúng chia màn hình thành những dải ngang.
2	vbTileVertical	Dàn đều các cửa sổ con sao cho chúng chia màn hình thành những dải dọc.
3	vbArrangeIcons	Các cửa sổ con được thu nhỏ thành những biểu tượng và được xếp thẳng hàng.

11 Công cụ trong VB6

§Ó t×m hiÓu vÒ c¸c c«ng cô trong VB chóng ta sÏ ®Ò cËp tíi
Add-in
C¸c c«ng cô trong add-in
Tr×nh ®ãng gãi vµ triÓn khai øng dông
Visual Basic cho phÐp n¹p vµ gì bá c¸c Add-in dïng ®Ó më réng m«i trêng trong ph¸t triÓn Visual Basic.

11.1 ADD-INS

Tõ menu Add-ins, chän Add-ins manager.. hép tho¹i Add-in xuÊt hiÖn



11.2 Các công cụ trong ADD-INS

11.2.1 Trình cài đặt ứng dụng

Lµ Tr×nh ®ãng gãi vµ triÓn khai øng dông sÏ ®îc nãi râ ë phÇn sau

11.2.2 Trình đối tượng dữ liệu tự động

ChØ cã trªn phiªn b¶n **Professional** vµ **Enterprise** cña VB 6
Tr×nh ®èi tîng d÷ liÖu (**Data Object Wizard**) tù ®éng t¹o ra c¸c ®èi tîng tÇng gi÷a (middle-tier object) r»ng buéc víi m«i trêng d÷ liÖu (**Data Environment**) hoÆc c¸c **UserControl.**
Nã cho phÐp ph¸t sinh c¸c ®o¹n ch¬ng tr×nh ®Ó t¹o ra nguån d÷ liÖu hiÖu chØnh vµ **User Control** ®Ó hiÓn thÞ vµ thao t¸c c¸c d÷ liÖu th«ng qua c¸c thñ tôc ®· lu tr÷.
Lu ý r»ng, tríc hÕt ta ph¶i t¹o mét **Data Environment** víi c¸c lÖnh ®Ó lÊy vÒ hoÆc thao t¸c trªn d÷ liÖu tríc khi dïng Wizard nµy.
C¸c lÖnh nµy bao gåm :

Dßng lÖnh **SELECT**
C¸c lÖnh tuú chän nh **INSERT, UPDATE, DELETE**
Add-In ®îc dïng khi ta cÇn :

- T¹o c¸c Recordset cho phÐp cËp nhËt tõ c¸c thñ tôc ®· lu tr÷
- T¹o c¸c User Control ®Ó hiÓn thÞ vµ thao t¸c víi CSDL
- Tù ®éng ph¸t sinh ch¬ng tr×nh cña VISUAL BASIC ®Ó thÓ hiÖn mèi liªn hÖ gi÷a c¸c d÷ liÖu.
- T¹o c¸c ®iÒu khiÓn ®Ó cho phÐp hiÓn thÞ vµ thao t¸c víi c¸c mèi liªn hÖ t×m kiÕm.
- Dïng m« t¶ b»ng v¨n b¶n dÔ hiÓu thay v× mét gi¸ trÞ khã hiÓu.
- Cã m« t¶ v¨n b¶n dÔ hiÓu cho gi¸ trÞ Null.

11.2.3 Trình xây dựng dữ liệu tự động

ChØ cã trªn phiªn b¶n **Professional vµ Enterprise** cña VB 6

Tr×nh x©y dùng biÓu mÉu d÷ liÖu (**Data Form Wizard**) ®îc kÕt hîp víi tr×nh t¹o øng dông (**Application wizard**), t¹o biÓu ®å (**Chart**) vµ líi (**FlexGrid**). Wizard nµy ®îc thiÕt kÕ ®Ó tù ®éng ph¸t sinh c¸c biÓu mÉu Visual Basic chøa c¸c ®iÒu khiÓn r»ng buéc d÷ liÖu vµ c¸c thñ tôc dïng qu¶n lý th«ng tin dÉn xuÊt tõ c¸c b¶ng vµ c¸c c©u truy vÊn.
Ta cã thÓ dïng Wizard ®Ó t¹o c¸c biÓu mÉu tõ mét b¶ng hay mét c©u truy vÊn, hoÆc mét biÓu mÉu kiÓu **Master/Detail** chøa c¸c mèi quan hÖ phøc t¹p lo¹i mét – nhiÒu. NÕu sö dông ®iÒn khiÓn ta cã thÓ dïng biÓu mÉu kiÓu líi hay kiÓu m¶ng (datasheet) . Wizard chØ ®îc dïng kÕt nèi víi kiÓu ADO.

Dïng Wizard ®Ó :

- ThiÕt kÕ nhanh chãng c¸c biÓu mÉu víi c¸c ®iÒu khiÓn r»ng buéc víi nguån d÷ liÖu.
- T¹o c¸c biÓu mÉu Master/Detail kiÓu mét mÈu tin, kiÓu líi.
- T¹o nhanh chãng c¸c khu«n mÉu cho biÓu mÉu dùa trªn d÷ liÖu.

11.2.3.1 Tạo biểu mẫu với một bảng hay kiểu lưới (hay datasheet)

X¸c ®Þnh lo¹i c¬ së d÷ liÖu : Lµ **ODBC** hay **Access**

Chän tªn tÖp tin c¬ së d÷ liÖu vµ kiÓu rµng buéc ta muèn sö dông trªn biÓu mÉu. Bíc nµy dïng cho nh÷ng c¬ së d÷ liÖu kh«ng ph¶i **ODBC**.

Chän kiÓu biÓu mÉu : Cã nhiÒu kiÓu :
Mét mÈu tin : ë mét thêi ®iÓm, chØ mét mÈu tin hiÓn thÞ. §©y lµ mÆc ®Þnh.
B¶ng d÷ liÖu (**Datasheet**) :BiÓu mÉu hiÓn thÞ nh÷ng trêng ®îc chän theo d¹nh b¶ng d÷ liÖu (**Datasheet**) dïng ®iÒu khiÓn **DataGrid**.
Master / Detail : MÈu tin **Master** cã ®Þnh d¹ng lµ mét mÈu tin vµ mÈu tin **Detail** lµ mét b¶ng d÷ liÖu (**Datasheet**). Khi d÷ liÖu trong **Master** thay ®æi, d÷ liÖu trong **Detail** tù ®éng thay ®æi theo do liªn kÕt gi÷a 2 phÇn nµy.

MS Hflex Grid: BiÓu mÉu hiÓn thÞ d÷ liÖu xÕp theo b¶ng.
MS Chart: BiÓu mÉu hiÓn thÞ d÷ liÖu theo biÓu ®å.

Chän r»ng buéc
- §iÒu khiÓn ADO
- Dïng ch¬ng tr×nh cña ADO
- Dïng líp d÷ liÖu (Data class)
- Chän nguån cho mÈu tin

Chän ®iÒu khiÓn ta muèn xuÊt hiÖn trªn biÓu mÉu vµ cho phÐp ***Wizard*** t¹o ch¬ng tr×nh cho chóng. §ã lµ c¸c nót ***Add, Update, Edit, Refresh, Close, Show Data Control.***
KÕt thóc.

11.2.3.2 Tạo biểu mẫu Master/ Detail

- X¸c ®Þnh c¬ së d÷ liÖu lµ ODBC hay Access
- Chän biÓu mÉu (t¬ng tù phÇn t¹o biÓu mÉu tríc)
- X¸c ®Þnh nguån d÷ liÖu cho phÇn *Master*
- X¸c ®Þnh nguån d÷ liÖu cho phÇn Detail
- X¸c ®Þnh d÷ liÖu cho kÕt nèi mét-nhiÒu gi÷a *Master* vµ *Detail*.
- Chän ®iÒu khiÓn ta muèn xuÊt hiÖn trªn biÓu mÉu vµ cho phÐp Wizard t¹o ch¬ng tr×nh cho chóng. §ã lµ c¸c nót ***Add, Update, Edit, Refresh, Close, Show data control.***
- KÕt thóc

Biểu mẫu chứa điều khiển dữ liệu ADO

Thu thËp c¸c th«ng tin cÇn thiÕt ®Ó kÕt nèi víi mét nguån d÷ liÖu **ODBC** vµ cho phÕp chän kiÓu rµng buéc. Bíc nµy chØ xuÊt hiÖn nÕu ta chän Remote **ODBC** ë bíc ®Þnh d¹ng c¬ së d÷ liÖu.

11.2.4 Trình thiết kế Add-ins tự động

Có trên mọi phiên bản của VB 6
Trình tạo ứng dụng tự động (Application Wizard) cho phép ta lưu các chọn lựa thành một profile để dùng lại về sau, cho phép tạo nhiều ứng dụng với cùng một định dạng. Ta còn có thể phóng Data form Wizard và Toolbar Wizard từ trong Trình tạo ứng dụng tự động để tạo các biểu mẫu dữ liệu và thanh công cụ. Menu giờ đây hoàn toàn có thể được hiệu chỉnh.

11.2.5 Trình thiết kế tự động

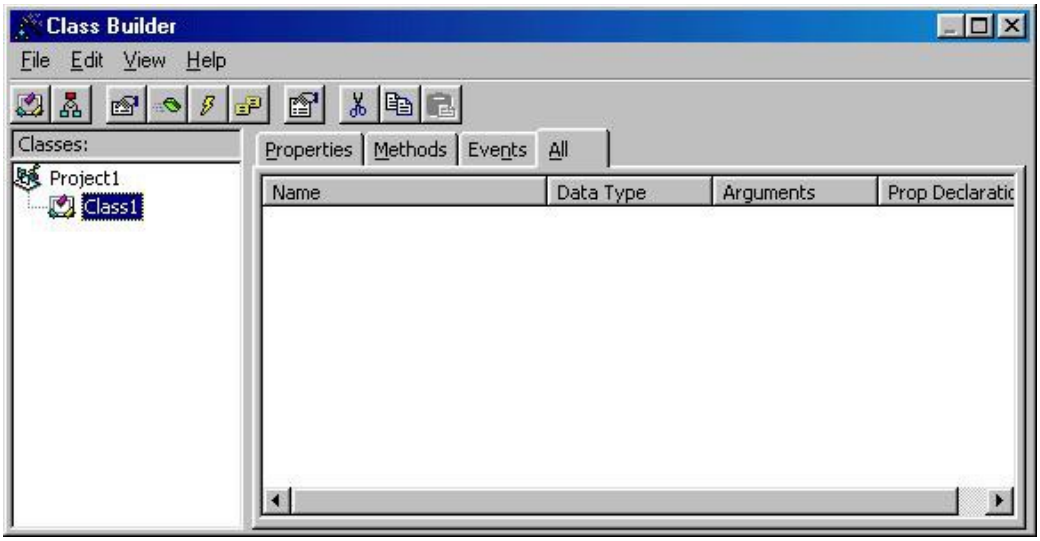
Chỉ có trên phiên bản **Professional và Enterprise** của VB 6.
Dùng trình thiết kế **Add-in** tự động (**Add-in Designer**) để bắt đầu quy trình lập trình cho một **Add-in** bằng cách chỉ ra cách nạp mặc định tên, mô tả, ứng dụng sau cùng, và các phiên bản. Trình thiết kế hỗ trợ một số đoạn chương trình cho các tập tin DLL hay EXE để đăng ký **Add-in** cho ứng dụng sau cùng.

11.2.5.1 Khởi tạo một add-in mới

Tạo một **Add-is** bằng cách chọn **New Project** từ menu File. Sau đó, chọn **Add-in** trong hộp hội thoại **New Project.**
Trên Tab General, đưa vào các thông tin cơ bản về mô tả add-in, các thức nạp và cho biết ứng dụng nào quản lý nó.
Chọn Tab **Advanced** để đưa vào những thông tin bổ sung về các tài nguyên và các giá trị trong **Registry**.
Để thêm chương trình cho add-in, nhấn đúp chuột lên **Add-in Designer**.
Để biên dịch add-in, chọn Meke exe hay Make add từ menu File

11.2.6 Tiện ích xây dựng lớp

Chỉ có trên phiên bản **Professional và Enterprise** của VB.



Tiện ích xây dựng lớp
Tiện ích xây dựng lớp (**Class Buider Utility**) hỗ trợ ***paramArray***, ***Optional***, ***ByVal*** và các giớ trị **Default** trong danh sách tham số và hỗ trợ các Enums. Tiện ích tổ chức các lớp theo phân nhánh trong một đề án của **Visual basic**. Nó theo dõi các lớp và phát sinh các đoạn chương trình đơn giản để thao tác trên các lớp, bao gồm thuộc tính, phương thức, sự kiện và enum.
Nhấn lên một lớp và tập hợp cho phép sửa đổi nó. Tương ứng với lớp được chọn bên trái và danh sách các thuộc tính, phương thức và sự kiện xuất hiện bên phải. Enum hiển thị trong Tab All.

11.2.7 Trình tạo thanh công cụ tự động

Có trên mọi phiên bản của VB 6.
Khi ta nạp tạo ứng dụng tự động (**Application Wizard**), trình thanh công cụ tự động (**Toolbar Wizard**) tự động mở khi ta muốn thêm một thanh công cụ có thể hiệu chỉnh vào biểu mẫu.
Ta có thể:

- Tạo một thanh công cụ.
- Đổi thứ tự các nút nhấn.

- Thêm Bitmap hay biểu tượng mà ta muốn thêm chức năng.

11.3 Trình đóng gói và triển khai ứng dụng

11.3.1 Phát hành ứng dụng

Sau khi viết xong một chương trình Visual Basic ta cần phát hành nó. Ta có thể phát hành qua đĩa, CD, qua mạng, Internet hay Intranet.
Có 2 bước để thực hiện việc phát hành :

- **Đóng gói** : đóng gói các tập tin của chương trình thành những tập tin.CAB để có thể triển khai chúng ở những nơi được chọn, và ta phải tạo chương trình cài đặt tương ứng với kiểu đóng gói.
- **Triển khai**: Chuyển ứng dụng đã được đóng gói đến nơi mà người sử dụng có thể cài đặt từ đó. Điều này có nghĩa là sao chép phần đóng gói xuống đĩa mềm, ổ mạng hay triển khai trên Web site nào đó.

11.3.2 Trình đóng gói và triển khai ứng dụng

Trình đóng gói và triển khai ứng dụng (**Package and deployment Wizard**) có sẵn trên mọi ấn bản VB6.
Trong các phiên bản cũ, nó là **Setup Wizard**, cho phép triển khai các tập tin .CAB lên Web server. Ổ đĩa mạng hay những thư mục khác. Giờ đây, Trình đóng gói và triển khai ứng dụng hỗ trợ cả ADO, OLE DB, RDO, ODBC và DAO cũng như các ứng dụng IIS và HTML. Nó cũng xử lý tốt hơn các nhóm trong menu Start và các biểu tượng cho chương trình cài đặt. Nó có thể được chay trong VB như một Add-in, hay trên dòng lệnh với tập tin.BAT.
Ngoài ra, ta có thể dùng **Setup Toolkit** (được cung cấp khi ta cài đặt VB) để hiệu chỉnh các htể hiện trong quy trình cài đặt.

11.3.3 Mở trình đóng gói và triển khai trong VB

Mở đề án ta muốn phát hành

> Lưu ý rằng nếu bạn đang làm việc với một nhóm các đề án hoặc có nhiều đề án đang được nạp, bạn phải bảo đảm rằng đề án đem đóng gói là đề án hiện hành trước khi mở Wizard.

Dùng Add-in Manager để nạp trình đóng gói và triển khai ứng dụng, nếu cần : Từ menu Add-ins, chọn **Add-in Manager**, chọn **Package and Deployment Wizard** từ danh sách, nhấn **OK**.

Chọn **Package and Deployment Wizard** từ menu **Add-ins** để phóng Wizard.
Trên màn hình chính chọn một trong các tuỳ chọn sau:

- Nếu ta muốn tạo một đóng gói chuẩn, đóng gói kiểu Internet hay các tập tin liên quan cho ứng dụng, chọn **Package**.
- Nếu muốn triển khai ứng dụng, chọn **Deploy**.
- Nếu muốn xem, soạn thảo, hay xoá kịch bản, chọn **Manager Scripts**.
- Thực hiện lần lượt qua các màn hình của Wizard.

11.3.4 Mở trình đóng gói và triển khai như một ứng dụng độc lập.

Nếu đề án ta muốn đóng gói đang mở,lưu nó lại và thoát khỏi Visual Basic.

Nhấn nút Start, chọn vào Package and Deployment Wizard từ nemu con của VB. Trong danh sách Project của màn hình khởi tạo, chọn đề án ta muốn đóng gói.

Lưu ý : Ta có thể nhấn nút Browse nếu đề án chưa có trong danh sách.
Trên màn hình chính, chọn một trong những tuỳ chọn sau :
Nếu muốn tạo một đóng gói chuẩn, đóng gói kiểu Internet, hay các tập tin kiên quan, chọn Package.

Nếu muốn triển khai ứng dụng, chọn Deploy.
Nếu muốn xem, soạn thao, hay xoá kịch bản, chọn Manage Scripts.
Thực hiện lần lượt qua các màn hình của Wizard.

11.3.5 Thi hành Wizard dưới chế độ silent

Sử dụng kịch bản ta có thể đóng gói và triển khai ứng dụng dưới chế độ **Silent**. Trong đó, Wizard tự động thi hành và ta không cần tương tác để chọn lựa hay di chuyển trên màn hình. Nó sử dụng các chọn lựa chứa trong kịch bản.
Chế độ Silent đặc biệt hữu dụng nếu ta đóng gói và triển khai ứng dụng bằng tập tin. BAT. Ta có thể dùng nó để kiểm tra kết quản với một thưu mục tạm thời.
Mở dấu nhắc DOS

Nhập vào tên tập tin thi hành (. EXE) của trình Wizard, pdcmdln. Exe, kế tiếp là đường dẫn và tên tập tin của đề án VB, các đối dòng lệnh tương ứng. Ví dụ :
pdcmdln.Exe c:\project1\project1.vbp /p “Internet Package “ /d Deploymentl /L “c:\project1\Silent mode.log “
Lưu ý : bạn chó thể dùng cả /p và / d để thi hành trong chế độ Silent. Nếu không, dùng một trong hai.

Tham số	Mô tả
/p **Packageingscript**	Theo sau /p là tên kịch bản đóng gói chứa các lựa chọn để thi hành trong chế độ silent.
/d **Deploymentscript**	Theo sau /d là tên kịch bản đóng gói chứa các lựa chọn để thi hành trong chế độ silent.
/l **Path**	Wizard sẽ chứa tất cả kết quả của nó chẳng hạn như là thông báo lỗi, báo cáo thành công vào một tập tin thay vì hiển thị ra màn hình. Sau /l là đường dẫn và tên của tập tin đó . Nếu tập tin này chưa có. wizard tự động tao ra.

Lưu ý : Tên tập tin hoặc tên kịch bản có chứa khoảng trắng đặt trong dấu trích dẫn (dấu nhảy kép).

11.3.6 Setup toolkit

Setup toolkit là một đề án cài đặt với VB và được sử dụng bởi Trình đóng gói và triển khai khi nó tạo chương trình **setup**. Để án **Setup toolkit** chứa các biểu mẫu và chương trình mà chương trình **setup** dùng cài đặt tập tin cho người sử dụng. Khi ta

dùng Trích đóng gói và triển khai. **Wizard** bao gồm setup1. exe mà đề án setup toolkit tạo ra. Tập tin này được dùng làm một tập tin cài đặt chính.
Setup toolkit còn dùng để điều chỉnh các màn hình hiển thị trong quá trình cài đặt nếu ta cần thêm những tính năng không cấp sẵn bởi Wizard.
Setup Toolkit chứa trong thư mục con ***\Wizards\PDWizard\Setup1*** của thư mục cài đặt VB.
Thận trọng: Các tập tin trong đề án này sẽ được sử dụng bởi kết quả của trình đóng gói và triển khai. Trước khi sửa đổi cần phải sao một bản dự phòng trong một thư mục khác. Nếu bạn sửa đổi Setup1. exe, chương trình setup được tạo bởi Trình đóng gói và triển khai sẽ dùng bản chỉnh sửa này thay vì bản gốc.
Sử dụng Setup Toolkit bằng các nạp đề án Setup1. vbp vào **Visual Basic** và tiến hành sửa đổi trên cách hiển thị cũng như tính năng của đề án. Khi làm việc này, ta cần lần lượt đi qua từng bước.

11.3.6.1 Các bước sửa đổi trình đóng gói và triển khai

Khi muốn sửa đổi **Setup Toolkit** nhằm thay đổi kết quả tạo ra bởi Trình đóng gói và triển khai, ta làm như sau :
Sửa đề án **Setup Toolkit** để chứa các lời nhắc, màn hình, chức năng chương trình hay những thông tin khác. Khi hoàn tất, biên dịch đề án để tạo setup1. exe.
Thi hành trình đóng gói, là theo các lời nhắc trên màn hình để tạo ra môi trường phát hành.

11.3.6.2 Các bước tạo một chương trình Setup hiệu chỉnh

Khi muốn tạo chương trình setup một cách thủ công dùng setup Toolkit thay vì Trình đóng gói và triển khai, ta làm như sau:

1. Nếu cần sửa đề án Setup Toolkit để chứa các lời nhắc, màn hình, chức năng, chương trình và các thông tin khác.
2. Xác định tập tin nào cần phân phát, bao gồm các tập tin thi hành, cài đặt và các tập tin liên quan.
3. Xác định thư mục cài đặt trên máy người dùng
4. Tạo thủ công tập tin setup.lst để đưa ra tên và thư mục cài đặt của tất cả các tập tin chứa trong đề án.
5. Xác định cách thức cài đặt tập tin.
6. Tạo tệp tin.CAB cho đề án, dùng tiện ích **Makecab.**

Mẹo: Bạn có thể dùng Trình đóng gói và triển khai để tạo tập tin.CAB sau đó sửa chúng bằng tay. Khi Wizard tạo một tập tin.CAB, nó tạo một tập tin. DDF và một tập tin.BAT trong thư mục con \Support của thư mục đề án. Để sửa đổi tập tin.CAB, sửa trên tập tin. DDF, sau đó chạy tập tin.BAT. Tập tin.BAT sẽ chạy Makecab.exe để tạo lại tập tin.CAB.

7. Tạo setup1.exe cho đề án bằng cách biên dịch dùng **Setup Toolkit**.
8. Sao chép tập tin vào môi trường phân phát, hoặc đưa lên Web side dùng trình phát hành Web (Web publishing Wizard)

11.4 Bài tập

Dùng trình tạo ứng dụng tự động tạo ứng dụng kiểu Explorer.
Dùng trình xây dựng dữ liệu tự động tạo biểu mẫu **Master/Detail**.

12 Những khái niệm cơ bản về CSDL

- Cơ sở dữ liệu là gì ?
- Sử dụng cửa sổ xem dữ liệu
- Tạo trình thiết kế môi trường dữ liệu
- Sử dụng các điều khiển dữ liệu để tạo giao diện người sử dụng

Cơ sở dữ liệu là phần cốt lõi của nhiều ứng dụng phần mềm kinh doanh. Cơ sở dữ liệu rất phổ biến trong thế giới kinh doanh vì chúng cho phép truy cập tập trung đến các thông tin theo một các nhất quán, hiệu quả và tương đối dễ dàng cho việc phát triển và bảo trì. Chương này tìm hiểu về các khái niệm cơ bản để thiết lập và bảo trì một cơ sở dữ liệu cho một doanh nghiệp, bao gồm cơ sở dữ liệu là gì, tại sao cơ sở dữ liệu hữu dụng và dùng cơ sở dữ liệu như thế nào để tạo ra các giải pháp cho doanh nghiệp.

Nếu bạn đã từng dùng Visual Basic và từng lập trình với cơ sở dữ liệu. Bạn sẽ thấy chương trình này khá cơ bản, tuy nhiên nó sẽ giúp bạn có được một nền tảng tốt của một hệ quản trị cơ sở dữ liệu nói chung.

Mặc dù các khái niệm cơ sở dữ liệu gần như tương tự giữa các hệ quản trị cơ sở dữ liệu, các nhà cung cập các hệ quản trị cơ sở dữ liệu thường có các tên gọi khác nhau cho các sản phẩm riêng của họ. Ví dụ, nhiều nhà lập trình Client / Server đề cập đến truy cấn chứa trong cơ sở dữ liệu như là **View** ; trong khi các nhà lập trình Access và Visual Basic lại gọi truy vấn là **QueryDef**. Cả hai khái niệm này đều là như nhau.

Nếu bạn đã từng dùng phiên bản cũ của VB - nhất là Visual Basic 3, ta cần biết một vài điểm mới trong lập trình cơ sở dữ liệu. Visual Basic chứa phiên bản mới nhất của Bộ máy cơ sở dữ liệu Jet (Visual Basic chia sẻ với Microsoft Access). Phiên bản này của Jet đưa ra các bổ sung cho bộ máy cơ sở dữ liệu sẽ được trình bày trong chương này. Ngoài ra, việc bổ sung **ADO** (Đối tượng dữ liệu ActiveX – ActiveX Data Object), cũng như các công cụ liên quan trong môi trường phát triển, thể hiện những thay đổi cho các nhà lập trình Visual Basic. Nếu đã quen với phát triển dữ liệu 32 bit trong Visual Basic, Bạn có thể nhảy thẳng đến chương "Đối tượng dữ liệu ActiveX – ADO".

12.1 Cơ sở dữ liệu là gì?

Cơ sở dữ liệu là một kho chứa thông tin. Có nhiều loại cơ sở dữ liệu, nhưng ta chỉ đề cập đến cơ sở dữ liệu quan hệ, là kiểu cơ sở dữ liệu phổ biến nhất hiện nay.

Một cơ sở dữ liệu quan hệ:

- Chứa dữ liệu trong các bảng, được cấu tạo bởi các dòng còn gọi là các mẩu tin, và cột gọi là các trường.
- Cho phép lấy về (hay truy vấn) các tập hợp dữ liệu con từ các bảng
- Cho phép liên kết các bảng với nhau cho mục đích truy cập các mẩu tin liên quan với nhau chứa trong các bảng khác nhau.

12.1.1 Bộ máy (Engine) cơ sở dữ liệu là gì?

Chức năng cơ bản của một cơ sở dữ liệu được cung cấp bởi một bộ máy cơ sở dữ liệu, là hệ thống chương trình quản lý cách thức chứa và trả về dữ liệu.

Bộ máy cơ sở dữ liệu trình bày trong tài liệu này là Microsoft Jet, Jet không phải là một thương phẩm, thay vào đó, nó là một hệ thống con được nhiều ứng dụng của Microsoft sử dụng. Microsoft lần đầu tiên đưa bộ máy này vào sử dụng với Visual Basic 3.0 và Access 1. Sau nhiều lần nâng cấp, phiên bản Jet dùng với quyển sách này là Jet 3.51, đi kèm với Visual Basic và Access.

Chú ý : Ngoài Jet, còn nhiều bộ máy cơ sở dữ liệu khác, như vì Visual Basic hỗ trợ Jet một các nội tại nên quyển sách này ưu tiên nói về Jet. Hơn nữa Jet còn hỗ trợ các bộ máy cơ sở dữ liệu khác. Trong chương "Làm quen với SQL Server " giới thiệu một bộ máy hoàn toàn khác SQL Server 6.5

12.1.2 Bản và trường

Các cơ sở dữ liệu được cấu tạo từ các bảng dùng thể hiện các phân nhóm dữ liệu. Ví dụ, nếu ta tạo một cơ sở dữ liệu để quản lý tài khoản trong công việc kinh doanh ta phải tạo một bảng cho khách hàng, một bảng cho Hoá đơn và một bảng cho nhân viên. Bảng có cấu trúc định nghĩa sẵn và chứa dữ liệu phù hợp với cấu trúc này.

Bảng: Chứa các mẩu tin là các mẩu riêng rẽ bên trong phân nhóm dữ liệu.

Mẩu tin: Chứa các môi trường. Mỗi trưởng thể hiện một bộ phận dữ liệu trong một mẩu tin. Ví dụ như mỗi mẩu tin thể hiện một mục trong danh bạ địa chỉ chứa trong trường Tên và họ, địa chỉ, thành phố, tiểu bang, mã ZIP và số điện thoại.

Ta có thể dùng chương trình Visual Basic để tham chiếu và thao tác với cơ sở dữ liệu, bảng, mẩu tin và trường.

12.1.2.1 Thiết kế cơ sở dữ liệu

Để tạo một cơ sở dữ liệu, trước hết ta phải xác định thông tin gì cần theo dõi. Sau đó, ta thiết kế cơ sở dữ liệu, tạo bảng chứa các trường định nghĩa kiểu dữ liệu sẽ có. Sau khi tạo ra cấu trúc cơ sở dữ liệu, tạo bảng chứa các trường định nghĩa kiểu dữ liệu sẽ có. Sau khi tạo ra cấu trúc cơ sở dữ liệu, cơ sở dữ liệu có thể chứa dữ liệu dưới dạng mẩu tin. Ta không thể đưa dữ liệu vào mà không có bảng hay định nghĩa trường vì dữ liệu sẽ không có chỗ để chứa.Do đó, thiết kế cơ sở dữ liệu cực kỳ quan trọng, nhất là rất khó thay đổi thiết kế một khi ta đã tạo xong nó.

Ví dụ ta tạo một bảng sau :

Bảng khách hàng	Bảng tblRegion
tblCustomer	TblRegion
ID	State
FirstName	RegionName
LastName	
Company	
Address	
City	

State	
Zip	
Phone	
Fax	
Email	

Có quan hệ giữa 2 bảng thông qua trường State (Trạng thái). Đây là mối quan hệ một - nhiều, đối với một mẩu tin trong tblRegion, có thể không có, hoặc có nhiều mẩu tin tương ứng trong bảng tblCustomer.

Cụm từ "tbl" thể hiện tên bảng, tên trường hiển thị đầy đủ, không chứa khoảng trắng hay những ký tự đặc biệt khác như dấu gạch dưới.

Bảng hoá đơn :

TblOrder
ID
CustomerID
OrderDate
ItemID
Amount

12.1.3 Recordset là gì ?

Một khi ta có khả năng tạo bảng, ta cần phải biết cách thao tác chúng.

Thao tác trên các bảng liên quan đến việc nhập và lấy về dữ liệu từ các bảng khác cũng như việc kiểm tra và sửa đổi cấu trúc bảng. Để thao tác với cấu trúc bảng, ta dùng các câu lệnh đinh nghĩa dữ liệu hoặc một đối tượng **TableDef** (được giới thiệu trong chương " Các đối tượng truy cập dữ liệu"). Để thao tác dữ liệu, trong một bảng, ta dùng Recordset.

Một Recordset là một cấu trúc dữ liệu thể hiện một tập hợp con các mẩu tin lấy về từ cơ sở dữ liệu. Về khái niệm, nó tương tự một bảng, nhưng có thêm một vài thuộc tính riêng biệt quan trọng.

Các RecordSet được thể hiện như là các đối tượng, về khái niệm tương tự như là các đối tượng giao diện người sử dụng (như là các nút lệnh và hộp văn bản) mà ta đã làm quen với Visual Basic trong các chương trước. Cũng như các kiểu đối tượng khác trong Visual Basic, các đối tượng Recordset có các thuộc tính và phương thức riêng.

Lưu ý : Ta có thể lập trình để tạo và sử dụng các recordset theo một trong ba thư viện truy cập dữ liệu – Các đối tượng truy cập dữliệu (DAO), các đối tượng truy cập dữ liệu từ xa (RDO) và các đối tượng dữ liệu ActiveX (ADO).

12.1.4 Các kiểu cơ sở dữ liệu

Cơ sở dữ liệu nội tại của Visual Basic, Jet, cung cấp 21 kiểu dữ liệu khác nhau.

Kiểu dữ liệu	Mô ta
Binary	Dùng để chứa các khối dữ liệu lớn như là đồ hoạ và các tập

	tin âm thanh số hoá.
Boolean	Giá trị logic đúng hoặc sai
Byte	Giá trị số nguyên một byte từ 0 đến 255
Currency	Trường số có thuộc tính đặc biệt để chứa các trí trị tiền tệ
Date/Time	Giá trị 8 byte thể hiện ngày hoặc giờ từ ngày 1/1/100 đến ngày 31/12/9999
Double	Kiể dữ liệu số 8 byte dấu phẩy động
GUID	Là một số 128 bye, cũng được gọi là định danh toàn thể duy nhất. Ta dùng số này để nhận dạng duy nhất các mẩu tin. Số này được dùng trong các bản sao
Integer	Số 2 byte đầu đủ từ -32768 đến 32767
Long	Số 4 byte đầy đủ từ -2,147,483,648 đến 2,147,483,647
Long binary	Trường giá trị lớn nhất có thể chứa các giá trị thập phân như là hình ảnh hay tập tin
(OLE Object)	Kiểu OLE Object nhúng trong cơ sở dữ liệu có thể lên tới 1 gigabyte.
Meno	Trường gí trị lớn có thể chứa đến 65,535 ký tự. Ta không cần thiết phải khai báo trước chiều dài của trường này.
Single type	Dữ liệu số 4 byte, dấu phẩy đơn
Text	Kiểu dữ liệu cáo chiều dài cố định, đòi hởi ta phải khai báo chiều dài của trường khi ta khai báo kiểu dữ liệu. Trường văn bản có thể dài từ 1 đến 255 ký tự.
VarBinary	Mẫu dữ liệu nhị phân biến đổi (dùng với ODBCDirect)

Không có sự tương đương một-một giữa kiểu dữ liệu Visual Basic và kiểu dữ liệu trường cơ sở dữ liệu. Ví dụ, ta không thể quy định một trường cơ sở dữ liệu là kiểu định nghĩa bởi người dùng hay biến Object của Visual Basic. Hơn nữa nếu ta dùng Microsoft Access để tạo cơ sở dữ liệu sử dụng trong các ứng dụng Visual Basic, lưu ý rằng một số kiểu dữ liệu hữu dụng trong ứng dụng Visual Basic không xuất hiện trong thiết kê bảng của Microsoft Access. Bởi vì Visual Basic hỗ trợ lập trình cơ sở dữ liệu khác với những gì tạo bằng Microsoft Access.

12.1.5 Tạo lược đồ cơ sở dữ liệu

Mặc dù việc tạo danh sách các bảng và trương là cách tốt nhất để xác định cấu trúc cơ sở dữ liệu, ta còn có một cách để xem các bảng và tường dưới dạng đồ hoạ. Sau đó, không chỉ xem được các bảng và trường hiện có mà còn thấy được mối quan hệ giữa chúng. Để làm được điều này, ta tạo lược đồ.

Lược đồ là sơ đồ các con đường trong cơ sở dữ liệu. Lược đồ thể hiện các bảng, trường và mối quan hệ trong cơ sở dữ liệu. Có lược đồ cơ sở dữ liệu là phần quan trọng trong thết kế phần mềm bởi vì nó cho ta một cách nhìn nhanh về những gì trong cơ sở dữ liệu.

Các lược đồ vẫn có vị trí quan trọng lâu dài sau khi quá trình thiết kế cơ sở dữ liệu hoàn tất. Ta sẽ cần đến lược đồ để thi hành các câu truy vấn trên nhiều bảng. Một lược đồ tốt sẽ trả lời được các câu hỏi như là, " Những bảng nào cần nối với nhau để đưa ra danh sách các hoá đơn trên $50.00 từ các khách hàng ở Minnesota trong 24 giờ qua ?"

Không có phương pháp chính thức để tạo lược đồ, mặc dù cũng có nhiều công cụ để thực hiện. Công cụ vẽ Visio rất uyển chuyển, nhanh và dễ dùng. Hơn nữa nó tích hợp với các ứng dụng Windows khác, nhất là Microsoft Office.

Phần này xem visio như một công cụ vẽ để ghi chếp về cơ sở dữ liệu. Nhung ta còn có thể dùng Visio như một công cụ phát triển. Với Visio Professional, ta có thể thiết kế cơ sở dữ liệu bằng đồ hoạ. Sản phẩm có khả năng lấy thiết kế đồ hoạ và tạo ra cơ sở dữ liệu thực sự. Tham khảo thông tin về Visio tại địa chỉ http://WWW.Visio.Com

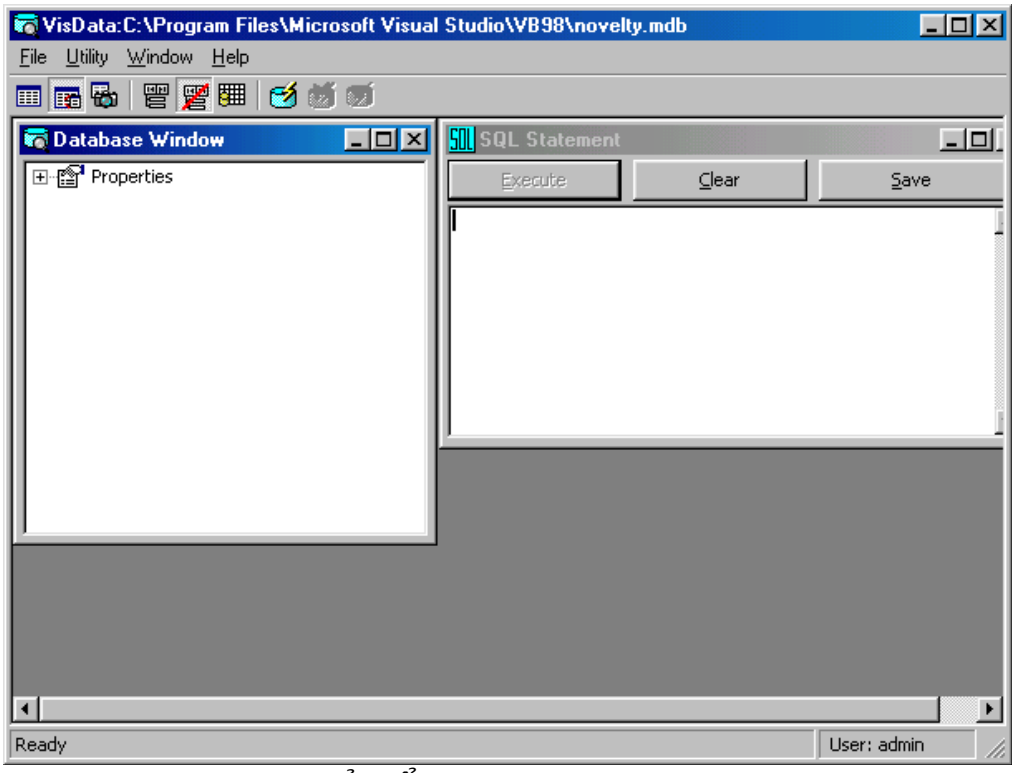
12.1.6 Dùng Visual Basic để tạo một cơ sở dữ liệu

Sau khi tạo xong lược đồ và chỉnh sửa thiết kế, đã đến lúc ta phải tạo cơ sở dữ liệu thực sự. Đạo cơ sở dữ liệu Jet dùng Visual Basic, ta có thể dùng tiện ích gọi là ***Visual Data Manager***. Tiện ích này trong ấn bản Visual Basic Professional và Enterprise cho phép ta tạo các cơ sở dữ liệu tương thích với ***Microsoft Access.***

> *Lưu ý : Do Visual Basic và Access 97 chia sẻ cùng bộ máy cơ sở dữ liệu ta có thể hoặc là dùng Visual Basic hoặc là dùng Access để tạo một cơ sở dữ liệu. Cơ sở dữ liệu cuối cùng đều như nhau.*

Để chạy Visual Data Manager, ta theo các bước sau :

1. Từ menu của Visual Basic chọn mục **Add-ins**, **Visual Data Manager**, cửa sổ **Visual Data Manager** sẽ xuất hiện.
2. Từ menu của **Visual Data Manager**, chọn File, New. Từ menu con, chọn Microsoft Access, Version 7.0 MDB. Một hộp thoại tập tin xuất hiện :
   Lưu ý : " Version 2.0 MDB " là phiên bản của Jet tương thích với phiên bản 16 –bit của Access và Visual Basic
3. Chọn thư mục ta muốn lưu cơ sở dữ liệu cơ sở dữ liệu mới rồi gõ tên. (Vì mục đích minh hoạ cho cuốn sách này, bạn có thể chọn tên cơ sở dữ liệu là **novelty.mdb**)
4. Nhấn chuột vào nút Save. Cơ sở dữ liệu mới được tạo và Visual Data Manager sẽ hiển thị một vài cửa sổ cho phép ta làm việc với cơ sở dữ liệu được hiển thị như hình dưới đây.

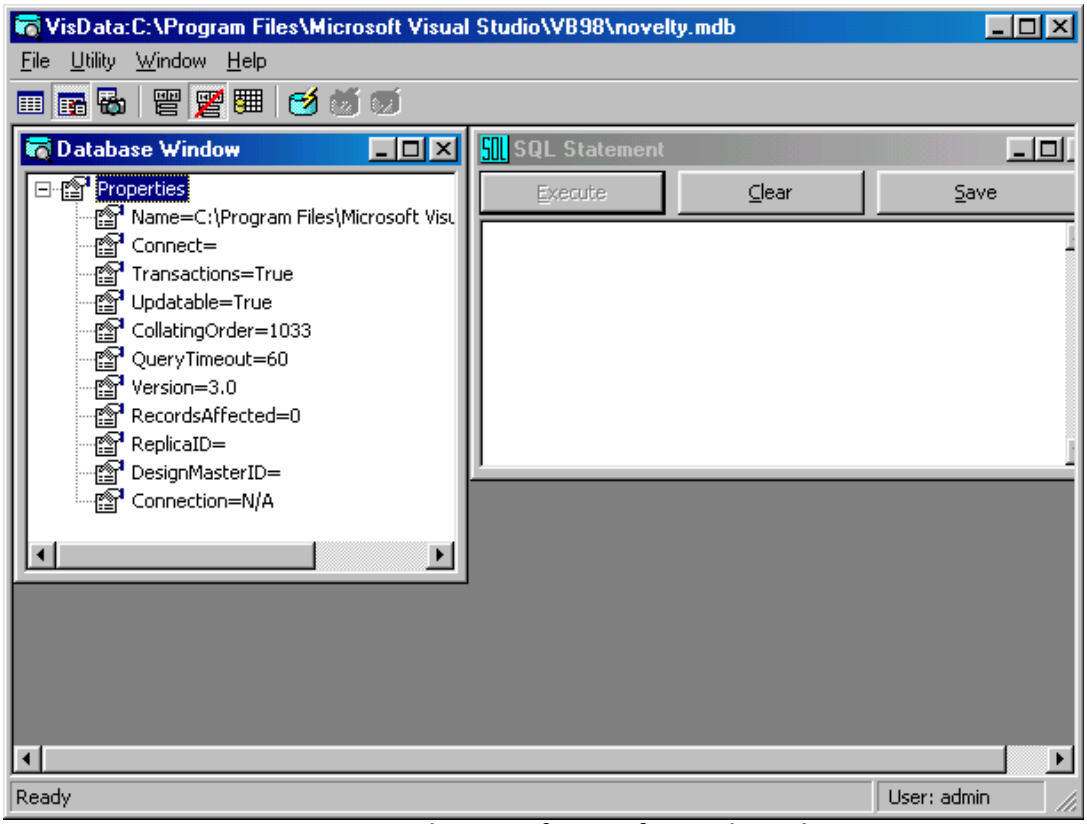


Cửa sổ Visual Data Manager

12.1.6.1 Sử dụng cửa sổ cơ sở dữ liệu

Cửa sổ DataBase của Visual Data Manager chứa tất cả các thành phần của cơ sở dữ liệu. Trong cửa sổ này ta có thể xem các thuộc tính, kiểm tra các bảng và các phần tử khác và thêm các thành phần mới vào cơ sở dữ liệu.

Để xem các thuộc tính ta vừa tạo, nhấn chuột vào dấu cộng ở bên trái của mục Properties. Mục này sẽ mở ra như hình dưới đây.

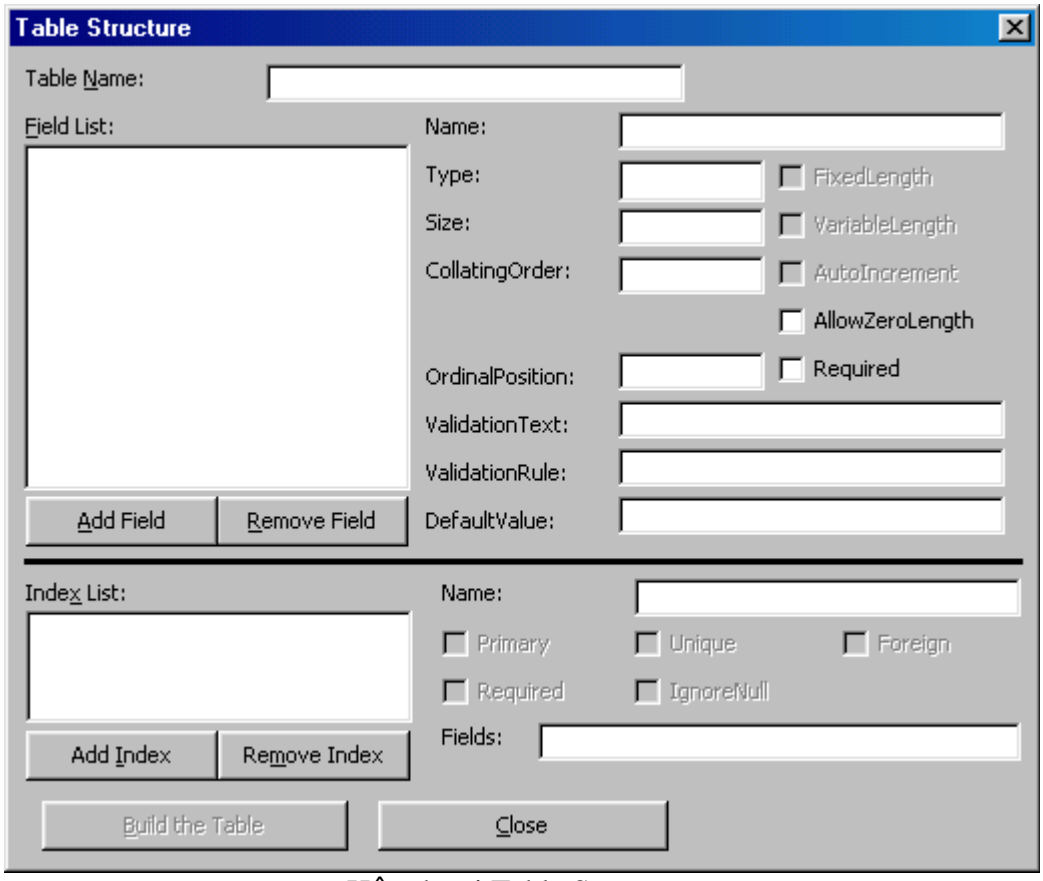


Xem các thuộc tính của cơ sở dữ liệu mới.

12.1.6.2 Tạo bảng

Một đặc tính của Visual Data Manager là nó không thể cho ta cách rõ ràng để tạo bảng mới trong cơ sở dữ liệu mà ta vừa tạo. Bởi vì các phần từ xuất hiện trong cửa sổ Database của Visual Data Manager rất nhạy với việc nhấn chuột phải. Nhưng một khi ta dùng nút chuột phải việc tạo một bảng mới thật là đơn giản.
Ví dụ : Để tạo một bảng mới ta theo các bước sau:

1. Trong cửa sổ Database của Visual Data Manager, nhấn chuột phải vào Properties. Menu ngữ cảnh của cửa sổ sẽ xuất hiện.
2. Chọn New Table. Hộp thoại Table Structure sẽ xuất hiện như hình dưới đây.

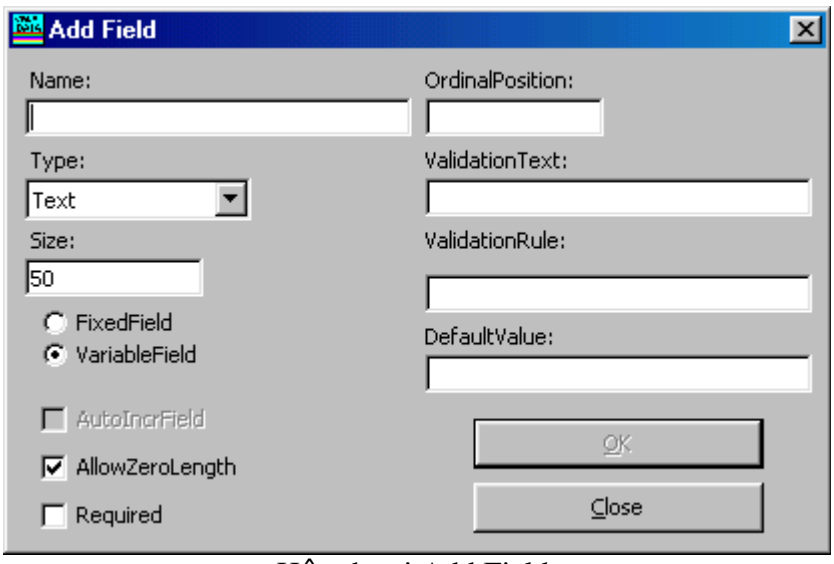


Hộp thoại Table Structure.

Trong hộp thoại Table Structure, ta có thể tạo cấu trúc bảng, chỉ định các trường, kiểu dữ liệu và chỉ mục. Ví dụ, ta sẽ tạo cấu trúc bảng để chứa khách hàng.

Để làm được điều này, theo các bước sau:

1. Gõ tblCustomer trong ô Table Name.
2. Nhấn chuột vào nút Add Field. Hộp thoại Add Field sẽ xuất hiện, được hiển thị như hình dưới đây.



Hộp thoại Add Field.

Hộp thoại Add field cho phép ta thêm một trường vào một bảng tạo bởi hộp thoại Table structure của Visual Data Manager.

3. Trong ô Name gõ First Name. Đây sẽ là tên của trường mà ta tạo trong bảng khách hàng.
4. Trong ô size gõ 25. Điều này chỉ ra rằng tên có thể lên đến 25 ký tự, nhưng không thể dài hơn. Điều này có nghĩa là cơ sở dữ liệu sẽ chứa các tên hiệu quả hơn.
5. Chọn Fixed Field để chỉ ra rằng đâu không phải là trường có chiều dài biến đổi, rồi nhấn nút OK. (Lưu ý rằng rất khó sửa đổi một trường một khi ta đã tạo xong nó.Vì vậy, phải chắc chắn rằng mọi thứ ta quy định là chính xác.) Trường được thêm vào cấu trúc cơ sở dữ liệu. Các hộp văn bản trong hộp thoại Add Field sẽ được xoá. Cho phép ta thêm vào một trường khác ngay lập tức.
6. Bây giờ ta có thể thêm các trường khác vào cấu trúc bảng. Sử dụng Add Field, thêm các trường vào tblCustomer các trường sau đây :

Tên trường	Kiểu dữ liệu	Kích cỡ dữ liệu	Fixed
First Name	Text	25	Yes
ID	Long, AutoInerField=true	N/A	N/A
LastName	Text	45	Yes
Company	Text	100	Yes
Address	Text	100	Yes
City	Text	100	Yes
State	Text	2	Yes
Zip	Text	9	Yes
Phone	Text	25	Yes
Fax	Text	25	Yes
Email	Text	255	Yes

7. Cần kiểm tra hộp AutoInerField khi tạo trường ID để đảm bảo rằng mọi khách hàng ta tạo đều có số hiệu duy nhất. Bởi vì bộ máy cơ sở dữ liệu tăng số trong trường một cách tự động, ứng dụng cơ sở dữ liệu sẽ không phải tự sinh ra số hiệu duy nhất.
8. Khi ta hoàn tất việc nhập trường, nhấn nút bấm Close.

12.1.6.3 Chỉ định chỉ mục và khoá chính

Đến đây ta vừa tạo xong một bảng cơ bản, phần còn lại là ta cần chỉ ra các chỉ mục. Một chỉ mục là một thuộc tính ta có thể gán cho một trường để tạo sự dễ dàng cho bộ máy cơ sở dữ liệu khi lấy về thông tin chứa trong trường đó. Ví dụ, trong cơ sở dữ liệu theo dõi khách hàng, ứng dụng có thể tìm kiếm các khách hàng theo họ, mã Zip và các số hiệu ID cá nhân. Do đó, cần thiết phải tạo các chỉ mục trên những trường này để giúp cho quy trình lấy mẩu tin dựa trên các trường này nhanh hơn.

Một khi ta đã nhận ra lợi ích của các chỉ mục trong việc thiết kế cơ sở dữ liệu, ta có thể tự đặt ra các câu hỏi như : Nếu các chỉ mục giúp việc tìm kiếm nhanh hơn, tại sao ta không đặt một chỉ mục trong tất cả các trường của mọi bảng ? Câu

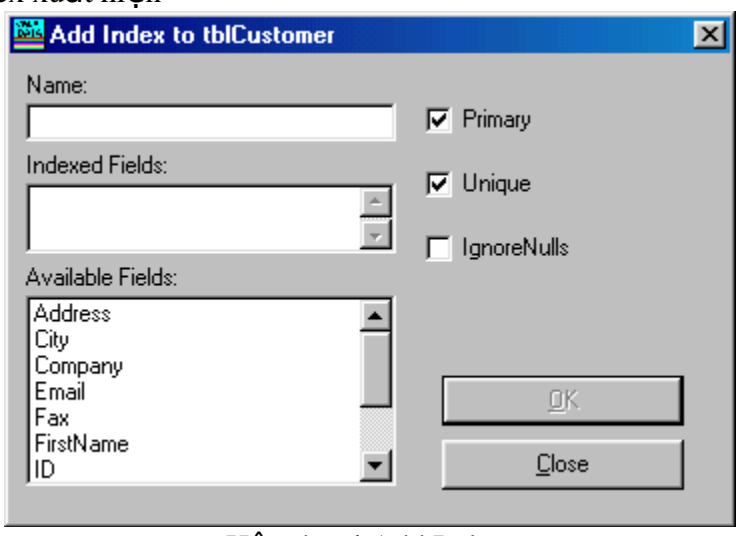
trả lời là các chỉ mục làm cho cơ sở dữ liệu của ta phình to hơn về mặt vật lý, vì vậy, nếu ta có quá nhiều chỉ mục, sẽ lãng phí bộ nhớ và làm cho máy tính của ta chạy chậm hơn. Điều này hiển nhiên làm mất đi các lợi thế ban đầu. Không có quy định về việc nên tạo bao nhiêu chỉ mục cho mỗi bảng, nhưng nói chung, ta nên tạo một chỉ mục dựa trên các trường mà ta nghĩ là dùng thường xuyên trong các câu truy vấn.

Hình 1. *Khoá chính* **Là kiểu chỉ mục đặc biệt. Một trường được coi là một khoá chính vủa bảng phục vụ cho việc xác định duy nhất mẩu tin. Vì vậy, không như các kiểu chỉ mục khác, sẽ không có hai mẩu tin trên cùng một bảng mà có cùng giá trị cho trường khoá chính. Tương tự, khi thiết kế một trường làm khoá chính, không có mẩu tin nào chứa giá trị rỗng, giá trị NULL ở trường này. Khi chỉ ra một trường làm khoá chính của bảng, ta có thể tạo mối quan hệ giữa bảng này với các bảng khác trong cơ sở dữ liệu.**

Mỗi bảng mà ta thiết kế phải có ít nhất một khoá chính, và nó phải được đánh số chỉ mục trên những trường mà ta mong đợi sẽ được truy vấn nhiều nhất. Trong trường hợp của bảng tblCustomer, cũng như với nhiều bảng cơ sở dữ liệu, khoá chính sẽ là trường ID. Các chỉ mục phụ là trường LastName và FirstName

Để tạo các chỉ mục và các khoá chính, theo các bước sau :

1. Trong hộp thoại Table Structure, nhấn chuột vào nut Add Index. Hộp thoại Add Index xuất hiện



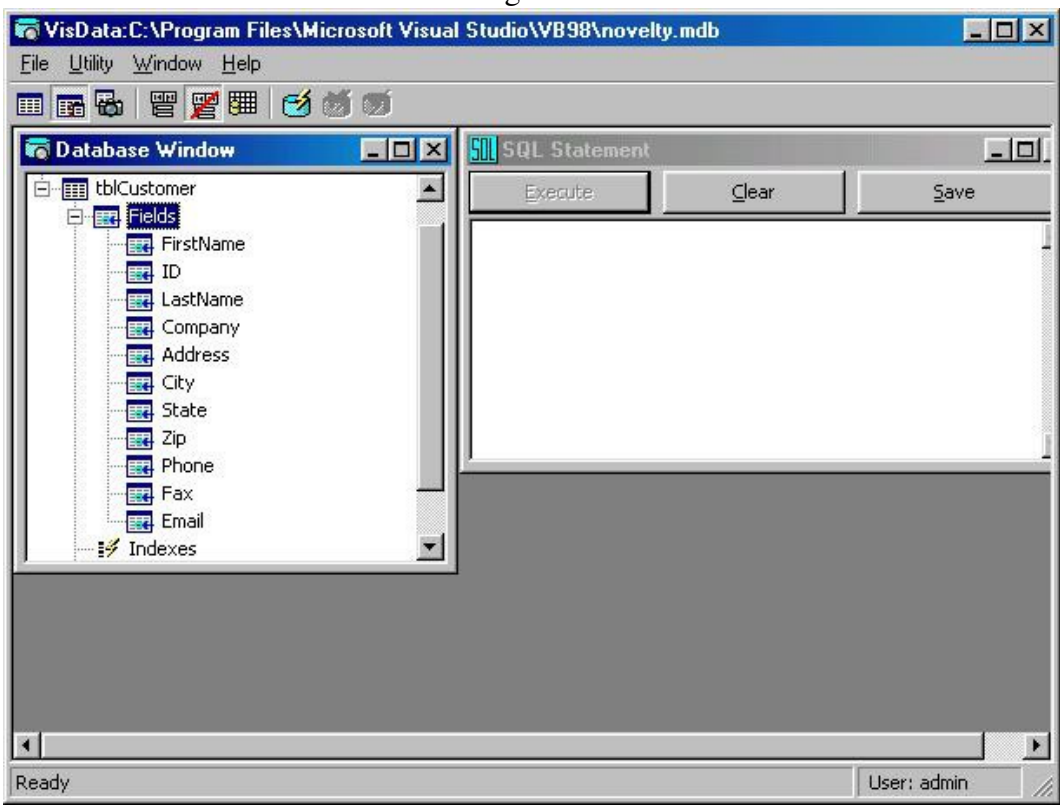
Hộp thoại Add Index.

Trước hết ta sẽ tạo khoá chính cho bảng.

2. Gõ Primary Key trong hộp văn bản Name
3. Nhấn đúp chuột vào trường ID trong danh sách các trường có sẵn. ID được thêm vào danh sách các trường có chỉ mục. Hộp đánh dấu Primary và Unique phải được chọn theo mặc định.
4. Nhấn OK, Hộp văn bản bị xoá và khoá chính được thêm vào thiết kế bảng. Lưu ý rằng chỉ mục có tên như tên trường (mặc dù đã quen với Microsoft Access m ta có thể không biết điều này, bởi vì Access che tên chỉ mục trên giao diện người sử dụng). Truy cập đến tên của một trường chỉ mục thì thật thuận tiện cho một mục đính nào đó.

Giờ đây ta có thể tạo thêm hai chỉ mục cho các trường FirstName và LastName. Để làm được điều này ta làm theo các bước sau :

1. Gõ tên chỉ mục FirstNameIndex trong hộp văn bản Name
2. Nhấn đúp chuột trên trường FirstName trong danh sách các trường hiện có, FirstName được thêm vào trong danh sách các trường có chỉ mục.
3. Bỏ chọn các hộp đánh dấu Primary và Unique, sau đó nhấn nut OK.
   Cảnh Báo *: Nếu ta để hộp đánh dấu Unique được chọn, ta sẽ không thể thêm hai người có cùng tên vào cơ sở dữ liệu.*
4. Lặp lại quy trình này với trường LastName, tạo một chỉ mục là LastIndex.
5. Nhấn nút chuột Close. Ta sẽ gặp lại hộp hội thoại Table Structure.
6. Để tạo bảng, nhấn nút Build the Table. Bảng sẽ được tạo và thêm vào cửa sổ Database của Visual Data Manager



Tạo bảng cho cơ sở dữ liệu.

12.1.6.4 Thay đổi thuộc tính của các trương có sẵn

Visual Data manager có vẻ hơi khó để sửa đổi phần lới các thuộc tính quan trọng của một bảng (không giống như Microsoft Access cho phép ta thay hầu hết cấu trúc bảng mọi lúc). Nói chung, khi ta muốn sửa đổi thuộc tính của trường bằng cách sử dụng Visual Data Manager ta phải xoá trường để tạo lại.

12.1.6.4.1 Sửa đổi chiều dài của trường LastName

Giả định rằng ta muốn sửa đổi chiều dài của trường LastName. Để làm được điều này, theo các bước sau :

1. Trong các cửa sổ Database của Visual Data Manager, nhấn chuột phải lên tblCustomer.

2. Từ menu ngữ cảnh, chọn Design. Hộp thoại Table Structure xuất hiện.

12.1.6.4.2 Xoá chỉ mục

Để xoá trường LastName, ta phải xoá chỉ mục của nó trước. Để làm được điều này, theo các bước sau :

1. Chọn **LastNameIndex** trong danh sách các chỉ mục
2. Nhấn nút **RemoveIndex**
3. Khi một thông điệp hỏi ta muốn xoá chỉ mục này không, nhấn Yes. Chỉ mục được xoá

12.1.6.4.3 Xoá trường LastName

Bây giờ ta có thể xoá trường này. Để làm được điều này, theo các bước sau:

1. Chọn trường LastName trong danh sách các trường.
2. Nhấn nút Remove Field. Khi xuất hiện thông điệp hỏi ta muốn xoá trường này hay không, nhấn Yes. Trường này sẽ bị xoá khỏi bảng.

Bây giờ thì ta có thể sửa đổi trường bằng cách thêm nó trở lại bảng, lần này với chiều dài 50. Đừng quên thêm chỉ mục cho trường này sau khi thêm nó trở lại bảng.

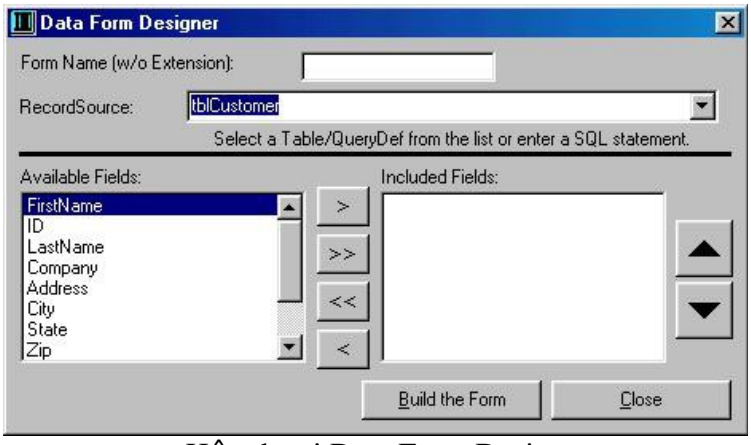
> ***Mẹo vặt** : Tiến trình sửa một trường có sẵn trong Visual Data Manager có vẻ khá phức tạp. Trong Microsoft Access, việc sửa đổi trên những trường cõ sẵn thật dễ dàng, vì lý do đó, các nhà lập trình cơ sở dữ liệu khéo léo trong Visual Basic sẽ giữ một bản sao của Access ở đâu đó để phòng hờ.*

12.1.6.5 Dùng Visual Data Manager để tạo giao diện

Một ưu điểm của Visual Data Manager so với Microsoft Access là khả năng tạo các biểu mẫu Visual Basic dựa trên cấu trúc dữ liệu được tạo.

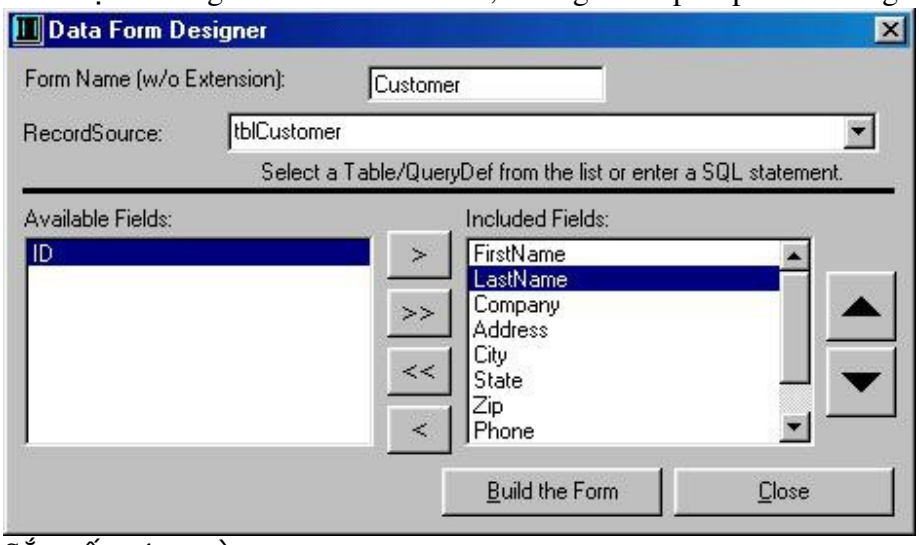
Giả định rằng ta đã hoàn tất khâu thiết kế tblCustomer và muốn thêm một biểu mẫu Visual Basic vào đề án dựa trên thiết kế bảng . Để làm được điều này, theo các bước sau :

1. Từ menu Visual Data Manager chọn Utility, ***Data Form Design***. Hộp thoại Data Form Design xuất hiện.
2. Trong hộp thoại văn bản Form name, gõ Customer
3. Trong hộp kết hợp RecordSource, chọn tblCustomer, ***Data Form Design*** điền danh sách các trường tìm thấy trong tblCustomer vào **Available Fields**



Hộp thoại Data Form Design

4. Nhấn nút mũi tên phải cho tất cả các trường hiện có trừ ID để thêm chúng vào biểu mẫu. (Không được thêm trường ID vào biểu mẫu vì người sử dụng không thể sử trường ID).
5. Chọn trường và nhấn mũi tên lên, xuống để sắp xếp các trường.

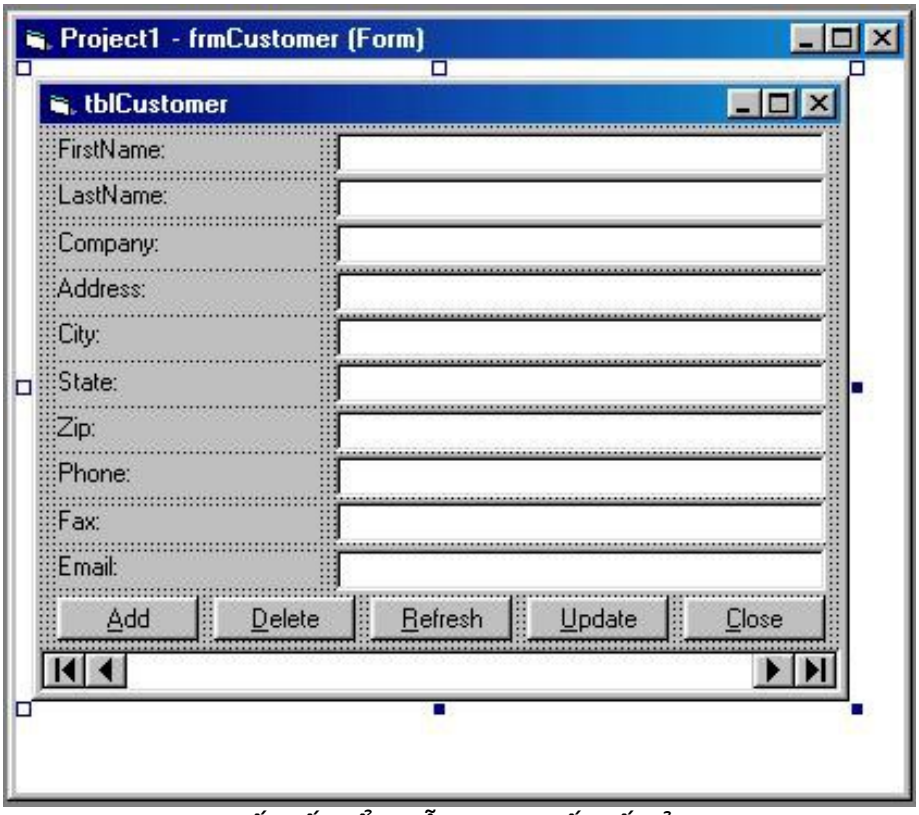


Sắp xếp các trường

6. Nhấn nút Build the Form. Biểu mẫu được tạo trong Visual Basic.
7. Nhấn Close.

Kế tiếp ta muốn thoát khỏi Visual Data Manager để xem biểu mẫu của ta như thế nào. Nhưng ta muốn quay trở lại để thêm các phần tử mới vào cơ sở dữ liệu hoặc muốn sử đổi những cái ta vừa làm. Để thông báo cho Visual Data Manager la ta muốn mở cơ sở dữ liệu lại trong lần kế tiếp, theo các nước sau :

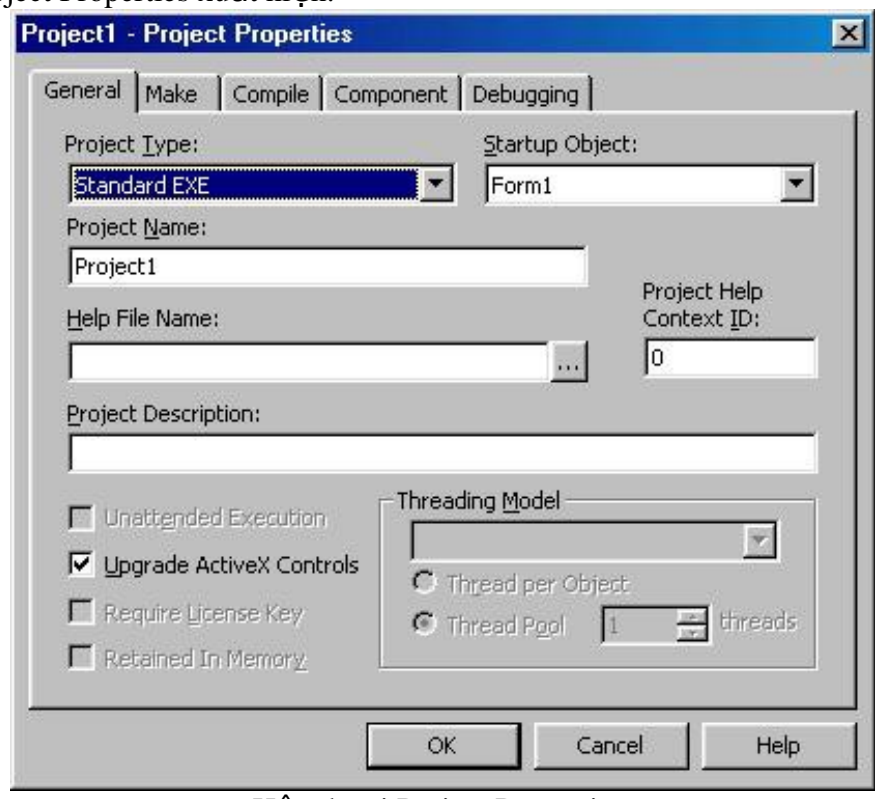
1. chọn Utility, Preferences. Từ Menu con, chọn Open Last Database từ Startup.
2. Thoát Visual Data Manager bằng cách chọn File, Exit, Khi ta trở về Visual Basic m ta sẽ thấy biểu mẫu mới gọi là frmCustomer



Thiết kế biểu mẫu theo thiết kế bảng.

Để làm việc với biểu mẫu mới được tạo, ta sẽ phải đặt nó làm biểu mẫu khởi động của đề án. Để làm được điều này, theo các bước sau:

1. Từ menu Project của Visual Basic, chọn Project1 Properties. Hộp thoại Project Properties xuất hiện.



Hộp thoại Project Properties.

2. Trong hộp kết hợp StartUp Object, chọn frmCustomer và nhấn nút OK.
3. Từ menu Run của Visual Basic, chọn Start. Ứng dụng thi hành, nó sẽ hiển thị giao diện nhập liệu trong frmCustomer.

Bây giờ ta có thể nhập liệu vào giao diện mà Visual Basic cung cấp cho ta. Để làm được điều này, theo các bước sau :

1. Nhấn nút Add. Ta sẽ thấy rằng ứng dụng không có phản hồi một cách trực quan để cho thấy rằng một điều gì đó đã thay đổi . Tuy nhiên, phải tin rằng bạn đang sửa đổi một mẩu tin mới.
2. Nhập dữ liệu vào mỗi hộp văn bản trong biểu mẫu.
3. Khi ta làm xong, nhấn nút Update. Mẩu tin sẽ được lưu trữ. Chỉ có một thông tin phản hồi ta thấy là điều khiển dữ liệu hiển thị "Record 1 "

Chương trình lúc thi hành.

Giao diện nhập liệu cơ bản được tạo bởi Data Form Designer cho ta ý nghĩa của chương trình mà ta phải viết để có một ứng dụng mạnh mẽ bằng chách sử dụng điều khiển Data.Mặc dù điều khiển data được coi là một giải pháp " Không cần lập trình ", nếu ta cần mởi rộng tính năng của nó (như thi hành các hành động tìm kiếm, xoá các mẩu tin) chương trình có thể không trực quan đối với những người mới học. Chúng ta sẽ tìm hiểu thêm thông tin về chương trình này, cách hoạt động và chỉnh sửa nó như thế nào để tạo ra một ứng dụng với đầy đủ tính năng hơn sau này.

12.1.7 Các mối quan hệ

Mối quan hệ là một cách đinh nghĩa chính thức hai bảng liên hệ với nhau như thế nào . Khi ta định nghiêa một mối quan hệ, ta đã thông báo với bộ máy cơ sở dữ liệu rằng hai trường trong hai bảng liên quan được nối với nhau.

Hai trường liên quan với nhau trong một mối quan hệ là khoá chính đã được giới thiệu ở phần trước và khoá ngoại. Khoá ngoại là khoá trong bảng kiên quan chứa bản sao của khoá chính của bảng chính.

Ví dụ, giả định rằng ra có các bảng cho phòng ban và Nhân viên. Có một mối quan hệ một - nhiều giữa một phòng ban và một nhóm nhân viên. Mỗi phòng ban có một ID riêng, tương tự với nhân viên. Tuy nhiên, để chỉ ra một nhân viên làm việc ở phòng ban nào, ta cần phải tạo một bản sao của ID của phòng ban cho mỗi mẩu tin của nhân viên. Vì vậy, để phân biệt mỗi nhân viên là một thành viên của một phòng ban, bảng **Employees** phải có một trường gọi là DepartmentID để chứa ID của phòng ban mà nhân viên đó làm việc. Trường DepartmentID trong bảng **Employees** được tham chiếu như 1 khoá ngoại của bảng Employees bởi vì nó sẽ chứa bản sao của khoá chính của bảng **Departments**.

Sau đó mối quan hệ báo cho bộ máy cơ sở dữ liệu hai bảng liên quan với nhau trong mối quan hệ và khoá ngoại nào liên quan với khoá chính nào. Bộ máy Access/Jet không đòi hỏi ta phải khai báo tường minh các mối quan hệ này, nhưng sẽ có lợi hơn nếu làm điều này bởi vì nó làm đơn giản hoá công việc lấy về dữ liệu dựa trên các mẫu tin nối qua hai hay nhiều bảng.

Ngoài việc ghép các mẩu tin liên quan trong các bảng riêng biết, ta còn định nghĩa mối quan hệ để tận dụng thế mạnh của tính toàn vẹn tham chiếu, một thuộc tính của bộ máy cơ sở dữ liệu duy trì các dữ liệu trong một cơ sở dữ liệu nhiều bảng luôn luôn nhất quán. Khi tính toàn vẹn tham chiếu tồn tại trong một cơ sở dữ liệu, bộ máy cơ sở dữ liệu sẽ ngăn cản ta xoá một mẩu tin khi có các mẩu tin khác tham chiếu đến nó trong cơ sở dữ liệu.

Sau khi đã định nghĩa một mối quan hệ trong cơ sở dữ liệu, việc định nghĩa mối quan hệ này sẽ được lưu trữ cho đến khi ta xóa nó.

Lưu ý : Ta không thể tạo một mối quan hệ cơ sở dữ liệu bằng cách dùng Visual Data Manager tuy nhiên ta có thể tạo một mối quan hệ sử dụng Microsoft Access hay Lập trình.

12.1.7.1 Sử dụng tính toàn vẹn tham chiếu để duy trì tính nhất quán

Khi các bảng nối kết với nhau thông qua mối quan hệ, dữ liệu trong mỗi bảng phải duy trì sự nhất quán trong các bảng liên kết. Tính toàn vẹn tham chiếu quản lý công việc này bằng cách theo dõi mối liên hệ giữa các bảng và ngăn cấm các kiểu thao tác nào đó trên các mẩu tin.

Ví dụ, giả định rằng ta có một bảng gọi là tblCustomer và một bảng khác là tblOrder. Hai bảng này quan hệ với nhau qua trường chung là ID.

Giả thiết ở đây là ta tạo các khách hàng chứa trong tblCustomer rồi tạo các hoá đơn trong tblOrder. Nhưng điều gì sẽ xảy ra nếu ta tiến hành xoá một khách hàng có hoá đơn chưa xử lý chứa trong bàng hoá đơn ? Hoặc là nếu ta tạo một hoá đơn mà không có một CustomerID hợp lệ gắn liền với nó ? Mục hoá đơn không có CustomerID thì không thể gửi đi, bởi vì địa chỉ gửi là một trường của mẩu tin trong tblCustomer. Khi dữ liệu tron các bảng quan hệ gặp phải rắc rối này, ta nói nó ở trọng thái không nhất quán.

Một điểu rất quan trọng là cơ sở dữ liệu không được trở nên không nhất quán, bộ máy cơ sở dữ liệu Jet cung cấp một cách để ta định nghĩa mối quan hệ trong các

bảng. Khi ta định nghĩa một mối quan hệ chính thức giữa hai bảng bộ máy cơ sở dữ liệu sẽ giám sát mối quan hệ mày và ngăn cấm bất kỳ thao tác nào vi phạm tính toàn vẹn tham chiếu. Hoạt động của cơ chế toàn vẹn tham chiếu là phát sinh ra lỗi mỗi khi ta thi hành một hành động nào đó làm cho dữ liệu rơi vào trạng thái không nhất quán. Ví dụ, trong cơ sở dữ liệu có tính toàn vẹn tham chiếu đang hoạt động, nếu ta cố tạo một hoá đơn chứa một ID của khách hàng đối với một khách hàng không tông tại, ta sẽ nhận một thông báo lỗi và hoá đơn sẽ không thể tạo ra.

12.1.8 Chuẩn hoá

Chuẩn hoá là một khái niệm liên quan đến mối quan hệ. Về cơ bản, nguyên tắc của chuẩn hoá phát biểu rằng các bảng cơ sở dữ liệu sẽ loại trừ tính không nhất quán và giảm thiểu sự kém hiệu quả.

Các cơ sở dữ liệu được mô tả là không nhất quán khi dữ liệu trong một bảng không tương ứng với dữ liệu nhập vào trong bảng khác. Ví dụ, Nếu phân nửa số người nghĩ rằng A ở miền Trung Tây và một nửa nghĩ rằng nó nằm ở phía Nam và nếu cả hai nhóm nhân viên nhập liệu theo ý kiến riêng của họ, khi ấy báo cáo cơ sở dữ liệu trình bày những việc xảy ra ở miền Trung Tây là vô nghĩa.

Một cơ sở dữ liệu kém hiệu quả không cho phép ta trích ra csc dữ liệu chính xác mà ta muốn. Một cơ sở dữ liệu chứa toàn bộ dữ liệu trong một bảng có thể buộc ta phải vất vả duyệt qua một lượng lớn tên các khách hàng, địa chỉ và lịch sử liên hệ chỉ để lấy về 1 số điện thoại của một khách hàng nào đó. Mặt khác, một cơ sở dữ liệu được chuẩn hoá đầy đủ chứa từng mẩu thông tin của cơ sở dữ liệu trong bảng riêng và xa hơn, các định từng mẩu thông tin duy nhất thông qua khoá chính của nó.

Các cơ sở dữ liệu chuẩn hoá cho phép ta tham chiếu đến một mẩu thông tin trong một bảng bất kỳ chỉ bằng khoá chính của thông tin đó.

Ta quyết định cách thức chuẩn hoá của một cơ sở dữ liệu khi ât thết kế và khởi tạo một cơ sở dữ liệu. Thông thường, mọi thứu về ứng dụng cơ sở dữ liệu - từ thiết kế bảng cho đến thiết kế truy vấn, từ giao diện người sử dụng đến cách hoạt động của báo cáo - đều xuất phát từ cách chuẩn hoá cơ sở dữ liệu.

> *Lưu ý : Là một lập trình viên cơ sở dữ liệu, thỉnh thoảng bạn sẽ chợt nảy ra ý nghĩ về cơ sở dữ liệu vẫn chưa được chuẩn hoá vì lý do này hay lý do khác. Sự thiếu chuẩn hoá có thể do chủ ý, hoặc có thể là do kết quả của sự thiếu kinh nghiệm hoặc sự khộng thận trọng trong việc thiết kế cơ sở dữ liệu ban đầu. Dù ở mức độ nào, nếu đã chọn chuẩn hoá một cơ sở dữ liệu đã tồn tại, ta nên thực hiện sớm (bởi vì mọi thứ khác thực hiện trong cơ sở dữ liệu đều phụ thuộc vào cấu trúc bảng của cơ sở dữ liệu). Hơn nữa, ta sẽ thấy những câu truy vấn hành động là công cụ rất hữu ích trong việc sắp xếp lại một cơ sở dữ liệu thiết kế thiếu sót. Truy vấn là hành động cho phép ta đi chuyển các trường từ bảng này sang bảng khác chư là thêm, cập nhậtvà xoá mẩu tin từ các bảng dựa trên các tiêu chỉ nêu ra.*

12.1.8.1 Quan hệ Một - Một

Là loại quan hệ dễ hiểu và dễ thực hiện nhất, bởi vì trong những mối quan hệ như vậy, một bảng sẽ lấy vị trí của một trường trong một bảng khác, trường liên quan cũng dễ nhận dạng. Tuy nhiên, quan hệ một - một không phải là mối quan hệ thông dụng nhất trong ứng dụng cơ sở dữ liệu. Do 2 nguyên nhân:

1. Hầu như ta không cần biểu diễn mối quan hệ một một với hai bảng. Ta có thể dùng nó để cải tiến khả năng hoạt động, ví dụ ta mất tính linh hoạt khi chứa các dữ liệu liên hệ trong một bảng tách biệt. Trong ví dụ trước, thay vì có các bảng nhân viên và công việc chứa trong bảng nhân viên.
2. Thể hiện quan hệ một - nhiều thì cũng khá dễ (nhưng linh hoạt hơn nhiều) quan hệ một một.

12.1.8.2 Quan hệ một - nhiều

Phổ biến hơn quan hệ một - một, trong đó, mối mẩu tin trong một bàng này không có, hoặc có một, hoặc nhiều mẩu tin trong một bảng liên hệ.

Ví dụ, ta gán từng khách hàng cho một người bán hàng. Để thực hiện điều này, ta cần một bảng cho người bán hàng :

Bởi vì một người bán hàng có trách nhiệm với nhiều khách hàng, ta có thể nói đã có mối quan hệ một - nhiều giữa người bán hàng và khách hàng.

Để thực hiện mối quan hệ này trong thiết kế cơ sở dữ liệu, ta phải copy khoá chính của phía một đến bảng chứa phía nhiều trong quan hệ.

Trong một thiết kế giao diện người sử dụng, ta thực hiện quá trình copy khoá chính của một bảng đến khoá ngoại của một bảng liên hệ nhờ một điều khiển hộp danh sách hay hộp kết hợp. Để tìm hiểu thêm thông tin về tổ chức giao diện.

12.1.8.3 Quan hệ nhiều nhiều

Quan hệ nhiều - nhiều là bước phát triển của quan hệ một - nhiều ví dụ cổ điển của quan hệ nhiều nhiều là học sinh và lớp. Mỗi học sinh có nhiều lới, mỗi lớp có nhiều học sinh (Tuy nhiên,có lớp không có hoặc chỉ có một học sinh, và có thể học sinh chỉ có một hoặc không có lớp).

Để thiết lập quan hệ nhiều nhiều, ta có thể sửa lại ví dụ trước sao cho cơ sở dữ liệu có thể chứa nhiều người bán cho một người mua. Mỗi người bán có nhiều khách hàng, và mỗi khách hàng có nhiều người bán.

Giao diện người sử dụng phát triển trong Microsoft Access chủ yếu thực hiện quan hệ một - nhiều và quan hệ nhiều - nhiều bằng cách dùng biểu mẫu con. Đối với một nhà lập trình VB, một biểu mẫu con Access tương tự một biểu mẫu trong biểu mẫu : Biểu mẫu chính hiển thị phía một trong quan hệ một nhiều, trong khi biểu mẫu con hiển thị các mẩu tin ở phía nhiều . Thuận tiện của biểu mẫu con là nó không đòi hỏi phải dữ quan hệ giữa 2 bảng được nhất quán ; ta chỉ quy định thuộc tính để trình bày khoá chính và khoá ngoại.

Khác với **Microsoft Access**, VB không hỗ trợ biểu mẫu con để tự động hiển thị tất cả các mẩu tin liên quan với một mẩu tin nhất định . Thay vào đó, ứng dụng Visual Basic chủ yếu yêu cầu ta lập trình để thực hiện một giao diện người sử dụng dựa trên quan hệ nhiều nhiều.

12.2 Sử dụng cửa sổ xem dữ liệu

Điểm mới trong Visual Basic 6.0 là cửa sổ Data View, cho phép ta làm việc với một cơ sở dữ liệu không cần phải sử dụng công cụ bên ngoài hay công cụ bổ xung **Add-in**.

1. Để dùng cửa sổ **Data View**, hay nhấn nút **Data View** trên thanh công cụ chuẩn của VB.
2. Cửa sổ **Data View** xuất hiện, Cửa sổ cho ta hai thư mục, **Data Links** và **Data Environment Connections**.

Để kiểm tra mối quan hệ hoạt động như thế nào, ta theo các bước sau:

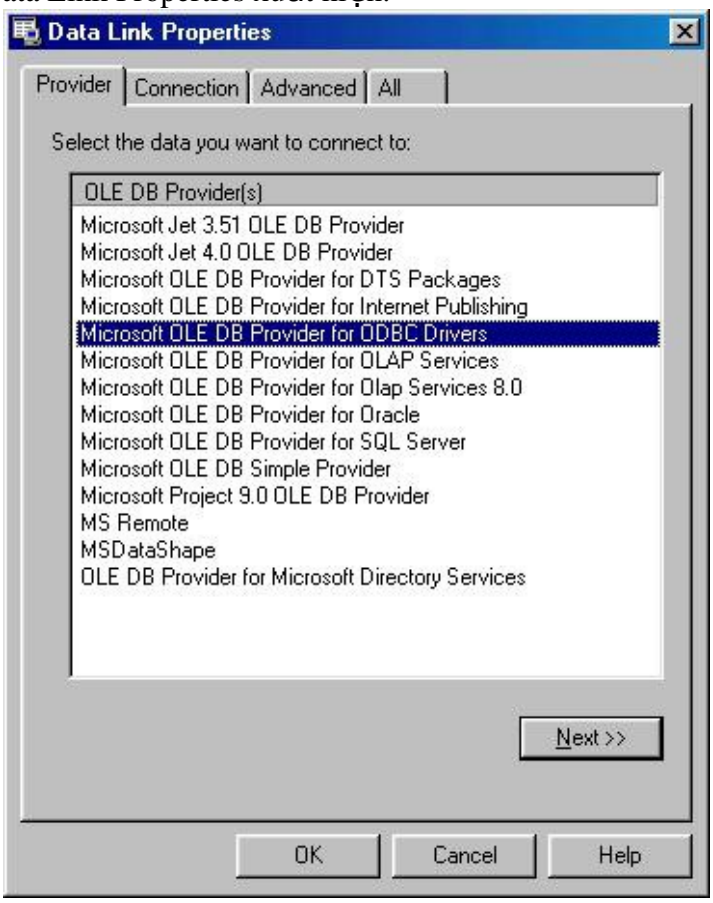
1. Đóng và lưu cửa sổ **Relationship**.
2. Mở bảng tblOrder và nhập và một mẩu tin cho một khách hàng không tồn tại trong bảng tblCustomer.

Bộ máy cơ sở dữ liệu sẽ sinh ra lỗi. Bởi vì lỗi này được sinh ra ở mức bộ máy cơ sở dữ liệu, giống như loại lỗi sinh ra khi có vấn đề về tính toàn vẹn tham chiếu trong Access hoặc trong một ứng dụng Visula Basic sử dụng cơ sở dữ liệu này.

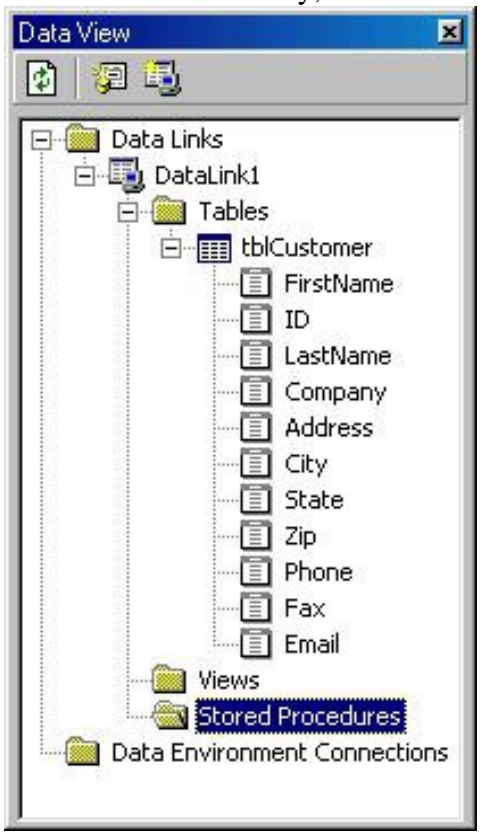
Liên kết dữ liệu (data link) là một cách kết nối môi trường phát triển của Visual Basic với một cơ sở dữ liệu nào đó. Một kết nối dữ liệu (**Data Environment connection**) là một cách sử dụng cơ sở dữ liệu trong một đề án VB. Điểm khác biệt giữa hai thành phần này là khi tạo một liên kết dữ liệu,nó xuất hiện trong cửa sổ data view mỗi khi cửa sổ hiển thị trong Visual Basic, ngay cả khi ta đóng đề án hiện hành và mở một đề án mới. Trái lại, trình thiết kế nối kết môi trường dữ liệu gắn liền với đề án ta đang làm việc. Nó trở thành một sản phẩm nhị phân tạo ra khi biên dịch và có thể được chia sẻ giữa nhiều đề án.

Muốn dùng một liên kết dữ liệu để duyệt dữ liệu, theo các bước sau:

1. Trong cửa sổ Data View, nhấn nút phải chuột lên thư mục Data Links. Từ menu bật ra, chọn Add a Data Link.
2. Cửa sổ Data Link Properties xuất hiện.



3. Chọn trình cung cấp Microsoft Jet, rồi nhấn Next.
4. Tab Connection xuất hiện. Nhập đường dẫn và tên tập tin cơ sở dữ liệu ta muốn dùng.
5. Nhấn nút Test Connection tạo phần dưới của cửa sổ . Ta sẽ có một thông điệp thông báo kết nối đến cơ sở dữ liệu thành công .
6. Nhấn OK, liên kết dữ liệu được thiết lập, và cửa sổ Data View nhắc ta nhập vào tên của liên kết. Gõ vào Novelty, rồi nhấn Enter.



Liên kết dữ liệu cung cấp một cách nhìn tóm lược về nguồn dữ liệu. Mỗi lần ta tạo một liên kết dữ liệu, ta có thể duyệt bằng cách sử dụng phần tử trong danh sách tóm lược. Thực hiện điều này bằng cách nhấn vào dấu cộng bên trái mỗi phần tử. Cơ sở dữ liệu mở rộng đầy đủ trong cửa sổ xem dữ liệu.
(Tuỳ theo công cụ ta dùng để tạo cơ sở dữ liệu, ta có thể thấy thêm một số bảng trong danh sách.
Giờ đây, ta có thể xem các dữ liệu sống động. Thực hiện điều này bằng cách nhấn đúp chuột lên bảng tblCustomer trong cửa sổ Data view.
Cách thể hiện khởi đầu của dữ liệu thì không đặc sắc lắm, vì ta chưa nhập mẩu tin nào cả. Tuy nhiên, ta có thể nhập mẩu tin bằng cách gõ vào ô trên lưới.

12.3 Tạo trình thiết kế môi trường dữ liệu

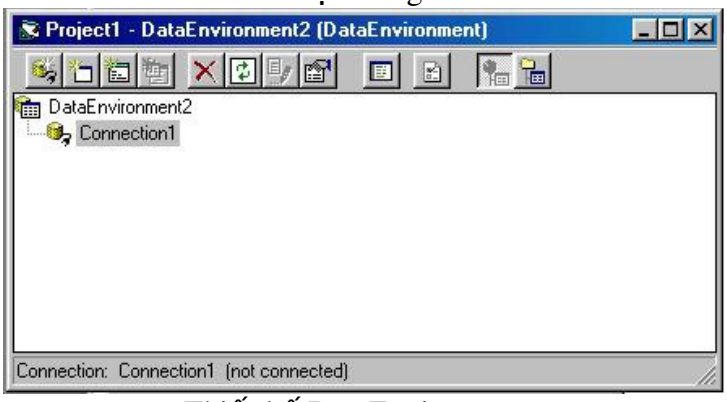
Ta có thể tạo một thiết kế **DataEnvironment** để quản lý một cách trực quan kết nối với một cơ sở dữ liệu. Khi ta có một thiết kế **DataEnvironment** được chứa trong tập tin nhị phân của ứng dụng lúc biên dịch, vì vậy không cần lo ngại về những phụ thuộc bên ngoài.

Lưu ý: Là điểm mới trong VB6, thiết kế DataEnvironment vờ một quan niệm tương tự như thiết kế UserConnection của RDO (Đối tượng dữ liệu từ xa – Remove Data Object) ta từng dùng trong VB5. Tuy nhiên, thiết kế DataEnvironment dựa trên ADO và cung cấp nhiều chức năng hơn. Nếu ta cso một ứng dụng hiện hành dùng RDO, ta có thể tiếp tục dùng thiết kế UserConnection.

Chương này trình bày cách dùng thiết kế DataEnvironment để tạo một giao diện người sử dụng được điều khiển với cơ sở dữ liệu. Tuy nhiên, có nhiều cách thực hiện.

Để thêm một thiết kế DataEnvironment vào ứng dụng dùng cửa sổ Data View, theo các bước sau :

1. Trong của sổ Data View, nhấn nút Add Data Environment.
2. Thiết kế DataEnvironment mới sẽ xuất hiện trong đề án. Một kết nối mặc định, gọi là Connection1 xuất hiện trong thiết kế

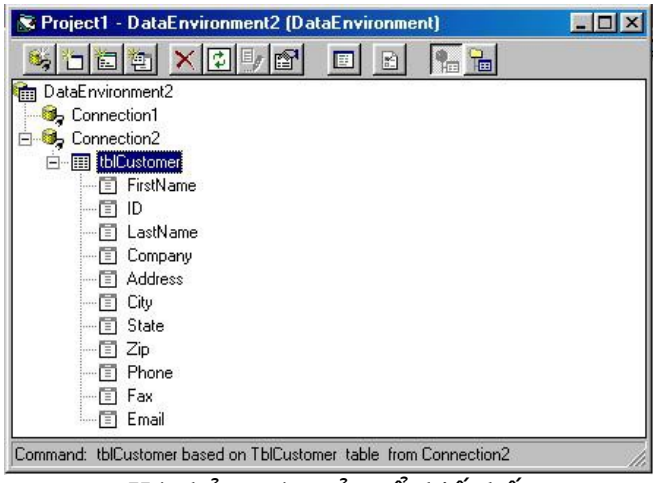


Thiết kế DataEnvironment.

Có thể điều chỉnh một cách thủ công kết nối mặc định trong một thiết kế dataenvironment để nó trỏ đến cơ sở dữ liệu. Nhung nếu có sở dữ liệu đã có sẵn trong cửa sổ Data View, ta chỉ cần kéo và thả bảng vào thiết kế . Để thực hiện điều này, ta làm như sau:

1. Khởi động cửa sổ Data view, chọn một bảng trong thư mục Tables (như là tblCustomer)
2. Kéo bảng lên trên thiết kế DataEnvironment.
3. Một kết nối mới gọi là Connection2 xuất hiện trong thiết kế, với bảng xuất hiện dưới đây.

Đến đây, ta có thể kéo các bảng khác vào thiết kế nếu thích. Khi hoàn tất, ta có :



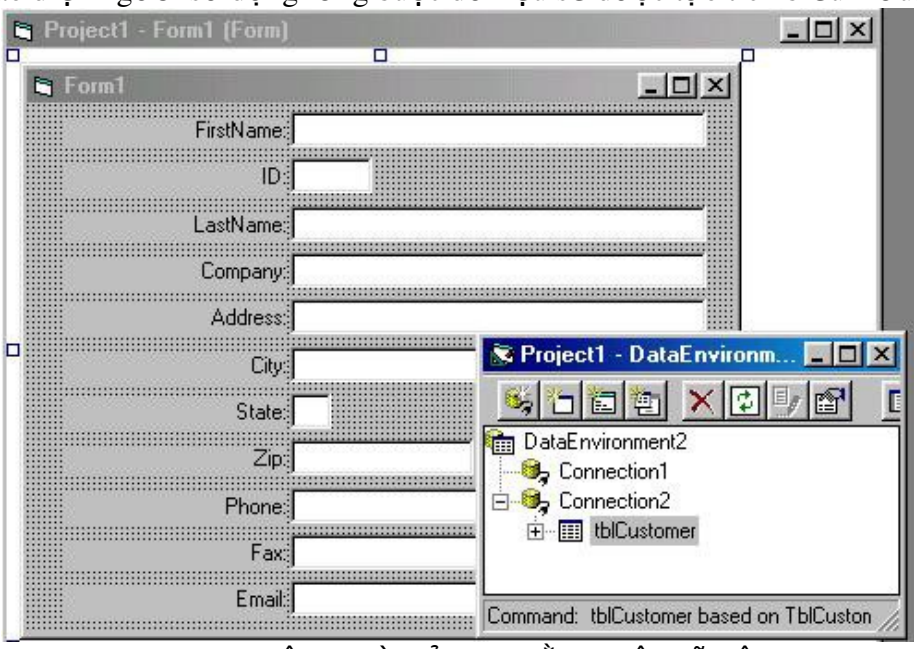
Kéo bảng vào cửa sổ thiết kế.

12.3.1 Tạo một giao diện người sử dụng với thiết kế DATAENVIRONMENT

Ta có thể tạo một giao diện người sử dụng nhanh chóng bằng cách dùng thiết kế DataEnvironment. Thiết kế kết hợp với cơ chế biểu mẫu của VB, cho phép ta dùng kỹ thuật kéo và thả để tạo một giao diện người sử dụng điều khiển bằng cơ sở dữ liệu. Để thực hiện điều này, ta theo các bước sau :

1. Mở biểu mẫu ta muốn dùng làm giao diện người sử dụng.
2. Chọn bảng trong thiết kế Data Environment (Không phải trong cửa sổ Data View).
3. Thả bảng vào biểu mẫu.

Một giao diện người sử dụng rằng buộc dữ liệu sẽ được tạo trên biểu mẫu.



Tạo dao diện người sử dụng rằng buộc dữ liệu.

Thi hành ứng dụng để xem mẩu tin thứ nhất trong cơ sở dữ liệu. Tuy nhiên, không có chức năng duyệt từ mẩu tin này sang mẩu tin khác. Để thực hiện điều đó, ta phải lập trình hoặc dùng một điều khiển dữ liệu, mô tả trong phần sau.

12.4 Sử dụng điều khiển dữ liệu để tạo giao diện người sử dụng

Ta có thể dùng một điều khiển dữ liệu để quản lý kết nối giữa biểu mẫu Visual Basic và một cơ sở dữ liệu. Điều khiển dữ liệu còn cung cấp chức năng duyệt dữ liệu đơn giản, cho phép ứng dụng duyệt qua một recordset, thêm và cập nhật mẩu tin. Phiên bản trước của VB cung cấp 2 loại điều khiển dữ liệu. **DAO Data**, thường được kết nối với cơ sở dữ liệu trên máy cá nhân như Microsoft Access và điều khiển Remove Data (RDC), dùng cho dữ liệu Client / Server. VB6 thêm một điều khiển dữ liệu mới , ADO Data, cho phép ta truy cập mọi loại dữ liệu và không thuộc mô hình quan hệ.

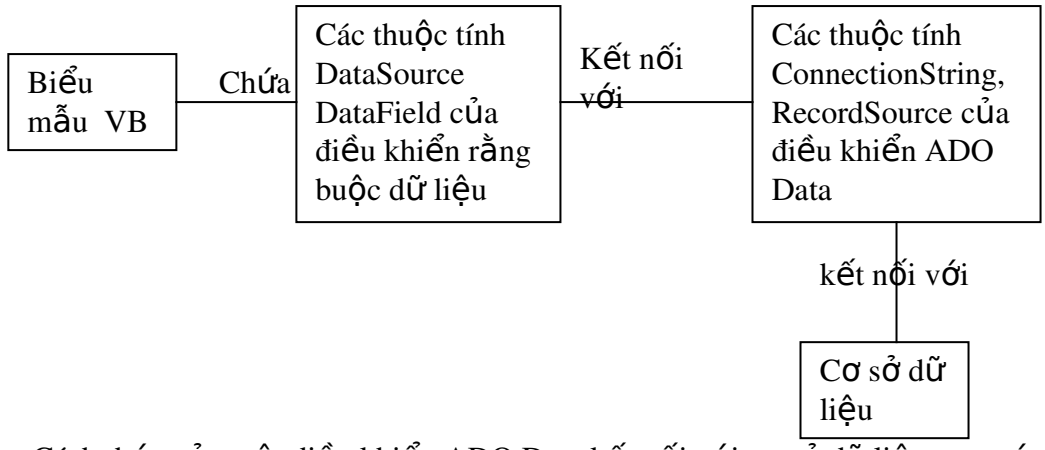
Vậy ta nên dùng điều khiển dữ liệu nào ? Đối với các ứng dụng cơ sở dữ liệu mới trong VB6, ta có thể dùng ADO Data. Nhung ta cũng có thể dùng mô hình cũ hơn, như là DAO hay RDO để bảo trì một cơ sở chương trình hiện hành. Vì các kiểu điều khiển dữ liệu hoạt động tương tự nhau, ta chỉ trình bày điều khiển dữ liệu ADO.

Lưu ý : Điều khiển Data có sẵn cho mọi ấn bản của Visual Basic. Trong phiên bản Learning, tính năng của điều khiển này rất hạn chế. Ví dụ, ta không thể dùng đối tượng Recordset của điều khiển Data để tạo một đối tượng recordset khác.

Do hạn chế này của điều khiển data trong ấn bản Learning, để sử dụng điều khiển với đầy đủ chức năng, ta giả sử rằng điều khiển data ở đây là bản dành cho ***Professional và Enterprise.***

Điều khiển dữ liệu là cách đơn giản nhất để truy cập đến cơ sở dữ liệu trong Visual Basic, dù cho đó là Access hay một hệ Client / Server.

Hình sau đây minh hoạ cách thức điều khiển ADO Data đã kết nối ứng dụng với một cơ sở dữ liệu.



Cách thức của một điều khiển ADO Data kết nối với cơ sở dữ liệu trong ứng dụng.

Lưu ý: Mặc dù điều khiển ADO Data là giải pháp dễ dàng để kết nối ứng dụng với một cơ sở dữ liệu, nhung đó không phải là giải pháp duy nhất. Sau khi đã quen với cách truy cập cơ sở dữ liệu trong Visual Basic, ta sẽ xem xét việc dùng chương trình để quản lý kết nối với vơ sở dữ liệu.

12.4.1 Kết nối với một cơ sở dữ liệu và làm việc với các mẩu tin

Tạo một ứng dụng dùng điều khiển ADO Data rất đơn giản . Thực ra, nếu những gì ta quan tâm chỉ là duyệt cơ sở dữ liệu thì ta không cần phải lập trình gì cả . Đây là một quy trình gồm 2 bước – quy định thuộc tính ConnectionString và RecordSourse của điều khiển Data, sau đó rằng buộc điều khiển với điều khiển giao diện người sử dụng. Để thực hiện điều này ta theo các bước sau :

1. Bắt đầu một đề án mới của Visual Basic.
2. Dùng menu Project Components, lập một tham chiếu đến " Microsoft ADO Data Control 6.0 (OLEDB)" bằng cách chọn vào hộp đánh dấu trong danh sách.
3. Nhấn nút OK, điều khiển ADO Data xuất hiện trên thanh công cụ của Visual Basic. Nhấn đúp chuột lên điều khiển để tạo một instance của điều khiển trên biểu mẫu.
4. Di chuyển và điều chỉnh điều khiển để cho nó nằm ở góc phải dưới của biểu mẫu, chiếm càng ít khoảng trống càng tốt.
5. Nhấn nút phải chuột lên điều khiển. Từ menu bật ra, chọn ***ADODC Properties.***
6. Hộp thoại *Properties* của điều khiển xuất hiện. Chọn nút tuỳ chọn "*User Connection String* " rồi nhấn Build.
7. Hộp thoại *Data Link Properties* xuất hiện. Đây là hộp thoại ta dùng để kết nối với cơ sở dữ liệu trong ví dụ "*Sử dụng Data View*" ở phần trước. Sử dụng cùng các bước kế nối đến cơ sở dữ liệu ***Novelty*** và nhấn OK khi hoàn tất.
8. Đến đây bạn đã quay lại hộp thoại ***Properties Pages*** của điều khiển *ADO Data*. Kế tiếp, ta thông báo cho điều khiển bảng nào sẽ được dùng. Chọn tab ***RecordSource***, rồi chọn 2 –adCmdTable từ hộp kết hợp *CommandType*.
9. Chọn hộp kết hợp *Table or Stored Procedure Name*. Hộp kết hợp hiển thị danh sách tất cả các bảng trong cơ sở dữ liệu. Chọn **tblCustomer** rồi nhấn OK.

Kết nối đến cơ sở dữ liệu xảy ra khi ứng dụng thi hành. Tuy nhiên, kết nối không có thông báo, bởi vì không có cách nào để hiển thị dữ liệu. Để hiển thị dữ liệu trả về từ một điều khiển dữ liệu, ta phải tạo các điều khiển kết nối rằng buộc với điều khiển dữ liệu. Để thực hiện điều này, theo các bước sau :

1. Tạo 2 hộp văn bản trên biểu mẫu.
2. Quy định thuộc tính DataSource của hộp văn bản là ADODC1, tên của điều khiển.
3. Chọn hộp văn bản thứ nhất và quy đinh thuộc tính **DataField** của nó là một trường của bảng trong cơ sở dữ liệu, chẳng hạn ta chọn *FistName*, một lần nữa giống như với thuộc tính **RecordSource** của cơ sở dữ liệu ta chọn một thuộc tính ***DataField*** của điều khiển rằng buộc sẽ hiển thị là một danh sách xổ xuống trình bày danh sách những gì có sẵn trong cơ sở dữ liệu.
4. Quy định thuộc tính ***DataField*** của hộp văn bản thứ hai là tên của một trường khác trong cơ sở dữ liệu, chẳng hạn như là *LastName*.
5. Thi hành ứng dụng.

12.4.1.1 Sử dụng điều khiển Data để kết nối với một cơ sở dữ liệu.

Sau khi đặt điều khiển ADO Data trên biểu mẫu, người sử dụng có thể duyệt qua các mẩu tin bằng cách nhấn các nút của điều khiển. Điều khiển gồm 4 nút. Lưu ý rằng ở trạng thái mặc đinh, điều khiển dữ liệu không cho phép người sử dụng thêm,xoá mẩu tin. Nếu muốn thực hiện điều này hay làm một hành động nào khác, ta phải lập trình.

Chú ý: Một số điều khiển ActiveX của các nhà cung cấp thứ 3 được thiết kế để thay thế và mở rộng khả năng của điều khiển dữ liệu cung cấp bởi Visual Basic.

12.4.1.2 Sử dụng điều khiển ADO Data để cập nhật mẩu tin

Ta không cần lập trình với điều khiển ADO Data để thực hiện việc cập nhật một cơ sở dữ liệu. Khi người sử dụng sửa đổi một mẩu tin hiển thị bởi điều khiển, mẩu tin đó được cập nhật ngay khi người sử dụng di chuyển sang mẩu tin khác (giả định rằng recordset có thể cập nhật được). Nếu đã quen thuộc với cách cập nhật mẩu tin dưới dạng biểu mẫu và lưới của Microsoft Access, chắc chắn bạn sẽ chờ đợi một phản ứng như vậy.

Còn có những cách khác để thao tác trên một RecordSet bằng chương trình. Cách dễ nhất để thực hiện điều này là sửa đổi giá trị của điều khiển giao diện người sử dụng rằng buộc với điều khiển dữ liệu ; ta còn có thể thao tác với đối tượng Recordset chứa trong điều khiển dữ liệu để cập nhật mẩu tin.

12.4.2 Tạo một giao diện người sử dụng cơ bản

Trong phần trước, ta đã tìm hiểu về cách dùng trang thuộc tính của điều khiển ADO Data để tạo một giao diện người sử dụng đơn giản. Trong phần này, ta tự tạo một giao diện người sử dụng, cũng dùng điều khiển ADO Data, nhưng bằng cách đặt thủ công các thuộc tính quả lý đường dẫn của cơ sở dữ liệu. Qunh trình này cho phép ta hiệu chỉnh ứng dụng và thêm chức năng bổ xung cho nó.

Để kết nối một điều khiển cần dùng dữ liệu với một điều khiển dữ liệu, ta theo các bước sau:

1. Bảo đảm rằng biểu mẫu chứa một điều khiển ADO Data mà thuộc tính RecordSource và ConnectionString của chúng được quy định là nguồn dữ liệu hợp lệ. Chuỗi kết nối tối thiểu ta cần dùng là :

   ```
   Provider = Microsoft.Jet.OLEDB.3.51; Data
   Source = App.parth & "\novelty.mdb"
   ```
2. Quy định thuộc tính DataSource của điều khiển cần dùng dữ liệu là tên của điều khiển Data. (Khi ta dùng cửa sổ Properties của Visual Basic để thực hiện điều này, thuộc tính DataSource hiển thị tên của tất cả các điều khiển Data trên biểu mẫu hiện hành).
3. Nếu điều khiển cần dùng dữ liệu có một thuộc tính DataField, quy định nó là tên trường ta muốn điều khiển hiển thị. Một lần nữa, ta lưu ý rằng nếu mọi thứ được thiết lập đúng, một danh sách các trường sẽ được xổ xuống trong thuộc tính DataField khi ta chọn thuộc tính này trong cửa sổ Properties của Visual Basic.

Lưu ý: Phần lớn nhưng không phải mọi điều khiển cần dùng dữ liệu đều có thuộc tính DataField. Ví dụ điều khiển DataGrid đi kèm với Visual Basic không có thuộc tính DataField, bởi vì điều khiển có thể hiển thị tất cả các trường trong một nguồn dữ liệu.

12.4.2.1 Điều khiển cần dùng dữ liệu

Một điều khiển cần dùng dữ liệu là một điều khiển bất kỳ có thuộc tính DataSource. Thuộc tính DataSource tham chiếu đến một điều khiển dữ liệu ; thuộc tính này kết nối điều khiển giao diện người sử dụng với điều khiển dữ liệu (đến phiên nó lại kết nối, hay rằng buộc giao diện người sử dụng với cơ sở dữ liệu). Điều khiển giao diện người sử dụng đó được nói là rằng buộc với cơ sở dữ liệu thông qua điều khiển dữ liệu.

Sau đây là danh sách của các điều khiển cần dùng dữ liệu đi kèm với Visual Basic:

- **CheckBox** : Điều khiển cung cấp một điều kiện đúng / sai. Nó chủ yếu rằng buộc với trường Boolean, hay Yes/ No trong một cơ sở dữ liệu.
- **ComboBox** : Đây là hộp kết hợp xổ xuống chuẩn của Visual Basic. Ta không dùng điều khiển này cho mục đính truy cập dữ liệu, bởi vì phần danh sách của nó không thể rằng buộc với một nguồn dữ liệu, chỉ có phần văn bản mà thôi. Nếu ta muốn dùng một điều khiển giao diện người sử dụng rằng buộc cho mục đích này, ta cân nhắc để dùng điều khiển mạnh hơn, DBCombo.
- **DBCombo :** Điều khiển cần dùng dữ liệu này hỗ trợ một danh sách xổ xuống tương tự điều khiển hộp kết hợp chuẩn của Visual Basic, nhưng nó có thể điền vào danh sách các dữ liệu lấy từ một bảng của cơ sở dữ liệu.
- **DataGrid :** Lưới hiển thị dữ liệu của cơ sở dữ liệu theo dòng và cột. Phiên bản thương phẩm của điều khiển này là điều khiển Apex True DB Grid.
- **DateTimePicker :** Điều khiển này có thể rằng buộc ngày hoặc giờ trong một cơ sở dữ liệu. Nó giúp người sử dụng chọn ngày, giờ một cách dễ dàng dưới dạng đồ hoạ.
- **DBList** : Điều khiển hộp danh sách này tương tự điều khiển hộp danh sách chuẩn của Visual Basic, nhưng nó có thể điền dữ liệu vào danh sách từ một bảng cơ sở dữ liệu.
- **Hierarchical FlexGrid** : Tương tự điều khiển FlexGrid trong VB5, điều khiển này cho phép thao tác với nhiều mẩu tin quan hệ trong một điều khiển lưới.
- **Image** : Điều khiển này tương tự điều khiển PictureBox, nhưng thiếu một vài tính năng của nó.
- **Label** : Điều khiển này cho phép trình bày văn bản từ một trường cơ sở dữ liệu, nhnhưng ngăn cản người sử dụng sử đổi nó.
- **ListBox** : Đây là hộp danh sách chuẩn của Visual Basic, ta không sử dụng điều khiển này cho mục đích truy cập cơ sở dữ liệu mà sử dụng điều khiển mạnh hợ **DBList**.

- **MaskedEdit** : Điều khiển này tương tự một hộp văn bản, nhưng cung cấp một chức năng kiểm tra nội tại cũng như một hiển thị mặc định gợi ý cho người sử dụng điều kiện nhập vào hộp văn bản.
- **MSChart** : Điều khiển Mschart là một điều khiển chuẩn đi kèm với Visual Basic. Cái khác trong VB6 là khả năng rằng buộc biểu đồ trực tiếp với một điều khiển dữ liệu.
- **MSFlexGrid** : Điều khiển này cho ta trình bày dữ liệu dưới dạng lưới. Ta còn có thể sử dụng điều khiển để xử lý dữ liệu, gộp nhóm và sắp xếp. Phiên bản thương phẩm của điều khiển này là VideoSoft VSLEX.
- **TextBox** : Điều khiển thông dụng này cho phép người sử dụng nhập dữ liệu trực tiếp.

12.4.2.2 Các điều khiển cần dùng dữ liệu của các nhà cung cấp thứ 3

Ngoài các điều khiển cần dùng dữ liệu đi kèm với Visual Basic, ta còn có các điều khiển được cung cấp bởi các nhà cung cấp thứ ba. Thông thường, khi một điều khiển là nhận thức dữ liệu, nhà cung cấp thường dùng từ " cần dùng dữ liệu" hay "rằng buộc", để chỉ rõ điều khiển có thể rằng buộc với một nguồn dữ liệu.

12.4.3 Thao tác trên các mẩu tin thông qua điều khiển ADO Data

Ngoài khả năng cho phép duyệt qua Recordset, điều khiển ADO Data cho phép ta thi hành các hoạt động không đòi hỏi lập trình. Ta có thể dùng chương trình với điều khiển dữ liệu để duyệt qua từng mẩu tin, xoá mẩu tin, và tạo mẩu tin mới.

Phần lớn, chương trình phải viết khi làm việc với điều khiển dữ liệu đều tập trung trên đối tượng Recordset. Một đối tượng Recordset trở nên sẵn sàng khi ta quy định thuộc tính ConnectionString và RecordSource cho nó. Để truy cập một thuộc tính hay phương thức của một đối tượng Recordset của điều khiển dữ liệu trong chương trình, ta tham chiếu đến điều khiển dữ liệu, rồi tham chiếu đến đối tượng Recordset.

Ví dụ: Để di chuyển đến mẩu tin thư nhất của Recordset chứa trong điều khiển dữ liệu tên là datCustomer, ta dùng đoạn chương trình sau:

```
DatCustomer.Recordset.MoveFirst
```

Tại sao không dụng datCustomer.MoveFirst ? Câu trả lời là điều khiển dữ liệu không giống như dữ liệu, thao vào đó, điều khiển dữ liệu dưới dạng đối tượng Recordset. Các thuộc tính bản thân của điều khiển dữ liệu gắn liền với sự xuất hiện và các phản ứng của nó, trong khi đối tượng Recordset có các thuộc tính và phương thức gắn liền với chính dữ liệu.

12.4.3.1 Dùng điều khiển dữ liệu để tạo mẩu tin mới

Để tạo một mẩu tin mới ta có 2 tuỳ chọn:

- Quy định thuộc tính EOFAction của điểu khiển dữ liệu là 2 AddNew. Giải pháp này không đòi hỏi lập trình.

- Dùng phương thức AddNew và Update của đối tượng Recordset của điều khiển dữ liệu. Giải pháp này phức tạp hơn, nhung cho ta khả năng điều khiển trên những gì xảy ra khi người sử dụng muốn tạo mẩu tin mới. Nó cũng thích hợp cho trường hợp ta muốn che dấu điều khiển dữ liệu đối với người sử dụng.

Để cho phép điều khiển dữ liệu tạo mẩu tin mới mà không cần lập trình, ta làm như sau:

1. Trong đề án điều khiển Data, quy định thuộc tính EOFAction của điều khiển Data là 2 – AddNew.
2. Thi hành đề án.
3. Nhấn nút MoveLast của điều khiển data, rồi nhấn Next, Thay vì di chuyển đến mẩu tin cuối cùng trong Recordset, điều khiển dữ liệu tạo một mẩu tin mới. Ta có thể nói rằng mẩu tin này mới vì tất cả các điều khiển rằng buộc trên biểu mẫu đều rỗng.
4. Nhập dữ liệu trong các điều khiển rằng buộc.
5. Dùng nút Previous của điều khiển Data, di chuyển đến mẩu tin trước đó. Mẩu tin mới được lưu vào cơ sở dữ liệu.

Muốn sử dụng phương thức AddNew và Update để tạo một mẩu tin mới, ta làm như sau :

1. Thêm các nút lệnh và các điều khiển khác và giao diện để thể hiện phương thứuc AddNew và Update.
2. Trong sự kiện Click của nút Update, đưa vào dòng chương trình sau DatCustomer.Recordset.Update
3. Trong sự kiện Click của nút Update, đưa vào dòng chương trình sau: DatCustomer.Recordset.AddNew
4. Khi người sử dụng nhập liệu, họ có thể tuỳ chọn nhấn **Update Record** để ghi nhận mẩu tin mới vào cơ sở dữ liệu. Họ còn có thể di chuyển sản mẩu tin khác để lưu nó, điều này cũng đúng với cập nhật mẩu tin.

Điểm quan trọng cần cần hiểu là khi người sử dụng tạo một mẩu tin mới trong giao diện nhập liệu sử dụng điều khiển dữ liệu, nhiều hoạt động sẽ là không hợp lệ bởi vì chưa có mẩu tin hiện hành.

Ví dụ, nếu ứng dụng cho phép người sử dụng tạo một mẩu tin bằng cách quy định thuộc tính **EOFAction** của điều khiển dữ liệu là **AddNew**, rồi cho phép người sử dụng thi hành phương thức **Delete** trên mẩu tin hiện hành, ứng dụng sẽ báo lỗi.

Lỗi xảy ra bởi vì không có mẩu tin bị xoá. Để tránh tình huống này, ta có một vài lựa chọn. Nếu ta đã quen với Visual Basic, lựa chọn hiển nhiên là bẫy lỗi và cấm phương thức Delete. Nhưng có một phương thức tốt hơn để tránh rắc rối này : vô hiệu hoá nút Delete để ngăn cản người sử dụng nhấn nó trong lần đầu tiên .

12.4.3.2 Dùng sự kiện moveComplete để cập nhật giao diện người sử dụng

Ta có thể dùng sự kiện MoveComplete của điều khiển ADO Data để khởi động sửa đổi trong ứng dụng khi người sử dụng di chuyển từ mẩu tin này sang mẩu tin khác.

Sự kiện MoveComplete được kích hoạt sau khi một mẩu tin mới trở thành hiện hành. Đây là một trong vài sự kiện được kích hoạt khi điều khiển di chuyển từ một mẩu tin này sang mẩu tin khác. Các sự kiện khác bao gồm WillChange, được kích hoạt khi điều khiển di chuyển từ mẩu tin này san mẩu tin khác, hay thay đổi một mẩu tin và sự kiện **RecordChangeComplete**, xảy ra khi một mẩu tin được sửa đổi thành công trong cơ sở dữ liệu như một kết quả của hoạt động trong điều khiển dữ liệu.

Ta chủ yếu dùng sự kiện **RecordChangeComplete** để thực hiện các tác vụ sau:

- Thi hành một câu truy vấn trên các mẩu tin liên quan đến mẩu tin chính Microsoft Access gọi nó là giao diện "biểu mẫu chính/ biểu mẫu con".
- Tính toán một giá trị dẫn xuất từ một hay nhiều giá trị trong mẩu tin.
- Quản lý nhiều vấn đề về giao diện người sử dụng để đáp ứng với trạng thái Recordset của điều khiển dữ liệu, thi hành những công việc như là che giấu hoặc vô hiệu hoá tính năng nào đó nếu một mẩu tin hợp lệ vắng mặt.

  ***Lưu ý**: Điều khiển DAO Data cung cấp các sự kiện tương tự những sự kiện mô tả ở đây. Các sự kiện "Will" của điều khiển ADO Data (như là WillMove và WillChange) gắn giống với sự kiện Validate của điều khiển DAO Data, trong khi các sự kiện "Complete" của điều khiển ADO Data (như là RecordChangeComplete và MoveComplete) thì tương tự với sự kiện Reposition của điều khiển DAO Data.*

  Ví dụ : Ta quan tâm đến vùng mà khách hàng cư ngụ. Ta có thể viết chương trình để quy định chia các Tiểu bang, mỗi tiểu bang là một vùng. Các Tiểu bang không có quan hệ kinh doanh sẽ được gắn giá trị Aunassigned. Ta có thể dùng điều khiển ADO Data để trình bày vùng của từng mẩu tin trong biểu mẫu như sau:

  1. Trên biểu mẫu của ứng dụng điểu khiển dữ liệu, ta tạo một điều khiển nhân và đặt tên nó là lblRegion.
  2. Trong thủ tục sự kiện MoveComplete của điều khiển ADO Data, đưa vào đoạn chương trình sau:

```
Option Explicit
Private Sub datCusstomer_MoveComplete ( ByVal AddReason
as ADODB.EventReasonEnum, ByVal pError Ass ADODB.Error,
asStatus As ADODB.EventStatusEnum, ByVal Rs as
ADODB.Recordset )
Dim strST as String
Dim StrRegion as String
If rs.BOF = false and Rs.EOF =False Then
      strST = RS.Field("State") & ""
end if
'Display region
```

```
Select Case strST
      Case “VT”, “NH”, “CT”
      strRegion = “Northeast”
      Case “NC”, “KY”, “AR”
      strRegion = “South”
      Case “OK”, “MN”, “MI”, “OH”
      strRegion = “Midwest”
      Case “MT”
      strRegion = “West”
      Case Else
      strRegion = “Unassigned”
End Select
lblRegion.Caption = strRegion
End Sub.
```

3. Thi hành ứng dụng, ta sẽ thấy các vùng được hiển thị khi ta thay đổi từ màu này sang màu khác.

12.4.3.3 Dùng điều khiển Data để xoá mẩu tin

Để xoá mẩu tin trong một ứng dụng sử dụng điều khiển dữ liệu, ta dùng phương thức Delete của đối tượng Recordset của điều khiển dữ liệu:

```
datCustomer.Recordset.Delete
```

Có một cảnh báo quan trọng liên quan đến việc sử dụng phương thức Delete của đối tượng Recordset với điều khiển Data. Khi xoá một mẩu tin, không có mẩu tin hiện hành xuất hiện để thay thế, recordset không có chỗ đứng. Vì vậy, để giải quyết rắc rối này, ta phải di chuyển sang mẩu tin khác trong Recordset (chủ yếu dùng phương thức MoveNext hay MoveLast của Recordset)

> ***Lưu ý**: Như đã nói trong phần tạo mẩu tin và dùng các sự kiện của điều khiển dữ liệu, ta phải bảo đảm rằng có một mẩu tin hiên hành trong recordset của điều khiển dữ liệu khi ta thi hành phương thức Delete, hoặc là ứng dụng sẽ báo lỗi. Để tránh lỗi này ta phải thiết kế giao diện người dùng sao cho người sử dụng không thể xoá mẩu tin không hiện hữu . Giải pháp hữu hiệu là kiểm tra thuộc tính EOF và BOF của recordset trước khi tiến hành phương thức Delete, nếu BOF hay EOF là True, thì phương thức Delete sẽ thất bại.*

12.4.3.4 Dùng sự kiện WillChangeRecord để bảo đảm dữ liệu hợp lệ

Trong lập trình cơ sở dữ liệu, việc kiểm tra dữ liệu hợp lệ (**Validation**) để đảm bảo rằng dữ liệu nhập vào hệ thống tuân thủ các điều kiện xác định bởi thiết kế ứng dụng.

Một cách để thi hành việc kiểm tra này khi lập trình với điều khiển ADO Data là viết chương trình trong sự kiện WillChangeRecord của điều khiển. Sự kiện này được kích hoạt ngay sau khi mẩu tin được hiển thị bởi điều khiển dữ liệu bị thay đổi. Một tình huống hay gặp là người dùng kích hoạt sự kiện bằng cách di chuyển sang mẩu tin khác sau khi sửa đổi hay tạo một mẩu tin.

Khác với phiên bản trước của Visual Basic vốn sử dụng điều khiển DAO Data, điều khiển **ADO Data** báo lỗi theo từng kiểu hoạt động của điều khiển Data. Điều

khiển DAO Data chỉ phát sự kiện **Validate** và **Reposition** vốn được kịch hoạt với một số lý do. Chương trình phải xử lý thêm để xác định tại sao sự kiện được kích hoạt.

12.4.3.5 Validation ở mức bộ máy cơ sở dữ liệu

Ngoài việc xử lý các **Validation** bổ xung khi dữ liệu được nhập, ta còn có thể thi hành **Validation** ở mức bộ máy cơ sở dữ liệu. Các **validation** này tin cậy hơn, bởi vì nó được áp dụng bất chấp quá trình sửa đổi dữ liệu gì. Nhưng **validation** ở mức này kém linh hoạt, bởi vì nó gần như không can thiệp được. Hơn nữa ta chỉ có thể thi hành validation trên cơ sở dữ liệu chỉ với mức trường, ta không thể tiến hành validation để so sánh giữa 2 trường.

Validation ở mức bộ máy cơ sở dữ liệu là một chức năng của thiết kế cơ sở dữ liệu. Đối với cơ sở dữ liệu Jet, tạo các quy tắc cho **validation** trong phần design view của bảng trên Access là dễ dàng nhất.

Ví dụ, ta muốn bảo đảm mẫu tồn kho không bao giờ được nhập vào bảng **Inventory** mà không có số **catolog**. Để thực hiện điều này đối với Microsoft Access ta đặt trong thuộc tính ***Allow Zero Length*** là No.

12.4.3.6 Làm cho validation rõ ràng hơn bằng Validation Text

Do người nhập liệu thiếu kinh nghiệm sẽ gặp thông báo lỗi do bộ máy cơ sở dữ liệu phát ra khi họ vi phạm quy tắc Validation. ta có thể hiển thị thông báo quen thuộc hơn khi người dùng nhập dữ liệu sai. Ta thực hiện điều này bằng thuộc tính Validation Text của cửa sổ định nghĩa bảng.

Ví dụ : Để thông báo một cách thân thiện hơn khi người sử dụng phạm quy tắc validation của CatalogNumber, ta làm như sau:

1. Trong Access, mở thiết kế của bảng tblInventory.
2. Trong thuộc tính Validation Text của trường CatalogNumber, nhập vào chuỗi ký tự sau :
   Chú ý: Bạn phải gõ một số catalog bắt đầu bằng một ký tự từ A đến M
3. Lưu và đóng thiết kế bảng và trở về Visual Basic.

Khi ta cố sửa giá trị trong trường Catolog Number thành một giá trị hợp lệ -. Ví dụ, sửa thành "Z12" một thông báo lỗi xuất hiện.

12.4.4 Các thuộc tính quan trọng khác của điều khiển ADO DATA

Điều khiển ADO Data có một số thuộc tính bổ sung quản lý cách hoạt động của nó, ta có thể quy đinh hầu hết các thuộc tính vào lúc thiết kế. Vì vậy, ta không cần lập trình.

12.4.4.1 Thuộc tính CommandType

Thuộc tính **CommandType** xác định kiểu lệnh mà điều khiển ADO Data phát ra trên nguồn dữ liệu để lấy về các mẩu tin. Ví dụ trong chương này sử dụng

CommandType là 2- adCmdTable để mở và làm việc trực tiếp với bảng. Tuy nhiên, ta có thể dùng lệnh dạng văn bản hay thủ tục đã lưu trữ để cung cấp dữ liệu cho điều khiển dữ liệu.

Lệnh dạng văn bản là một chuỗi, được tạo ra trong mã nguồn của ứng dụng, và được đưa vào bộ máy cơ sở dữ liệu để xử lý.Đối với các cơ sở dữ liệu quan hệ (như Microsoft Jet, cũng như nhiều hệ cơ sở dữ liệu khác), chuỗi này được cấu tạo dưới câu truy vấn SQL. Tuy nhiên ADO cho phép ta dùng ngôn ngữ bất kỳ mà nguồn dữ liệu có thể hiểu được như là lệnh danh văn bản.

Một thủ tục đã lưu trữ là một câu truy vấn hay các lệnh khác được nhúng trong bản thân cơ sở dữ liệu. Ta chủ yếu tạo một thủ tục đã lưu trữ (store procedure) để tận dụng khả năng quản lý tạp trung của thủ tục truy cập cơ sở dữ liệu, cũng như cải tiến khả năng hoạt động của cấu trúc truy vấn. Cơ sở dữ liệu Jet của Microsoft cung cấp một dạng cơ bản của thủ tục đã lưu trữ gọi là **QueryDef**. **Microsoft SQL Server** cung cấp một bộ máy đầy đủ các mở rộng đến SQL để cho phép ta lập trình các thủ tục đã lưu trữ.

12.4.4.2 Thuộc tính EOFAction

Thuộc tính EOFAction xác định những gì điều khiển dữ liệu thực hiện khi người sử dụng di chuyển đến cuối của Recordset. Nếu ta quy định thuộc tính là 2 – AddNew, điều khiển tạo một bản ghi mới khi người sử dụng đi qua phần cuối cùng của mẩu tin hiện hành. (Nói cách khác, giá trị này làm cho giao diện hoạt động tương tự biểu mẫu của Microsoft Access). Tuy nhiên, nhớ rằng giá trị này không phải là hoạt động mặc định của điều khiển dữ liệu của Visual Basic ; ta phải thay đổi thuộc tính lúc thiết kế để đảm bảo rằng điều khiển có phản ứng như vậy.

Để tạo một mẩu tin mới khi thuộc tính **EOFAction** của điều khiển dữ liệu được quy đinh **AddNew**, ta nhấn nút **MoveLast**, rồi nhấn nút **MoveNext**.

12.4.4.3 Dùng thuộc tính Mode để kiểm soát truy cập đến dữ liệu

Bằng cách quy định thuộc tính Mode của điều khiển ADO Data, ta có thể kiểm soát xem những người sử dụng khác có truy cập cơ sở dữ liệu hay không khi ứng dụng đang thi hành. Ví dụ, bằng cách quy định thuộc tính Mode là 12 – adModeShareExclusive, ứng dụng sẽ được tăng cường khả năng truy cập loại trừ đến dữ liệu - không người sử dụng nào khác có thể truy cập đến nó khi ứng dụng đang thi hành.

Ta còn có thể mở một nguồn dữ liệu chỉ đọc (Read only) bằng cách quy định thuộc tính Mode là 1 – adModeRead ; ứng dụng của bạn sẽ nhận được khả năng truy cập chỉ được đọc dữ liệu. Ưu điểm của giá trị này là, cải tiến khả năng hoạt động, bởi vì bộ máy cơ sở dữliệu không cần quan tâm đến những vấn đề rắc rối như là khoá mẩu tin hay kết nối nhiều người sử dụng xảy ra khi có nhiều hơn một ứng dụng truy cập đến cùng một mẩu tin.

12.5 Tổng kết

Chương này trình bày những khái niệm cơ bản của cơ sở dữ liệu nói chung, cũng như cách thức kết nối dễ dàng nhất của ứng dụng Visual Basic với cơ sở dữ liệu Microsoft Access.

Cần nhớ rằng, mặc dù Visual Basic và Microsoft Access chia sẻ cùng một bộ máy cơ sở dữ liệu, cơ sở dữ liệu kiểu Access không phải là khả năng duy nhất của Visual Basic.

12.6 Hỏi và Đáp

Hỏi : Ta thấy rằng Visual Data Manager không mạnh và dễ sử dụng như Microsoft Access, vậy tại sao ta lại dùng nó ?
Đáp : Nếu ta không có hoặc không thể dùng Microsoft Access thì ta có thể dùng nó.

Hỏi : Thế còn điều khiển DAO Data thì sao ?

Đáp : Trong quyển sách này, ta tập trung vào phiên bản ADO của điều khiển bởi vì nó mạnh hơn điều khiển DAO Data. Mặc dù vậy, ta vẫn có thể dùng DAO khi không thể dùng DAO khi không thể dùng ADO. Để tìm hiểu chi tiết về điều này, tham khảo một số mẹo liên quan đến điều khiển DAO Data.

Hỏi : Điều khiển dữ liệu có vẻ dễ sử dụng nhưng chúng có vẻ bị hạn chế và rối rắm trong vài trường hợp. Có cách nào để thực hiện các chức năng cơ sở dữ liệu trong Visual Basic.
Đáp : Chắc chắn bạn đã được Microsoft cung cấp điều khiển DAO Data (kể từ Visual Basic 3.0) như là một giải pháp không cần lập trình. Đối với các hạn chế của điều khiển dữ liệu, ta có thể có giải pháp là kết hợp 1 hay nhiều điều khiển dữ liệu với lập trình hoặc chỉ sử dụng chương trình.
Hỏi : Có cách nào dùng các điều khiển cần dùng dữ liệu mà không dùng điều khiển dữ liệu không ?

Đáp : Có, mặc dù, nếu ta không dùng điều khiển dữ liệu, ta sẽ phải viết chương trình để quản lý kết nối với cơ sở dữ liệu một cách thủ công hay dùng thiết kế DataEnvironment (như trình bày trong phần "Dùng thiết kế DataEnvironment để tạo giao diện người sử dụng " trong chương này). Điều này không phải là không làm được nhưng hơi phức tạp. Ta sẽ dùng chương trình đối tượng (DAO, RDO hay ADO) để xử lý việc trả về và cập nhật mẩu tin. (Ta còn có thể dùng kiểu lập trình này để cho phép các điều khiển không cần dùng dữ liệu, như là điều khiển TreeView của Visual Basic, để trình bày dữ liệu từ cơ sở dữ liệu).

Hỏi : Ta có thể dùng đối tượng Recordset của điều khiển dữ liệu vào mục đích khác không ?

Đáp : Được, đối tượng Recordset của điều khiển ADO Data tương tự các đối tượng Recordset khác trong ADO. Ta có thể gán một recordset tạo ra trong chương trình ADO với một đối tượng Recordset của đối tượng ADO Data.

13 Các đối tượng truy cập dữ liệu

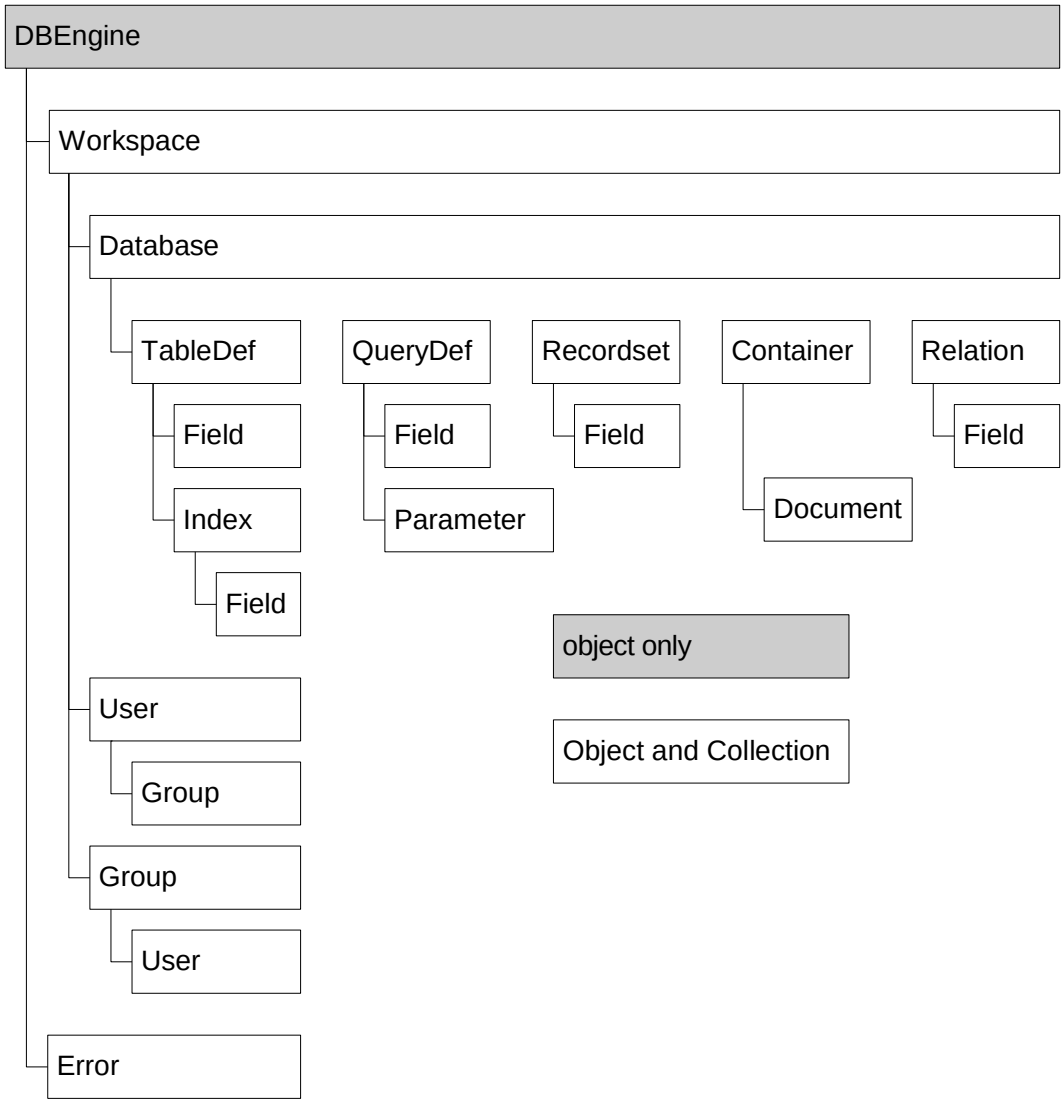
Sử dụng mô hình đối tượng DAO.
Sử dụng DAO để làm việc với dữ liệu.
Tạo đối tượng để thao tác với cấu trúc cơ sở dữ liệu.

Ta có thể dùng **DAO** (*đối tượng truy vấn cơ sở dữ liệu –Data Access Object)* để thao tác với cơ sở dữ liệu thông qua lập trình với Visual Basic. Với DAO, ta có thể thi hành các câu truy vấn, cập nhật giá trị trong các bảng cơ sở dữ liệu và tạo cấu trúc cơ sở dữ liệu bao gồm các bảng , các *câu truy vấn chưa sẵn* và *mối quan hệ* giữa các bảng.

DAO được các nhà lập trình Visual Basic sử dụng để truy cập các cơ sở dữ liệu trên máy tính cá nhân hay Client / Server. Nhưng với sự ra đời của ADO, giờ đây nó chỉ thích hợp để dùng cơ sở dữ liệu Jet mà thôi. Còn đối với việc phát triển các hệ *Client / Server* mới, chủ yếu ta tận dụng thế mạnh của ADO.

13.1 Sử dụng mô hình đối tượng DAO

Mô hình đối tượng **DAO** khá phức tạp với hàng trăm yếu tố. Hàng tá kiểu tập hợp chứa hàng tá đối tượng, mỗi đối tượng lại có thuộc tính, phương thức và các đối tượng con của riêng nó.
Sau đây là mô hình cây phân cấp của đối tượng **DAO:**



Hình 13.1 *Mô hình cây phân cấp DAO, trình bày mối qua hệ giữa các đối tượng cơ sở dữ liệu.*

Một cách để phân loại sự phức tạp của mô hình đối tượng **DAO** là bắt đầu bằng đối tượng ***Database*** và xem xét các tập hợp của nó. Tập hợp là những bộ các đối tượng quan hệ với nhâu; đối tượng ***Database*** có các tập hợp các đối tượng như sau:

Thông qua các tập hợp sở hữu bởi đối tượng ***Database,*** ta có thể thao tác trên dữ liệu và cấu trúc của một cơ sở dữ liệu, tạo các đối tượng cơ sở dữ liệu mới, và kiểm tra cấu trúc và dữ liệu chứa trong một cơ sở dữ liệu.
Trong lập trình DAO, có một tập hợp cốt lõi gồm các kỹ thuật thông dụng được sử dụng gần như cho mọi chương trình. Chúng bao gồm:

Thi hành câu truy vấn SELECT để lấy về các dữ liệu từ cơ sở dữ liệu
Duyệt qua từng mẩu tin trong một *recordset.*
Thi hành câu truy vấn hành động (bao gồm các câu truy vấn *update, delete và append).*
Sửa đổi cấu trúc cơ sở dữ liệu.

Xử lý lỗi phát sinh bởi truy cập cơ sở dữ liệu.

Kỹ thuật này làm việc rất tốt với cơ sở dữ liệu Access.

13.1.1 Lập trình với đối tượng

Để sử dụng tốt các đối tượng của DAO bạn phải hiểu căn bản về đối tượng. Tất cả đối tượng bao gồm các phần tử sau:

Các thuộc tính : là những dữ liệu gắn liền với đối tượng gồm 1 kiểu dữ liệu chuẩn (như interger, string..) hay 1 tập hợp đối tượng trong nó. Ví dụ : ***Recordset*** chứa trong nó là tập hợp *Fields.*

Các phương thức : là các hành động mà đối tượng có thể thi hành. Chúng có thể hoạt động như các hàm (trả về dữ liệu) hay các thủ tục con

Các sự kiện: là các thông điệp mà đối tượng có thể gửi đến các ứng dụng đang sử dụng đối tượng. **DAO** cũng có các sự kiện riêng.

13.1.2 Sử dụng điều khiển DAO Data

Ta có thể sử dụng điều khiển DAO data để kết nối với một cơ sở dữ liệu Jet của Microsoft. Mặc dù với sự xuất hiện của điều khiển mạnh hơn, ***ADO Data,*** việc sử dụng điều khiển ***DAO Data*** bị giảm bớt, nhưng vẫn tồn tại lý do để sử dụng điều khiển cổ điển DAO data. Ngoài ra, nó còn có khả năng kết nối với các nguồn dữ liệu như các tập tin dBASE, văn bản, bảng tính Excel mà không cần phải dùng ODBC (nối kết cơ sở dữ liệu mở).

Lưu ý : *Nguyên lý cơ bản của điều khiển DAO tương tự với điều khiển* ***ADO Data****, chỉ khác ở phần chi tiết. Để sử dụng điều khiển, ta gán tên của tập tin cơ sở dữ liệu vào thuộc tính* ***DatabaseName*** *rồi chọn một bảng hoặc một nguồn dữ liệu nào khác trong thuộc tính* ***RecordSource.*** *Sau cùng, ta sẽ ràng buộc các điều khiển giao diện người sử dụng như là các hộp văn bản với điều khiển bằng cách chỉ ra giá trị cho thuộc tính* ***DataSource*** *và* ***DataField*** *của từng điều khiển giao diện người sử dụng*

13.1.3 Sử dụng thuộc tính Connect của điều khiển DAO Data để truy cập nguồn dữ liệu bên ngoài

Thuộc tính ***Connect*** xác định loại cơ sở dữ liệu kết nối đền điều khiển *Data.* Theo mặc định, thuộc tính này được chỉ định là Microsoft Access., nhưng ta có thể sửa đổi chỉ định này nếu ta muốn kết nối với một kiểu dữ liệu không phải Access. Các kiểu dữ liệu này được gọi là các kiểu dữ liệu bên ngoài.

Jet hỗ trợ các kiểu cơ sở dữ liệu trên máy tính cá nhân sau đây:

dBASE III,IV và 5.0

Phiên bản Excel 3.0, 4.0, 5.0 và 8.0

Phiên bản Foxpro 2.0, 2.5, 2.6 và 3.0

Lotus spreadsheet với định dạng WK1, WK3 và WK4.

Phiên bản Paradox 3.x, 4.x và 5.x

Tập tin văn bản ASCII có phân cách.

Ví dụ : Sử dụng điều khiển Data để nối với file excel 5.0 là excel-db.xls
Thuộc tính : Connect = Excel 5.0; DatabaseName= excel-db.xls; RecordSource = Sheet$ ..
Lưu ý : DAO không hỗ trợ một số hoạt động trên cơ sở dữ liệu bên ngoài. Đặc biệt, cơ sở dữ liệu đó không hỗ trợ các thủ tục DAO như tạo cơ sở dữ liệu, trường , định nghĩa truy vấn. Chúng chỉ hỗ trợ với cơ sở dữ liệu Jet.

13.2 Sử dụng DAO để làm việc với dữ liệu

DAO được dùng chủ yếu để thao tác trên dữ liệu trong một cơ sở dữ liệu đã có. Thi hành truy vấn, cập nhật mẩu tin, và thi hành các hoạt động bảo trì là sở trường của DAO.

13.2.1 Dùng đối tượng DataBase để kết nối với một CSDL

Đối tượng *Database* là nơi để ứng dụng bắt đầu phần lớn các truy cập cơ sở dữ liệu của nó. Để dùng đối tượng ***Database,*** ta bắt đầu lập tham chiếu đến Microsoft DAOs trong menu *Project References* của Visual Basic .. Điều này sẽ cho phép ứng dụng tăng cường truy cập đến mọi đối tượng do DAO cung cấp.
Để thực hiện điều này, ta theo các bước sau:

1. Từ menu *Project,* chọn *References.*
2. Hộp thoại ***References*** xuất hiện. Từ danh sách các thư viện, chọn vào hộp đánh dấu "*Microsoft DAO 3.5.1 Object Library"*.
3. Nhấn OK. Giờ đây, ta có thể dùng đối tượng được cung cấp bởi thư viện đối tượng ***DAO***.

Bước kế tiếp là khai báo một biến đối tượng cơ sở dữ liệu trong chương trình :

```
Dim db As Database
```

Nếu ứng dụng được thiết kế để làm việc với một cơ sở dữ liệu duy nhất, tốt nhất ta nên khai báo biến *Database* ở mức mô-dun của biểu mẫu chính của ứng dụng.
Nếu nhiều biểu mẫu ứng dụng cần truy cập đến một cơ sở dữ liệu, ta nên tạo một lớp quản lý kết nối đến cơ sở dữ liệu.

13.2.1.1 Sử dụng phương thức OpenDatabase để tạo một đối tượng Database

Đây là phương thức của đối tượng ***Workspace***
Ta dùng phương thức ***OpenDatabase*** để tạo một đối tượng ***Database***. Phương thức này trả về một đối tượng ***Database*** (do đó, trước khi dùng ***OpenDatabase,*** ta phải khai báo một biến đối tượng có kiểu là ***Database*** để chứa giá trị trả về của phương thức).

```
Dim db As Database
Set db = OpenDatabase("..\..\DB\novelty.mdb")
Msgbox "The database " & db.Name & "is now open."
```

Phương thức ***OpenDatabase*** có tham số bắt buộc – tên của cơ sở dữ liệu ta muốn mở.

Cú pháp : (Đối tượng ***Workspace*** được mặc định)
OpenDatabase(dbName, *[options], [readonly],[connect]*)

Tham số	Mô tả
Options	Nếu tham số này là *true,*cơ sở dữ liệu mở trong chế độ loại trừ; không ai có thể mở cơ sở dữ liệu trong chế độ loại trừ. Nếu giá trị này là *false*, những người khác có thể mở cơ sở dữ liệu.
Readonly	Nếu tham số này là *True,* ta không thể sửa đổi cơ sở dữ liệu.
Connect	một chuỗi chỉ ra cách thức kết nối với cơ sở dữ liệu; chuỗi thường chỉ được dùng cho những nguồn dữ liệu *Client / Server* và *ODBC.*

13.2.1.2 2.1.2 Sử dụng phương thức Execute để thi hành truy vấn hành động

Sử dụng *Execute* của đối tượng ***Database*** để thi hành một câu lệnh SQL trên SQL. Phương thức này nên dùng để:

Cập nhật, xoá hay sao chép mẩu tin (trong Access / Jet, ta gọi là các truy vấn hành động)

Sửa cấu trúc dữ liệu

Các câu truy vấn SELECT được thi hành qua phương thức ***OpenRecordset*** của ***Database.***

13.2.2 Sử dụng đối tượng Recordset

Ta sử dụng đối tượng *Recordset* để thao tác với các mẩu tin trong DAO. Đối tượng ***Recordset*** cung cấp một giao diện hướng đối tượng cho mô hình cơ sở dữ liệu quan hệ liên quan đến các bảng được chia thành những mẩu tin và trường.
Để tạo một *recordset*, ta chủ yêu sử dụng phương thức ***OpenRecordset***.

13.2.2.1 Sử dụng phương thức OpenRecordset để tạo một đối tượng Recordset

Trong DAO, các đối tượng *Database, Connection, QueryDef, TableDef,* và *Recordset* đều có phương thức ***OpenRecordset***. Tất cả đều được dùng với cùng mục đích; truy cập dữ liệu chứa trong cơ sở dữ liệu.
Phương thức ***OpenRecordset*** thực sự là một hàm trả về một đối tượng *Recordset*, ta cần khai báo một đối tượng *Recordset* trước khi sử dụng *OpenRecordset.*

```
Dim db As Database
Dim rs As Recordset

Set db = OpenDatabase("..\..\db\novelty.mdb")
Set rs = db.OpenRecordset("tblCustomer")
```

Tham số bắt buộc duy nhất của phương thức ***OpenRecordset*** là nguồn dữ liệu.
Đây là chuỗi ký tự, theo nguyên tắc là tên của một bảng hoặc một định nghĩa truy vấn chứa sẵn. Nhưng nó cũng có thể là một câu lệnh SELECT SQL:

```
Dim db As Database
Dim rs As Recordset
Set db = OpenDatabase("..\..\db\novelty.mdb")
Set rs = db.OpenRecordset("SELECT * FROM tblCustomer" & _
        " Order By [LastName]")
```

13.2.2.2 Tránh dùng dấu trích dẫn

Ta thường có câu lệnh SQL:

```
SELECT * FROM tblCustomer WHERE [LastName]="Smith"
```

Làm thế nào để đưa vào *OpenRecordset* mà không bị nhầm dấu ":

Giải pháp sử dụng dấu đơn :

```
Dim db As Database
Dim rs As Recordset
    Set db = OpenDatabase("..\..\db\novelty.mdb")
Set rs = db.OpenRecordset("SELECT * FROM tblCustomer" & _
        " WHERE [LastName]='Smith'")
```

Nếu tên người sử dụng là nhập vào từ text box txtLastName.Text thì đưa vào thế nào :

```
Dim db As Database
Dim rs As Recordset
Set db = OpenDatabase("..\..\db\novelty.mdb")
Set rs = db.OpenRecordset("SELECT * FROM tblCustomer" & _
        " WHERE [LastName]='" & txtLastName.Text &"'")
```

13.2.3 Chỉ ra các tuỳ chọn cho Recordset

Tham số *Options* của phương thức ***OpenRecordset*** xác định một số cách thức thao tác với mẩu tin.

Hằng	**Ý nghĩa**
dbOpentable	Trong một workspace của Microsoft Jet, tạo một đối tượng Recordset kiểu bảng (bảng được đánh chỉ mục chỉ sử dụng một bảng)
dbOpenDynamic	Trong một workspace kiểu ODBCDirect, mở một đối tượng Recordset kiểu dynamic (cập nhật được dữ liệu khi người khác sửa đổi và truy vấn dữ liệu qua nhiều bảng)
dbOpenDynaset	Mở một đối tượng kiểu Dynaset (tham chiếu đến

	recordset của nhiều bảng hoăc từ nhiều cơ sở dữ liệu khác)
dbOpenForwardonly	Mở một đối tượng Recordset mà con trỏ của nó chỉ có thể di chuyển tới

13.3 Sử dụng đối tượng Field để thao tác với các trường

Đối tượng ***Field*** thể hiện một trường trong một cấu trúc dữ liệu. Các đối tượng ***TableDef, Recordset, Relation*** và ***Index*** đều chứa các tập hợp trường.

Ta có thể lấy giá trị của một trường bằng cách kiểm tra giá trị của thuộc tính ***Value*** của một đối tượng ***Field*** . (Bởi vì thuộc tính *Value* là thuộc tính mặc định của đối tượng *Field,* ta chỉ cần tham chiếu đến đối tượng *Field;* ta không nhất thiết phải tham chiếu tường minh đến thuộc tính Value).

13.4 Sửdụng các phương thức duyệt với đối tượng Recorset

Sau khi tạo xong một đối tượng **Recordset,** ta có thể dùng các phương thức duyệt để di chuyển qua từng mẩu tin trong *recordset*. Ta chủ yếu thực hiện điều này trong trường hợp cần lấy về dữ liệu từ mọi mẩu tin trong một recordset, mặc dù ta còn có thể sử dụng chúng để cho phép người sử dụng chỉ duyệt qua các mẩu tin.

Các phương thức duyệt của một đối tượng *Recrdset* gồm có:

MoveFirst : di chuyển đến mẩu tin đầu tiên trong recordset.
MoveNext : di chuyển đến mẩu tin kế tiếp trong recordset.
MovePrevious : di chuyển về mẩu tin trước đó trong recordset.
MoveLast: di chuyển đến mẩu tin cuối cùng trong recordset.
Move : di chuyển một số mẩu tin đã được chỉ định trước.

13.4.1 Sử dụng BOF và EOF để duyệt qua Recordset

Ngoài các phương thức trên, đối tượng ***Recordset*** cung cấp 2 thuộc tính cho ta biết khi ta di chuyển về đầu hoặc về cuối *recordset.*

*Thuộc tính **EOF**(end of file) là True* khi ta di chuyển quá mẩu tin cuối cùng của *recordset.*
*Thuộc tính **BOF**(begin of file)* là *True* khi ta di chuyển đến một vị trí trước mẩu tin thứ nhất trong *recordset.*

Hình 13.2 *BOF và EOF trong một Recordset.*

```
Do until EOF
        'perform action on the data or read values from fields
        rs.MoveNext
Loop
```

13.4.2 Dùng BOF và EOF để xác định một Recordset có rỗng hay không

Thuộc tính ***BOF*** và ***EOF*** luôn có sẵn, thậm chí trong một *recordset* không có mẩu tin. Trên thực tế, cách tốt nhất để xem *recordset* có chứa mẩu tin hay không là kiểm tra giá trị của cả hai thuộc tính ***EOF*** và ***BOF***. Nếu cả ***BOF*** và ***EOF*** đều là *True*, *recordset* không chứa mẩu tin.

13.4.3 Dùng thuộc tính RecordCout để xác định số mẩu tin trong một recordset

Ta dùng thuộc tính RecordCount để xác định số mẩu tin trong một recordset. Nhưng lưu ý rằng giá trị của RecordCount sẽ không hợp lệ nếu ta chưa chuyển đến mẩu tin cuối cùng trong *Recordset.*

Bởi vì Jet xử lý truy vấn trên 2 giai đoạn. Giai đoạn đầu chỉ trả về dữ liệu đủ cho ứng dụng hoạt động mà không bị chậm trễ trong xử lý. Giai đoạn thứ hai thi hành bên dưới, trả về toàn bộ dữ liệu yêu cầu để truy vấn hoàn chỉnh. Jet thực hiện điều này chỉ nhằm mục tiêu tăng khả năng hoạt động, vì vậy, việc thi hành ứng dụng không bị cản trở trong khi chờ trả về một khối lượng dữ liệu lớn. Ta không điều khiển trực tiếp quá trình này, nhưng ta cần biết rằng nó tồn tại và cách thức làm việc với nó khi ta muốn biết chính xác có bao nhiêu mẩu tin đang có trong một *recordset* chẳng hạn.

```
Option Explicit
' References: Microsoft DAO 3.51 Object Library
Dim db As Database
Private Sub Form_Load()
```

```
    Set db = OpenDatabase("..\..\db\novelty.mdb")
End Sub
Private Sub cmdCount_click()
    Dim rs As Recordset
    Set rs = db.OpenRecordset("tblInventory")
    rs.MoveLast
    MsgBox "There are " & rs.RecordCount & _
        " items in inventory.", vbInformation
End Sub
```

Lưu ý : Nếu muốn xác định số mẩu tin là 0 dùng thuộc tính ***BOF*** và ***EOF*** thay vì dùng **RecordCount**

13.4.4 Dùng phương thức Edit để sửa đổi giá trị trong một mẩu tin

Ta có thể sửa đổi mâu tin hiện hành trong một đối tượng *Recordset* cập nhật được bằng cách dùng phương thức ***Edit*** và ***Update*** của *recordset*. muống sửa đổi giá trị của một trường trong một *Recordset,* theo các bước sau :

1. Dùng các phương thức duyệt của đối tượng ***Recordset*** để di chuyển đến mẩu tin cần sửa đổi.
2. Thi hành phương thức ***Edit*** của *recordset.*
3. Dùng tập hợp ***Fields*** của đối tượng ***Recordset*** để gán giá trị cho trường trong mẩu tin :

   ```
   rs.Fields("LastName")= "Smith"
   ```

   hoặc

   ```
   rs!LastName = "Smith"
   ```

4. Lưu mẩu tin vào cơ sở dữ liệu bằng cách dùng phương thức ***Update*** của *Recordset.*

13.4.5 Sử dụng phương thức AddNew và Update để tạo mẩu tin mới

Ta có thể tạo một mẩu tin mới trong đối tượng ***Recordset*** cập nhật được bất kỳ bằng cách dùng phương thức ***AddNew*** và ***Update***. Tạo một mẩu tin mới trong một *recordset* là quá trình 3 bước:

1. Thi hành phương thức ***AddNew*** của *RecordSet.* Nó thêm một mẩu tin mới, trắng vào cuối của *recordset.*
2. Gán giá trị cho mẩu tin mới bằng cách sử dụng câu lệnh gán mà ta thường dùng với các trường cơ sở dữ liệu.
3. Dùng phương thức ***Update*** để ghi mẩu tin vào cơ sở dữ liệu.

```
Option Explicit
' References: Microsoft DAO 3.51 Object Library
Private Enum TextBoxes
    txtProduct = 0
    txtCatalogNumber = 1
    txtWholesalePrice = 2
    txtRetailPrice = 3
End Enum
Private db As Database
Private rs As Recordset
Private x As Integer
Private Sub Form_Load()
    Set db = OpenDatabase("..\..\DB\novelty.mdb")
    Set rs = db.OpenRecordset("tblInventory")
    For x = txtProduct To txtRetailPrice
        TextBox(x).Enabled = False
    Next x
    cmdSave.Enabled = False
    cmdNew.Enabled = True
End Sub
Private Sub cmdNew_Click()
    ' Create a new record
    rs.AddNew
    ' Enable data entry controls
    For x = txtProduct To txtRetailPrice
        TextBox(x).Enabled = True
    Next x
    TextBox(txtProduct).SetFocus
    cmdSave.Enabled = True
    cmdNew.Enabled = False
End Sub
Private Sub cmdSave_Click()
    ' Map UI controls to fields
    rs.Fields("Product") = TextBox(txtProduct)
```

```
rs.Fields("CatalogNumber") = TextBox(txtCatalogNumber)
rs.Fields("WholesalePrice") = TextBox(txtWholesalePrice)
rs.Fields("RetailPrice") = TextBox(txtRetailPrice)
' Commit data to database
rs.Update
' Clear out UI
For x = txtProduct To txtRetailPrice
   TextBox(x).Text = ""
   TextBox(x).Enabled = False
Next x
cmdSave.Enabled = False
cmdNew.Enabled = True

End Sub
```

13.4.6 Sử dụng AppendChunk để nối dữ liệu vào một trường nhị phân

Ta có thể chứa dữ liệu nhị phân vào trong cơ sở dữ liệu. Dữ liệu nhị phân bao gồm hình ảnh hay tập tin âm thanh – nghĩa là bất kỳ những gì ta muốn chứa không chỉ là giá trị số hay chuỗi ký tự.

Lưu ý: *Microsoft Access tham chiếu đến trường nhị phân như là trường OLE Object (đối tượng OLE). Ngoài ra, ta có thể dùng kỹ thuật* ***AppendChunk*** *này để nối thêm dữ liệu vào một trường memo cũng như trường nhị phân. Các phương thức thao tác trên trường nhị phân tồn tại gần như đồng dạng trong cả 3 mô hình đối tượng truy cập dữ liệu –***DAO, RDO** *và* **ADO.**

Khi ta gán một giá trị cho trường nhị phân trong chương trình, ta cần thi hành một số bước bổ sung để lấy dữ liệu trong trường. Bởi vì dữ liệu nhị phân không có chiều dài cố định như các kiểu dữ liệu khác; một mẩu dữ liệu nhị phân có thể lên đến hàng mega- byte hoặc hơn.

Vì vậy, để đặt một mẩu dữ liệu nhị phân trong cơ sở dữ liệu, trước hết ta phải chia nó thành nhiều đoạn dữ liệu nhị phân, ta nối nó vào trường bằng cách dùng phương thức ***AppendChunk*** của đối tượng ***Field*** của *recordset.*

13.4.7 Sử dụng phương thức Close để đóng Recordset

Ta thực hiện điều này khi chương trình hoàn tất sử dụng đối tượng ***Recordset***:

```
rs.Close
```

Điểm đặc biệt quan trọng khi đóng một đối tượng *Recordset* là nếu đối tượng đặt một khoá (*lock)* trên bảng . Đóng một *Recordset* sẽ nhả khoá ứng dụng thiết lập trên đối tượng cơ sở dữ liệu, cho phép người sử dụng khác truy cập đến nó.

Lưu ý rằng trong **DAO,** *Workspace, Connection, Database và QueryDef* đều có phương thức *Close*

13.5 Tìm kiếm dữ liệu trong Recordset và bảng

Sau khi tạo một cơ sở dữ liệu và nhập liệu, ta cần một cách để định vị các mẩu tin riêng rẽ trong một *recordset.* Quá trình định vị một mẩu tin riêng rẽ trong recordset theo một tiêu chí chỉ định trước nào đó trong chương trình gọi là tìm kiếm.

Tìm kiếm khác với truy vấn là một truy vấn thì trả về một *recordset.* Tìm kiếm duyệt qua các mẩu tin trong recordset hiện hành để tìm ra một mẩu tin thoả mãn tiêu chí chỉ định.

Có một số kỹ thuật tìm kiếm dữ liệu. Sử dụng phương thức nào là tuỳ thuộc vào loại cấu trúc dữ liệu ta đang truy cập :

> Nếu ta đang làm việc với một recordset, ta dùng các phương thức ***Find-*** *FindFirst, FindNext, FindLast* và *FindPrevious.*
> Nếu ta truy cập trực tiếp đến một recordset kiểu bảng, ta có thể dùng phương thức ***Seek*** để định vị mẩu tin. Phương thức này khó lập trình hơn, nhưng nó nhanh hơn vì ta có thể dùng một chỉ mục của bảng với phương thức ***Seek.***

13.5.1 Sử dụng phương thức Find để định vị mẩu tin trong một recordset

Để tìm một mẩu tin trong một *Recordset*, ta dùng một trong bốn phương thức tìm kiếm của đối tượng ***Recordset :***

FindFirst
FindLast
FindNext
FindPrevious

Cú pháp của bốn phương thức như nhau - để sử dụng một trong bốn phương thức tìm kiếm này, ta truyền một mệnh đề WHERE của SQL vào phương thức chỉ ra thông tin ta cần tìm kiếm. Sau khi thi hành phương thức, mẩu tin hiện hành trong đối tượng *Recordset* trở thành mẩu tin thoả mãn tiêu chí WHERE. Nếu tìm kiếm không định vị được mẩu tin yêu cầu, thuộc tính *NoMatch* của đối tượng *Recordset* có giá trị là *True*.

Loại phương thức ta dùng cũng xác định cách thức tìm thấy mẩu tin. Ví dụ, nếu dùng *FindFirst* , bộ máy cơ sở dữ liệu sẽ di chuyển đến mẩu tin thứ nhất trong *recordset* thoả tiêu chí. *FindNext* và *FindPrevious*, ngược lại, tìm các mẩu tin hiện hành.

Điểm quan trọng cần lưu ý là, khác với một câu truy vấn SELECT của SQL, tìm kiếm không sinh ra một *recordset*. Khi bộ máy cơ sở dữ liệu tìm ra mẩu tin thoả

tiêu chí tìm kiếm, nó di chuyển đến mẩu tin đó; mẩu tin trở thành mẩu tin hiện hành. Nếu không có mẩu tin tìm thấy, mẩu tin hiện hành sẽ được giữ nguyên và thuộc tính ***NoMatch*** của đối tượng *Recordset* có giá trị là *True.*

13.5.2 Sử dụng phương thức Seek để thi hành tìm kiếm theo chỉ mục

Để tiến hành một tìm kiếm trên một chỉ mục, trước hết bảng phải có một chỉ mục.

Ví dụ sau đây, giả định rằng ta có một bảng *tblCustomer* với một chỉ mục trên trường *LastName.* Tên của chỉ mục là *LastNameIndex.*

```
Option Explicit
' References MS DAO 3.51 Object Library
Private db As Database
Private rs As Recordset
Private Sub Form_Load()
   Set db = OpenDatabase("..\..\db\novelty.mdb")
   Set rs = db.OpenRecordset("tblCustomer", dbOpenTable)
End Sub
Private Sub cmdSeek_Click()
   rs.Index = "LastNameIndex"
   rs.Seek "=", txtLastName.Text
   If rs.NoMatch Then
        ' not found
        MsgBox "No customer by that name was found.", vbExclamation
   Else
        ' return info
        MsgBox rs!Address & vbCrLf & _
             rs!City & ", " & rs!State & "  " & _
             rs!Zip & vbCrLf & _
             rs!Phone, _
             vbInformation, _
             rs!FirstName & " " & rs!LastName
   End If
End Sub
```

Điểm hạn chế của phương thức *Seek* là khả năng tìm kiếm trên một trường duy nhất và nó phải được đánh chỉ mục. Ngoài ra nó chỉ sử dụng các toán tử sau : >;<;>=;<=;=

Khác với phương thức *Seek* , vốn hạn chế thì phương thức Find cho phép ta sử dụng các toán tử có sẵn của SQL như *Like , in* .

13.5.3 Lặp qua suốt tập hợp Indexes của TableDef

Để xác định các chỉ mục tồn tại trong một bảng, ta có thể lặp qua suốt tập hợp *Indexes* của đối tượng *TableDef.*

```
Option Explicit
' References MS DAO 3.51 Object Library
Private db As Database
Private rs As Recordset
Private Sub Form_Load()
   Set db = OpenDatabase("..\..\db\novelty.mdb")
End Sub
Private Sub cmdShowIndexes_Click()
   Dim td As TableDef
   Dim ind As Index
   Dim f As Field
   Set td = db.TableDefs("tblCustomer")
   For Each ind In td.Indexes
      Debug.Print ind.Name
      For Each f In ind.Fields
         Debug.Print "   On field: " & f.Name
      Next
   Next
End Sub
```

Lưu ý:*Có thể dùng đoạn chương trình tương tự như trên để lặp qua suốt các tập hợp khác của DAO. Ví dụ, có thể dùng chương trình này để xem tất cả cơ sở dữ liệu trong một workspace, các định nghĩa truy vấn QueryDef trong một cơ sở dữ liệu, hoặc là các trường trong một* ***TableDef.***

13.5.4 Sử dụng thuộc tính Bookmark để ghi nhớ vị trí trong một Recordset

Khi ta tiến hành các tác vụ trên một đối tượng *Recordset,* ta thường di chuyển đây đó trong *recordset*. Dùng thuộc tính ***Bookmark*** của đối tượng ***Recordset*** để giữ lại một vị trí sao cho ta có thể quay lại ví trí đó sau này.

Thuộc tính ***Bookmark*** cho ta một cách hiệu quả di chuyển giữa hai hay nhiều mẩu tin. Sử dụng *bookmark* thì nhanh hơn là lặp đi lặp lại phương thức *Find* để di chuyển qua từng mẩu tin.

Mỗi mẩu tin trong một *recordset* có thể đánh dấu *bookmark* được có một số hiệu *bookmark* duy nhất để ta có thể đọc ra hay ghi vào tại một thời điểm bất kỳ. Tuy nhiên, *bookmark* không được chứa trong cơ sở dữ liệu; nó được phát sinh tự động khi một đối tượng *recordset* được tạo và bị huỷ bỏ khi đối tượng *recordset* bị huỷ.

Để sử dụng *bookmark,* ta theo các bước sau:

1. Di chuyển đến vị trí trong *recordset* ta muốn đánh dấu *bookmark.*
2. Để gán giá trị thuộc tính *bookmark* của đối tượng *Recordset* cho một biến chuỗi. Nó sẽ lưu *bookmark* duy nhất cho mẩu tin hiện hành.
3. Khi ta muốn quay lại mẩu tin, đổi thuộc tính *Bookmark* của *recordset* là giá trị của biến chuỗi. Mẩu tin hiện hành sẽ được đổi sang mẩu tin *bookmark.*

Lưu ý: *Không phải mọi kiểu đối tượng* ***Recordset*** *đều hỗ trợ thuộc tính* ***Bookmark.*** *Để xác định xem ta có thể đánh dấu bookmark trên một kiểu recordset hay không , kiểm tra giá trị của thuộc tính* ***bookmarkable.*** *Nếu thuộc tính này là* ***True****, ta có thể đánh dấu bookmark trên* ***Recordset.***

13.5.5 sử dụng tập hợp Errors và đối tượng Error để xử lý lỗi

Ứng dụng có thể xử lý lỗi trong **DAO** bằng cách sử dụng đối tượng ***Error*** và tập hợp ***Errors***. Bởi vì ***Errors*** có sẵn trong lập trình cơ sở dữ liệu, một hoạt động nào đó đều có thể phát sinh nhiều hơn một lỗi. (Điều này chính xác với lập trình *Client / Server* , trong đó, ví dụ như khi server bị treo, tầng trung gian bị ảnh hưởng theo và trình điều khiển *ODBC* sẽ bị thất bại). Cả 3 tầng này đều phát sinh thông báo lỗi của riêng chúng, nhưng ta sẽ không thấy các thông báo lỗi này trừ phi ta có một tập hợp để duyệt qua).

Tập hợp *Errors,* thay vì đối tượng ***Error*** duy nhất, cho phép ta duyệt qua tất cả lỗi để xác định nguyên nhân vấn đề.

Ví dụ sau đây trình bày cách lặp xuyên qua tập hợp ***Errors*** của ***DBEngine*** để xem các lỗi phát sinh.

```
Option Explicit

' References: MS DAO 3.51

Dim db As Database

Dim rs As Recordset

Private Sub cmdBadFileName_Click()

On Error GoTo ErrHandler

   Set db = OpenDatabase("..\..\DB\slez.mdb")

Exit Sub

ErrHandler:

Dim DBError As Error
```

```
    Debug.Print "Contents of DBEngine Errors Collection"
    Debug.Print "-------------------------------------"
    For Each DBError In DBEngine.Errors
        Debug.Print DBError.Description
    Next
End Sub
```

13.6 Tạo đối tượng để thao tác trên cấu trúc của một CSDL

DAO cung cấp các thủ tục rất phong phú dùng để tạo cơ sở dữ liệu, bảng, trường, và các định nghĩa truy vấn. Ngoài ra, DAO còn cho phép ta tạo kiểu dữ liệu hiệu chỉnh mới trong ứng dụng, chẳng hạn như các thuộc tính hiệu chỉnh của các đối tượng cơ sở dữ liệu và những kiểu tài liệu cơ sở dữ liệu mới.

13.6.1 Tạo một CSDL

Ta có thể sử dụng phương thức ***CreateDatabase*** của đối tượng ***DBEngine*** trong **DAO** để tạo cơ sở dữ liệu Jet của Microsoft.

Khi tạo một cơ sở dữ liệu, ta phải cung cấp tên một tập tin (thường có phần mở rộng là MDB) và một *locale. Locale* là một chức năng của ngôn ngữ được dùng bởi người sử dụng ứng dụng; nó chỉ ra thứ tự đối chiếu cho cơ sở dữ liệu. Thứ tự đối chiếu xác định cách thức lưu trữ các dữ liệu dạng văn bản trong cơ sở dữ liệu theo mặc định.

Lưu ý: *Phương thức* ***CreateDatabase*** *của đối tượng* ***DBEngine*** *thay thế dòng lệnh* ***CreateDatabase*** *thường dùng trong các phiên bản cũ của* ***DAO****. Nhưng bởi vì* ***DBEngine*** *là một trong những đối tượng DAO mà ta không cần tham chiếu tường minh trong chương trình, ta có thể tiếp tục sử dụng* ***CreateDatabase*** *như cách cũ, không có tham chiếu đến đối tượng* ***DBEngine*** *trước nó. Tuy nhiên, nếu muốn chính xác 100%, ta dùng DBEngine.CreateDatabase thay vì CreateDatabase.*

Ví dụ sau tạo một cơ sở dữ liệu. Ngoài ra, nó còn nối thêm một bảng duy nhất với 2 trường vào cơ sở dữ liệu .

```
Option Explicit
' References MS DAO 3.51
Private db As Database
Private td As TableDef
Private f As Field
Private Sub cmdCreate_Click()
On Error GoTo ErrHandler
    Set db = DBEngine.CreateDatabase("..\..\DB\newdb.mdb", dbLangGeneral)
    Set td = New TableDef
    Set f = td.CreateField("LastName", dbText, 50)
```

```
    td.Fields.Append f
    Set f = td.CreateField("FirstName", dbText, 50)
    td.Fields.Append f
    td.Name = "tblSupplier"
    db.TableDefs.Append td
    Set db = Nothing
    MsgBox "The database newdb.mdb has been created."
Exit Sub
ErrHandler:
    If Err = 3204 Then
        MsgBox "Try deleting the database 'newdb.mdb' first, pal."
    Else
        MsgBox Err.Description
    End If
End Sub
```

Bởi vì phương thức ***CreateDatabase*** phát ra một lỗi nếu cơ sở dữ liệu đã có sẵn, chương trình này chứa phần xử lý lỗi thông báo cho người sử dụng những gì cần làm khi cơ sở dữ liệu đã có sẵn. Cũng lưu ý rằng ta không cần tạo một đối tượng ***DBEngine.*** Nó luôn có sẵn để sử dụng bởi ứng dụng.

13.6.2 Sử dụng đối tượng TableDef để thao tác với bảng

Ta sử dụng đối tượng *TableDef* để thao tác với cấu trúc của các bảng trong ứng dụng. Ta có thể dùng đối tượng *TableDef* để tạo bảng mới hoặc thay đổi bảng hiện hành.

13.6.2.1 Sử dụng đối tượng TableDef để tạo một bảng mới

Để tạo một bảng mới, ta tạo một đối tượng kiểu ***TableDef*** và sau đó nối nó vào tập hợp ***TableDefs*** bằng cách sử dụng phương thức ***Append*** của tập hợp. (Cách làm này tương tự cho nhiều kiểu đối tượng DAO khác dùng để tạo các thành phần cố định của một cơ sở dữ liệu).

Ví dụ sau đây dùng *TableDef* để tạo một bảng mới. Đoạn chương trình này sử dụng đối tượng ***Field*** và tập hợp *Fields;* sẽ được trình bày trong phần sau. Lưu ý rằng nếu ta dùng đoạn chương trình này để tạo một bảng có sẵn, ta sẽ bị báo lỗi thi hành. Do đó, nếu ta có sẵn một bảng là *tblEmployee,* ta chỉ ra giá trị cho thuộc tính ***Name*** của đối tượng ***TableDef*** là *tblEmployeeNew.*

```
Option Explicit
' References: Microsoft DAO 3.51 Object Library
Dim db As Database
```

```
Private Sub Form_Load()
   Set db = OpenDatabase("..\..\db\novelty.mdb")
End Sub
Private Sub cmdCreate_Click()
On Error GoTo ErrHandler
Dim td As TableDef
Dim f As Field
   Set td = New TableDef
   Set f = New Field
   f.Name = "FirstName"
   f.Type = dbText
   td.Name = "tblEmployeeNew"
   td.Fields.Append f
   db.TableDefs.Append td
   MsgBox "Lo, the table has been created."
Exit Sub
ErrHandler:
   If Err.Number = 3010 Then
      MsgBox "You can't create the table twice, chief."
   Else
      MsgBox Err.Description
   End If
End Sub
```

Ta có thể tạo bảng và trường theo cách chính quy. Ta có Microsoft Access và *Visual Data Manager* của Visual Basic để làm việc này.

Ngoài ra, Để tạo bảng và các trường trong bảng cũng như mối quan hệ của bảng bằng chương trình ta dùng thêm phương thức *CreateField* của *TableDef* để tạo đối tượng *Field* và sử dụng phương thức *CreateRelation* của đối tượng ***Database***. Để tạo chỉ mục sử dụng phương thức *CreateIndex* của đối tượng ***TableDef***.

13.6.2.2 Sử dụng đối tượng QuyryDef để thao tác trên truy vấn chứa sẵn

Ta có thể sử dụng đối tượng ***QueryDef*** của DAO để tạo và sửa đổi các truy vấn chứa sẵn. Ngoài việc tạo các truy vấn chứa sẵn, đối tượng *QueryDef* còn cho phép

thi hành các truy vấn tham số hoá; đây là một láy do phổ biến để truy cập các đối tượng *QueryDef* thông qua lập trình DAO.

Ta còn có cách khác để tạo các truy vấn chứa sẵn thông qua DAO. Đó là sử dụng Microsoft Access hay *Visuald Data Manager* của Visual Basic để tạo các truy vấn chứa sẵn.

Ta tạo định nghĩa truy vấn trong cơ sở dữ liệu bằng cách sử dụng đối tượng ***QueryDef*** của DAO. Các định nghĩa này có thể là tạm thời, nghĩa là chúng sẽ biến mất khi hoàn tất hoặc là vĩnh viễn chứa trong cơ sở dữ liệu.

Ta tạo một đối tượng *QueryDef* bằng đoạn chương trình sau:

```
Option Explicit
' References MS DAO 3.51
Dim db As Database
Dim qd As QueryDef
Sub Form_Load()
  Set db = OpenDatabase("..\..\DB\novelty.mdb")
End Sub
Private Sub cmdCreate_Click()
On Error GoTo ErrHandler
  Set qd = New QueryDef
  qd.SQL = "SELECT * FROM tblCustomer " & _
        "WHERE LastName Like 'L*' " & _
        "ORDER BY [LastName], [FirstName]"
  qd.Name = "qryCustomerSortName"
  db.QueryDefs.Append qd
  MsgBox "Whaddya know, the query was created."
Exit Sub
ErrHandler:
   MsgBox "There was an error creating the querydef. (" & _
        Err.Description & ")"
End Sub
```

Ta chỉ ra tên của *QueryDef* cần tạo bằng cách sử dụng thuộc tính *Name* của đối tượng *QueryDef.* Nếu ta cố tạo một *QueryDef* có sẵn, nó sẽ kích hoạt một lỗi bẫy được.

Đừng quên đặt tên cho *QueryDef* mới; nếu không, ta sẽ không có cách nào tham chiếu đến nó (và Jet không thể lưu nó). Nhớ rằng, ta cần xác định truy vấn nào có sẵn trong cơ sở dữ liệu tại thời điểm bất kỳ bằng cách lặp qua tập hợp *QueryDefs.*

13.6.2.3 Sử dụng đối tượng QueryDef để thi hành truy vấn

Ta có thể thi hành truy vấn bằng cách sử dụng các phương thức của đối tượng *QueryDef* . Ta thường thực hiện điều này trong những trường hợp mà ta cần làm một công việc bất thường nào đó trước khi thi hành truy vấn, như là chỉ định một tham số, hoặc khi ta cần thi hành một truy vấn hành động dùng để cập nhật, xoá hay sửa đổi các mẩu tin trong cơ sở dữ liệu.

Để thi hành một ***QueryDef***, ta sử dụng phương thức ***OpenRecordset*** của đối tượng *QueryDef* (để thi hành một truy vấn hành động).

Để thi hành một *QueryDef* , ta theo các bước sau:

1. Tạo một *instance* của đối tượng *QueryDef* bằng lập trình
2. Tạo một chuỗi SQL định nghĩa những gì *QueryDef* sẽ làm khi nó thi hành.
3. Gán chuỗi SQL cho thuộc tính ***SQL*** của *QueryDef*.
4. Nối *QueryDef* vào tập hợp ***QueryDefs*** của đối tượng ***Database***.

Ví dụ :

```
Private Sub cmdRun_Click()
On Error GoTo ErrHandler
  Dim rs As Recordset
  Dim qd As QueryDef
  Set qd = db.QueryDefs("qryCustomerSortName")
  Set rs = qd.OpenRecordset
  Do Until rs.EOF
    lstCustomer.AddItem rs!LastName & " " & _
                rs!FirstName & " " & _
                rs!Address
    rs.MoveNext
  Loop
Exit Sub
ErrHandler:
    MsgBox "There was an error running the query. " & _
        "Are you sure you created it?"
End Sub
```

13.6.2.4 Sử dụng đối tượng QueryDef để tạo truy vấn

Ta có thể tạo một *QueryDef* bằng cách lập trình. Để tạo một *QueryDef*, ta tạo một đối tượng *QueryDef* mới bằng cách sử dụng phương thức ***CreateQueryDef*** của đối tượng ***Database***. Phương thức này cho phép cung cấp một tên cho truy vấn và gán cho nó một chuỗi SQL. Thi hành phương thức này cũng lưu *QueryDef* vĩnh viễn trong cơ sở dữ liệu.

```
Dim db As Database
Dim qd As QueryDef
Set db = OpenDatabase("..\..\DB\novelty.mdb")
Set qd = db.CreateQueryDef(“qryCustomer” , “Select * from tblCustomer”)
```

Mỗi lần tạo *QueryDef*, ta có thể thao tác với nó trong chương trình như ta vẫn thường làm. Cho *QueryDef* một tên là để xác định xem truy vấn có được chứa vĩnh viễn trong cơ sở dữ liệu hay không. Nếu ta muốn tạo một *QueryDef* mà không chứa vĩnh viễn trong cơ sở dữ liệu, chỉ cần truyền một chuỗi rỗng thay cho tham số thứ nhất của phương thức ***CreateQueryDef***.

13.6.2.5 Sử dụng đối tượng Parameter để tạo truy vấn tham số hoá

Tập hợp *Parameters* của một *QueryDef* cho phép ta tiến hành một truy vấn tham số hoá. Các truy vấn này được xây dựng với một hay nhiều thành phần cố ý bỏ qua; các thành phần này được phải cung cấp đủ khi truy vấn thi hành.

Ta tạo các truy vấn tham số hoá bởi vì chúng thi hành nhanh hơn các truy vấn xây dựng trong SQL chứa trong chương trình Visual Basic. Điều này xảy ra vì bộ máy cơ sở dữ liệu biên dịch một truy vấn trước khi ta thi hành, giúp tối ưu hoá thi hành của truy vấn.

Chương 12 trình bày cú pháp của một truy vấn tham số hoá

```
SELECT * FROM tblCustomer WHERE ID=pID
```

Tham số trong truy vấn này là *pID*. Dưới cú pháp SQL của Jet, pID được thông dịch thành một tham số bởi vì nó không hưởng ứng tên của trường hoặc bảng bất kỳ trong cơ sở dữ liệu.

Để truy vấn thi , ta cung cấp một giá trị cho tham số này. Ta cung cấp một tham số trong chương trình bằng cách chỉ ra thuộc tính *Value* của đối tượng *Parameter* thích hợp.

```
Option Explicit
' References MS DAO 3.51
Private db As Database
Private qd As QueryDef
Private rs As Recordset
Private Sub Form_Load()
   Set db = OpenDatabase("..\..\DB\novelty.mdb")
   Set qd = db.QueryDefs("qryCustomerByID")
End Sub
Private Sub cmdQuery_Click()
   qd.Parameters("pID").Value = txtID.Text
   Set rs = qd.OpenRecordset
```

```
    If rs.EOF And rs.BOF Then
        MsgBox "Sorry, no customers with that " & _
            "ID are in the database."
    Else
        MsgBox rs!Address & vbCrLf & _
            rs!Phone, _
            vbInformation, _
            "Info for " & rs!FirstName & _
            " " & rs!LastName
    End If
End Sub
```

Để tìm ra một khách hàng với ứng dụng này, người sử dụng nhập số ID của khách hàng trong hộp văn bản. Nếu ứng dụng tìm thấy một khách hàng với ID đó trong cơ sở dữ liệu, nó sẽ hiển thị địa chỉ và số điện thoại cảu khách hàng đó. Nếu ứng dụng xác định rằng không có mẩu tin đáp ứng tiêu chí tìm kiếm, nó sẽ thông báo cho người sử dụng.

Lưu ý: *Bởi vì tham số của một QueryDef là một hàm của câu lệnh SQL định nghĩa truy vấn, ta không thể tham số mới vào tập hợp* ***Parameters*** *của một QueryDef như cách ta thường làm để thêm một đối tượng vào tập hợp trong DAO.*

13.7 Làm việc với tài liệu và nơi chứa CSDL

Tài liệu cơ sở dữ liệu là một cấu trúc DAO cho phép ta tham chiếu đến các phần tử của cơ sở dữ liệu một cách tổng quát. Chúng còn cung cấp khả năng mở rộng các tính năng của cơ sở dữ liệu Jet. Ta truy cập các thuộc tính của một tài liệu cơ sở dữ liệu thông qua đối tượng ***Container.***

Đối tượng ***Container*** sở hữu bộ các đối tượng ***Document.***

Tương tự như nhiều đối tượng khác có sẵn trong DAO, ta có thể dùng lập trình DAO để lặp xuyên qua nơi chứa và tài liệu. Ví dụ sau đây trình bày cách lặp xuyên qua các đối tượng ***Collection*** và ***Document*** trong một cơ sở dữ liệu Jet.

```
Option Explicit
' References DAO 3.51
Private db As Database
Private con As Container
Private doc As Document
Private Sub Form_Load()
  Set db = OpenDatabase("..\..\DB\novelty.mdb")
End Sub
```

```
Private Sub cmdView_Click()
  lstOutput.Clear
  For Each con In db.Containers
    lstOutput.AddItem con.Name
      For Each doc In con.Documents
        lstOutput.AddItem "    " & doc.Name
    Next
  Next
End Sub
```

Cần phải hiểu điểm khác biệt giữa tài liệu cơ sở dữ liệu và tập hợp DAO. Ví dụ, một tập hợp của ***TableDef*** tham chiếu đến tất cả các *TableDef* ta mở ra trong chương trình. Trái lại, tài liệu ***Tables*** chứa các tham chiếu đến tất cả các tài liệu bảng trong cơ sở dữ liệu mà ta mở. Trong Jet, nơi chứa *Tables* bao gồm các tài liệu như là truy vấn chứa sẵn và các bảng hệ thống (bắt đầu bằng "MSys").

Ta sử dụng các đối tượng ***Container*** và ***Document*** trong những trường hợp sau:

> Ta muốn gán việc cho phép bảo mật cho một đối tượng trong một cơ sở dữ liệu bảo mật .
>
> Ta muốn tạo hoặc lấy về các thuộc tính hiệu chỉnh cho tất cả các thành phần của một cơ sở dữ liệu bằng cách lặp xuyên qua các tập hợp

Ngoài việc kiểm tra nội dung của các tại liệu cơ sở dữ liệu hiện hành, DAO còn cho phép ta định nghĩa và tự tạo các tài liệu, chúng có thể chứa trong cơ sở dữ liệu cùng với tài liệu mặc định.

Các tài liệu hiệu chỉnh bao hàm trong mô hình đối tượng Jet để hỗ trợ các mở rộng. Ý tưởng chung là để ngăn ngừa mô hình đối tượng hiện hành cấm các tính năng mới.

13.8 Tạo và sử dụng các thuộc tính hiệu chỉnh của đối tượng DataBase

Ta có thể tham chiếu đến các thuộc tính của DAO một cách tổng quát. Điều này cho phép khả năng mở rộng, cho phép ta thêm các thuộc tính riêng và đọc thuộc tính của các đối tượng hiện hành dù cho ta có biết những gì chúng được gọi hay không.

Ta truy cập đến danh sách tổng quát các thuộc tính của DAO thông qua tập hợp *Properties* và đối tượng *Property.* Ta có thể tạo một thuộc tính mới bằng cách sử dụng phương thức *CreateProperty* của đối tượng ***Document.*** Các thuộc tính được cung cấp bởi tập hợp *Properties* được gọi là thuộc tính động bởi vì chúng có thể khác nhau giữa một cơ sở dữ liệu này và một cơ sở dữ liệu khác hay giữa các phiên bản khác nhau của bộ máy cơ sở dữ liệu.

Ví dụ sau đây trình bày cách thức tập hợp ***Properties*** hoạt động. Đây là thủ tục phát sinh danh sách các thuộc tính mặc định hiện có trong một cơ sở dữ liệu.

```
Option Explicit
```

```
' References: Microsoft DAO 3.51
Dim db As Database
Dim pr As Property
Private Sub Form_Load()
   Set db = OpenDatabase("..\..\DB\novelty.mdb")
End Sub
Private Sub cmdShow_Click()
On Error Resume Next
   lstOutput.Clear
   For Each pr In db.Properties
      With pr
         lstOutput.AddItem .Name & _
                     ": " & .Value
      End With
   Next
End Sub
```

Có 13 thuộc tính mặc định của một đối tượng *Database* hiện có thông qua tập hợp ***Properties***, bao gồm những phần cố định như là ***Name***, ***Version,*** và ***Connect,*** cũng như các thuộc tính không thể truy cập trực tiếp, như là ***AccessVersion*** và ***Build***.

Vì vậy, ta có thể viết chương trình như sau:

```
MsgBox db.Name
```

Vậy, tại sao ta không viết chương trình để truy cập một thuộc tính của cơ sở dữ liệu :

```
MsgBox db.Properties ("Name")
```

Câu trả lời là ta có thể tựtạo các thuộc tính hiệu chỉnh và gán chúng đến bất kfy DAO nào (không chỉ đến đối tượng ***Database)***, Ví dụ sau tạo một thuộc tính mới trong cơ sở dữ liệu.

```
Private Sub cmdCreate_Click()
On Error Resume Next
Dim prp As Property
  Set prp = db.CreateProperty("DateLastBackedUp", dbDate, Now)
  db.Properties.Append prp
  cmdShow_Click   ' update list of properties
End Sub
```

Sau đây là cú pháp đầy đủ của phương thức ***CreateProperty:***

obj.CreateProperty ([propertyName], [datatype] , [value], [ddl])

Tham biến *PropertyName* là một chuỗi chỉ ra tên của một thuộc tính mới

Tham biến *DataType* chỉ ra kiểu dữ liệu của thuộc tính mới.

Tham biến *Value* cung cấp một giá trị khởi tạo cho thuộc tính.

Tham biến *ddl* là một giá trị boolean chỉ ra thuộc tính mới có phải là một đối tượng DDL hay không. Nếu đúng, những người sử dụng không có quyền sửa đổi cấu trúc cơ sở dữ liệu, sẽ không thể sửa đổi thuộc tính hiệu chỉnh.

Để lấy về thuộc tính hiệu chỉnh sau khi tạo ra, ta chỉ cần truy cập nó thông qua tập hợp *Properties* của đối tượng *Database*.

13.9 Tổng kết

Chương này giải thích về lý thuyết thống nhất và quan trọng của lập trình DAO. Nếu là người xử lý giỏi, ta có thể tiến hành phần lớn các hoạt động ta từng làm trong thế giới truy cập cơ sở dữ liệu của Visual Basic.

13.10 Hỏi và đáp

Hỏi: *Dùng DAO có phù hợp với lập trình CLient / Server không ?*

Đáp: Có thể sử dụng DAO cho lập trình Client / Server, nhưng tốt hơn nên dùng ADO. ADO cung cấp một giao diện chương trình tương tự như DAO, nhưng một số tính năng mới bổ sung (như là khả năng truy cập đến trình cung cấp dữ liệu không quan hệ, các kết nối và truy vấn không đồng bộ, và một mô hình đối tượng đơn giản hơn.)

Hỏi: *Tôi đang tạo một ứng dụng dựa trên Jet. Vậy tôi nên tiếp tục dùng DAO hay là chuyển sang dùng ADO ?*

Đáp: Nếu ứng dụng của bạn hoạt động tốt với DAO, bằng mọi cách, giữ lại kỹ thuật DAO. Đó là mô hình đối tượng dành cho cơ sở dữ liệu Jet. ADO có một số chỗ không thể làm được như DAO đã làm, ví dụ, bạn có thể tạo ra một cơ sở dữ liệu Jet trong DAO, nhưng bạn không thể làm được điều đó với ADO.

14 Thiết lập báo cáo và Xuất thông tin

Sử dụng thiết kế DataReport

sử dụng Microsoft Access để làm báo cáo

Sử dụng Crystal report để lập báo cáo

Thiếy lập báo cáo trên cơ sở dữ liệu không chỉ đơn giản là hiển thịdữ liệu từ cơ sở dữ liệu. hầu hết các báo cáo cơ sở dữ liệu liên quan một số hoạt động khác trên dữ liệu.

a. **Truy vấn** dữ liệu để trả về, hiển thị và in ra những phần ta muốn. ta gọi là lọc
b. **Sắp xếp** Sao cho nó xuất hiện heo mmột thứ tự có ý nghĩa
c. **Phân nhóm** dữ liệu để hiển thị một cách gọn gàng.

Chương này sẽ phân biệt giữa các báo cáo và xuất thông tin

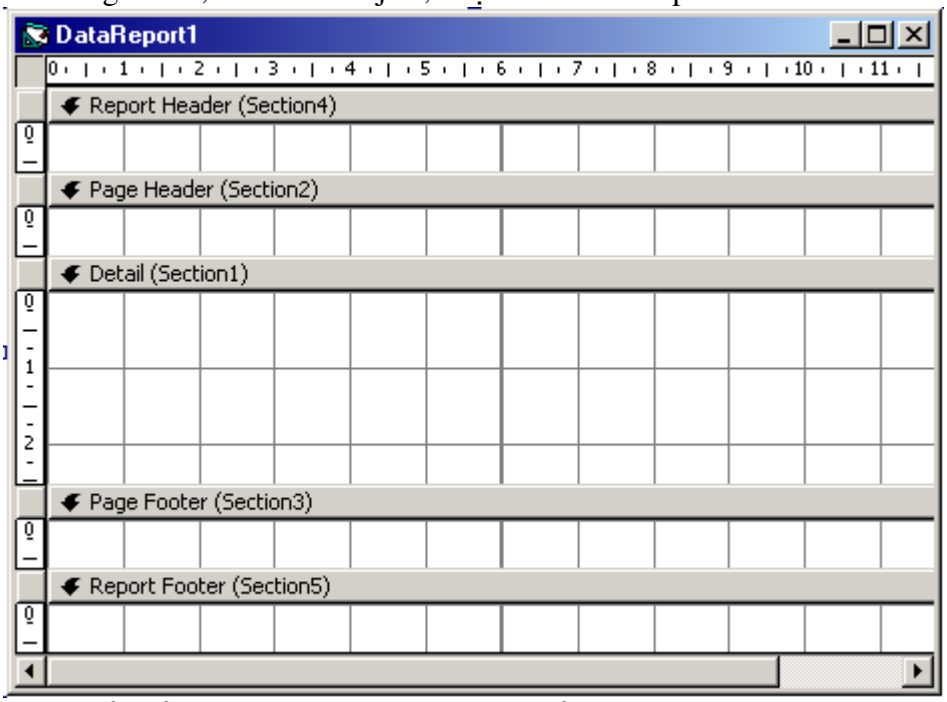
Visual Basic cho phép truy cập đến một số kỹ thuật *Client / server* mới hỗ trợ truy cập cơ sở dữ liệu hiệu quả hơn - Đặc biệt khi nó dùng để xuất thông tin hay thiết lập báo cáo

14.1 Sử dụng thiết kế DataReport

Là điểm mới trong Visual Basic 6, thiết kế DataReport là cách trực quan để tạo ra những báo cáo thích hợp trong môi trường phát triển Visual Basic. Thiết kế DataReport, cung cấp các chức năng hết sức cơ bản, nhưng nó có ưu điểm rất dễ dùng.

Để tạo thiết kế DataReport, ta theo các bước sau:

1. trong đề án, từ menu Project, chọn Add DataReport.



2. Thiết kế DataReport được thêm vào đề án.

DataReport chứa một số phân đoạn;

Report Header hiển thị một lần ở đầu báo cáo.

Report foodter hiển thị một lần ở cuối báo cáo

Page header hiển thị tại đầu mỗi trang

page footer hiển thị tại cuối mỗi trang

Detail section hiển thị các dòng dữ liệu

một hay nhiều group header hay foodter hiển thị tại đầu hoặc cuối của một phân đoạn nhóm.

Dễ nhất là tạo DataReport dùng trên thiết kế DataEnvironment. Với DataEnvironment, ta có thể dùng cách kéo thả để thiết kế báo cáo

14.1.1 Thiết kế với DataReport

Mỗi lần tạo ra thiết kế DataReport, ta phải ràng buộc nó với một cơ sở dữ liệu để hiển thị dữ liệu. ta thực hiện điều này thông qua một bộ gồm các điều khiển ràng buộc chỉ hoạt động trong ngữ cảnh thiết kế DataReport.

Các điều khiển trực quan của thiết kế DataReport bao gồm:

Điều khiển nhãn

điều khiển hộp văn bản

điều khiển ảnh

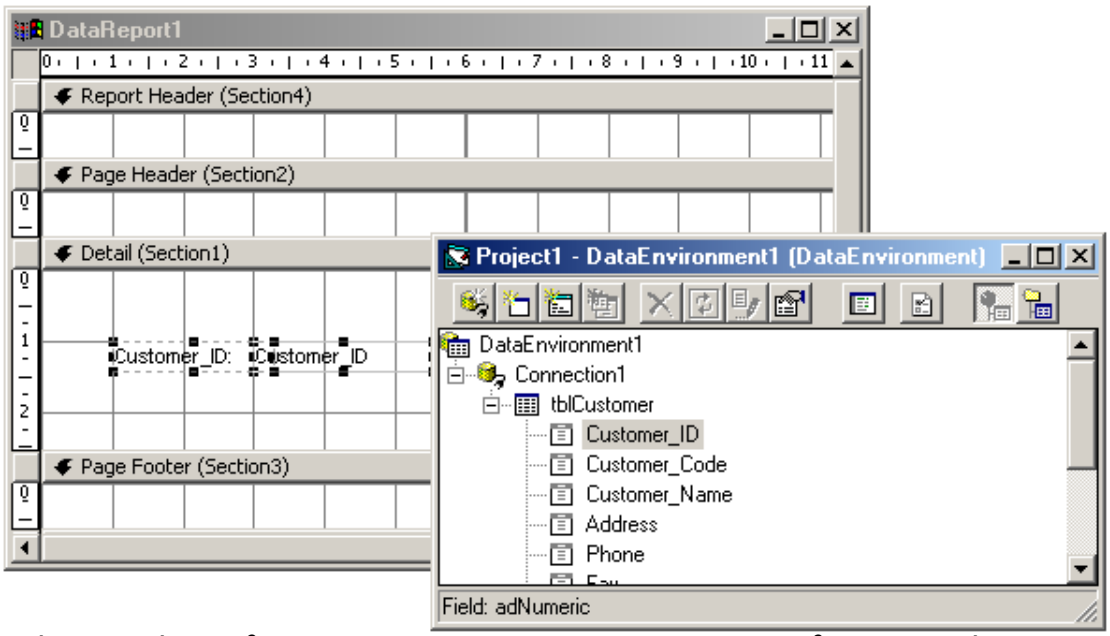
điều khiển đoạn thẳng và điều khiển hình dạng

điều khiển hàm cho phép chèn các tính toán tóm tắt và báo cáo

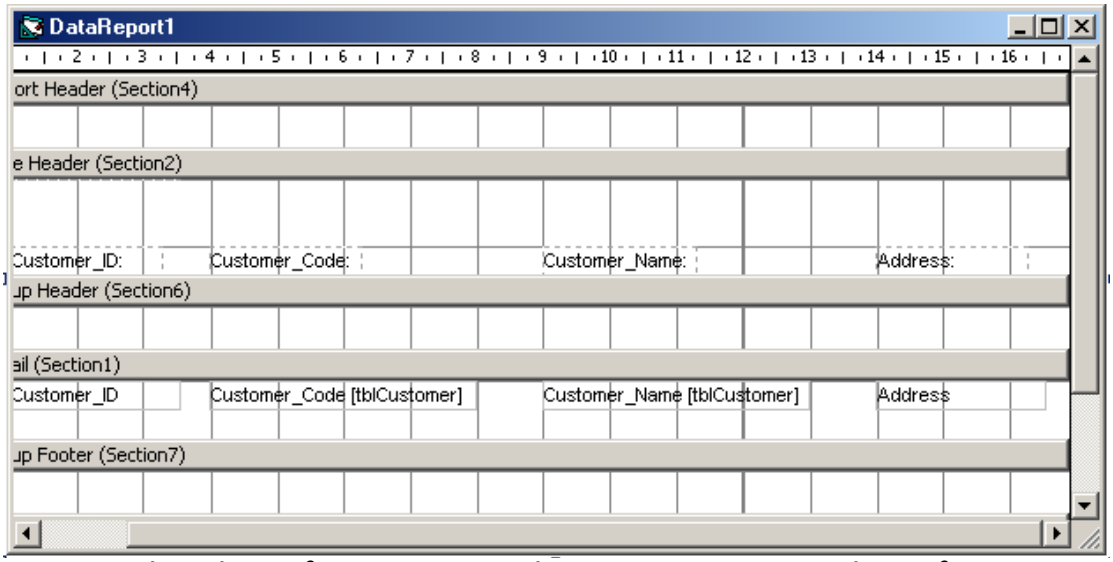
bởi vì các điều khiển của DataReport được thiết kế đặc biệt để hoạt động trong ngữ cảnhcủa thiết kế DataReport, chúng có những tên gọi khác nhau. Ví dụ tên lớp chính thức của điều khiển hộp văn bản là Rpttextbox; tên lớp điều khiển ảnh là RptImage

Sẽ đơn giản hơn nhiều nếu ta dùng kỹ thuật kéo thả để tạo báo cáo

1. Từ menu Project, chọn Add DataReport
2. Tạo thiết kế Data Enviroiment
3. Trong thiết kế Data Enviroiment, tạo một câu lệnh truy vấn dựa trên bảng tblCcustomer
4. Thiết kế Data Enviroiment hiển thị danh sách các trường dữ liệu trong bảng tblCustomer. Chọn và kéo rê trường Customer_ID từ thiết kế Data Enviroiment vào phân đoạn Detail của thiết kế DataReport
5. Một điều khiển nhãn và một hộp văn bản ràng buộc với trường dữ liệu Customer_ID xuất hiện trên thiết kế DataReport



Nếu kéo điều khiển phân đoạn Page Header, nó sẽ được hiển thị một lần trên mỗi trang. Nếu thực hiện việc này với nhiều trường, ta sẽ có báo cáo tương tự như sau:



Khi có nhiều điều khiển được chọn, điều chỉnh kích cỡ một điều khiển sẽ làm kích cỡ các điều khiển khác bị diều chỉnh theo

Kéo và thả trường dữ liệu từ thiết kế Data Enviroiment vào báo cáo, ta phải thêm một số bước để chỉ ra nguồn dữ liệu cho ràng buộc

Bước 1: Chọn thiết kế DataReport bằng cách chọn vào nó trong danh sách drop-down của cửa sổ Properties

bước 2: Trong cửa sổ Properties, quy định thuộc tính DataSource của DataReport là tên của trình thiết kế Data Enviroiment

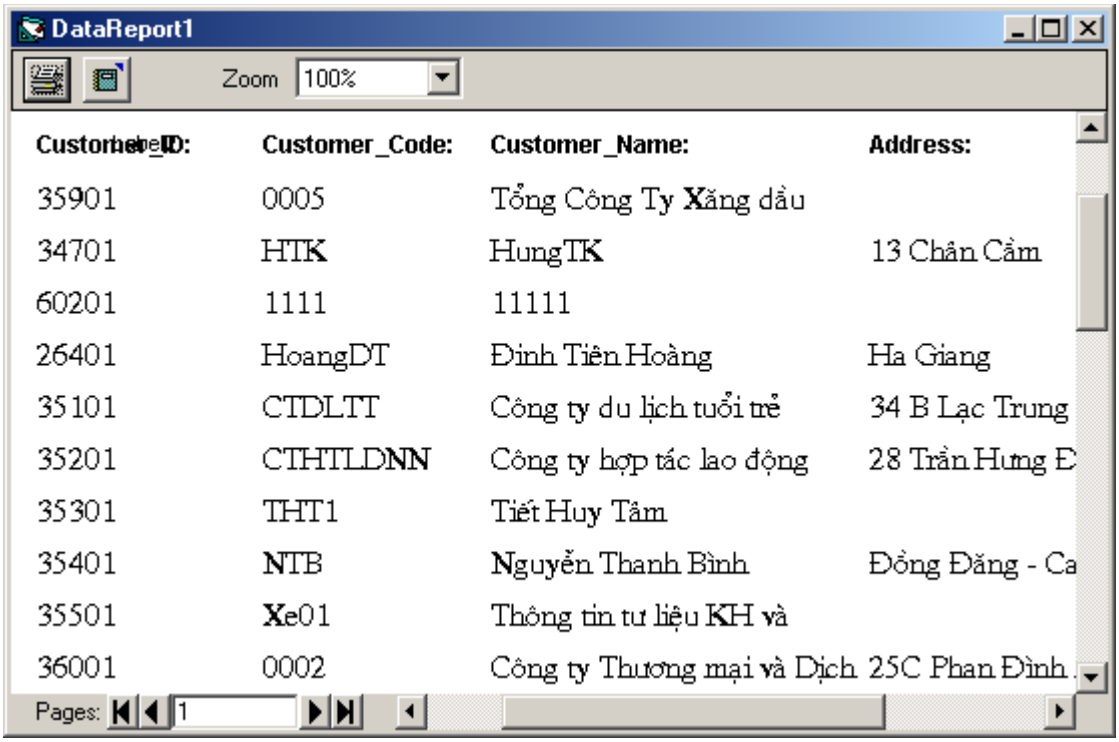
Bước 3: Nếu đang sử dụng trình thiết kế Data Enviroiment, quy định thuộc tính Data member của DataReport là lệnh dữ liệu mà ta muốn

14.1.2 Xem và xuất DataReport

Ta có thể xem báo cáo trong chế độ Print Preview bằng cách thi hành phương thức **Show.** Ví dụ nếu DataReport được gọi là drCustomer thì ta có thể chạy nó bằng dòng lệnh:

drCustomer.Show

báo cáo được hiển thị nhe hình dưới đây:



14.2 Sử dụng Microsoft Access để làm báo cáo

Microsoft Access cho phép viết báo cáo cơ sở dữ liệu. Nó hỗ trợ giao diện dễ dùng và trực quan mà hầu hết lập trình viên Visual Basic đều ưa thích. tương tự Crystal Report, báo cáo Microsoft Access cho phép phân nhóm và sắp xếp dữ liệu, cũng như sử dụng các biểu thức hiệu chỉnh trong báo cáo.

14.2.1 Thi hành báo cáo của Access từ Visual Basic

Có hai kỹ thuật thực hiện điều này:

sử dụng Automation để phóng một instance của Microsoft Access, thi hành báo cáo trực tiếp từ trong ứng dụng. Automation là một kỹ thuật cho phép giao tiếp giữa các ứng dụng trong Windows. Ở đây Access sẽ làm Automation server

Dùng sản phẩm VSREPORT của Videosoft cho phép người sử dụng thi hành báo cáo của Microsoft Access bất kể họ có cài đặt Access hay không. Đây là điều khiển ActiveX chuyển đổi báo cáo từ tập tin MDB thành một định dạng mà ta có thể cung cấp với các ứng dụng

14.2.1.1 sử dụng AUTOMATION để thi hành báo cáo Access

Ta có thể phóng một instance của Microsoft Access từ ứng dụng viết bằng Visual Basic. Với kỹ thuật này, ta có thể lập trình Microsoft Access tương tự như với đối tượng khác ngay trong Visual Basic

Bất lợi của kỹ thuật này là buộc người sử dụng phải chạy ột instance của Microsoft Access mỗi khi họ muốn xem hoặc in ấn báo cáo. Dĩ nhiên nó cũng đòi hỏi phải có Microsoft Access nạp sãn trên máy. Nếu bạn muốn khắc phục trở ngại này nhưng vẫn dùng báo cáo Access trong ứng dụng, bạn có thể xem xét giải pháp thiết lập báo cáo của Video soft VSREPORT

Để lập trình Access thông qua Automation, ta làm một tham chiếu đến Access bằng cách từ menu Project chọn References chọn hộp dánh dấu "Microsoft Access 8.0 Object Library"

14.2.1.1.1 Tránh dùng dàng buộc trễ với Automation

Nếu bạn dùng qua Automation qua OLE Automation, bạn sẽ thấy điểm khác biệt quan trọng giữa kỹ thuật Automation trong phần này và OLE Automation. Trong Visual Basic ta cần tránh định nghĩa đối tượng Automation theo kiểu Object tổng quát. Ví dụ trong Visual Basic 3.0 ta có thể viết chương trình như sau:

```
Dim appAcces as Object
Set appAcces =CreateObject("Access.Application")
```

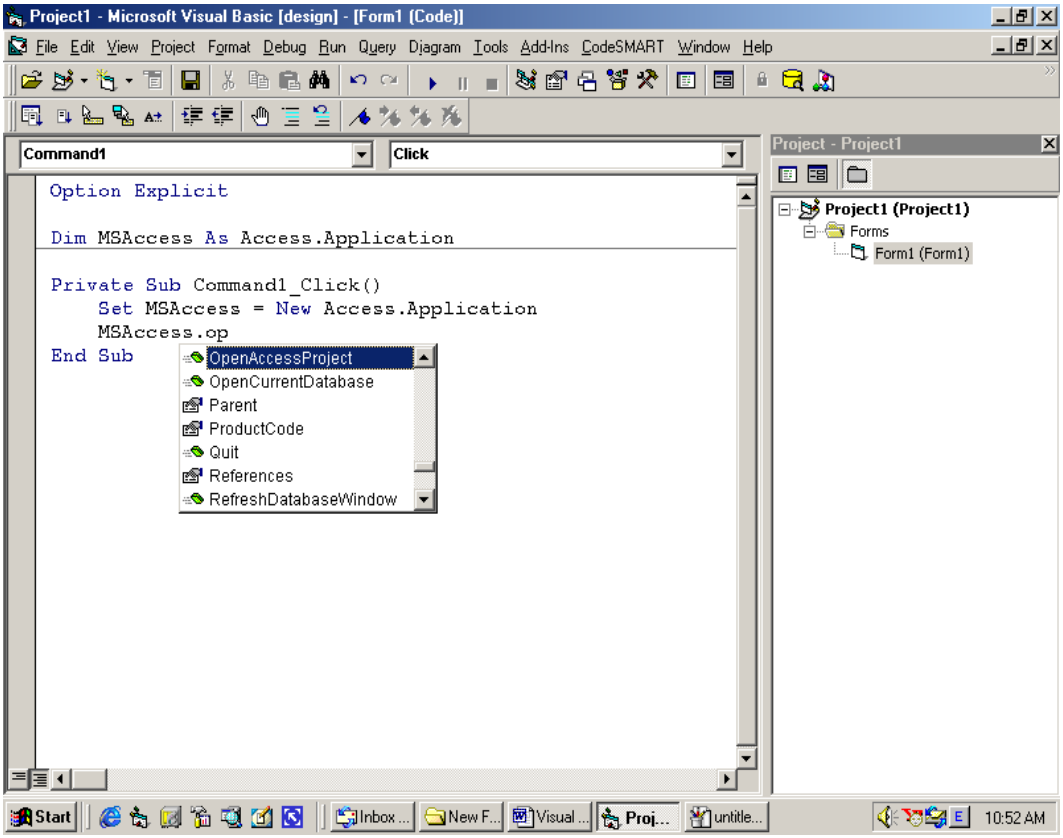
Đoạn chương trình trên hoạt động tốt với Visual Basic 3.0 nhưng có cách khác tốt hơn. Thay vì dùng kiểu Object ta dùng kiểu dữ liệu đối tượng Automation Server cung cấp. Bởi vì khi đó Visual Basic không cần thi hành câu lệnh truy vấn trên Automation Server mỗi khi ta truy nhập nó để xác định kiểu đối tượng cần tạo. kỹ thuật này gọi là ràng buộc trế, giờ đây chỉ phù hợp với hai tình huống:

ta không biết trước kiểu đối tượng Automation Server khi cần tạo đối tượng

Ta đang sử dụng một môi trường phát triển ứng dụng không hỗ trợ ràng buộc sớm sẽ cải tiến đáng kể tốc đọ chương trình, nhất là những lệnh cần nhiều sử lý để gọi đến Microsoft Office Automation Server

14.2.1.1.2 Thi hành Automation để điều khiển Microsoft Access

Sau khi thiết lập tham chiếu đến Microsoft Access từ trong ứng dụng, ta viêt chương trình để tạo instance cho đối tượng. khi đó có một đối tượng tham chiếu đến server, Visual Basic tích hợp mô hình đối tượng của server vào tính năng liệt kê thành phần tự động



Ngoài ra ta có thể xem toàn bộ thuộc tính, phương thức và sự kiện của đối tượng Access trong cửa sổ Object Browser

Đoan chương trình sau tạo một instance của Microsoft Access trong Visual Basic. Nó sử dụng Automation để thi hành phương thức Docmd của Access

```
Option Explicit

Dim MSAccess As Access.Application

Private Sub Command1_Click()
   Set MSAccess = New Access.Application
   MSAccess.OpenCurrentDatabase ("D:\TungPT\Temp\reins.mdb ")
   MSAccess.DoCmd.OpenReport "rptEmployee", acViewNormal
   MSAccess.CloseCurrentDatabase
   Set MSAccess = Nothing
End Sub
```

14.2.1.2 sử dụng VSREPORTS để thi hành báo cáo Access

VSREPORT được thực hiện theo 2 phần:

Trình tiện ích chuyển đổi dùng tập tin MDB của Access và tạo một tập tin báo cáo từ đối tượng Report chứa trong cơ sở dữ liệu

điều khiển ActiveX dùng để mở tập tin báo cáo và thi hành báo cáo trong ứng dụng

Cấu trúc này tương tự Crystal Reports. Với cả hai hệ thống, ta phải tạo một tệp tin thông báo, rồi thêm một điều khiển và viết đoạn chương trình trong ứng dụng để in báo cáo. mỗi lần sửa đổi trong báo cáo, ta phải lưu và phân phát lại cho người sử dụng.

Có nhiều điểm khác nhau giữa Crystal Reports và VSREPORT. Crystal Reports có một tiện ích để thiết kế báo cáo riêng, trái lại, VSREPORT sử dụng Microsoft Access. chọn lựa các sản phẩm tuỳ thuộc vào các ứng dụng.

Nếu bạn muốn dùng báo cáo của Access, để ý rằng báo cáo của VSREPORT dùng trên Automation là người sử dụng không phải khởi động một instance của Access mỗi lần họ muốn thi hành báo cáo. Thực vậy, với mỗi VSREPORT, người sử dụng không cần nạp sẵn Access trên máy, điều này rất tiện nếu bạn là người quản lý một cơ quan lôncs định hướng sử dụng đối với nhiều loại người sử dụng. Nếu bạn khổn chắc chắn rằng toàn bộ người sử dụng có cài sẵn phoên bản Access mới nhất, bạn có thể đưa cho họ một tệp tin VSREPORT mà không cần bận tâm người sử dụng đã cài Access hay chưa.

14.2.1.2.1 Dùng VSREPORT để chuyển đổi báo cáo Access

Trước hết tạo một báo cáo trong Microsoft Access, sau đó thi hành trình thông dịch VSREPORT trên đó. Kết quả của thông dịch là một tệp tin có thể dùng ActiveX VSREPORT trong ứng dụng

Có 2 trình thông dịch:

TRANS95.EXE: chuyển đổ báo cáo Access 95

TRANS97.EXE : chuyển đổ báo cáo Access 97

Các tiện ích này chứa trong thư mục cài đặt VSREPORT

14.2.1.2.2 Tạo ứng dụng dùng VSREPORT

1. phóng tiện ích thông dịch Access 97, TRANS97.EXE
2. Nhấn đúp chuột lên More File để hiển thị hộp thoại tệp tin
3. Chọn tệp tin cơ sở dữ liệu Novelty
4. Trình thông dịch mở cơ sở dữ liệu và hiển thị danh sách các báo cáo. Chọn rptCustomer, sau đó nhấn mũi tên phải phía trên ở dưới hộp thoại. nhấn Next
5. trình thông dịch yêu cầu cung cấp đường dẫn và tệp tin cần chọn. ta cũng có thể đổi tên tệp tin kết quả vào lúc này.
6. nhấn Translate. trình thông dịch sinh ra tập tin.VSR chứa định nghĩa báo cáo

14.2.1.2.3 Dùng điều khiển ActiveX VSREPORT để thi hành báo cáo có Access

Để đưa báo cáo vào ứng dụng, ta dùng điều khiển ActiveX VSREPORT. Điều khiển này không xuất hiện lúc thi hành; nó chỉ đọc và xuất ra tập tin.VRS

Để cài đặt bản DAO 3.5 của điều khiển VSREPORT, ta làm như sau:

1. Cài đặt tập tin cập nhật lấy từ CD hoặc tải xuống từ Internet, Web side *http://www.videosoft.com*

2. Bảo đảm tập tin VSREP351.OCX có sẵn trong thư mục system của windows
3. Dùng tiện ích regsvr32 để dăng ký tệp tin OCX mới
4. Từ menu Project, chọn Components để thêm điều khiển VSREPORT

Đưa điều khiển vào ứng dụng

1. Từ menu Project, chọn Components để thêm điều khiển VSREPORT vào đề án
2. điều khiển VSREPORT xuất hiện trong hộp công cụ. Nhấn đúp chuột vào điều khiển VSREPORT trong hộp công cụ để đưa nóp vào biểu mẫu. điều khiển sẽ không hiển thị lúc thi hành, ta không cần bận tâm về vị trí điều khiển
3. Quy định thuộc tính Report File Name của điều khiển là tệp tin.VSR ta đã tạo trước đó. Sau đó, quy định thuộc tính DataBase Name của điều khiển là tên tập tin.MDB. Ta cũng có thể dùng cửa sổ Properties hoặc dùng chương trình
4. Thi hành phương thức PrintReport để thi hành báo cáo. Ta còn tuỳ chọn cho phép gửi kết qủa ra màn hình hoặc máy in, tuỳ theo giá trị thuộc tính của máy in PrintDevice.

Đoạn chương trình sau để in trong chế đọ PrintPreview và thi hành báo cáo cơ sở dữ liệu dùng điều khiển ActiveX VSREPORT

```
Option Explicit

' Requires Jet 3.5 version of VSREPORTS (vsrep351.ocx)

Private Sub Form_Load()
            vsReport1.ReportFileName   =   App.Path   &
"\rptCustomer.vsr"
    vsReport1.DatabaseName = "..\..\DB\novelty.mdb"
End Sub

Private Sub cmdPreview_Click()
    vsReport1.PrintDevice = vsrPrintDeviceScreen
    vsReport1.Zoom = 50   ' Percent
    vsReport1.PrintReport
End Sub

Private Sub cmdPrint_Click()
    vsReport1.PrintDevice = vsrPrintDevicePrinter
    vsReport1.PrintReport
End Sub

Private Sub vsReport1_LoadingDoc(ByVal Page As Integer,
ByVal Of As Integer, Cancel As Boolean)
    Debug.Print "Loading: " & Page & " of " & Of
End Sub
```

14.2.1.2.4 Tạo ứng dụng báo cáo nâng cao với VSREPORT

VSREPORT có một số tính năng khác cho phép tạo ứng dụng báo cáo mạnh mẽ. Sau đây là các tính năng nâng cao của điều khiển VSREPORT:

Thuộc tính Zoom; cho phép phóng to hoặc thu nhỏ theo đơn vị %

các thuộc tính Printer để điều khiển cách thức in ấn của báo cáo

Hỗ trợ duyệt trangđối với báo cáo nhiều trang: Dùng thuộc tính PrevewPage để cho phép người sử dụng lật trang

Tích hợp với đối tượng DataBase của DAO: Thay vì gán DataBase Name cho điều khiển, ta có thể gán đối tượng DataBase của DAO cho điều khiển. giải pháp này được dùng khi ứng dụng đang dùng DAO để kết nối với cơ sở dữ liệu và ta không muốn có một tham chiếu rõ ràng đến cơ sở dữ liệu từ trong điều khiển VSREPORT. Để gán một đối tượng DataBase cho điều khiển VSREPORT , ta quy định thuộc tính DataBase Access Mode của điều khiển là 1 – vsr DataBase Objiect. Sau đó ta có thể gán đối tượng DataBase của DAO cho thuộc tính DataBase Objectcủa điều khiển VSREPORT.

Ngoài các thuộc tính trên, báo cáo thi hành với điều khiển VSREPORT có thể kích hoạt sự kiện, như cách thực hiện của báo cáo trong Access. nó cho phép ta viết chương trình đáp ứng các sự kiện sảy ra trong khi in báo cáo. tuy nhiên, mô hình sự kiện của điều khiển VSREPORT khác với mô hình sự kiện của Access, vì vậy ta phải giữ bản gốc báo cáo Access cẩn thận trước khi chuyển sang dùng VSREPORT.

14.3 Sử dụng Crystal report để lập báo cáo

Crystal Reports cho phép tạo báo cáo cơ sở dữ liệu trong ứng dụng viết bằng Visual Basic. nó gồm hai phần chủ yếu:

trình thiết kế báo cáo xác định dữ liệu sẽ đưa vào báo cáo và cách thể hiện của báo cáo

Một điều khiển ActiveX cho phép thi hành, hiển thị, inấn điều khiển lúc thi hành ứng dụng

Đối với nhiều người lập trình Visual Basic, Crystal Reports là tất cả những gì cần khi muốn thiết lập báo cáo cơ sở dữ liệu. Bởi vì phiên bản Crystal Reports đi kèm với Visual Basic cực kỳ đẽ dùng.

Có 2 bước để tạo một báo cáo dùng Crystal Reports: tạo báo cáo và thêm điều khiển ActiveX của Crystal Reports vào ứng dụn. Bạn tạo báo cáo dùng trình thiết kế báo cáo của Crystal Reports. Ứng dụng này để tạo các tài liệu báo cáo thi hành trong ứng dụng. Sau đó ta mở tài liệu báo cáo trong ứng dụng bằng cách sử dụng điều khiển Crystal Reports.

14.3.1 Cài đặt Crystal Reports

Khác với phiên bản cũ của Visual Basic, Crystal Reports không được cài đặt tự động khi ta cài Visual Basic

Để cài đặ Crystal Reports phóng trình cài đặt Crystl32.exe chứa trong thư mục \COMMON\TOOL \VB\CRYSREPT trên đĩa VB6. các tệp tin tương ứng sẽ

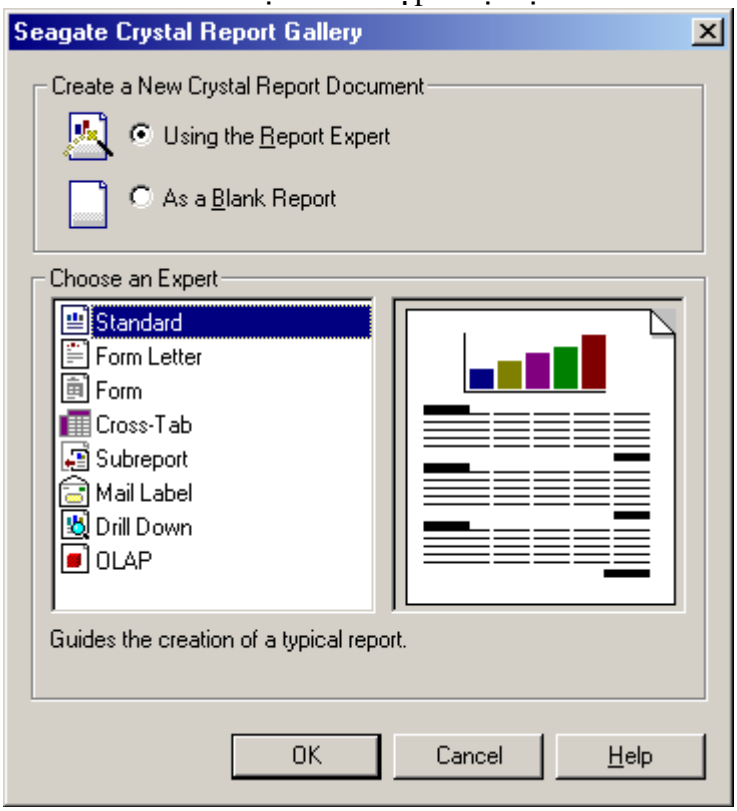
được copy vào hệ thống và đăng ký. sau đó ta mới có thể sử dụng Crystal Reports trong ứng dụng

14.3.2 Dùng Crystal Reports tạo báo cáo

ta không thể tạo báo cáo băng chương trình, mà thay vào đó ta dùng Crystal Reports đe tạo báo cáo. sau khi tạo báo cáo ta lưu nó và phân phát cùng với ứng dụng cho người sử dụng.

Để phóng trình thiết kế Crystal Reports ta theo các bước sau:

1. phóng trình thiết kế Crystal Reports
2. Trình thiết kế hoạt động
3. Từ menu file chọn New. hộp thoại tạo báo cáo mới xuất hiện;



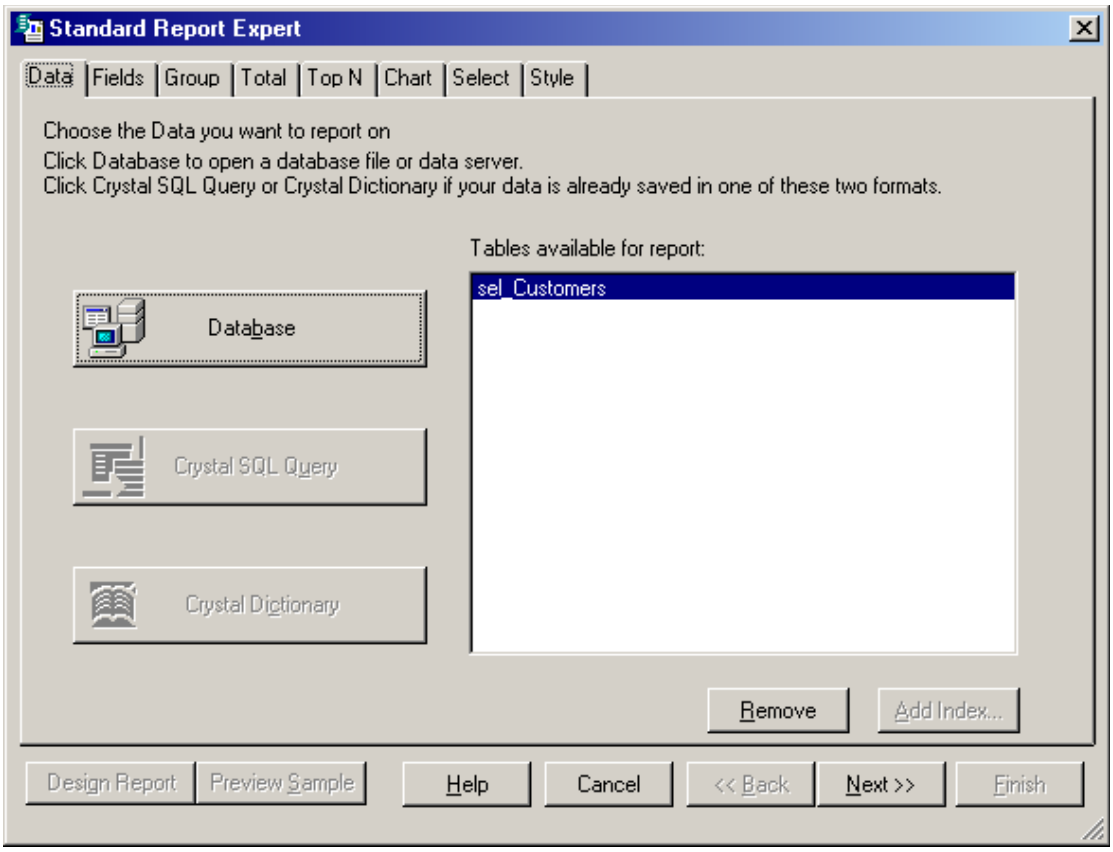
Với hộp thoại này, Visual Basic cung cấp một số khiuôn mẫu báo cáo để ta chọn dùng xuất dữ liệu ta có thể dùng báo cáo do mình thiết kế làm khuôn mẫu cũng như tạo các báo cáo hiệu chỉnh không có trong khuôn mẫu có sẵn

các kiểu báo cáo có sẵn trong Crystal Reports

Kể báo cáo	mô tả
standar	báo cáo liệt kê theo dòng và cột, cho phép sắp xếp và gộp dữ liệu
listing	báo cáo là danh sách dữ liệu liên tục không có tổng kết hay trường tổng cộng. Ta có thể dùng kiểu báo cáo này để in danh bạ điện thoại
cross- Tab	Tổ hợp dữ liệu theo hai chiều
mail Lable	báo cáo thiết kế để in dữ liệu theo cột cho nhãn thư
Summary	báo cáo chỉ hiển thị thông tin tổng quát, không chứa dữ liệu chi tiết

graph	báo cáo thể hiện dữ liệu một cách trực quan
Top N	báo cáo cho phép chỉ hiển thị một số bản ghi được chọn
drill down	báo cáo cho phéơ nhấn đúp chuột lên dữ liệu tổng quát để hiển thị các thông tin chi tiết
Các kiểu báo cáo khác	báo cáo sử dụng khuôn mẫu bạn đã tạo trước đó

Ví dụ mẫu – dùng Crystal Reports để tạo báo cáo



1. Khởi động Crystal Reports và tạo một báo cáo mới. Chọn kiểu Standar
2. Ở bước một của Standar, nhấn nút Data file.
3. Trong hộp thoại tập tin chọn cơ sở dữ liệu muốn báo cáo. Nhấn Add. danh sách các bảng được chỉ ra. Nó chuyển sang bước hai, hiển thị các quan hệ giữa các bảng trong cơ sở dữ liệu

Bởi vì các mối quan hệ được định nghĩa sẵn cho cơ sở dữ liệu đã được xác định ở mức bộ máy cơ sở dữ liệu, ta không cần định nghĩa lại. Nhưng nếu cần thiết phải thiết lập hoặc xoá một quan hệ ở mức báo cáo thay vì ở mức bộ máy cơ sở dữ liệu ta theo các bước sau:

1. Nhấn chuột vào đoạn thẳng thể hiện quan hệ giữa các bảng
2. Nhấn Delete quan hệ bị xoá bỏ
3. Nhấn và rê một trường nào đó từ một bảng và thả ở bảng khác quan hệ giữa hai bảng lại được thiết lập
4. Nhấn Next. chuyển sang bước 3. Đến đây ta xác lập trường nào sẽ hiển thị trong báo cáo.
5. Nhấn vào Tab Sort. Bước này cho phép xác định cách sắp xếp dữ liệu

6. Chọn vào Tab Total. Cho phép tóm lược dữ liệu trong báo cáo
7. nhấn vào Tab Style. Cho phép xác định cáh thức thể hiện báo cáo
8. Nhấn Preview Report

14.3.3 Thi hành báo cáo trong ứng dụng với điều khiển ActiveX của Crystal Reports

Việc cho phép người sử dụng ứng dụng thi hành Crystal Reports là hoàn toàn đơn giản; nó liên quan đến việc điều khiển ActiveX của Crystal Reports vào đề án và viết đoạn chương trình xử lý. Để làm được điều này, ta làm như sau:

1. Tạo một ứng dụng Visual Basic mới với một nút lệnh duy nhất
2. Thêm điều khiển Crystal vào ứng dụng thông quan menu Project Components
3. Tạo một instance của một điều khiển Crystal Reports trên biểu mẫu bằng cách nhấn đúp lên thanh công cụ. điều khiển kết quả được gọi là Crystal Reports 1
4. trong sự kiện click của nút lệnh dưa vào đoạn chương trình sau

```
Private Sub cmdReport_Click()

        CrystalReport1.ReportFileName = App.Path &
"\product.rpt"
    CrystalReport1.PrintReport

End Sub
```

5. Thi hành nút lệnh và ấn nút Run. báo cáo thi hành hiển thị cho người sử dụng trong cửa sổ preview. Đến đây người sử dụng có thể xuất báo cáo ra máy in bằng cách nhấn nút Print. dùng thuộc tính Distination của điều khiển Crystal Reports, ta có thể gửi dữ liệu trực tiếp đến máy in bỏ qua cửa sổ Preview.

14.3.4 Sử dụng bản mới hơn của Crystal Reports

Phiên bản Crystal Reports mới đưa ra một số tính năng mới

báo cáo con
tuỳ chọn định dạng mới
báo cáo có điều kiện
Trình điều khiển cơ sở dữ liệu trực tiếp
Xuất ra Word và Excel
Hỗ trợ Web
Hỗ trợ những nguồn dữ liệu không phải là quan hệ

15 ODBC và các đối tượng dữ liệu từ xa

15.1 Định cấu hình và sử dụng ODBC

ODBC là một công nghệ Windows cho phép ứng dụng client nối với CSDL từ xa. Lưu trú trên máy client, ODBC tìm cách làm cho nguồn dữ liệu quan hệ trở thành tổng quát đối với ứng dụng Client. Điều này có nghĩa là ứng dụng Client không cần quan tâm kiểu cơ sở dữ liệu mà nó đang nối là gì.

Bởi vì đây là công nghệ ở phía Client, ODBC không đòi hỏi ta phải xử lý trên Server của cơ sở dữ liệu.

ODBC gồm 3 phần:

Trình quản lý điều khiển (driver manager)
Một hay nhiều trình điều khiển (driver)
Một hay nhiều nguồn dữ liệu

15.1.1 Kiến trúc của ODBC

Hình Cấu trúc ODBC trình bày kết nối giữa ứng dụng Client và cơ sở dữ liệu Server thông qua ODBC Driver Manager

Kiến trúc ODBC chứa kết nối giữa ứng dụng Client và cơ sở dữ liệu server thông qua Trình quản lý điều khiển ODBC.

LƯU Ý Nguồn dữ liệu ODBC được tạo để dụng với RDO có thể được dùng mà không cần thay đổi với **ADO** - Thực vậy, ODBC là một trình cung cấp tự có của ADO, giúp việc chuyển đổi từ RDO sang dễ dàng hơn.

15.1.2 Tạo nguồn dữ liệu

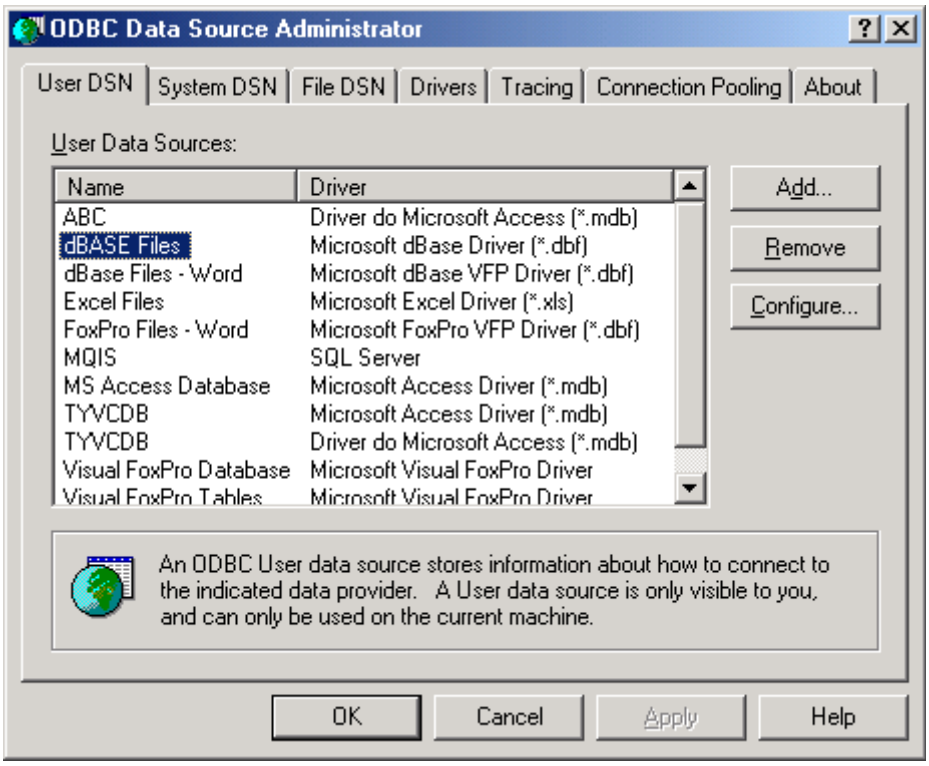
Để một ứng dụng *Client* nối với cơ sở dữ liệu *Client / Server* dùng ODBC, trước hết, ta phải cung cấp thông tin về nguồn dữ liệu ODBC trên *Client.* Mỗi server yêu cầu những gói thông tin khác nhau để nối với *Client.* ODBC cung cấp thông tin này một tên đơn giản để ta có thể tham chiếu đến nó, thay vì phải thiết lập gói thông tin từ đầu mỗi lần ta cần đến nó. Điều này cung cấp cho ứng dụng *Client* khả năng tham chiếu một cách dễ dàng đến tổ hợp của một điều khiển, một cơ sở dữ liệu và có thể có thêm tên một người sử dụng và mật khẩu. Tên này chính là tên của nguồn dữ liệu hay DSN.

Ví dụ trong phần này được tạo với phiên bản 3.51 của *Trình quản lý điều khiển ODBC* và phiên bản 3.6 của điều khiển *SQL Server.* Nếu bạn dùng một phiên bản

cũ hơn của ODBC, bạn sẽ thấy có một vài chỗ khác nhau ở hộp thoại của Trình quản lý điều khiển, cũng như thiếu một số tính năng. Hơn nữa, phiên bản cũ của ODBC không có khả năng kiểm nghiệm một kết nối ở trong trình quản lý điều khiển. Bạn có thể tải xuống phiên bản mới nhất của trình quản lý điều khiển ODBC như một phần của Thành phần truy cập dữ liệu của Microsoft (*Microsoft Data Access Components- MDAC*), chứa ở địa chỉ *http:// www.microsoft.com/data/*.

Để tạo một tên nguồn dữ liệu ODBC trên máy *Client,* ta theo các bước sau:

Bảo đảm rằng ta có một *SQL Server* đang hoạt động, và chỉ có thể truy cập nó từ máy *Client.* Điều này để tránh những vấn đề không thuộc ODBC có thể xảy ra như là nối mạng, bảo mật, v.v...
Phóng *Control Panel* từ menu Start.
Từ *Control Panel,* nhấn đúp chuột lên biểu tượng ODBC. Hộp thoại Quản trị Nguồn dữ liệu xuất hiện:



Hình Hộp thoại Quản trị nguồn dữ liệu ODBC

Danh sách các nguồn dữ liệu có thể thay đổi theo máy. Đến đây, ta có thể tạo một trong ba kiểu nguồn dữ liệu ODBC:

User DSN : Chỉ có người tạo ra nó mới sử dụng nó và chỉ trên máy đang dùng.
System DSN: bất kỳ ai sử dụng máy này cũng có thể dùng được. Đây cũng là kiểu nguồn dữ liệu mà ta cần tạo khi cài đặt ứng dụng cơ sở dữ liệu Web.
File DSN: có thể được copy và sử dụng dễ dàng bởi máy khác.

15.1.2.1 Tạo System DSN

Chọn vào tab System DSN trong cửa sổ **ODBC Data Source Administrator**

Nhấn nút Add.

Hộp thoại Create New Data Source xuất hiện. Chọn tên của điều khiển cơ sở dữ liệu ta muốn dùng (trong ví dụ này là SQL Server).

Nhấn **Finish.** Trình tạo nguồn dữ liệu mới đến SQL Server xuất hiện.

Trong ô **Name**, nhập tên của nguồn dữ liệu. Tên này sẽ được dùng trong ứng dụng Client để tham chiếu đến cơ sở dữ liệu, vì vậy, nên đặt tên sao cho gợi nhớ, có thể lấy tên của cơ sở dữ liệu. (Ví dụ là Novelty).

Điền vào ô **Description** là tuỳ chọn. Nó cung cấp thêm thông tin gắn liền với tên nguồn dữ liệu ODBC. Thông tin này chỉ hiển thị trong cửa sổ ODBC của Control Panel, và được xem như một thông tin về tên nguồn dữ liệu.

Trong hộp kết hợp **Server,** chọn SQL Server chứa cơ sở dữ liệu mà ta đang làm việc.

Nếu máy tính mà ta đang dùng cũng đang nối với SQL Server, tên của server sẽ xuất hiện trong danh sách thả xuống. Nếu SQL Server chứa trên cùng máy, ta có thể dùng local để thay thế tên của server

Nhấn **Next.** Màn hình kế tiếp của Trình tự động xuất hiện, hỏi ta cách login vào server. Ta có thể chọn cơ chế login của WinNT, trong đó, login mạng được dùng như là ID của người sử dụng và mật khẩu. Nhấn **Next.**

Có thể không cần nhập mật khẩu. Ta chỉ dùng đến nó khi mở một kết nối đến cơ sở dữ liệu. Điều này ngăn cản những rắc rối bảo mật tiềm ẩn, vì mọi thông tin của ODBC DSN đều được chứa trong **registry** của máy tính.

Màn hình kế tiếp của Trình tự động xuất hiện. Chọn vào hộp dánh dấu "Change the default database to", rồi chọn cơ sở dữ liệu Novelty từ hộp kết hợp. Mặc dù bước này là tuỳ chọn, ta nên luôn liên kết tên cơ sở dữ liệu với tên nguồn dữ liệu ODBC.

Nhấn **Next.** Màn hình kế xuất hiện, nhắc ta chọn thông dịch bộ ký tự. Thông thường, ta không cần đổi bất kỳ mục nào trong màn hình này, trừ phi ta đang sử dụng bộ ký tự khác trên server, vì vậy, nhấn **Next.**

Màn hình kế cho ta chọn khả năng kích hoạt tác vụ ghi nhật ký. Nó cho phép ta xem những hành động bên trong mà ODBC làm khi gải quyết một truy vấn. Thông thường, ta chỉ chuyển nó thành on nếu ta đang gặp lỗi hay tìm kiếm những nguyên nhân ách tắc trong ứng dụng.

Hình 23.3 Sử dụng cửa sổ điều khiển ODBC để chọn một cơ sở dữ liệu SQL Server mặc định

Nhấn **Finish.** Hộp thoại xuất hiện, mô tả chi tiết của nguồn dữ liệu mà ta vừa tạo. Sau đó, nhấn **nút Test Data Source.** Trình điều khiển sẽ đáp ứng bằng cách thông báo một kết nối vừa được thiết lập thành công.

Sau khi nhấn OK, tên nguồn dữ liệu mới xuất hiện trong cửa sổ ODBC Data Source Administrator.

15.1.2.2 Kiểm nghiệm kết nối cơ sở dữ liệu với ODBCPING

Phiên bản cũ của ODBC không có khả năng nối vào nguồn dữ liệu để kiểm định những thông tin kết nối mà ta cung cấp cho trình quản lý điều khiển có hợp lệ không.

Công việc này hữu dụng khi ta cần xác định nguyên nhân trục trặc trong một kết nối *Client / Server* là do server, do kết nối mạng, do ODBC, hay do ứng dụng *Client.* Tiện ích ***ODBCPING*** không giải quyết được trục trặc nhưng tối thiểu nó thông báo cho ta biết có kết nối được với server thông qua ODBC hay không.

Cú pháp:

odbcping /Uusername /Ppassword /Sserver

Ví dụ, để kiểm định kết nối với *SQL Server* tên là BEDROCK dùng tên login là *randy* và mật khẩu là *prince,* ta dùng dòng lệnh sau:

odbcping /Urandy /Pprince /SBEDROCK

Từ menu *Start,* phóng *dấu nhắc DOS.*

Trong cửa sổ DOS, nhập vào dòng lệnh:

odbcping /Urandy /Pprince /SBEDPOCK

ODBCPING thiết ập nối kết với *server* rồi thoát.

15.1.3 Truy cập nguồn dữ liệu với điều khiển DAO DATA và ODBCDIRECT

Tạo một ứng dụng Visual Basic với điều khiển ***ADO Data*** nối với nguồn dữ liệu ODBC rất dễ. Trong thực tế, nó không đòi hỏi phải lập trình. Ta có thể dùng điều khiển ***Data*** để nhanh chóng kiểm tra kết nối đến cơ sở dữ liệu *Client / Server,* hoặc ta có thể dùng nó để tạo thử một giao diện người sử dụng với kết nối trực tiếp đến cơ sở dữ liệu.

Vì những lý do về khả năng hoạt động, ta có thể dùng điều khiển dữ liệu từ xa, hoặc là điều khiển ***ADO Data*** thay vì điều khiển ***DAO Data,*** bởi vì những điều khiển này chuyên dùng cho truy cập dữ liệu trên *Client / Server.* Trong những phiên bản DAO trước phiên bản 3.5, DAO tự động nạp bộ máy cơ sở dữ liệu Jet mỗi khi nó truy cập dữ liệu *Client / Server* -thậm chí khi ta không thực sự dùng cơ sở dữ liệu Jet/Access. Trong VB5, ta có thêm tuỳ chọn sử dụng ODBCDIRECT để truy cập dữ liệu Client / Server. ODBCDIRECT chỉ là một chuyển đổi để truy cập server thực tiếp thông qua DAO mà không cần nạp bộ máy cơ sở dữ liệu Jet. Điều này tiết kiệm bộ nhớ và giảm thời gian nạp ứng dụng trên máy Client. Nó thích khi ta phải dùng DAO để truy cập dữ liệu Client / Server nhưng không bận tâm về tính linh hoạt, khả năng sử dụng lại và tính dễ bảo trì của chương trình.

VÍ DỤ MẪU *Tham khảo ví dụ mẫu của phần này trong đề án* **ODBCDAO.vbp,** *chứa trong thư mục \samples\PhanIV\23-rdo\ODBCDAO*

15.1.3.1 Kiểm nghiệm SQL Server DSN

Tạo nguồn dữ liệu ODBC tên là *Novelty.*

Phóng Visual Basic và tạo đề án mới kiểu ***Standard EXE.***

Từ menu ***Project Components,*** lập một tham chiếu đến điều khiển lưới ràng buộc dữ liệu (*Microsoft Data Bound Grid Control*).

LƯU Ý *Điều khiển* ***DBGrid*** *ta đang dùng ở đây chỉ để duyệt dữ liệu. Nó có nhiều cách dùng khác nhau. Tham khảo chương 30 "Sử dụng điều khiển DBGrid và điều khiển Apex True DBGrid".*

Điều khiển ***DBGrid*** xuất hiện trong hộp công cụ. thêm một *instance* của điều khiển ***DBGrid*** và một điều khiển ***Data*** chuẩn vào biểu mẫu

Đổi thuộc tính ***DefaultType*** của điều khiển *Data* thành 1- *Use ODBC*. Nó khiến cho ứng dụng bỏ qua bộ máy Jet, và do đó, làm ứng dụng khởi động nhanh hơn và tốn ít bộ nhớ hơn.

Trong thuộc tính ***Connect*** của điều khiển ***Data,*** đưa vào chuỗi ký tự sau:

ODBC; UID=randy; PWD = prince;DSN = Novelty

Trong thuộc tính ***RecordSource*** của điều khiển *Data,* đưa vào câu truy vấn:

Select * from qryCustomer

Bởi vì ta hạn chế quyền truy cập trực tiếp đến bảng trên cơ sở dữ liệu, ta chỉ truy vấn trên ***View*** của cơ sở dữ liệu (*qryCustomer*).

Quy định thuộc tính ***DataSource*** của điều khiển *DBGrid* là *Data1*.

Thi hành ứng dụng. Lưới hiển thị toàn bộ dữ liệu trong bảng *tblCustomer*.

15.2 Truy cập dữ liệu dùng điều khiển dữ liệu từ xa

*Điều khiển dữ liệu từ xa (**Remote Data Control -** RDC)* là một cách truy cập dữ liệu từ xa trong ứng dụng viết bằng Visual Basic. Điều khiển này dùng một giao diện lập trình tương tự điều khiển ***ADO Data,*** hay là điều khiển ***DAO Data.*** Trừ một vài ngoại lệ nhỏ, RDC hoạt động tương tự các điều khiển dữ liệu khác – ta cung cấp cho điều khiển một số thông tin về nơi chứa dữ liệu, nó sẽ trả về dữ liệu và cung cấp cho các điều khiển giao diện người sử dụng quan tâm đến dữ liệu. Người sử dụng sau đó có thể tương tác với dữ liệu thông qua các điều khiển trên giao diện - duyệt dữ liệu, cập nhật, thêm bản ghi mới – và ta không phải lập trình về các hoạt động này.

*Lưu ý rằng ta không hạn chế vào **RDO** khi muốn sử dụng dữ liệu Client / Serve. VB6 giới thiệu một điều khiển dữ liệu mới cho nguồn dữ liệu* **ADO** *được thiết kế để thay thế RDC. Tìm hiểu chi tiết thêm trong chương 27.*

15.2.1 Sử dụng RDC

7. Trong Visual Basic, chọn menu *Project, Components.*
8. Từ danh sách các thành phần, chọn *Microsoft Remote Data Control.* Thiết lập tham chiếu đến *Microsoft Data Bound Grid Control.*

9. Nhấn OK. Điều khiển ***Remote Data*** *và* ***Data Bound Grid*** xuất hiện trên hộp công cụ.

Hình *Biểu tượng của điều khiển* **RDC** *(Microsoft Remote Data Control) và điều khiển Microsoft Data Bound Grid xuất hiện trên hộp công cụ.*

4. Tạo *instance* cho điều khiển ***Remote Data*** và điều khiển ***Data Bound Grid*** trên biểu mẫu.

5. Nếu chưa cung cấp tên người sử dụng và mật khẩu khi tạo tên nguồn dữ liệu ODBC, ta đổi thuộc tính ***UserName*** của điều khiển ***Remote Data*** là tên người sử dụng để truy cập cơ sở dữ liệu, chẳng hạn như là "*randy*" hay "*sa*".

6. Đổi thuộc tính ***Password*** của điều khiển ***Remote Data*** là mật khẩu của người sử dụng truy cập cơ sở dữ liệu.

7. Quy định thuộc tính ***SQL*** của điều khiển *Remote Data* là:

```
Select * from qryCustomer
```

8. Trong thuộc tính ***DataSourceName*** của điều khiển *Remote Data* chọn *DSN* cho cơ sở dữ liệu *SQL Server.* DSN ta tạo trước đây sẽ xuất hiện trong danh sách thả xuống.

9. Chỉ định thuộc tính ***DataSource*** của điều khiển *DBGrid* là *MSRDC1,* tên của điều khiển *Remote Data.*

10. Thi hành ứng dụng. Ứng dụng trả về nội dung của bảng *tblCustomer* và hiển thị nó trong lưới dữ liệu.

Ngiên cứu kỹ chương này, bạn sẽ thấy rằng sử dụng điều khiển ***Data*** với tuỳ chọn **ODBCDIRECT** và điều khiển *Remote Data* để truy cập dữ liệu về cơ bản là như nhau. Cả hai đều cho phép truy cập đến cơ sở dữ liệu ODBC không cần lập trình.

VÍ DỤ MẪU *Tham khảo ví dụ mẫu của phần này trong đề án* **RDCTest.vbp,** *chứa trên thư mục \samples*
PhanIV\23-rdo\RDCTest

15.3 Sử dụng RDO trong chương trình

Đối tượng dữ liệu từ xa(Remote Data Object-RDO) được sắp xếp trong cấu trúc phân nhánh đối tượng dữ liệu tương tự như **DAO(Đối tượng truy cập dữ liệu –** Data Access Object **).**

Lưu ý rằng mô hình đối tượng của **RDO** đơn giản hơn **DAO.** Nhiều chức năng của **DAO** (như là đáp ứng về bảo mật và khả năng sửa đổi thiết kế cơ sở dữ liệu) được xử lý bởi bộ máy cơ sở dữ liệu. Mặt khác, trong RDO, chức năng này được xử lý bởi server.

LƯU Ý ADO cung cấp mô hình đối tượng đơn giản hơn **RDO,** trong khi vẫn đảm bảo mọi chức năng của **RDO.**

Điểm hạn chế này có nghĩa là ta không thể dùng **RDO** để tạo đối tượng cơ sở dữ liệu như là bảng, view, và thủ tục chứa sẵn (ít ra, ta không thể sử dụng đối tượng để thực hiện điều này – Tuy nhiên, ta có thể dùng RDO để truyền trực tiếp câu lệnh SQL theo thủ tục đến server, nghĩa là hầu hết những gì có thể làm với ISQL/w, ta cũng có thể làm với RDO).

1. Lập một tham chiếu đến **RDO** trước khi sử dụng nó. Chọn menu Project, References. Hộp thoại **References** xuất hiện:
2. Chọn Microsoft Remote Data Object 2.0 từ danh sách
3. Nhấn OK. **RDO** xuất hiện trong ứng dụng.

Thực hiện các bước này khi ứng dụng cần truy cập **RDO**. Nếu ứng dụng dùng **RDC**, ta cần phải lập một tham chiếu đến **RDO** và thêm **RDC** vào đề án.

Hình 23.5 *Kiến trúc đầy đủ của RDO 2.0*

15.3.1 Quy định thuộc tính bộ máy cơ sở dữ liệu dùng đối tượng RDOENGINE.

Đối tượng ***rdoEngine*** là đối tượng ở mức cao nhất trong mô hình đối tượng **RDO**. Nói chung, ta dùng đối tượng rdoEngine để chỉ định thuộc tính mặc định cho các đối tượng RDO khác mà ứng dụng tạo ra.

Đối tượng ***rdoEngine*** còn hữu dụng trong trường hợp ta muốn quy định hay kiểm tra kiểu con trỏ được dùng bởi bộ máy cơ sở dữ liệu. Con trỏ là cách truy cập từng dòng của bộ kết quả. Con trỏ khác nhau có những khả năng khác nhau; ví dụ, con trỏ cuộn tới cho phép truy cập các dòng tuần tự, nhưng không được phép quay ngược lại và tham chiếu đến những dòng trong bộ kết quả ta vừa dịch chuyển qua.

*Đối tượng **rdoEngine** tương đương đối tượng **DBEngine** của **DAO**. Không có điểm tương tự trực tiếp với rdoEngine trong **ADO**; chức năng của nó bị phân chia giữa các đối tượng **Connection** và **Recordset** của ADO.*

Ngoài khả năng thể hiện nguồn dữ liệu, đối tượng ***rdoEngine*** còn chứa tập hợp ***rdoErrors*** cho phép ta lặp xuyên qua toàn bộ thông báo lỗi phát sinh bởi một *transaction* của một cơ sở dữ liệu nhất định.

Đối tượng ***rdoEngine*** chứa tập hợp các đối tượng ***Enviroment***, cũng như tập hợp các đối tượng ***rdoError***.

*Ví dụ, nếu ứng dụng cần tạo một số đối tượng dựa trên login thông thường và mật khẩu của người sử dụng, ta có thể chỉ ra các yếu tố này khi ta bắt đầu sử dụng **RDO** trong ứng dụng. Các giá trị mặc định ta quy định (dùng thuộc tính rdo**DefaultPassword** và **rdoDefaultUser**) là dành cho các đối tượng rdoEnvironment mà ta tạo ra trong ứng dụng.*

Ta cũng có thể dùng đối tượng ***rdoEngine*** để diều khiển cách thức tạo và duy trì các con trỏ trong ứng dụng.

Ngoài việc cung cấp các giá trị mặc định cho những đối tượng tạo bởi **RDO**, đối tượng ***rdoEngine*** tự động tạo đối tượng ***rdoEnvironment***. Ta có thể tham chiếu đến đối tượng này trong chương trình như là *rdoEnvironments(0).*

Hình 23.6 *Tập hợp **Environment** và đối tượng **Environment** trong cây phân cấp* **RDO** *(Đối tượng dữ liệu từ xa)*

15.3.2 Truy cập môi trường đối tượng rdoEnvironment

Đối tượng ***rdoEnvironment*** thể hiện môi trường cơ sở dữ liệu. Đây là một cách tham chiếu đến bộ kết nối cơ sở dữ liệu (theo dạng của đối tượng rdoConnection).

*Đối tượng **rdoEnvironment** tương tự đối với Workspace trong **DAO**. Với **ADO**, rdoEnvironment tương tự với đối tượng Connection của **ADO**.*

Cũng như đối tượng ***rdoEngine***, không chắc rằng ứng dụng cần tạo nhiều hơn một *instance* của đối tượng ***rdoEnvirment*** - trừ phi ta cần hỗ trợ nhiều *transaction* đồng thời trên nhiều kết nối cơ sở dữ liệu. Thay vì tạo một *instance* mới của đối tượng *rdoEnvironment*, ta có thể truy cập đối tượng *rdoEnvironment* hiện hành, *rdoEnvironment(0)*, được tạo bởi đối tượng *rdoEngine* khi ta lập một tham chiếu đến RDO từ ứng dụng.

Đối tượng rdoEnvironment thuộc về tập hợp rdoEnvironmént và chứa tập hợp gồm các đối tượng rdoConnection:

Hình 23.7 *Tập hợp* ***rdoEnvironments*** *và đối tượng* ***rdoEnvironment*** *trong cây phân cấp* **RDO** *(Đối tượng dữ liệu từ xa)*

15.3.3 Thiết lập kết nối dùng đối tượng rdoConnection

Ta dùng đối tượng ***rdoConnection*** để thiết lập một kết nối đến *server* cơ sở dữ liệu từ xa trong RDO. Sau khi ta có một đối tượng ***rdoConnection*** hợp lệ, ứng dụng có thể bắt đầu tương tác với cơ sở dữ liệu.

Đối tượng rdoConnection tương tự đối với ***Database*** *trong lập trình với DAO. Nó cũng tương tự theo một nghĩa nào đó với đối tượng Connetion của ADO.*

Đối tượng ***rdoConnection*** thuộc về tập hợp ***rdoConnections*** và chứa tập hợp các đối tượng ***rdoQuery***, đối tượng ***rdoResultset***, và đối tượng ***rdoTable***.

Hình 23.8 *Tập hợp* ***rdoConnections*** *và đối tượng* ***rdoConnection*** *trong cây phân cấp* **RDO** *(Đối tượng dữ liệu từ xa)*

Để tạo một kết nối dùng đối tượng rdoConnection, ta bắt đầu bằng cách tạo chuỗi kết nối ODBC. Khác với điều khiển Data, ta bỏ qua ODBC; cũng như mệnh đề của chuỗi kết nối của đối tượng rdôCnnection. (Nó ám chỉ rằng vì sử dụng RDO, ta sẽ dùng ODBC).

Dưới đây là danh sách các phần tử thông dụng nhất trong chuỗi kết nối ODBC khi kết nối với SQL Server.

Tham biến	**Mô tả**
UID	Tên login của người sử dụng
PID	Mật khẩu của người sử dụng
DSN	Tên nguồn dữ liệu ta tạo trong Trình quản lý điều khiển ODBC
DRIVER	Điều khiển ODBC mà ta muốn kết nối
DATABASE	Tên cơ sở dữ liệu mà ta muốn kết nối
APP	Tên ứng dụng kết nối với cơ sở dữ liệu
LANGUAGE	Ngôn ngữ quốc tế được dùng bởi server
SERVER	Tên của SQL Server mà ứng dụng kết nối đến

Tuy nhiên, đây không phải toàn bộ các tham biến. Và không nhất thiết phải dùng tất cả các tham biến này trong mọi trường hợp. Cơ sở dữ liệu có thể sử dụng nhiều hoặc ít hơn.

Ví dụ, để nối đến cơ sở dữ liệu SQL Server Novelty, ta dùng chuỗi kết nối:

```
DSN = Novelty; UID = randy; PWD = prince
```

Nếu không cung cấp tên login và mật khẩu của người sử dụng trong chuỗi kết nối, khi ứng dụng nối đến server, trình điều khiển của ODBC sữ hiển thị hộp thoại yêu cầu người sử dụng cung cấp thông tin login.

15.3.3.1 Tạo chuỗi kết nối không chứa DSN

Ngoài cách tạo kết nối sử dụng tên nguồn dữ liệu ODBC, ta còn có một tuỳ chọn để tạo một chuỗi kết nối không cần có DSN. Khi đó, chuỗi kết nối bao gồm toàn bộ thông tin cần thiết để login, kể cả tên trình điều khiển ODBC, chẳng hạn như:

Driver = SQL Server ; SERVER = bedrock; DATABASE = bedrock; UID = randy; PWD = prince;

Lưu ý rằng, thứ tự các phần tử trong chuỗi không quan trọng, nhưng cách đánh vần là có quan trọng. Một trong những lỗi phổ biến nhất mà nhà lập trình thường gặp khi xây dựng chuỗi là bỏ qua ký tự (chấm phẩy (;) hay ký tự ngoặc cong), hoặc chèn một ký tự lạ (như khoảng trắng trước và sau dấu bằng) trong chuỗi kết nối. ODBC rất kén chọn với định dạng của chuỗi kết nối, vì thế nên thận trọng.

Dùng chuỗi không có DSN trong trường hợp ta không cần điều khiển toàn bộ qua cấu hình phía Client. Nó cũng nhanh hơn một ít khi log vào cơ sở dữ liệu. Bởi vì DSN được chứa trong Registry của Windows, và rất tốn kém truy cập vào đấy.

15.3.3.2 Mở cơ sở dữ liệu

Sau khi đã có thông tin cần để thiết lập kết nối đến cơ sở dữ liệu, ta có 2 tuỳ chọn để thiết lập kết nối:

Dùng phương thức OpenConnection của đối tượng rdoEnvironment.

Dùng phương thức EstablishConnection của đối tượng

Cả hai phương thức đều như nhau, chúng chỉ hoạt động hơi khác nhau một chút.

15.3.3.2.1 Dùng phương thức OpenConnection

```
Option Explicit
' References RDO 2.0
Private MyConn As rdoConnection
Private Sub Form_Load()
Dim strConnect As String
strConnect = "DSN=Novelty;" & _
             "PWD=prince;" & _
             "UID=randy;"
Set  MyConn = rdoEnvironments(0).OpenConnection("Novelty", _
,, strConnect)
End Sub
```

Phương thức OpenConnection có 5 tham biến : Tham biến thứ nhất, tên nguồn dữ liệu, không phải là tuỳ chọn. Bốn tham biến còn lại (Prompt, Readonly, Connect, và Options) là tuỳ chọn. Điều này giải thích cú pháp lạ của phương thức - tham biến thiếu đặt giữa các dấu phẩy. (Lưu ý rằng đoạn chương trình này tạo

ngầm đối tượng rdoEnvironment, tham chiếu đến rdoEnvironment(0), để tạo một kết nối đến cơ sở dữ liệu).

Mặt khác, phương thức EstablishConnection, thực hiện yương tự như OpenConnection. Nhưng nó hoạt động theo cách trực tiếp hơn:

```
Option Explicit
' References RDO 2.0
Private MyConn As rdoConnection
Private Sub Form_Load()
Dim strConnect As String
strConnect = "DSN=Novelty;" & _
                "PWD=prince;" & _
                "UID=randy;"
MyConn.EstablishConnection
End Sub
```

Ví dụ mẫu Tham khảo ví dụ trong đề án projectConnect.vbp trong thư mục \samples\PhanIV\23-rdo\Connect.

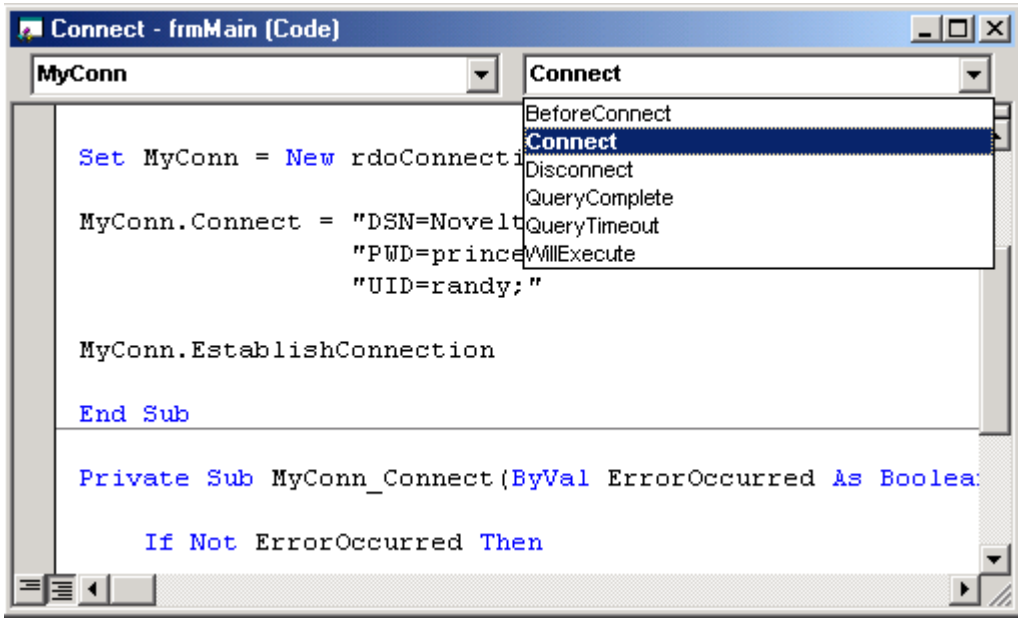
15.3.4 Đáp ứng sự kiện trong RDO

Đối tượng RDO có khả năng sinh ra sự kiện. Sự kiện còn cho phép ứng dụng thi hành những hành động mất nhiều thời gian (như là kết nối đến cơ sở dữ liệu hay thi hành câu truy vấn) mà không cần giữ điều khiển của ứng dụng. Thay vì chờ cho hành động của RDO xảy ra, ứng dụng có thể làm việc khác cho đến khi kết nối được thiết lập hay truy vấn hoàn tất. Ta gọi đây là thi hành không đồng bộ. Sự kiện là cách thức đối tượng RDO thông báo cho ứng dụng rằng một hoạt động bất đồng bộ vừa hoàn tất.

Để khiến một đối tượng dữ liệu đặc biệt sinh ra sự kiện, ta phải khai báo nó theo cách đặc biệt: dùng từ khoá WithEvents. Ví dụ, khai báo đối tượng rdoConnection có tên là MyConn phát sinh sự kiện, ta dùng dòng lệnh:

Private WithEvents MyConn As rdoConnection

Sau khi khai báo, các sự kiện của nó sẽ xuất hiện trong cửa sổ Code của Visual Basic.



Hình Truy cập thủ tục sự kiện của một đối tượng RDO được khai báo bằng từ khoá WithEvents.

Sau đó, ta có thể viết các thủ tục xử lý sự kiện. Các thủ tục này tương tự thủ tục xử lý sự kiện của các đối tượng cơ bản khác, như thủ tục sự kiện Load của biểu mẫu hoặc thủ tục sự kiện Click của nút lệnh.

Cách dễ thất để minh hoạ cách thức hoạt đọng của các sự kiện RDO là viết một thủ tục thông báo khi một kết nối đến cơ sở dữ liệu được thiết lập.

```
Option Explicit
 ‘referene rdo 2.0
WithEvents Myconn As rdoconnection

Private Sub Form_Load()
   Set Myconn = New rdoconnection
   Myconn.conect = "DSN=novelty;" & _
                   "PWD=rince;" & _
                   "UID=randy;"
   Myconn.EstablishConnection
End Sub
Private Sub myconn_connect(ByVal ErrorOcurred As Boolean)
   If ErrorOcurred Then
      MsgBox "There was a problem" & _
         "connecing to the database.", vbExclamation
   Else
      MsgBox "connection established.", vbInformation
   End If

End Sub
```

15.4 Tạo kết nối với trình thiết kế uerconnecttion

Trình thiết kế kết nối người sử dụng (Uerconnection designer) tạo sự dễ dàng ho ngườI sử dụng Visual Basic để kết nối với cơ sở dữ liệu client/server dùng RDO. Nó phát sinh đốI tượng **rdoconnection** mà không phảI lập trình nhiều
Tương tự biểu mẫu trong đề án VB, ta thêm trình thiết kế **Uer_connecttion** vào đề án và dùng nó như dùng biểu mẫu. Khi ta biên dịnh tạp tin thi hành EXE trong VB, trình thiết kế userconnection cũng được biên dịch như biểu mẫu.

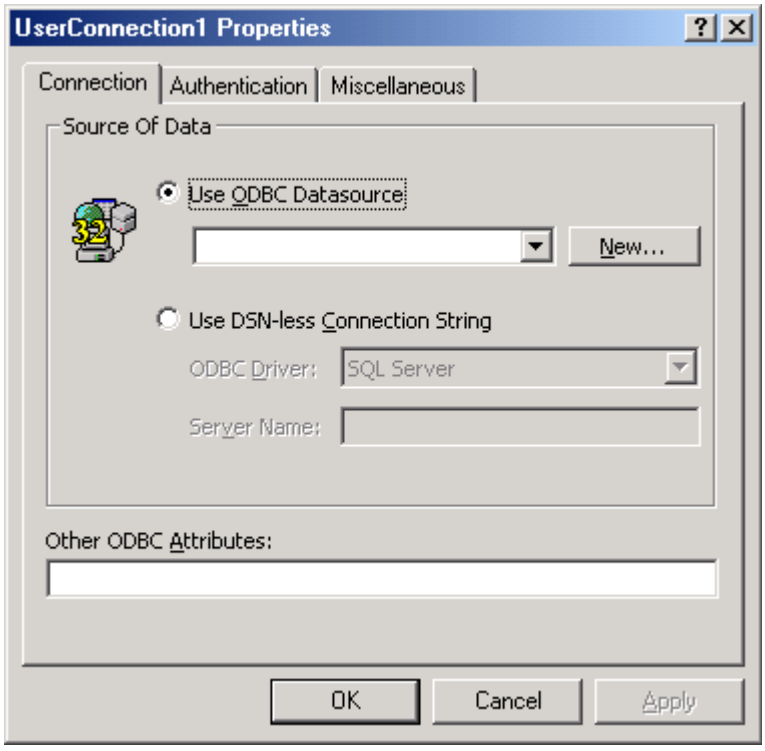
Lưu ý với ADO trong VB 6, ta có một kiểu thiết kế mới để tậo kết nối vớI cơ sở dữ liệu. thiết kế này gọI là **dataenviroment**, sẽ được trình bày trong chương 27 "ĐốI tượng dữ liệu ActiveX". Nó vượt xa khảng năng của thiết kế UserConnection trong một số sách. thiết kế UserConncton là một dạng mô-đun lớp.

Để dùng trình thiết kế UserConnection,ta làm như sau:

1. Thêm một trình thiết kế UserConnection mới vào đề án.
2. dùng giao diện đồ hoạ của UserConnection, thể hiện nguồn dũ liệu ODBC mà ta muốn nối đến, và cách thức kết nối.
3. trong chương trình, tạo một instance của đối tượng **rdoconnection** từ thiết kế UserConnection.

Ví dụ mẫu –dùng trình thiết kế UserConnection để két nối với cơ sở dữ liệu ODBC

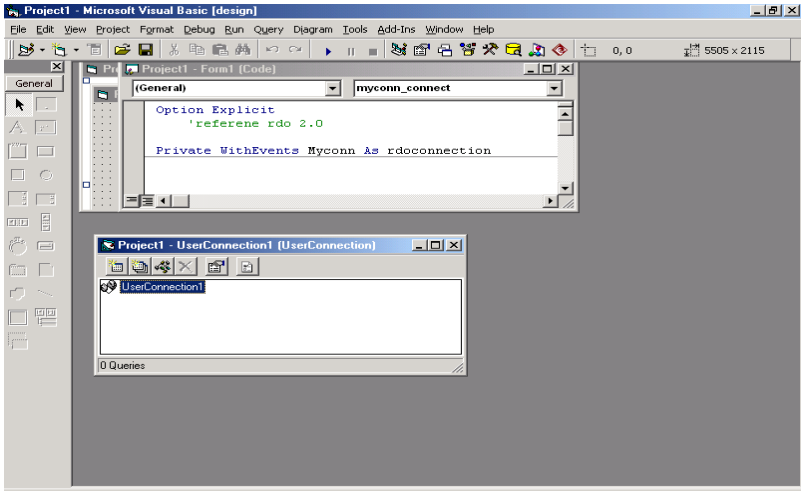
1. Trong đề án VB, chọn menu Project, more ActiveX Designers. Chọn Microsoft UserConnection.
2. Một trình thiết kế UserConnection được thêm vào đề án, và hộp thoại **UserConnection properties** xuất hiện:



3. Trong *tab Connection,* chọn nguồn dữ liệu ODBC hay xây dựng chuỗi kết nối không có DSN.
4. Chon tab Authentication, nhập tên và mật khẩu người sử dụng. Chọn vào hộp đánh dấu " Save Connection Information for Design=time". Như thế, ta có thể truy cập kết nối bất kỳ lúc nào ta muốn.

   Lưu ý trong trình thết kế UserConnection, cũng tương tự như với ODBC DSN, ta có tuỳ chọn là bỏ qua các thông tin gắn liền với việc thẩm định. Bỏ qua công việc này nghĩa là tên người login và mật khâu sẽ không được biên dịch vào ứng dụng và nó không được lưu cùng với trình thiết kế UserConnection. Mặc dù vậy, nhớ rằng thiết kế UserConnection không chứa dữ liệu cho thẩm định, trong khi bạn không cung cấp các thông tin đó trong chương trình khi thực thi, do đó, trình điều khiển ODBC sẽ hiển thị hộp thoại mỗi lần ứng dụngkết nối đến cơ sở dữ liệu.(Tắt hộp thoại login này trong thiết kế UserConnection bằng cách dùng hộp kết hợp Prompt Behavior trong tab Authentication của hộp thoại Properties của thiết kế).

   Trong ứng dụng, ta muốn giải quyết vấn đề thẩm định người sử dụng theo một cách khác - trả về tên người sử dụng và mật khẩu tại thời điểm người sử dụng khởi động ứng dụng, hay(đối với các ứng dụng hỗ trợ quyết định) tạo một người sử dụng chỉ đọc đặc biệt để duyệt dữ liệu.
5. Nhấn OK.
6. hộp thoại properties đóng. Trên màn hình xuất hiện tên trìh thiết kế UserConnection.



Hình Cửa sổ thiết kế UserConnection sau khi ta đặt thuộc tính khởi động nó.

7. Trong cửa sổ properties cảu VB, gõ vào một tên cho trình thiết kế UserConnection. Ta dùng tên này để tham chiếu đến trình thiết kế UserConnection tong chương trình. Ví dụ ta đặt tên trình thiết kế UserConnection két nối đến cơ sở dữ liệu company là conCompany.

15.5 Truy cập truy vấn với trìng thiết kế UserConnection

Ta có thể tạo câu truy vấn trong trình thiết kế UserConnection theo một tổng ba cách:

Gọi một thủ tục chứa sẵn hay view trên server cơ sở dữ liệu để trả vềcác bản ghi.

Phát một câu lệnh cho server cơ sở dữ liệu dưới dạng một câu truy vấn SQL động.

Xây dựng một câu truy vấn trong trìng thiết kế UserConnection dùng Microsoft query. Kỹ thuật này tương tự kỹ thuật truy vấn động, nhưng nó mạng tính đồ hoạ hơn, vì vậy đê xây dựng lúc thiết kế.

15.5.1 Gọi thủ tục chứa sẵn trong một trình thiết kế UserConnection

15.5.1.1 Thêm thủ tục vào tình thiết kế UserConnection

Để gọi thủ tục chứa sẵn tù một đối tượng được tạo ra từ trình thiết kế UserConnection, vào lúc thiết kế, ta pahỉ thêm thủ tục chứa sẵn vào trìn thiết kế UserConnection. Sau đó, ta có thể truy cập thủ tục như một phương thức của đối tượng kết nối.

1. Trên thanh công cụ của trình thiết kế UserConnection, nhấn nút **Insert Query**
2. Hộp thoại Query Properties xuất hiện. trong hộp **query name,** nhập tên truy vấn; có thể trùng với tên thủ tục chứa sẵn.
3. Hộp kết hợp *Based on stored Procedure* cung cấp một danh sách các thủ tục chứa sẵn trên cơ sở dữ liệu company. Ta chọn thủ tục **lastNameLookup.**
4. Chọn tab Prameters. thủ tục *LastNameLoopup* lấy tham số name, là tên người sử dụng đang tim kiếm.
5. nhấn OK. thủ tục *LastnameLoopup* được thêm vào trìh thiết kế **UserConnection**

/**********

anh

**********/

15.5.1.2 Gọi thủ tục

Tạo một instence của *UserConnection* trong chương trình, sau đó, thi hành thủ tục chúa sẵn như là một phương thức của đối tượng **rdoConnection**.

1. Thêm một hộp danh sách chuẩn, một nút lệnh chuân, va một hộp văn bản chuẩn vào biểu mẫu trong đề án chứa UserConnection. Dùng hộp văn bản để nhập tham ssố cho truy vấn. Chương trinh thi hành câu truy ấn được kích hoạt bằng cách nhấn vào nut lệnh.
2. trong phần *Declatians* khai báo một biến chứa rdoConnection và **rdoresultset**

```
Private Conn As concompany
Orivate rs As rdoresultSet
```

3. Trong suwj kieenj click của nut lệnh, lập trìh để tạo instance của UserConnection, nối với cơ sở dữ liệu và thi hành thủ tục chứa sẵn.

```
Private Sub command1_click()
        Set conn = New conCompany
        conn.EstablishConnection
        conn.lastnameloopup Text.tex
        Set rs = con.LastQueryResults
        list1.Clear
        Do util rs.eof
                list1.additem rs!lastName&" "&rs!fistName
                rs.MoveNext
        Loop
        conn.Close
End Sub
```

Lưu ý rằngkhi nhấn ký tự dấn chấm sau đối tượng Conn, thủ tục chứa sẵn được tham chiếu bơi UserConnection xuất hiện trong danh sách các thành phần của đối tượng.

4. thi hành ứng dụng. Nhập tên Smith vào hộp văn bản, rồi nhấn nút lệnh. Danh sách liệt kê những tên chưa từ "Smith" như phần đầu.

15.5.2 Dùng Microsotf Query để xây dựng chuỗi SQL trong trình thiết kế UserConnection.

Ta vừa thấy tham chiếu đến một câu truy vấn chứa trên server dùngtrnhf thiết kế UserConnection rất rễ. Nếu câu truy vấn không có sẵn trên server, ta phải phát sinh trên client rồi sau đó sẽ thi hành trên server, kết quả tra về Client.

Ta có thể nhập chuỗi SQL một cách thủ công hoặc dùng công cụ đồ hoạ trên Client. Microsoft Query là một công cụ gần gũi nhất ới trình thiết kế UserConnection.

```
/*************************

**************************/
```

15.5.2.1 Thêm truy vấn vào trình thiết kế UserConnection dùng Microsoft Query

1. Nhấn nút **Insert Query** trên thanh công cụ của trình thiết kế UserConnection.
2. hộp thoại Properties của truy vấn xuất hiện.
3. Nhập tên của truy vấn trong hộp văn bản Query Name.
4. chon khả năng tuỳ chọn *Based on User-Defined SQL*,rồi nhấn **Build.**
5. Microsoft Query phóng lên nguồn dữ liệu ta muốn kết nối, rồi nhấn OK.

/**********

**********/

Ta phải chỉ ra nguồn dư liệu trở lại khi ta thực hiện trong phần trước. ta co một DSN của người dùng cho cơ sở dữ liệu company. Vì một lý do nào đó, nó muốn dùng DSN dựa trên tập tin thay vì Uer-DSN hay System-DSN. May mắn là,MS query cho ta khả năng tạo DSN tong chương trình, vì thế, đây không là vấn đề lớn. Tuy nhiên, ta vẫn muốn thiết lập mọi thứchứa trong một nơi.

Một vấn đề khác với MS query la nos cai đặt để tìm kiếm các tập tin DSN trong thư mục trong \Program Files \Common Files\Microsoft Shared\Vba, trong khi ODBC tạo DSN dựa trên tập tin \Program Files \Common Files \ODBC\Data Sources. Trong hộp thoại **choose Data Source** của Ms Query, ta giải quyết vấn đề nàt bằng cách nhấn nút Option để chi ra thư mục tìm kiếm DSN. Để lam điêu này, nhấn **Browse** trong hộp thoại Data Source Options, chọn đúng thư mục, nhấn OK rồi nhấn **Add.** Khi ấy, thư mục mới sẽ được nhận ra bởi MS Query về sau.

6. trình tự động Query Winzard thi hành, hiện thị dữ liệu có sẵn trong nguồn dữ liệu. Hộp đầu tiên của Winzard hiển thị.
7. Nhấn đúp chuột lên truy vấn hay view mà ta định dùng (ví dụ,chon *qryCustomer*). Truy vấn hay View được mở rộng để hiển thị danh sách các trường chứa trong truy vấn.
8. Nhấn đúp lên mỗi trường ta muốn lấy về, ví dụ chon *FirstName* và *LastName.*
9. Nhấn **Next.**
10. Bước Lọc dữ liệu (Filter Data) xuất hiện. Ở bước này, ta xácđịnh cách thức lọc bản ghi. Ví dụ, để trả về những khách hàng tên Jones, chon trường Lastname từ danh sách các trường. Sau đó, trong hộp kết hợp bên phải danh sách cột,chọn Equal. Trong hộp kết hợ kế tiép, chọn Jones.
11. Nhấn **Next**. Bước kế tiếp cho phép ta sắp xếp dữ liệu. Trong hộpkết hợp Sort, chon FirstName.

12. Nhấn **Next. Khi** hoàn tất, ta có thể tuỳ chọn để xem dữ liệu trả về hay trở lại thiết kế UserConnection. Chọn View Data or Edit Query trong MS Query, rồi nhấn **Finish**.
13. Dữ liệu trả về hiển thị trong MS Query.

Ví dụ mẫu Tham khảo ví ụ mẫu trong đề án Query.vbp, trong thư mục \sample\PhanIV\23-rdo\Query.

Khi hoàn tất xem dữ liệu, chọn menu File, **Exit to Connection Designer.** Microsoft Query thoát ra và trả truy vấn ta xây dựng về hộp thoại Properties của truy vấn của trình thiết kế UserConnection dưới dạng chưỗi SQL.

Nhấn OK để đóng hộp thoại Query Properties. Truy vấn được chứa trong trình thiết kế UserConnection.

Có vài ưu điểm khi chứa truy vấn trong trình thiết kế UserConnection:

- Dùng lại thiết kế trong nhiều đề án với nhiều người lập trình mà không cần thay đổi về phía server.
- Kiểm soát được độ phức tạp của ứng dụng vì không có nhiều dòng lệnh để bảo trì.

Tuy nhiên, nhược điểm của kỹ thuật này là truy vấn trên Client, chung sẽ không truy cập được từ mọi Client khác. Như thế,từng Client có cách truy nhập khác nhau, dễ dấn tớitình trạng không nhất quán, nhất là khi thiết kế cơ sở dữ liệu phát triển theo thời gian. Nếu tạo truy vấn trên Client, ta nên nạp nó vào Server.

15.6 Sử dụng dữ liệu với đối tượng rdorerultset

Ta sử dụng đối tượng rdoresultset để thao tác với dữ liệu trả về do tương tác với server.

Mỗi đối tượng rdoresultset thuộc về tập hợp rdoresultsets. Đối tượng rdoresultset chứa tập hợp rdoColumn.

Đối tượng rdoresultset gần như đồng nhất với đối tượng Recordset cuả DAO. Có thể tạo đối tượng rdoresultset theo một số cách điển hình, nó được tạo như một kết quả của câu truy vấn.

Hình

15.7 Thi hành truy vấn với đối tượng rdoQuery

Trong RDO 2.0 đối tượng RDOQuery là một thay thế cho đối tượng rdoprepareStatment trong RDO 1.0. Ta có thể vẫn dùng các đối tượng rdoPrepareStatment trong RDO 2.0 vì nó vẫn tương thích với các phiên bản trước. Tuy nhiên, với chương trình viết mới, ta nên dùng rdoQuery.

Từng đối tượng rdoQuery thuộc về tập hợp rodQueries. Đối tượng rodQuery chứa tập hợp các đối tượng rdoColumn và rdoParameter.

16 Truy cập cơ sở dữ liệu với lớp

Làm việc với lớp và đối tượng.
Sử dụng lớp và đối tượng với truy cập cơ sở dữ liệu.
Tạo lớp cần sử dụng dữ liệu.
Tạo lớp xuất dữ liệu.
Triển khai lớp như là các Activex Server.

Các ứng dụng truy cập dữ liệu thường phức tạp hơn nhiều so với các ứng dụng thông thường. Lý do Visual Basic đưa ra các công nghệ DAO, ADO, ODBC là nhằm giúp giải quyết tính phức tạp này.

Nhưng các công nghệ này chỉ mới giải quyết những phức tạp nảy sinh trong quá trình phát triển phần mềm. Mô hình đối tượng cơ sở dữ liệu như ADO chẳng hạn, giúp ta trừu tượng hoá cơ sở dữ liệu và do đó tạo sự dễ dàng khi cập nhật một bản ghi hoặc sửa đổi định nghĩa bảng trong chương trình. Tuy nhiên, nó không giúp ta tính thuế trên doanh thu bán hàng hay từ chối một mẩu dữ liệu khách hàng bởi vì nó không có ID hợp lệ.

Visual Basic cho phép ứng dụng củng cố các quy luật kinh doanh vào các lớp. Lớp là một kiểu mô – dun chương trình cho phép ta tạo đối tượng. Các đối tượng ta tạo với mô-dun lớp tương tự các đối tượng truy cập dữ liệu ta sử dụng để giao tiếp với cơ sở dữ liệu, ngoại trừ chúng được dùng cho mục đích bất kỳ. Trong ngữ cảnh truy cậ cơ sở dữ liệu, ta chú yếu dùng lớp cùng với các đối tượng cơ sở dữ liệu để tạo nên một ứng dụng truy cập cơ sở dữ liệu.

Giả sử ta tạo một ứng dụng xử lý hoá đơn và khách hàng. Trong một ứng dụng không theo hướng đối tượng, ta phải viết các hàm hay thủ tục ghi thông tin khách hàng và hoá đơn, trả về các thông tin từ cơ sở dữ liệu, in thông tin, v.v.. Nếu viết bằng Visual Basic, ta còn phải xử lý rải rác trên hàng chục thủ tục sự kiện.

Là người lập trình theo hướng đối tượng, ta sẽ bắt đầu bằng cách phân tích và thiết kế các thành phần, hay là đối tượng khái quát hoá vấn đề xử lý khách hàng và hoá đơn. Anh ta hay cô ta sẽ xác định một đối tượng khách hàng có những thông tin gì và đối tượng khách hàng đó có thể thực hiện những hành động nào trên dữ liệu; tương tự với đối tượng hoá đơn. Sau khi đối tượng được phân tích và thể hiện thành lớp, nó sẽ được sử dụng trong ứng dụng. Ta có thể dùng lại đối tượng khách hàng và đối tượng hoá đơn trong ứng dụng bất kỳ sau này. Bởi vì mã nguồn của đối tượng tồn tại trong nơi chứa rất dễ truy cập đến là mô-dun lơp thay vì trong rất nhiều thủ tục sự kiện rải khắp ứng dụng, ta cũng có thể dễ dàng gỡ rối và bảo trì các đối tượng này.

Ngoài ra, lớp và đối tượng còn tận dụng các tính năng ngôn ngữ mạnh của Visual Basic. Ví dụ, có một quan hệ giữa khách hàng và hoá đơn – ta có thể nói một khách hàng thuộc về một hoá đơn, hay chính xác hơn, tập hợp các hoá đơn thuộc về một khách hàng. Visual Basic hỗ trợ tập hợp các đối tượng.

Một hệ thống hướng đối tượng có 3 đặc điểm :

Trừu tượng hoá : rút gọn vấn đề đến mức dễ hiểu nhất.

Đa hình : cho phép đối tượng thi hành cùng phương thức và chứa cùng dữ liệu với các đối tượng khác. Điều này tạo sự dễ dàng khi lập trình, giúp ta

không cần xử lý lại với từng đối tượng, bởi vì các đối tượng vẫn có cùng phương thức và thuộc tính.

Tóm lược : đây là cơ chế qua đó *logic* chương trình và dữ liệu được nhóm lại với nhau.

Kế thừa : đối tượng hiện hữu có thể sinh ra một đối tượng mới. Tính năng này trong Visual Basic còn bị hạn chế, nó được cung cấp dưới dạng các giao diện và thông qua đại diện.

16.1 Làm việc với lớp và đối tượng

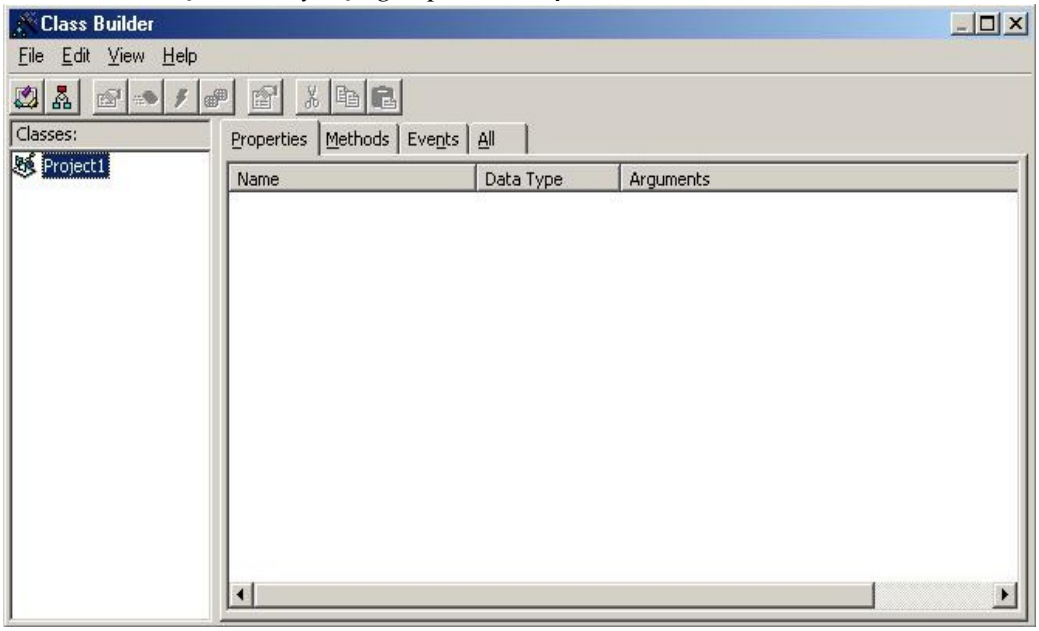
Bạn nên xem lại chương 2 : Tìm hiểu Visual Basic và chương 10 : Lập trình hướng đối tượng.

16.1.1 Tạo cây phân nhánh lớp với tiện ích xây dựng lớp

Khi ta thiết kế lớp chứa lớp, nhất là lớp có tập hợp – chương trình trở nên khó viết vì rất khó quản lý chúng. ***Tiện ích xây dựng lớp*** *(Class Builder utility)* giúp ta thiết lập và quản lý các quan hệ giữa tập hợp các lớp bao bọc chúng dễ dàng hơn.

Ví dụ mẫu - Tạo một lớp tập hợp cho phép làm việc với tập hợp của đối tượng Order

1. Khởi động Visual Basic và nạp đề án chứa lớp *COrder.*
2. Từ menu *Add-Ins,* chọn *Add-Ins Manager.*
3. Nhấn *Class Builder Utility* trong hộp thoại *Add-In Manager*. Trong bảng *Load Behavior*, nhấn **Loaded/Unloaded**.
4. Nhấn OK. *Tiện ích xây dựng lớp* được nạp và xuất hiện trên menu *Add-Ins*.
5. Để phóng *tiện ích xây dựng lớp,* chọn menu *Add-Ins,****Class Builder Utility***
6. **Tiện ích xây dựng lớp** cảnh báo rằng đề án hiện hành chứa lớp hiện hành chưa được xây dựng với tiện ích này. Nhấn OK.
7. Cửa sổ *Tiện ích xây dựng lớp* xuất hiện.

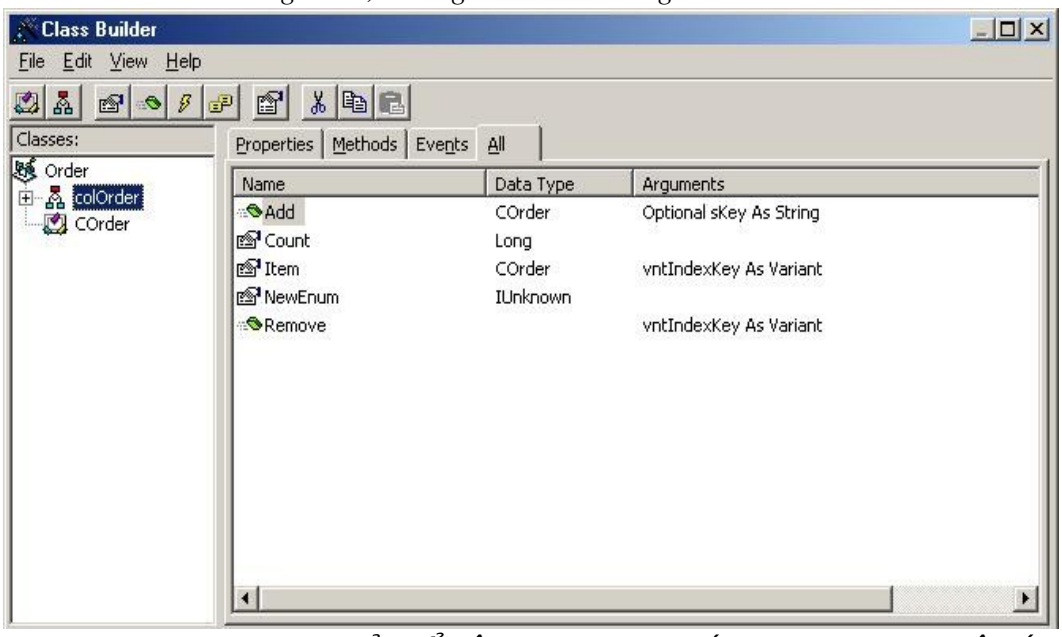


Hình 16.1 : Dùng nút **New Collection** (nút thứ hai từ trái sang) của tiện ích xây dựng lớp để tạo một lớp tập hợp.

8. Bây giờ, ta dùng *Tiện ích xây dựng lớp* để tạo lớp tập hợp. Nhấn nút ***Add New Collection*** trên thanh công cụ:

   Hình 16.2 : Nút Add New Collection 
9. Hộp thoại ***Collection Builder*** xuất hiện. Trong ô Name, nhập tên của lớp tập hợp – *colOrder*.
10. Chọn ***COrder*** trong bảng có chữ "*Collection Of* ".Điều này cho biết lớp tập hợp này là một tập hợp của đối tượng COrder. Nó tạo sự an toàn về kiểu cho lớp – khác với các tập hợp thông thường, chỉ đối tượng *COrder* mới được thêm vào tập hợp này.
11. Nhấn OK. Lớp tập hợp được thêm vào *tiện ích xây dựng lớp*.
12. Ta có thấy *tiện ích xây dựng lớp* phát sinh các thuộc tính và phương thức để hỗ trợ 4 phương thức của tập hợp (*Add, Count, Item và Remove)* cũng như là *NewEnum.*

    Lưu ý :*Tiện ích xây dựng lớp tạo thuộc tính* ***NewEnum*** *cho phép lớp tập hợp hỗ trợ phép lặp xuyên qua tập hợp dùng cấu trúc điều khiển* ***For Each..Next.*** *Không nhất thiết phải dùng thuộc tính này trong chương trình, nhưng ta biết sẽ dùng nó với* ***For Each..Next.***



Hình 16.3 *Cửa sổ tiện ích xây dựng lớp sau khi nó tạo một lớp tập hợp.*

13. Bây giờ ta định nghĩa lớp tập hợp, ta có thể yêu cầu *tiện ích xây dựng lớp* phát sinh chương trình bằng chọn menu *File Update Project* hoặc dùng phím tắt *Ctrl+S.*
14. Tiện ích tự động xây dựng lớp tập hợp. Đóng tiện ích bằng cách chọn *File,* ***Exit.*** Ta thấy lớp *Orders* được tạo ra do tiện ích xây dựng lớp.

Ta cũng thấy rằng tiện ích đã định nghĩa lại cú pháp của phương thức ***Add*** của tập hợp. Thay vì lấy đối tượng làm tham biến như phương thức ***Add*** của tập hợp quy ước, phương thức *Add* mới lấy biến làm tham biến. Những biến này gắn với các thuộc tính của đối tượng ***COrder.*** Điều này cho phép ta tạo đối tượng, thêm nó vào tập hợp, gán giá trị cho nó - những cải tiến đáng kể so với cách thức thực hiện thông thường.

```
Dim MyOrder as COrder
Dim MyOrders as colOrder
Set MyOrders= New colOrder
set MyOrder= MyOrders.Add(2.99, "Cheese", 201, #6/5/99#)
```

Mặc dù vậy, đây chưa phải là cú pháp tốt nhất để thêm mộ phần tử vào tập hợp. Có thể dùng cách gọn gàng hơn:

```
Dim MyOrder as COrder
Dim MyOrders as colOrder
Set MyOrders= New colOrder
set MyOrder= MyOrders.Add()
MyOrder.Price = 2.99
MyOrder.ItemOrdered = "Cheese"
MyOrder.CustomerID = 201
MyOrder.OrderDate = #6/5/99#
```

Cả hai cách đều đưa về cùng kết quả: một đối tượng được thêm vào tập hợp và thuộc tính được gán giá trị. Cách thứ hai, dùng thuộc tính thay vì tham biến trông rõ ràng, vì vậy ta sử dụng nó rộng rãi hơn.

Một trường hợp dùng tham biến thuận tiện hơn thuộc tính là khi lớp tập hợp tồn tại trong một thành phần triển khai từ xa qua mạng. Theo kịch bản đó, mỗi lần gọi đến thuộc tính là sinh ra một truy cập 2 lượt đi về qua mạng. Vì vậy, để ứng dụng hoạt động tốt hơn, ta nên dùng tham biến và chỉ gọi một lần.

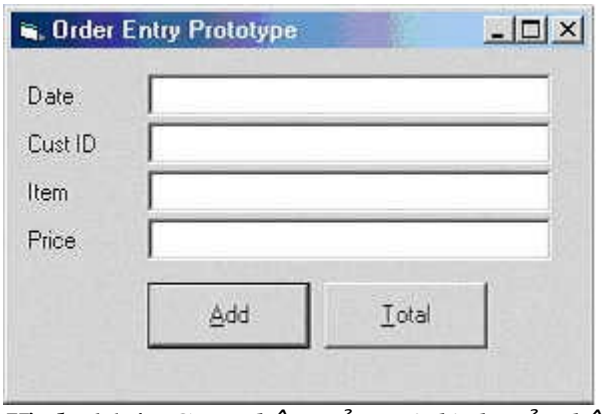
Ví dụ mẫu - Sửa đổi phương thức Add của lớp tập hợp để bỏ qua tham biến

```
Public Function Add(Optional sKey as String) as COrder
    Dim objNewMember as COrder
    Set objNewMember = New COrder
    If Len(sKey) =0 Then
        mCol.Add objNewMember
    Else
        mCol.Add objNewMember, sKey
    End If
    Set Add = objNewMember
    Set objNewMember = Nothing
End Function
```

16.1.1.1 Dùng lớp tập hợp để thao tác với các phần tử dữ liệu

Để xây dựng ứng dụng dùng lớp ***COrder*** và ***colOrder***, ta theo các bước sau :

1. Tạo đề án ***Standard Exe*** mới. Thêm lớp *COrder* và *colOrder.*
2. Trong đề án, tạo một biểu mẫu cho phép người sử dụng đưa vào *Date, CustomerID, Item, Price.* Thêm 2 nút lệnh vào giao diện, nút thứ nhất nhập dữ liệu, trong khi nút thứ hai tính toán tổng số hoá đơn được nhập.



Hình 16.4 *Giao diện của mô hình của hệ thống nhập hoá đơn hướng đối tượng có 2 nút lệnh.*

3. Trong biểu mẫu, khai báo 2 biến mức mô-dun, một cho đối tượng ***COrder*** và một cho tập hợp ***colOrder.***

```
Option Explicit
Private ThisOrder as COrder
Private AllOrders as colOrder
```

4. Tạo *instance* cho tập hợp ***colOrder*** trong sự kiện ***Load*** của biểu mẫu.

```
Private Sub Form_Load ()
    Set AllOrders = New colOrder
End Sub
```

5. Viết chương trình cho sự kiện ***Click*** của nút ***Add*** để thêm hoá đơn vào tập hợp, rồi khởi động lại giao diện người sử dụng của ứng dụng.

```
Private Sub cmdAdd_Click ()
    ‘ Creates a new order and adds
    ‘ it to the collection
    Set ThisOrder = AllOrders.Add

    ThisOrder.OrderDate = txtOrderDate.Text
    ThisOrder.CustomerID = txtCustomerID.Text
    ThisOrder.ItemOrdered = txtItemOrdered.Text
    ThisOrder.Price = txtPrice.Text
    ‘ Reset the user Interface
    txtOrderDate.Text = “ ”
    txtCustomerID.Text = “ ”
    txtItemOrdered.Text = “ ”
    txtPrice.Text = “ ”
    txtOrderDate.SetFocus

End Sub
```

6. Sau cùng, trong sự kiện ***Click*** của nút *Total,* viết chương trình để trả về tổng số hoá đơn đã nhập dùng vòng lặp ***For Each.. Next.***

```
Private Sub cmdTotal_Click()
```

```
Dim curTotal As Currency

   For Each ThisOrder In AllOrders
      curTotal = curTotal + ThisOrder.Price
   Next

   MsgBox "The total is " & curTotal, vbInformation

End Sub
```

16.1.1.2 Tham chiếu đến phần tử trong tập hợp

Dùng vòng lặp ***For Each.. Next*** để duyệt qua tập hợp không phải là cách duy nhất để làm việc với tập hợp. Sau khi thiết lập và đưa vào các đối tượng, ta có thể lấy ra phần tử từ tập hợp. Nếu thêm đối tượng vào tập hợp với một khoá hay giá trị chuỗi duy nhất, ta có thể lấy về đối tượng mà không không cần xác định vị trí của nó trong tập hợp - một ưu điểm so với mảng.

Ví dụ, Để lấy về hoá đơn thứ 4 trong tập hợp :

```
Set MyOrder = AllOrders(4)
```

Để tham chiếu đến thuộc tính của đối tượng ***Order*** thứ 4 trong tập hợp, ta truyền vào chỉ mục của tập hợp :

```
AllOrders(4).OrderDate = #6/5/98#
```

Dòng lệnh này hoạt động nhờ thủ tục mặc định của tập hợp trả về một tham chiếu đến phần tử được đánh số. Thủ tục mặc định này là phương thức ***Item***. Vì là mặc định, nên ta không cần gọi nó một cách tường minh. Nếu không, có thể dùng:

```
AllOrders.Item(4).OrderDate= #6/5/98#
```

> ***THẬN TRỌNG*** *Phần tử đầu tiên trong tập hợp được đánh số 1, không có phần tử thứ 0. Nó cũng khách với tập hợp của các đối tượng* ***Form*** *hay* ***Control*** *trong Visual Basic; các tập hợp này đánh số từ 0. Như vậy, tập hợp do ta tự tạo được đánh số từ 1; trong khi tập hợp do Visual Basic tạo đánh số từ 0. Đây cũng là trường hợp của tập hợp các đối tượng truy cập dữ liệu, như là tập hộp* ***TableDefs*** *của đối tượng* ***Database.***

Để trả về một phần tử từ tập hợp sử dụng một khoá trước đó, ta phải thêm nó vào tập với một khoá. Để thực hiện điều này, truyền một chuỗi duy nhất cho phương thức ***Add*** :

```
AllOrders.Add “ORD193”).prive
```

Ta sẽ gặp thông báo lỗi nếu như chuỗi này không duy nhất. Để tham chiếu đến một thuộc tính của đối tượng này, ta dùng dòng lệnh :

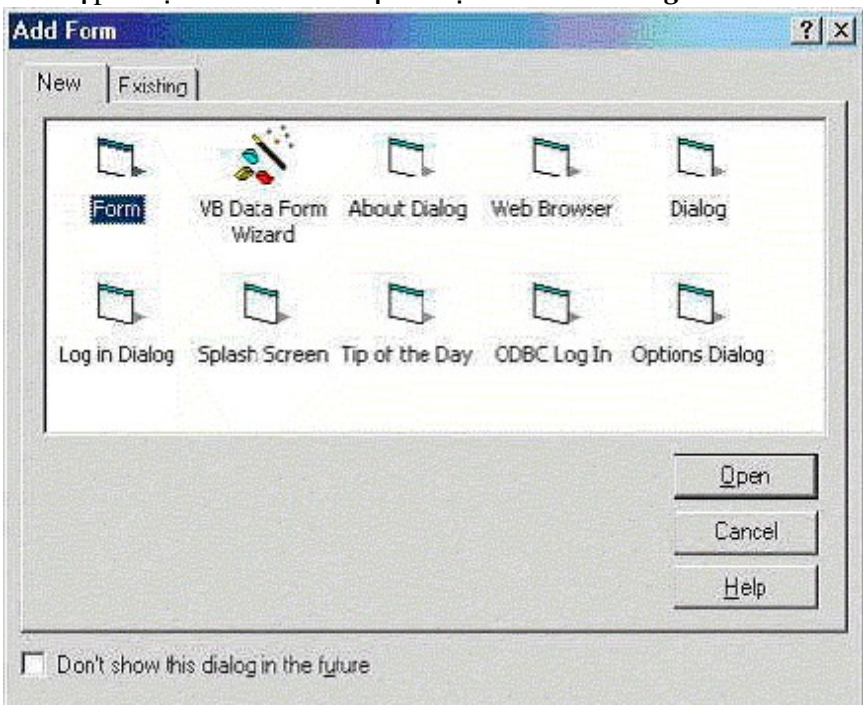
```
MsgBox AllOrders (“ORD193”).prive
```

Lưu ý rằng ta không dùng giá trị số cho khoá của một đối tượng, ngay cả khi ta truyền nó cho phương thức ***Add*** dưới dạng chuỗi. Bởi vì phương thức ***Item*** có thể trả về một đối tượng từ tập hợp bằng khoá hoặc bằng vị trí xếp thứ tự. Nếu phương thức ***Item*** nhận số n, nó trả về phần tử thứ n trong tập hợp. Nếu nó nhận một chuỗi, nó trả về đối tượng với khoá là chuỗi đó. Mặc dù vậy, Nếu chuỗi có thể chuyển thành số, ta sẽ có nguy cơ gặp lỗi.

16.1.2 Sử dụng biểu mẫu như lớp

-Ví dụ mẫu - Tạo một Instance của biểu mẫu Login

1. Trong đề án của Visual Basic, nhấn nút phải chuột trên cửa sổ *Project Explorer.*
2. Từ menu bật ra, chọn *Add.* Chọn ***Form***.
3. Hộp thoại *Form* xuất hiện. chọn biểu mẫu ***Login***.



Hình 16.5 :*Hộp thoại Add Form của Visual Basic cho phép chọn kiểu biểu mẫu định nghĩa sẵn.*

4. Trong hộp thoại *Form,* nhấn ***Open.*** Một biểu mẫu mới được tạo.
5. Trong cửa sổ ***Code,*** đưa vào các thủ tục thuộc tính :

```
Public Property Get UserName() As String
   UserName = txtUserName.Text
End Property

Public Property Let UserName(ByVal strNew As String)
   txtUserName.Text = strNew
End Property

Public Property Get Password() As String
   Password = txtPassword.Text
End Property

Public Property Let Password(ByVal strNew As String)
   txtPassword.Text = strNew
End Property
```

```
Public Sub Login()
   MsgBox "Logging in user " & UserName & "."
End Sub
```

6. Biểu mẫu *Login* chuẩn của Visual Basic chưa chương trình trong sự kiện nhấn nút OK và Cancel để người sử dụng Login vào hệ thống. Xoá đoạn chương trình đó đi và thay bằng đoạn chương trình sau:

```
Private Sub cmdCancel_Click()
   Me.Hide
End Sub
Private Sub cmdOK_Click()
   UserName = txtUserName.Text
   Password = txtPassword.Text
   Login
   Me.Visible = False
End Sub
```

Điều này sẽ tận dụng thế mạnh của các thuộc tính và phương thức hiệu chỉnh của chúng ta. Ngoài ra, Form này sẽ được ứng dụng rộng rãi trong ứng dụng "*Clien/Server*" sẽ được trình bày trong phần sau.

16.2 Tạo Intance bội cho biểu mẫu

Ta có thể tạo *instance bội (multiple instance)* cho biểu mẫu tương tự như việc tạo *instance bội* cho các đối tượng từ lớp. Mỗi *instance* sau đó của biểu mẫu có định danh riêng trong ứng dụng, với một bản sao của toàn bộ thuộc tính, phương thức, và các điều khiển giao diện người sử dụng chứa trong thiết kế gốc của biểu mẫu.

Tạo ra nhiều *instance* cho biểu mẫu *Login* của người sử dụng là vô nghĩa. Tuy nhiên, đối với các biểu mẫu khác, ta tạo ra *instance bội* là điều thường gặp như ví dụ sau cho thấy :

```
Dim f as frmMain
Set f = New frmMain
f.Show
```

16.2.1 Sử dụng lớp và đối tượng trong truy cập cơ sở dữ liệu

Có một số phương tiện giúp cho việc áp dụng kỹ thuật hướng đối tượng trong truy cập dữ liệu trong Visual Basic :

Gắn một bản ghi duy nhất với một đối tượng : Đây là kỹ thuật đơn giản nhất, không cần lập trình nhiều. Mỗi trường trong bản ghi trở thành một thuộc tính của đối tượng; lấy dữ liệu về từ cơ sở dữ liệu hay lưu dữ liệu vào cơ sở dữ liệu đều được xử lý qua đối tượng.

Uỷ nhiệm xử lý dữ liệu cho một đối tượng ***Recordset*** *chứa trong một đối tượng :* Đây là kỹ thuật tốt nhất khi ta cần xử lý một số không giới hạn các bản ghi. Kỹ thuật này cũng dễ lập trình, bởi vì có nhiều chức năng quản lý

được cung cấp sẵn trong các mô hình đối tượng được sử dụng (**DAO** hay **RDO**). Kỹ thuật đặc biệt hữu dụng khi ta dùng **ADO** bởi vì **ADO** cung cấp khả năng ngắt kết nối với nguồn dữ liệu, cho phép ứng dụng *Client* thao tác với dữ liệu không cần kết nối với Server. Bởi vì nhiều người sử dụng kết nối đồng thời là một điểm yếu của các máy tính *Client/Server,* ngắt kết nối có nghĩa là giải pháp sẽ linh hoạt hơn.

Gắn nhóm các bản ghi vào một tập hợp : Lập trình phức tạp hơn trường hợp gắn một bản ghi với một đối tượng, nhưng nó hữu dụng hơn trong trường hợp ta phải xử lý với nhóm các bản ghi có liên quan tại một thời điểm. Ta có thể gặp phải khó khăn liên quan đến khả năng hoạt động của ứng dụng nếu ta không cẩn thận giới hạn số bản ghi xử lý tại một thời điểm.

Chia truy cập dữ liệu giữa Client và Server : Kỹ thuật này thích hợp nhất khi ta cần đạt được khả năng linh hoạt và hiệu quả cao nhất trong ứng dụng. Nó liên quan đến việc chia Logic của tầng trung gian của ứng dụng thành 2 phần : thành phần phía server thi hành truy cập và trả về dữ liệu đến Clien và thành phần phía Client nhận kết quả này và gắn chúng vào các thuộc tính của đối tượng.

Các ví dụ còn lại trong chương sẽ đưa ra ý tưởng kỹ thuật nào sẽ phù hợp trong từng tình huống cụ thể.

16.2.1.1 Sử dụng các lớp xử lý bản ghi duy nhất

Lớp xử lý bản ghi trả về một bản ghi duy nhất từ cơ sở dữ liệu và đưa nó vào ứng dụng dưới dạng một đối tượng. Các trường trong bản ghi được trình bày như những thuộc tính của đối tượng. Bất kỳ hành động nào thi hành trên dữ liệu (như là lưu dữ liệu về cơ sở dữ liệu, in ấn, hay tiến hành tính toán) đều được trình bày như những phương thức của đối tượng xử lý bản ghi.

Để cho phép lớp điền giá trị cho thuộc tính, ta cung cấp cho nó một phương thức (đặt tên là ***GetData***) thi hành truy vấn một bản ghi duy nhất trên cơ sở dữ liệu, nó dùng các trường trong bản ghi được trả về để điền giá trị vào thuộc tính của đối tượng.

Khi giao cho đối tượng vai trò lấy bản ghi về từ cơ sở dữ liệu, ta cần cung cấp cho đối tượng ID của bản ghi, thường là khoá chính này; phương thức *GetData* của đối tượng sẽ dùng khoá chính này để lấy về bản ghi cần thiết.

Đối tượng cũng có thể có thêm phương thức ***Save*** cho phép người sử dụng của thành phần lưu bản ghi vào cơ sở dữ liệu.

16.2.1.2 Sử dụng lớp xử lý mảng(ARRAY-HANDLING CLASS)

Ta có thể tạo lớp truy vấn cơ sở dữ liệu, sau đó truyền dữ liệu về ứng dụng Client dưới dạng mảng. Ưu điểm của giải pháp này là nếu lớp xử lý mảng được biên dịch thành ActiveX Server DLL hay EXE, nó có thể tạo ứng dụng Client mà không đòi hỏi tham chiếu đến bộ máy của cơ sở dữ liệu. Tất cả những gì Client cần là một tham chiếu đến thành phần ActiveX server. Điều này làm cho ứng dụng Client tiêu tốn ít tài nguyên hơn; ứng dụng dễ phân phát và định cấu hình hơn.

Tạo một thành phần phía server trả về một mảng nghĩa là ứng dụng phía Client không cần duy trì kết nối liên tục đến server, giúp tăng cường tính linh hoạt của ứng dụng Client/Server.

16.2.1.3 Sử dụng phương thức xưởng sản suất(FACTORY METHOD)

Xưởng sản xuất đối tượng là một phương pháp tạo các đối tượng khác. Ta dùng *Xưởng sản xuất đối tượng* trong trường hợp một đối tượng giữ vai trò tạo ra các đối tượng khác. Để tạo ra các đối tượng khác, ta dùng phương thức; các phương thức này gọi là *phương thức xưởng sản xuất.*

Ví dụ, đối tượng ***Database*** của **DAO,** có một số phương thức xưởng sản xuất cung cấp đối tượng :

Phương thức	Mô tả
CreateQueryDef	Tạo một đối tượng ***QueryDef.***
CreateRelation	Tạo một đối tượng ***Relation***
CreateTableDef	Tạo một đối tượng ***TableDef***
OpenRecordset	Tạo một đối tượng ***Recordset***
CreateProperty	Tạo một đối tượng ***Property***

Ngoài việc sử dụng các *phương thức xưởng sản xuất* do Visual Basic cung cấp, ta có thể tạo ra các phương thức này trong các lớp tự tạo.

Một lý do phổ biến để dùng các phương thức xưởng sản xuất là yêu cầu thực hiện theo cách tạo đối tượng trong Visual Basic. Các ngôn ngữ lập trình khác sử dụng đối tượng (như Visual C++) có *contructor* đảm nhiệm việc tạo Instance của đối tượng từ lớp; những *constructor* còn có thể thi hành các tác vụ khác, như cung cấp các dữ liệu khởi tạo cho lớp. Lớp có nhiều *constructor* để người lập trình có thể chọn lựa và *constructor* có thể lấy và trả về tham số.

Tuy nhiên, không có tương quan trực tiếp với *constructor* trong Visual Basic. Thay vào đó, các lớp Visual Basic có sự kiện ***Initialize.*** Ta có thể lập trình trong sự kiện *Initialize* như trong *constructor* của Visual C++, nhưng không may là ta không thể truyền tham số cho sự kiện *Initialize* của Visual Basic, và ta chỉ có một sự kiện *Initialize* cho mỗi lớp. Nó làm tính tiện dụng của sự kiện *Initialize* bị hạn chế đối với mọi phần tử, ngoại trừ các giá trị cơ bản nhất là các giá trị mặc định. Các phương thức xưởng sản xuất cung cấp giải pháp khắc phục điều này.

```
Option Explicit

' Requires CCustomer class

Public Function CreateCustomer(Optional lngID As Variant) As CCustomer

Dim Cust As CCustomer

   If IsMissing(lngID) Then
      ' Create new customer
```

```
        Set CreateCustomer = New CCustomer
    Else
        ' Retrieve customer from DB
        Set Cust = New CCustomer
        Cust.GetData (lngID)
        Set CreateCustomer = Cust
    End If

End Function
```

Xưởng sản xuất đối tượng tạo một đối tượng khách hàng rất dễ dàng với đầy đủ tính năng. Đoạn chương trình sau tạo đối tượng ***CCustomer:***

```
Dim MyCust As CCustomer
Dim cf As CCustFactory
Set cf = New CCustFactory

' Retrieve data
Set MyCust = cf.CreateCustomer(txtID.Text)
```

Một lý do khách để dùng các *phương thức xưởng sản xuất* trong lập trình cơ sở dữ liệu Visual Basic là để tiết kiệm các tài nguyên kết nối, bởi kết nối đến cơ sở dữ liệu đòi hỏi tiêu tốn nhiều thời gian và bộ nhớ trên Client và Server.

Nếu dùng *lớp xưởng sản xuất* sẽ dễ dàng hạn chế số kết nối thực hiện với server. Ta chỉ kết nối một lần, trong lớp xưởng sản xuất thay vì trong mỗi mô-dun lớp.

16.3 Tạo các lớp cần sử dụng dữ liệu

Lớp có khả năng ràng buộc trực tiếp với cơ sở dữ liệu. Đây là tính năng mới trong Visual Basic 6.0. Ta không cần giới hạn các điều khiển dữ liệu thành nguồn dữ liệu cho ứng dụng cơ sở dữ liệu.

Ngoài việc cung cấp nguồn dữ liệu mới, Visual Basic 6 còn loại bỏ những hạn chế về cách thức kết nối của ứng dụng với nguồn dữ liệu. Ta không cần thiết lập ràng buộc dữ liệu vào lúc thiết kế, như với điều khiển ***DAO Data.*** Thay vào đó, ta có thể gán một nguồn dữ liệu (như là điều khiển *ADO Data,*thiết kế ***DataEnvironment,***hay lớp cung cấp dữ liệu) cho phần tiêu thụ dữ liệu (như là một điều khiển giao diện người sử dụng ràng buộc) lúc thi hành. Điều này cho phép đóng gói mô-dun lớp truy cập dữ liệu.

Để tạo một lớp làm nguồn dữ liệu truy cập cơ sở dữ liệu ***Novelty***, ta theo các bước sau:

1. Tạo một đề án *Standard EXE.*
2. Dùng menu *Project* để thêm một mô-dun lớp vào đề án. Đặt tên mô-dun lớp là ***CCustData.***
3. Dùng cửa sổ *Properties* để đổi thuộc ***DataSourceBehavior*** thành *1-vbDataSource.*
4. Dùng menu *Project,* ***References*** để thiết lập một tham chiếu đến *Microsoft ActiveX Data Objects.*

5. Trong phần khai báo của lớp, tạo một đối tượng ***Recordset*** *(private).* Đối tượng này sẽ xử lý truy cập cơ sở dữ liệu.

```
Private mrsCustomer As ADODB.Recordset
```

6. Trong sự kiện ***Initialize*** của lớp, viết chương trình để tạo đối tượng *Recordset*:

```
Set mrsCustomer = New ADODB.Recordset
mrsCustomer.Source = "select * from tblCustomer"
mrsCustomer.CursorType = adOpenKeyset
mrsCustomer.LockType = adLockOptimistic
mrsCustomer.ActiveConnection = "DSN=JetNovelty;"
mrsCustomer.Open
```

7. Trong sự kiện *Initialize* của lớp, viết chương trình đăng ký nguồn dữ liệu với tập hợp ***DataMembers*** do thư viện *Microsoft Data Binding* cung cấp. Điều này cho phép phần tiêu thụ dữ liệu (như là các điều khiển ràng buộc) dùng lớp này như là một nguồn dữ liệu :
8. Viết chương trình cho sự kiện ***GetDataMember*** của lớp để trả về một đối tượng ***Recordset*** dựa trên tham số *DataMember* của sự kiện (Sự kiện *GetDataMember* sẽ hiện diện sau khi ta quy định giá trị cho thuộc tính ***DataSourceBehavior***). Bởi vì sự kiện *GetDataMember* được tham số hoá, ta có thể kết nối ứng dụng với vô số các *recordset*;tuy nhiên, lớp này chỉ trả về một danh sách khách hàng.

   Lưu ý : Tham số *Data* của sự kiện *GetDataMember* là một đối tượng, không phải một *recordset*. Do đó, trong sự kiện có thể truy cập đối tượng khác thay vì đối tượng ***Recordset*** của **ADO.**
9. Kế đến, viết phương thức (*public)* trong lớp *CCustData* để điều khiển *recordset*. Trong ví dụ này, ta tạo 4 phương thức di chuyển được cung cấp bởi một điều khiển dữ liệu:

```
Public Sub MoveFirst()
  mrsCustomer.MoveFirst
End Sub

Public Sub MoveLast()
  mrsCustomer.MoveLast
End Sub

Public Sub MoveNext()
  mrsCustomer.MoveNext
  If mrsCustomer.EOF Then
    mrsCustomer.MoveLast
  End If
End Sub

Public Sub MovePrevious()
  mrsCustomer.MovePrevious
  If mrsCustomer.BOF Then
    mrsCustomer.MoveFirst
```

```
    End If
End Sub
```

10. Dùng menu *Project Components*, thêm một điều khiển *Microsoft DataGrid Control 6.0 (OLE DB)* vào đề án.Nó cho phép ta hiển thị dữ liệu trong một lưới (grid). Tạo một instance của điều khiển lưới trên biểu mẫu.
11. Trên biểu mẫu, tạo 4 nút lệnh để duyệt qua dữ liệu. Đặt tên chúng là ***cmdFirst, cmdPrevious, cmdNext, cmdLast.***
12. Trong sự kiện ***Load*** của biểu mẫu, khai báo một instance của lớp nguồn dữ liệu. Viết chương trình cho sự kiện ***Click*** để gọi các phương thức di chuyển :

```
Private Sub Form_Load()
    ' Create the data source object
    Set mCustData = New CCustData

    ' Bind the object to grid
    Set DataGrid1.DataSource = mCustData
    DataGrid1.DataMember = "Customers"

End Sub

Private Sub cmdFirst_Click()
    mCustData.MoveFirst
End Sub

Private Sub cmdLast_Click()
    mCustData.MoveLast
End Sub

Private Sub cmdNext_Click()
    mCustData.MoveNext
End Sub

Private Sub cmdPrevious_Click()
    mCustData.MovePrevious
End Sub
```

13. Thi hành ứng dụng.

16.3.1 Tạo lớp xuất dữ liệu

Có nhiều khả năng ta viết một thủ tục lấy một bảng cơ sở dữ liệu và chuyển đổi nó thành tập tin văn bản có phân cách, sau đó ta sẽ sử dụng lại thủ tục này nhiều lần.

Phần này xây dựng trên hàm xuất HTML mô tả trong chương 14 "Thiết lập báo cáo và xuất thông tin " để cung cấp việc xuất dữ liệu ta đã sử dụng lớp **CCustomer**

Để cung cấp cho lớp *CCustomer* khả năng xuất, ta thêm một thuộc tính và một phương thức cho lớp. Thuộc tính mới, ***HTMLText,*** lấy thuộc tính của đối tượng

CCustomer và định dạng nó thành một trang HTML. Thủ tục thuộc tính này gọi một hàm *(private)*, ***HTMLRow*** để định dạng từng dòng của bảng.

```
Public Property Get HTMLText() As String
Dim str As String
   str = "<html>" & vbCrLf
   str = str & "<head>" & vbCrLf
   str = str & "<title>" & "Customer: " & _
            FirstName & " " & _
            LastName & _
            "</title>" & vbCrLf
   str = str & "<body bgcolor=#ffffff>" & vbCrLf
   str = str & "<font face=Arial,Helvetica>" & vbCrLf
   str = str & "<table border>" & vbCrLf
   str = str & HTMLTableRow("First name:", FirstName)
   str = str & HTMLTableRow("Last name:", LastName)
   str = str & HTMLTableRow("Address:", Address)
   str = str & HTMLTableRow("City:", City)
   str = str & HTMLTableRow("State:", State)
   str = str & "</table>"
   str = str & "</font>" & vbCrLf
   str = str & "</body>" & vbCrLf
   str = str & "</html>" & vbCrLf
   HTMLText = str
End Property
```

Đoạn chương trình trên chỉ định dạng HTML mà thôi. Quá trình xuất gồm 2 phần : thứ nhất, định dạng dữ liệu trong HTML, thứ hai, ghi nó ra tập tin. Như vậy, ta phải viết thủ tục riêng để thi hành từng tác vụ.

Phương thức ***SaveHTML*** của đối tượng *Customer* lưu dữ liệu HTML được định dạng thành một tập tin.

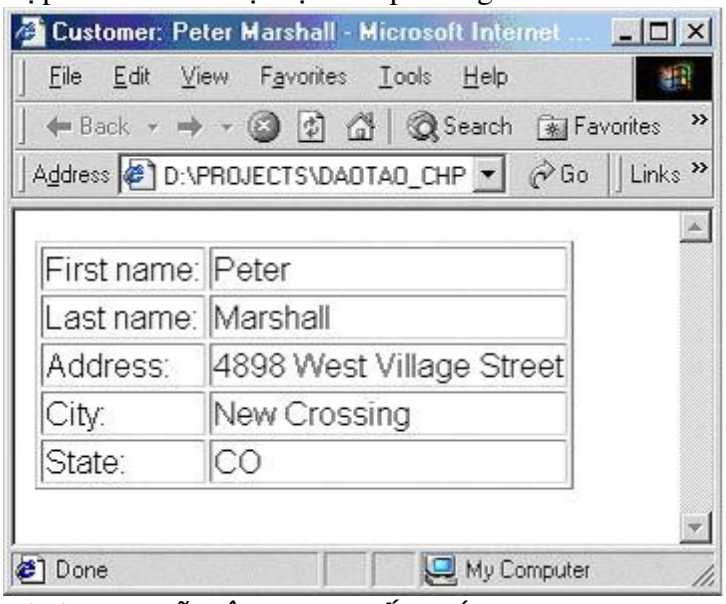
```
Public Sub SaveHTML(strFileName As String)
Dim fs As Scripting.FileSystemObject
Dim txt As Scripting.TextStream

   Set fs = New Scripting.FileSystemObject
   Set txt = fs.OpenTextFile(strFileName, ForWriting, True)

   txt.Write HTMLText
   txt.Close

   Set txt = Nothing
   Set fs = Nothing

End Sub
```

Đưa các thủ tục này thành một phần của lớp *CCustomer,* tạo một ứng dụng cho phép người sử dụng cung cấp ID của khách hàng và tên tập tin xuất. Khi người sử dụng nhấn một nút lệnh, khách hàng được chọn sẽ được xuất vào tập tin HTML chỉ ra bởi người sử dụng.

```
Option Explicit
```

```
Private Cust As CCustomer
Private Sub Form_Load()
   Set Cust = New CCustomer
End Sub
Private Sub cmdExport_Click()
   Cust.GetData txtID.Text
   Cust.SaveHTML App.Path & "\" & txtFilename.Text
   MsgBox "File saved.", vbInformation
End Sub
```

Tập tin HTML được tạo bởi phương thức xuất :



***Hình 16.6** Dữ liệu được xuất dưới dạng HTML và trình bày trong trình duyệt IE*

16.3.2 Triển khai lớp thành Active Server

Ta có thể dùng Visual Basic để biên dịch các đề án dựa trên lớp thành các thành phần ActiveX. Các thành phần này, dưới dạng các DLL hay EXE, cung cấp các chức năng của đối tượng mà không cần phân phát hay sao chép mã nguồn của lớp. Nó tiện dụng khi ta dùng lại chương trình trong nhiều đề án cũng như với nhiều lập trình viên. Tạo thành phần ActiveX từ lớp cũng cho ta khả năng phân phát đối tượng từ xa, như trong chương 15.

Để tạo đề án của thành phần ActiveX trong Visual Basic, ta bắt đầu tạo một đề án mới. Khi Visual Basic yêu cầu ta cung cấp kiểu đề sẵn, chọn ActiveX DLL hay ActiveX EXE. Một đề án mới được tạo với một mô-dun lớp duy nhất. Sau đó, ta có thể thêm các lớp khác để lập trình. Bước cuối cùng là biên dịch toàn bộ thành ActiveX DLL hay ActiveX EXE.

Biên dịch đề án ActiveX tương tự biên dịch với đề án thông thường. Điểm khác biệt là việc sử dụng ActiveX DLL và ActiveX EXE được thiết kế để dùng với các ứng dụng khác.

16.3.2.1 Sử dụng thành phần ActiveX trong một đề án thông thường STANDARD EXE

Sau khi biên dịch thành phần ActiveX, ta có thể lập một tham chiếu đến nó từ trong các đề án khác của Visual Basic. Khi ta thực hiện điều này, nghĩa là ta có thể dùng lớp chứa trong thành phần ActiveX để tạo đối tượng trong đề án. Điều này cho phép ta tận dụng tính năng của thành phần ActiveX một cách đơn giản, nhất quán, không bận tâm đến chương trình bên trong của thành phần.

Để thực hiện tham chiếu ActiveX Server dùng menu *Project References.*

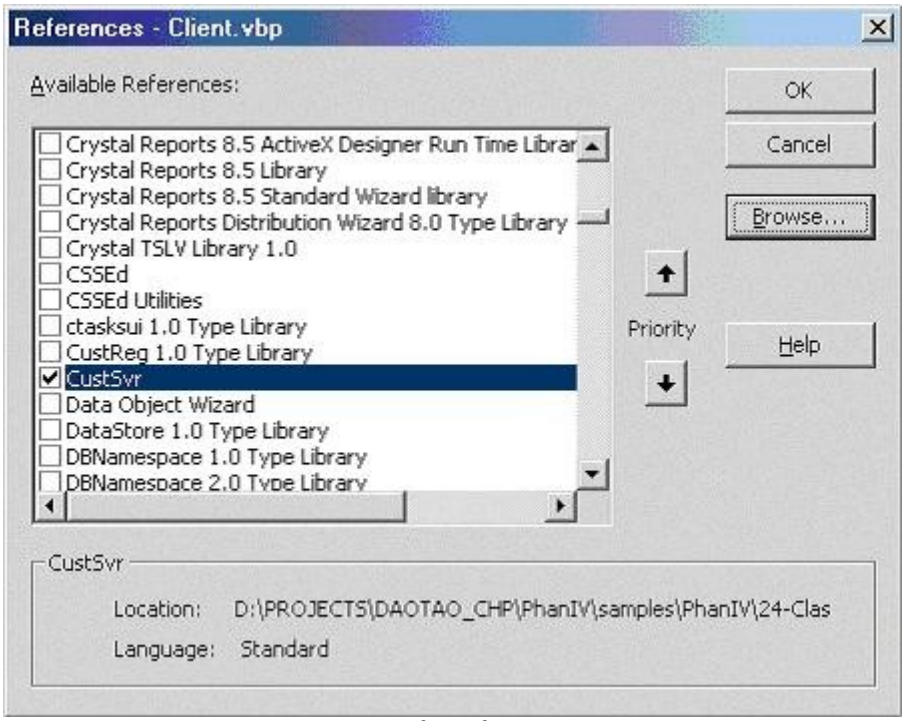
16.3.2.1.1 Biên dịch ActiveX Server

1. Tạo một đề án Visual Basic mới. Khi Visual Basic yêu cầu ta cung cấp kiểu đề án cần tạo, chọn ***ActiveX DLL.***
2. Trong cửa sổ *Properties,* sửa thuộc tính Name của đề án ActiveX DLL thành *CustSrv.*
3. Visual Basic thêm một lớp rỗng, gọi là Class1, vào đề án ActiveX DLL một cách mặc định. Nhấn nút phải chuột lên lớp, chọn Remove Class1 từ menu bật ra để loại bỏ lớp này.
4. Bởi vì ta sắp thêm vào đề án một lớp sử dụng **ADO,** ta dùng menu *Project Refenrences* để lập một tham chiếu đến *Microsoft ActiveX Data Objects 2.0.* Nếu muốn dùng phiên bản của *CCustomer* xây dựng trong phần xuất ra HTML trước đây, lập một tham chiếu đến *Microsoft Scripting Library.*
5. Kế đến, thêm lớp được tạo trước đó vào đề án ActiveX DLL. Để thực hiện điều này, nhấn nút phải chuột lên đề án CustSrv, chọn *Add* từ menu bật ra. Chọn ***Class Module.***
6. Hộp thoại ***Add Class Module*** xuất hiện. Chọn *tab Existing.* Chọn lớp *CCustomer.*
7. Trong thuộc tính của lớp *CCustomer,* đổi thuộc tính ***Instancing*** thành *5_Multiuse.*
8. Nếu cần, thêm trình thiết kế ***DataEnviroment*** ta tạo trước đó vào đề án. Thực hiện bằng cách chọn menu *Project, Add File,*rồi chọn tập tin *deNovelty.Dsr* từ hộp thoại tập tin.
9. Biên dịch đề án bằng cách chọn menu *File,* ***Make CustSrv.dll***.
10. Hộp thoại ***Make Project*** xuất hiện. Chọn thư mục lưu DLL và nhấn OK.

ActiveX DLL được biên dịch là *CustSrv.dll* ta có thể sử dụng nó trong bất kỳ đề án nào.

16.3.2.1.2 Sử dụng ActiveX Server

1. Tạo một đề án mới kiểu *Standard EXE.*
2. Tìm và lập tham chiếu đến *CustSrv* trong menu *Project References.*



Hình 16.7 *Thêm một tham chiếu đến một ActiveX Server được tạo ra từ đề án kiểu Standard EXE.*

3. Trong biểu mẫu, đưa vào đoạn chương trình tạo *instance* của đối tượng từ ActiveX server và trả về thông tin từ cơ sở dữ liệu dưới dạng một đối tượng.

```
Option Explicit

' References CustSvr

Private Cust As CCustomer

Private Sub Form_Load()
   Set Cust = New CCustomer
End Sub

Private Sub cmdGetCust_Click()

   Cust.GetData txtID.Text

   MsgBox Cust.FirstName & " " & _
        Cust.LastName & " " & _
        Cust.Address & " " & _
        Cust.City & " " & _
        Cust.State
End Sub
```

4. Thi hành ứng dụng. Nó sẽ hiển thị dữ liệu từ cơ sở dữ liệu khi ta nhập một số hợp lệ trong hộp văn bản và nhấn nút lệnh.

Đối với một ứng dụng độc lập thì ứng dụng này có lợi ích không rõ rệt nếu ta dùng trong ứng dụng phân tán thì nó có lợi ích đáng kể.

Để có thể triển khai từ xa qua mạng có thể dùng *Microsoft Transaction Server*.

16.3.2.2 Đăng ký thành phần trong ActivateX trên máy người sử dụng

Thành phần ActiveX cần được đăng ký trên máy tính để sử dụng lại trên máy đó. Việc đăng ký bảo đảm rằng hệ điều hành sẽ nhận ra sự tồn tại của thành phần mỗi khi ứng dụng tạo đối tượng từ đó.

Khi ta làm việc với ActiveX server trên một máy tính dùng vào việc phát triển phần mềm, việc đăng ký không là vấn đề. Bởi vì thành phần được tự động đăng ký với Visual Basic ngay lúc chúng được biên dịch. Tuy nhiên khi ta phân phát thành phần đến người sử dụng, ta phải bảo đảm rằng thành phần ActiveX được đăng ký trên máy người sử dụng. Có một số phương pháp để thực hiện điều này.

Sử dụng ***Trình đóng gói và triển khai ứng dụng*** *(Package and Deployment Wizard)* để phân phát người sử dụng.

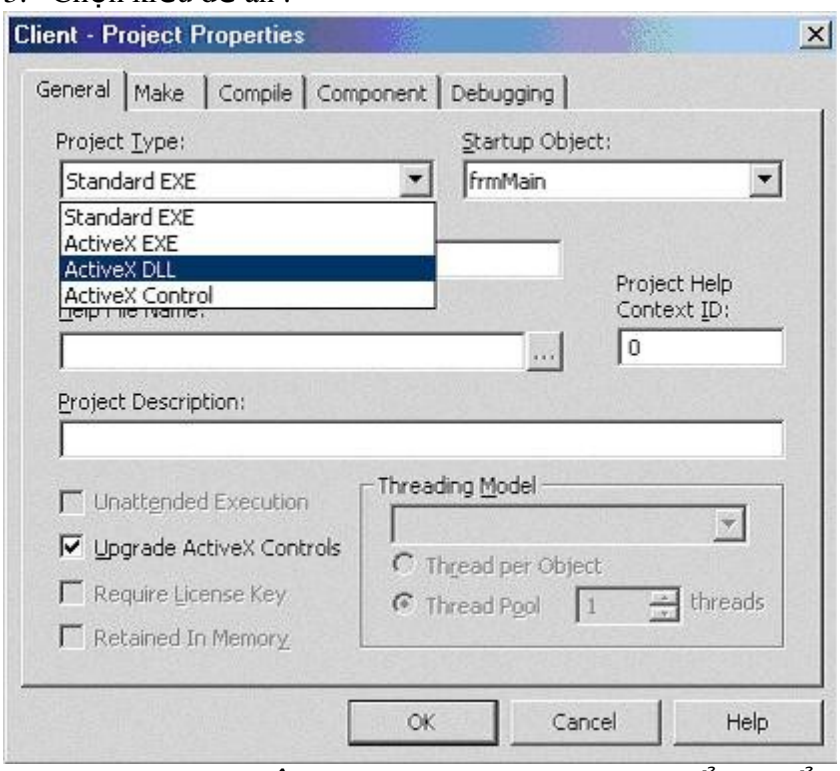
Sử dụng bằng phương pháp thủ công dùng tiện ích ***Regsrv32. EXE.***

16.3.2.3 Chuyển đổi đề án Standard EXE thành đề án ActivateX

Ta có thể chuyển đổi đề án *Standard EXE* thành đề án ActiveX khi ta dự định dùng lại một đề án có sẵn và biến nó thành thành phần ActiveX độc lập.

Ngoài ra, ta có thể chuyển đổi giữa 2 kiểu thành phần ActiveX : ***ActiveX DLL*** và ***ActiveX EXE)*** bằng cách đổi thuộc tính của đề án :

1. Trong Visual Basic, mở đề án ta dự định chuyển đổi.
2. Chọn menu *Project Properties.*
3. Chọn kiểu đề án :



Hình 16.8 *Dùng hộp thoại Project Properties để chuyển đổi một đề án Standard EXE thành một đề án ActiveX.*

16.3.2.4 Sử dụng thành phần ActiveX từ xa

Khả năng biên dịch bộ các lớp độc lập thành các thành phần ActiveX là phần quan trọng của truy cập cơ sở dữ liệu từ xa. Trong môi trường *Client / Server,* ta có thể dùng ActiveX Server để xử lý tương tác giữa ứng dụng Client và các *Server* cơ sở dữ liệu.

16.3.2.5 Tạo thành phần đa luồng

Visual Basic có khả năng tạo thành phần ActiveX ***đa luồng*** *(multi-thread).* Với đa luồng, thành phần thi hành theo cách đáp ứng hơn. Bởi vì hệ điều hành có thể cấp phát tài nguyên bộ xử lý cho nhiều luồng; nếu thành phần thi hành với nhiều luồng, chức năng của thành phần được chia thành nhiều khối cho nhiều luồng (một số luồng có thể được gán những mức ưu tiên cao bởi hệ điều hành). Ta có thể xem luồng như đơn vị điều khiển nhỏ nhất của hệ điều hành có khả năng lập lịch biểu thi hành.

Ngoài ra, trên những máy có nhiều bộ vi xử lý, hệ điều hành có khả năng gán những luồng nhất định để thi hành trên những bộ vi xử lý riêng. Cho thành phần khả năng đa luồng là giải pháp tận dụng tối đa năng lực máy tính với nhiều bộ vi xử lý.

Thêm hỗ trợ đa luồng trong thành phần chỉ đòi hỏi ta đổi thuộc tính đề án vào lúc biên dịch. Không có thay đổi nào khác trong chương trình.

1. Trong Visual Basic, mở đề án của thành phần (*ActiveX EXE, ActiveX DLL, điều khiển ActiveX).*
2. Từ menu *Project,*chọn *Properties*
3. Chọn ***Apartment Threaded*** trong bảng *Threading Model.*
4. Nhấn OK. Khi đề án được biên dịch, nó sẽ có hỗ trợ đa luồng.

Ta có thêm khả năng bổ sung cho ***ActiveX EXE*** là : ***Thread per Object*** hay ***Thread Pool.*** *Thread per Object* có nghĩa là *ActiveX EXE* sẽ sinh ra một luồng cho mỗi đối tượng được tạo. Điều này cung cấp khả năng hoạt động tốt nhất, nhưng có thể tiêu tốn nhiều tài nguyên máy tính.

ThreadPool, trái lại, cho phép ta kiểm soát có bao nhiêu luồng mà thành phần ActiveX có thể tạo cùng một lúc, cho phép ta hạn chế tài nguyên tiêu tốn bởi thành phần. Tuy nhiên, nếu số đối tượng yêu cầu vượt quá số luồng trong *Thread pool* của Server của thành phần, những yêu cầu đến sau trên đối tượng của Server sẽ bị ngăn lại cho đến khi quá trình của *Client* giải phóng một luồng.

16.3.2.6 Các hạn chế của thành phần đa luồng

Ngôn ngữ hạn chế hỗ trợ đa luồng: Đây là chuyển đổi biên dịch. Ta không thể viết chương trình để làm gì với nó cả.

Không có trình gỡ rối hỗ trợ đa luồng.

Không có hỗ trợ đa luồng trong ứng dụng MDI

Không hỗ trợ cho các điều khiển ActiveX đơn luồng trong ứng dụng đa luồng : Điểm hạn chế này bị áp đặt bởi Visual Basic do vấn đề lớn khả năng hoạt động sinh ra bởi tổ hợp của một điều khiển đơn luồng trong một nơi chứa đa luồng. Điều này có nghĩa là nếu ta tạo một điều khiển ActiveX

đa luồng sử dụng các điều khiển đơn luồng khác – nó sẽ không hoạt động. Tuy nhiên phần lớn các điều khiển trong Visual Basic là đa luồng.

Các thuộc tính và phương thức được khai báo là *Friend* không thể gọi qua luồng. Để goi thuộc tính và phương thức đặc biệt gọi qua *Public.*

16.4 Tổng kết

Lớp và đối tượng thể hiện lĩnh vực mới của lập trình Visual Basic.

Chương này cung cấp những thông tin để biến các kỹ thuật hướng đối tượng mở rộng thành những hành động, không chỉ là tận dụng những kỹ thuật cao cấp như đa luồng và thành phần, mà còn tạo sự đơn giản và ổn định theo thời gian.

7.HỎI VÀ ĐÁP

Hỏi: *Lớp và đối tượng có vẻ phải xử lý phức tạp. Một điều tôi thắc mắc là "Bạn có thể tạo những ứng dụng đơn giản hơn bằng cách viết thêm chương trình và thêm mô-dun nhiều hơn". Tôi có nghĩ sai không ?*

Đáp: Bạn có phần đúng là khi xây dựng một ứng dụng với lớp và đối tượng thì sẽ tốn nhiều thời gian cho thiết kế và xây dựng lần đầu. Lưu ý rằng một trong những mục tiêu của lập trình hướng đối tượng là tính năng sử dụng lại. Nó sẽ tiết kiệm cho bạn rất nhiều thời gian và sức lực sau này.

Hơn nữa, nó còn giúp bạn dễ bảo trì, dễ gỡ rối.

17 Truy cập dữ liệu từ xa

Client / Server và các thành phần
Cấu trúc Client / Server Three-Tier
Dùng ActiveX truy cập cơ sở dữ liệu
Chuyển đổi dữ liệu với bản sao cơ sở dữ liệu

17.1 Client / Server và các thành phần

Thuật ngữ thành phần tầng trung gian (*middle-tier)* đã thay đổi từ khi kỹ thuật này được giới thiệu lần đầu tiên. Ví dụ, ta tạo một điều khiển ActiveX nhúng trong một ứng dụng *Client* để nói chuyện với một thành phần ActiveX Server cũng chứa trên *Client.* Sau đó, thành phần ActiveX Server sẽ nói chuyện với ActiveX Server chứa trên máy thuộc tầng trung gian, đến lượt tầng trung gian sẽ nói chuyện với Server cơ sở dữ liệu chứa trên WinNT Server.

Ta lướt qua một số khái niệm cơ bản :

ActiveX : là khái niệm gắn liền với các đối tượng trong chương trình dùng giao tiếp với nhau. Nó không phải là một sản phẩm, cũng không là một kỹ thuật.

Điều khiển ActiveX : là một thành phần thường có giao diện(nhưng không phải các thành phần đều có). Thành phần lưới cơ sở dữ liệu *(DBGrid)* được cung cấp bởi Visual Basic là một thành phần ActiveX.

ActiveX Server: còn gọi là thành phần mã hoá ActiveX. Trong Visual Basic 4.0 nó gọi là OLE Server. ActiveX Server là một thành phần đưa ra một hay nhiều lớp chứa trong một khối đã được biên dịch. Ứng dụng sẽ dùng các đối tượng được sinh ra từ các lớp chứa trong thành phần ActiveX. Ta có thể truy cập khối biên dịch này thông qua kỹ thuật **DCOM** (Mô hình đối tượng thành phần phân tán – *Distributed Component Object Model).*

Chương này tập trung về vấn đề sử dụng ***ActiveX Server*** trên mạng sao cho các máy Client có thể tạo *instance* từ mạng.

17.1.1 Cấu trúc Cilent/Server Three- Tier

Những nguyên tắc mà người lập trình *Client / Server* phải tuân thủ :

Duy trì mộ hệ thống ổn định để đáp ứng với các quy luật kinh doanh thường xuyên thay đổi.

Cung cấp điểm khởi đầu đơn giản và nhất quán cho dữ liệu, trong khi cùng lúc bảo vệ cơ sở dữ liệu khỏi các ứng dụng Client, và trái lại, bảo vệ các ứng dụng Client khỏi tính phức tạp và bất thường của Server.

Ý tưởng xây dựng các quy luật kinh doanh trong ngôn ngữ lập trình bất kỳ, không chỉ với SQL.

Ý tưởng triển khai các quy luật kinh doanh trên một máy tính không phải là Server cơ sở dữ liệu, để bảo toàn năng lực xử lý của Server cơ sở dữ liệu.

Ý tưởng triển khai các quy luật kinh doanh vào Client để giảm thiểu lưu thông trên mạng.

Triển khai các quy luật kinh doanh tại một điểm đơn giản trong hệ thống (trái với ý nghĩa triển khai toàn bộ trên mọi máy Client mỗi lần có thay đổi).

Cấu trúc *Client / Server* giải quyết vấn đề này. Trong một cấu trúc 3 tầng *(three-tier),* các quy luật kinh doanh được đóng gói trong một thành phần tầng trung gian giữa ứng dụng Client và Server cơ sở dữ liệu.

17.1.1.1 THIẾT LẬP MỘT CẤU TRÚC PHẦN CỨNG CHO DCOM

Có nhiều phương pháp thực hiện cấu trúc phần cứng *Client / Server.* Chương này chỉ đưa ra cách đơn giản nhất để cung cấp cho bạn giải pháp trong trường hợp thường gặp nhất.

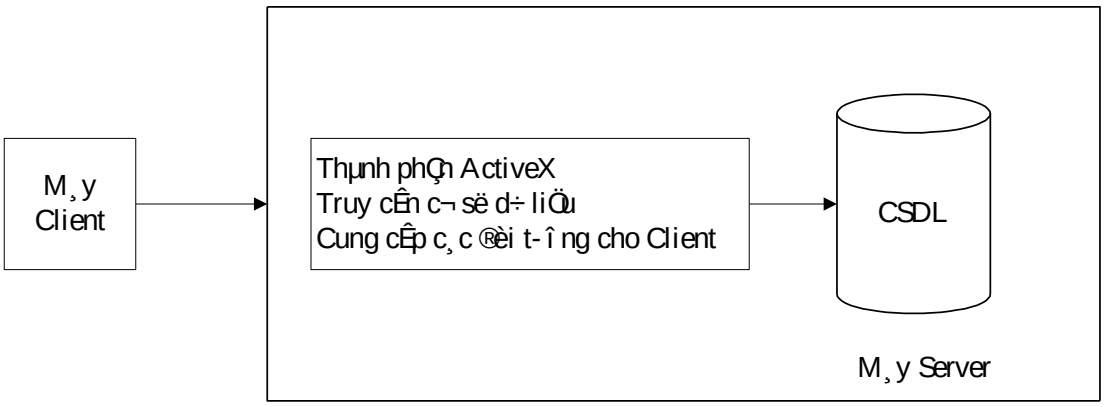
Giả sử rằng có 2 máy, trong đó một máy là WinNT Server chạy *Microsoft SQL Server.* Đây là Server. Máy Client là Win95, hoặc Win98, hoặc WinNT.

Có thể bạn không có sẵn các phần mềm và phần cứng mô tả như trên. Nếu không có 2 máy nối mạng, bạn vẫn có thể thực hiện trên cùng một máy. Nếu bạn không có máy WinNT, bạn nên giả lập nó bằng cách dùng cơ sở dữ liệu của *Microsoft Jet.* Toàn bộ ví dụ trong chương này được viết bằng ADO để phù hợp với điều này.

Nếu không có WinNT Server, ta có thể thiết lập **DCOM** cho Win95. Tập tin để thực hiện DCOM trên Win95 chứa trên đĩa một của bản Visual Basic *Enterprise,* trong thư mục *\Tools\DCOM98.* Muốn xem chi tiết, cũng như tải xuống các tập tin **DCOM,** ta tìm trong trang web *http:// www.microsoft.com /com /dcom/dcom1_2 / dcom1_2. asp*

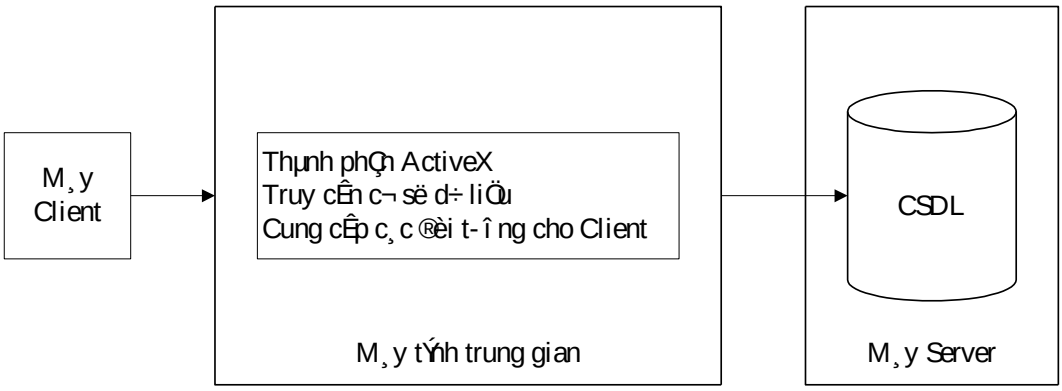
Ví dụ người sử dụng trên mạng muốn truy cập dữ liệu khách hàng. Để giữ được tính nhất quán, khả năng dùng lại, dễ lập trình và bảo trì, bạn nên truy cập đến cơ sở dữ liệu thông qua thành phần ActiveX. Thành phần này được biên dịch và thiết lập trên mạng sao cho ứng dụng Client có thể truy cập nó từ xa.

Hình sau đây là sơ đồ của cấu trúc 3 tầng tổng quát. Nó kết hợp cấu trúc vật lý (2 máy tính) và cấu trúc logic (từng tầng cung cấp các chức năng khác nhau).



***Hình 17.1** Lược đồ của một cấu trúc 3 tầng tổng quát hoà trộn cấu trúc vật lý (2 máy tính) với cấu trúc logic (mỗi tầng có 1 tính năng khác nhau).*

Ta có thể chia thành phần tầng trung gian qua nhiều máy. Vì vậy, cấu trúc Client / Server 3 tầng khi đó trở thành cấu trúc *n* tầng.



Hình 17.2 *Đây là ví dụ về cấu trúc Client / Server n tầng trình bày tầng trung gian thể hiện trên một máy.*

Ta triển khai các thành phần tầng trung gian trên một máy theo những tình huống sau:

Server cơ sở dữ liệu chạy trên hệ điều hành không hỗ trợ ActiveX.

Ta muốn giữ gìn năng lực của Server cơ sở dữ liệu.

Ta muốn mở rộng khả năng ứng dụng bằng cách triển khai nhiều bản sao của cùng một ActiveX Server đến nhiều hơn một máy.

Bất lợi của việc cung cấp cho thành phần ActiveX một máy tính riêng là cấu hình này đòi hỏi thêm một bước chuyển qua mạng từ Client đến Client và ngược lại. Điều này có thể khiến giảm khả năng hoạt động và tăng lưu lượng trên mạng. Trường hợp này có được chấp nhận hay không tuỳ thuộc chức năng định cấu hình mạng và các *prototype* ta làm trong giai đoạn thiết kế của đề án.

Thậm chí khi ta không có điều khiển trên hệ điều hành trên đó cơ sở dữ liệu thi hành, ta vẫn có thể sử dụng tầng trung gian - nếu ta triển khai nó đến một máy riêng. Tầng cơ sở dữ liệu là một khối UNIX hay bộ *mainframe* của công ty. Ứng dụng Client dùng kỹ thuật **DCOM** vì 2 lý do :

1. Ứng dụng *Client* không bao giờ truy cập đến Server cơ sở dữ liệu, thay vào đó, chúng truy cập dữ liệu thông qua thành phần ActiveX.
2. Đây là lý do quan trọng nhất, DCOM che chắn cho ứng dụng khỏi các phần rắc rối của giao thức qua mạng, các phần mềm nền (*platform),* và biên của máy tính.

Triển khai một hệ thống Client / Server trong đó, Server cơ sở dữ liệu không phải *Windows* thì dễ hơn trong thế giới 3 tầng, bởi vì, ta không cần phải định cấu hình cho *middleware (*phần mềm tầng giữa- tập hợp các chương trình điều khiển chuyên xử lý việc giao tiếp qua nhiều *platform* và các giao thức mạng) trên từng máy Client. Với kiểu 3 tầng, ta chỉ phải định cấu hình chương trình này một lần – trong phần cấu trúc giữa tầng giữa và Server cơ sở dữ liệu. Nhờ đó, có thể tiết kiệm thời gian và sơ sót khi ta định cấu hình và triển khai ứng dụng đến nhiều Client.

17.1.1.2 DÙNG DCOM TRÊN NHỮNG PLATFORM KHÁC

DCOM có thể làm việc trên những môi trường khác Windows. Microsoft có quan hệ với các nhà cung cấp hệ điều hành khác để nhờ họ làm cho DCOM xuất hiện trên *platform.*

COM và DCOM bắt đầu xuất hiện trên UNIX. ActiveX SDK (*Công cụ phát triển phần mềm – Software Development Kit)* trên Macintosh đã có vào cuối năm 1996; có thể đến lúc này, sử dụng ActiveX trên Macintosh là hiện thực.

17.1.1.3 TẠO ỨNG DỤNG DCOM ĐẦU TIÊN

Xem lại ví dụ của chương 16 đề án *CustSrv.* Đây là thành phần ActiveX có một lớp là *CCustomer,* dùng để truy vấn cơ sở dữ liệu và trả về một đối tượng *Customer.*

Phiên bản của đề án *CustSrv*2, loại bỏ hỗ trợ cho phương thức HTMLSave (không phù hợp cho một thành phần triển khai ở xa) và hỗ trợ thêm thuộc tính ***GetList.*** *GetList* hiển thị danh sách tóm tắt các khách hàng dựa trên tiểu bang mà họ sinh sống. Nó dùng OLE DB từ ***Trình cung cấp Jet cục bộ*** *(native Jet Provider).* Nhờ đó, cho phép dễ dàng thao tác qua mạng. (ODBC DSN được yêu cầu bởi thành phần Jet Novelty).

17.1.1.3.1 ActiveX EXE

Đừng dùng các ActiveX EXE triển khai từ xa. Ví tính năng của MTS đáp ứng rất tốt vấn đề triển khai từ xa.

17.1.1.3.2 TRIỂN KHAI ActiveX DLL TỪ XA DÙNG MTS (MICROSOFT TRANSACTION SERVER)

Triển khai một thành phần dùng MTS là cách dễ nhất để tạo một ứng dụng phân tán.

MTS chạy trên WinNT, Win95 và Win98; nhưng hiệu quả nhất là WinNT.

MTS có trong bộ WinNT 5.0. Với NT 4.0 (là môi trường chúng ta đang nói chuyện), ta phải cài đặt phiên bản mới nhất của MTS dùng *Windows Option Pack*, kèm theo đĩa CD của Visual Basic. Nó nằm trong đĩa 2 của bản *Enterprise,* thư mục *\NToptpak.* Hoặc là ta có thể tải xuống *Windows Option Pack* từ web site tại *http:// www.microsoft.com/windows/dowloads/contents/Updates/NT40ptPK*. Lưu ý rằng Option Pack được áp dụng cho Win95, Win98 – khi cài *NT Option Pack* trên máy không phải NT, ta có *Personal Web Server* đối với Win95 và một phiên bản hạn chế của MTS.

Phần này làm việc với MTS 2.0. Chương này chỉ nói về sử dụng MTS với vai trò là một trung gian yêu cầu đối tượng và theo dõi các thành phần phân tán từ xa. Ta không tìm hiểu các tính năng khác của MTS như là cho phép *transaction* giữa các thành phần, hay là mô hình bảo mật của MTS.

Muốn nghiên cứu thêm MTS vào http:// www.microsoft.com/ntServer/basics/appservices/transsvcs/

Triển khai một thành phần từ xa dưới MTS có 3 bước:

1. Tạo gói MTS để chứa thành phần.

2. Đặt thành phần vào gói.
3. Export gói ra từ MTS và cài đặt vào máy Client.

Tạo gói MTS

Gói MTS là một nơi chứa logic cho một hay nhiều mô-dun lớp chứa trong ***ActiveX DLL.*** Việc tạo gói cung cấp khả năng quản lý mọi lớp chứa trong gói, thậm chí các lớp của các thành phần khác.

1. Phóng MTS từ menu Start của Windows (Nó tồn tại trong các nhóm lập trình khác nhau tuỳ theo phiên bản và cách cài đặt).
2. MTS xuất hiện trong cửa sổ ***Microsoft Management Console:***

MMC cung cấp khả năng mở rộng thông qua *snap-in.* Snap-in là một thành phần phần mềm hoạt động như một phần nối giữa MMC và công cụ của Server. *Snap-in* được đóng gói như một ActiveX DLL, nhưng ta chưa thể xây dựng *snap-in* trong Visual Basic. Vì vậy, nếu bạn có một công cụ phần mềm mà bạn muốn quản lý trong MMC, bạn phải chờ đến khi nó hỗ trợ Visual Basic hoặc là bạn viết *snap-in* trong Visual C++.

3. Trong phần cửa sổ bên trái, mở mục *Microsoft Transaction Server,* sau đó mở *Computers,* rồi *My Computer.* Ta sẽ thấy mục ***Packages Installed.***
4. Nhấn vào mục này, Ta sẽ thấy một số gói có sẵn (mặc dù các gói này có thể thay đổi tuỳ vào hệ thống).
5. Nhấn nút phải chuột trên thư mục ***Packages Installed.*** Từ menu bật ra, chọn *New,* ***Package.***
6. Trình tự động *Package Wizard* xuất hiện. Chọn *Create an empty package.* Đặt tên nó là Novelty.
7. Màn hình ***Set Package Identify*** xuất hiện. Bởi vì ta không dùng bảo mật với thành phần này, hãy để nó quy định là *Interactive user,* rồi nhấn ***Finish.***
8. Ta sẽ thấy gói mới trong thư mục ***Package Installed.***

Mặc dù ta đã tạo xong gói, nhưng nó chưa làm gì cả, bởi vì ta chưa thêm thành phần cho nó.

Đưa thành phần vào gói

Công việc rất đơn giản chỉ cần kéo rê ***ActiveX DLL*** vào gói đã tạo sẵn.

1. Trong MTS, mở mục Novelty dưới mục *Packages Installed.*
2. Ta sẽ thấy 2 mục : ***Components*** và ***Roles.*** Nhấn vào mục *Components.* Mục này rỗng vì ta chưa thêm thành phần vào.
3. Sắp xếp các cửa sổ màn hình sao cho *Windows Explorer* mở sát bên cửa sổ MTS.
4. Chọn và kéo rê tập tin ***ActiveX DLL*** từ cửa sổ *Windows Explorer* vào phần bên phải của cửa sổ MTS. Các lớp được cung cấp bởi thành phần được thêm vào gói MTS.

Một biểu tượng quả bóng màu xanh lá cây và đen hiển thị trong mục *Components* cho biết thành phần đã được cài đặt và nó đang phục vụ yêu cầu. Ta có thể kiểm tra bằng cách :

1. Tạo một ứng dụng Visual Basic kiểu *Standard EXE*.
2. Dùng menu *Project References,* cho một tham chiếu đến thành phần *CustSrv* mà ta vừa đặt trong MTS. (Tên thư viện là CustSvr2)
3. Nhập đoạn chương trình sau đây:

```
Private mCust As CCustomer

Private Sub Form_Load()
   Set mCust = New CCustomer
End Sub

Private Sub Form_Unload(Cancel As Integer)
   Set mCust = Nothing
End Sub
```

Ứng dụng này không làm gì với thành phần cả, nó chỉ giữ một *instance* của đối tượng được mở để ta có thể xem xét cách MTS phục vụ các yêu cầu.

4. Thi hành ứng dụng. Nếu ta sắp xếp 2 cửa sổ của ứng dụng và cửa sổ của MTS sát cạnh nhau, ta sẽ thấy quả bóng đang quay, thể hiện nó đang phục vụ các yêu cầu của ứng dụng.
5. Đóng ứng dụng, quả bóng ngừng quay nó không phục vụ nữa.

Trong khi quả bóng đang quay, ta có thể nhấn nút phải chuột lên biểu tượng gói, khám phá cửa sổ *Properties* của nó. Trong cửa sổ này, chọn tab ***Advanced,*** MTS cho phép ta quy định MTS sẽ giữ thành phần nạp vào bộ nhớ trong bao nhiêu phút sau lần truy cập đầu tiên. Giá trị này có thể là 0, nghĩa là thành phần bị xoá khỏi bộ nhớ ngay khi ứng dụng nhả nó ra. Nếu giá trị này là 1440 phút, thành phần được *cache* vào bộ nhớ trong 24 giờ sau khi Client truy cập lần cuối cùng. Nếu muốn nạp thành phần vào bộ nhớ bất chấp Client có dùng nó hay không, chọn tuỳ chọn "*Leave running when idle."* Nó sẽ tăng khả năng hoạt động và tiêu tốn vùng bộ nhớ trên phía Server.

Export và cài đặt gói MTS trên máy Client

Để cho phép ứng dụng Client truy cập thành phần chạy dưới MTS, ta phải đăng ký vào *Registry* của máy Client để báo với nó rằng thành phần đang chạy từ xa.

Để Client nhận ra thành MTS phân phát từ xa, ta làm như sau :

Export gói : Tạo một tập tin cho phép cài đặt. Tập tin này thi hành mọi hoạt động cần thiết để Client nhận ra thành phần triển khai từ xa.

Cài đặt gói lên máy Client.

Để xem ví dụ ta làm như sau:

1. Trong cửa sổ MTS, nhấn nút phải chuột lên gói Novelty. Từ menu bật ra, chọn ***Export.***
2. Hộp thoại ***Export Package*** xuất hiện. Dùng nút *Browse* để chọn một thư mục rỗng để export gói. Cho vào tập tin Novelty, nhấn *Save.*
3. Trở về hộp thoại *Export Package.* Nhấn ***Export.*** Sau một lúc, thông báo "*The package was successfully exported"* xuất hiện.

Thư mục dùng để *export* gói giờ đây chứa các tập tin bao gồm:

Một tập tin gọi là Novelty.PAK

Bản sao của ActiveX DLL, CustSvr2.dll

Một thư mục gọi là Clients. Thư mục này chứa một tập tin EXE gọi là Novelty. exe

Tập tin Novelty.PAK và CustSvr2.DLL được dùng để tạo bản sao cho bộ cài đặt của gói Novelty vào máy MTS. Tập tin Novelty.exe chạy trên máy Client để chuyển yêu cầu của đối tượng chứa trong CustSvr2.dll vào máy MTS. Mỗi lần chạy Novelty.exe trên máy Client, ứng dụng tham chiếu đến thành phần CustSvr2 sẽ truy cập nó qua mạng thay vì truy cập nội bộ.

Để xoá một thành phần khỏi MTS, chỉ cần nhân nút phải chuột lên nó và và chọn Delete trong menu bật ra. Thực chất, hành động này không xóa ***Active DLL*** khỏi hệ thống; nó chỉ bỏ những gì đã đăng ký. Ứng dụng sẽ không thể dùng thành phần cho đến khi nó được đăng ký lại.

17.1.1.3.3 3. DÙNG ActiveX ĐỂ TẠO THUẬN LỢI CHO VIỆC TRUY CẬP CƠ SỞ DỮ LIỆU

Trong chương trước, bạn đã biết cách truy cập cơ sở dữ liệu dùng đối tượng và các lớp. Kỹ thuật này cho phép đóng gói logic chương trình trong mô-dun lớp để có thể dùng lại dễ dàng. Khả năng để biên dịch mô-dun lớp một cách độc lập thành một ***ActiveX DLL*** hay ***ActiveX EXE*** làm cho lớp dễ dùng hơn, cho phép tạo ra các ứng dụng truy cập cơ sở dữ liệu cực kỳ linh hoạt và mạnh mẽ.

Kỹ thuật này cũng được dùng khi cần tối ưu hoá tốc độ của ứng dụng Client. Tốc độ hoạt động bắt nguồn từ việc ứng dụng Client không phải nạp một bộ máy cơ sở dữ liệu (*database engine),* hoặc những tầng trung gian khác khi chúng khởi động. Nếu ta dùng ADO hay ODBC, ta sẽ có cùng thuận lợi, vì ta không phải phân phát các thư viện này hoặc định lại cấu hình trên máy Client. Nó tiết kiệm đáng kể bộ nhớ và ứng dụng chạy nhanh hơn nhiều.

Một thuận lợi khác của kỹ thuật này là các phần liên quan được phân phát cùng với ứng dụng cũng giảm nhỏ, vì ta không phải nạp các thư viện của bộ máy cơ sở dữ liệu Jet vào từng máy Client.

Thêm vào đó, bằng cách truyền mảng thay vì đối tượng qua mạng, ứng dụng không phải duy trì một kết nối thường xuyên đến Server. Bởi khi truyền đối tượng qua mạng, nó tồn tại trên Server. Tuy nhiên, khi truyền một khối dữ liệu chẳng hạn như một mảng qua mạng, dữ liệu không tồn tại trên Server. Nó hoàn toàn được truyền đến Client, nghĩa là Client không cần nối đến Server để làm việc với dữ liệu. Như vậy, ứng dụng có thể phục vụ nhiều người cùng một lúc, vì chẳng hạn như thay vì có 50 người sử dụng nối đến Server mà không làm gì, ta có 500 người sử dụng chỉ nối đến Server khoảng 10% thời gian.

17.1.1.3.3.1 SỬ DỤNG GETROWS ĐỂ TRẢ VỀ DỮ LIỆU MẢNG

Có thể dùng phương thức ***GetRows*** của đối tượng *Recordset* để trả về dữ liệu từ *ActiveX Server* đến ứng dụng Client dưới dạng mảng 2 chiều chứa các giá trị kiểu Variant. Kỹ thuật này được ưa thích vì ta không phải triển khai các thư viện bộ máy cơ sở dữ liệu cũng như định cấu hình cho cơ sở dữ liệu phía *Client. Client*

chỉ biết đến các mảng cung cấp bởi ActiveX Server đang được triển khai qua mạng.

1. Tạo một đối tượng *Recordset* (thường là kết quả của một yêu cầu từ Client đến thành phần ActiveX triển khai từ xa).
2. Nếu đang dùng **DAO,** ta phải xác định số dòng trong đối tượng *recordset.* Trong ADO, ***GetRows*** tự động quyết định kích cỡ của *recordset.*
3. Khai báo biến *Variant* để chứa mảng.
4. Thi hành phương thức *GetRows* của đối tượng *Recordset* để gán dữ liệu trong *Recordset* vào biến *Variant.*
5. Thi hành chương trình để chuyển đổi dữ liệu từ mảng *Variant* thành những gì ứng dụng cần.

Ví dụ sau trả về một mảng *Variant* sau khi gọi cơ sở dữ liệu. Phương thức này, chứa một mô-dun lớp gọi là *CCusData,* được thiết kế để sinh ra một hộp danh sách với các thông tin khách hàng.

```
Option Explicit

' References DAO 3.51

' Private variables
Private db As Database
Private rs As Recordset
'
Public Function GetList(strState As String) As Variant
' Retrieves a list of employees and
' places it into a variant array
' using GetRows.
Dim strSQL As String
   Set db = OpenDatabase("..\..\DB\novelty.mdb")
   strSQL = "SELECT ID, FirstName, LastName " & _
         "FROM tblCustomer " & _
         "WHERE State = '" & strState & "' " & _
         "ORDER BY LastName, FirstName"
   Set rs = db.OpenRecordset(strSQL)
   ' RecordCount isn't valid until
   ' you move to the end of the recordset
   rs.MoveLast
   rs.MoveFirst
   GetList = rs.GetRows(rs.RecordCount)

End Function
```

Xử lý một lớp xử lý mảng trong ứng dụng Client.

```
    Option Explicit

    ' References CustSrvA.
```

```
Private mCustData As CCustData

Const FIRSTNAME = 1
Const LASTNAME = 2

Private Sub Form_Load()

   Set mCustData = New CCustData

   With cboState
     .AddItem "CA"
     .AddItem "WA"
     .AddItem "NV"
   End With

End Sub

Private Sub cboState_Click()
Dim vData As Variant
Dim x As Long

   vData = mCustData.GetList(cboState.Text)

   lstCustomer.Clear

   For x = 0 To UBound(vData, 2)
      lstCustomer.AddItem vData(FIRSTNAME, x) & _
                  " " & vData(LASTNAME, x)
   Next x

End Sub

Private Sub Form_Unload(Cancel As Integer)
   Set mCustData = Nothing
End Sub
```

Bất lợi của kỹ thuật này là ta phải viết khá nhiều chương trình để xử lý mảng Variant trả về từ thành phần ActiveX trên phía Client. Một giải pháp cho vấn đề này là viết một thành phần ActiveX phía Client để xử lý với mảng.

Để thêm một phương thức trả về dữ liệu cho một khách hàng dùng mảng *Variant,* ta dùng phương thức ***GetData:***

```
Public Function GetData(lngID As Long) As Variant
    Dim strSQL As String
    strSQL = "SELECT * FROM tblCustomer " & _
             "WHERE ID= " & lgnID
    Set rs = db.OpenRecordset(strSQL)
    GetData = rs.GetRows
    rs.Close
```

```
    Set rs = Nothing
End Function
```

17.1.1.3.3.2 TẠO MỘT LỚP ĐỂ GIẢI MÃ MẢNG VARIANT

Khi trả về mảng *Variant* ta sẽ gặp một khó khăn. Làm sao xác định cột nào của mảng ứng dụng với trường nào trong cơ sở dữ liệu ? Quan trọng hơn, làm sao áp dụng kỹ thuật OOP để có thể dùng lại sau này ?

Một giải pháp là triển khai một thành phần ActiveX thứ hai, lần này trên Client, để giải mã mảng *Variant* được phục vụ bởi tầng trung gian. Thành phần này sẽ làm 2 việc : gọi *CCustData* và trả về đối tượng cấu tạo từ những dữ liệu trong mảng Variant được truyền về từ tầng trung gian.

CCustData thực hiện như một phần ứng dụng Client. Nó không tham chiếu đến bất kỳ cơ sở dữ liệu nào và tồn tại độc lập nhằm trả về mảng Variant từ CCustData và gán cho các thuộc tính của đối tượng *Customer.*

Ưu điểm của kỹ thuật này là ứng dụng Client xử lý riêng với đối tượng *Customer.* Không có truy cập cơ sở dữ liệu trực tiếp trên Client, lập trình viên phía Client không cần bận tâm loại cơ sở dữ liệu.

Sau khi có lớp *CCustData* thi hành truy cập dữ liệu và lớp *CCustomer* chuyển đổi mảng dữ liệu thành đối tượng, ta viết chương trình để trả về đối tượng *Customer.* Khi chạy chương trình, người sử dụng chọn một Tiểu bang để xem xét các khách hàng đang cư ngụ ở đó. Khi người sử dụng chọn một khách hàng, thông tin chi tiết về người đó xuất hiện.

Kết hợp kỹ thuật này với kỹ thuật ActiveX / DCOM, mang lại những điểm lợi đáng kể. Bằng cách biên dịch lớp *CCustData* thành một thành phần ActiveX và triển khai nó trên mạng, ta đã cho phép truy cập các quy tắc kinh doanh lưu trữ tập trung từ mọi nơi trên mạng. Và bởi vì thành phần này truyền dữ liệu đến Client qua DCOM trong mảng, ta không cần triển khai một thư viện truy cập dữ liệu vào từng máy Client. nó giúp cho chương trình trên Client chạy nhanh hơn, dễ định cấu hình và dễ quản lý hơn.

Trái lại, nêu không triển khai truy cập dữ liệu qua mạng, ta sẽ không đạt được những kết quả này.

17.1.1.3.4 TRUYỀN DỮ LIỆU VỚI BẢN SAO CƠ SỞ DỮ LIỆU

Khi cơ sở dữ liệu được tập trung hóa, toàn bộ dữ liệu chứa hết vào một nơi và cho phép truy cập đến nó từ mọi nơi trong mạng. Tuy nhiên, sẽ rất khó khăn nếu ta di chuyển hay sửa đổi cơ sở dữ liệu để không ảnh hưởng đến Client đang kết nối.

Để giải quyết vấn đề này, Microsoft Jet cung cấp khả năng sao chụp một cơ sở dữ liệu từ máy này sang máy khác gọi là *Replication.* Thay vì cho chụp toàn bộ cơ sở dữ liệu (ta có thể thực hiện bằng cách đơn giản nhất là copy tập tin cơ sở dữ liệu), kỹ thuật sao chụp này có logic khác hơn. Jet sẽ thi hành đồng bộ hoá trên cơ sở dữ liệu (*synchronization)*-so sánh từng mẩu tin để đảm bảo rằng chúng có cùng dữ liệu, sau đó, copy những thay đổi từ cơ sở dữ liệu gốc sang một số tuỳ ý các cơ sở dữ liệu bản sao.

Cơ sở dữ liệu gốc còn được xem là thiết kế gốc, bởi vì nó chứa thiết kế của cơ sở dữ liệu dùng làm nơi chứa tập trung cho các dữ liệu dùng chung. Trong một hệ thống sao chụp, những thay đổi trên thiết kế của cơ sở dữ liệu i như là thêm

hoặc xoá trường, bảng và các định nghĩa truy vấn chỉ có thể thực hiện trong thiết kế gốc. Tuy nhiên, khi có sửa đổi trên đối tượng của cơ sở dữ liệu chứa trong thiết kế gốc, những thay đổi này sẽ được phân phát đến các bản sao tại thời điểm cơ sở dữ liệu được đồng bộ hoá. Đây là cách duy nhất để sửa đổi trên một cơ sở dữ liệu bản sao.

Ví dụ, là người sử dụng của một ứng dụng cơ sở dữ liệu, bạn muốn làm việc ở nhà, nhưng ứng dụng cơ sở dữ liệu chứa trên server của cơ quan. *Replication* cho phép "lấy ra" bản sao của cơ sở dữ liệu bằng cách nạp nó trên máy tính xách tay chẳng hạn. Nhờ đó, bạn có thể làm việc ở nhà. Bản cơ sở dữ liệu chứa trong máy tính xách tay chính là một bản sao cơ sở dữ liệu.

Khi quay lại cơ quan, bạn đăng ký lại cơ sở dữ liệu. Lúc đó, cơ chế ***replication*** của Jet so sánh các mẩu tin trong bản sao với các mẩu tin trong cơ sở dữ liệu gốc. Nếu có thêm mẩu tin mới, chúng sẽ được copy đến hệ thống chính; nếu có sửa đổi trên các mẩu tin, những sửa đổi này cũng được cập nhật vào hệ thống chính. Tương tự, nếu có dữ liệu mới xuất hiện trong bản gốc, nó sẽ được copy vào bản sao cùng lúc đó.

Trong trường hợp cơ sở dữ liệu quá lớn, khoảng 200 MB chẳng hạn, việc tạo ra bản dự phòng cho nó là điều hết sức phiền phức; ta phải tốn khá nhiều thời gian chết để copy dữ liệu qua mạng. Trong thời gian đó, những người sử dụng khác cũng bị cấm truy cập cơ sở dữ liệu để bảo đảm dữ liệu không thay đổi.

Replication giải quyết vấn đề này bằng cách cho phép chỉ copy những tập tin mới hoặc có sửa đổi kể từ lần tạo bản dự phòng mới nhất. Do đó, ta không phải tạo bản dự phòng cho toàn bộ cơ sở dữ liệu.

Ta còn có thể dùng *Replication* để tiến hành cập nhật ứng dụng cơ sở dữ liệu. Nó được dùng trong trường hợp một số logic của ứng dụng được nhúng trong cơ sở dữ liệu dưới dạng câu truy vấn. Nếu cần thay đổi câu truy vấn để phản ánh một thay đổi trong ứng dụng hoặc thay đổi trong thực tế kinh doanh, *replication* có thể chuyển một cách tự động các sửa đổi cho các Client có yêu cầu.

17.1.1.3.4.1 THIẾT KẾ CƠ SỞ DỮ LIỆU CÓ SỬ DỤNG *REPLICATION*

Cũng như với những cơ sở dữ liệu nhiều người sử dụng, khi thiết kế cơ sở dữ liệu có dùng *replicaton,* cần phải dự kiến trước để mọi chuyện tiến hành suôn sẻ. Một trong những vấn đề quan trọng cần xem xét là khoá chính trong bảng. Nhất là khoá chính là một trường đánh số tự động. Bởi mặc định, trường này bắt đầu từ 1 và tăng dần lên cho từng mẩu tin, mỗi mẩu tin tăng thêm 1. Vì vậy, nếu một người sử dụng noà đó đưa thêm các mẩu tin trong bản sao cơ sở dữ liệu, và sau đó đồng bộ hoá dữ liệu về cơ sở dữ liệu gốc; sẽ xảy ra trường hợp hai hay nhiều người sử dụng đưa vào mẩu tin có ID là 1. Để tránh điều này, ta nên thiết kế trường khoá chính cho cơ sở dữ liệu theo một trong những kỹ thuật sau :

Quy định thuộc tính *New Values* của trường AutoNumber là ngẫu nhiên (*Random).*

Quy định thuộc tính *Field Size* của trường AutoNumber là *Replication ID.*

Replication còn có thể thêm trường cho từng bảng trong hệ thống, điều này có thể gây ra vấn đề nếu ta viết chương trình (hoặc phát triển giao diện người sử dụng) dựa trên tập hợp gồm một số trường. Bài học rút ra là ta nên tạo khả

năng *replication* cho cơ sở dữ liệu càng sớm càng tốt trong quy trình xây dựng chương trình, để tránh rắc rối kể trên.

17.1.1.3.4.2 THỰC HIỆN *REPLICATION* VỚI MICROSOFT ACCESS

Để bắt đầu với hệ cơ sở dữ liệu hỗ trợ sao chụp, trước hết, ta phải xác định cơ sở dữ liệu sẽ được đồng bộ hoá như thế nào và cơ sở dữ liệu gốc sẽ chứa ở đâu. Những vấn đề đặc biệt liên quan đến cách thức để những người sử dụng ở xa có thể nối mạng đến thiết kế gốc (qua mạng WAN, qua mạng điện thoại, thậm chí qua Internet). Điều quan trọng là có kế hoạch trước trong nỗ lực duy trì khả năng truy cập đến những phiên bản mới nhất cho hầu hết (nếu không phải tất cả) người sử dụng.

Sau khi có kế hoạch về cấu trúc của hệ cơ sở dữ liệu hỗ trợ sao chụp, ta bắt tay vào thiết lập cấu trúc đó, gồm các bước sau:

1. Tạo một cơ sở dữ liệu (hoặc dùng cơ sở dữ liệu hiện hành)
2. Chọn cơ sở dữ liệu làm *Bản thiết kế gốc.*
3. Tạo một hay nhiều bản sao từ *Bản thiết kế gốc.*

Ta có thể dùng Microsoft Access hoặc lập trình để tiến hành thiết lập ***replication,*** thông qua **DAO (*Đối tượng truy cập dữ liệu –* *Data Access Object).***

i. Dùng Microsoft Access để tạo Bản thiết kế gốc và bản sao

Trong khi ta có thể tiến hành ***Replication*** dùng chương trình của DAO, Microsoft Access là cách dễ hơn để bắt đầu *Replication.* Bởi vì Access có các lệnh menu quản lý việc tạo các ***Bản thiết kế gốc*** và bản sao cơ sở dữ liệu tự động.

Để tạo một *Bản thiết kế gốc* dùng Microsoft Access, ta cần cài đặt tính năng gọi là ***Sao chụp cặp đựng tài liệu-****Briefcase Replication.* Đây là tuỳ chọn lúc ta cài đặt Access; còn nếu chưa có sẵn, Access sẽ thông báo những gì cần làm khi ta tạo *Bản thiết kế gốc,* gồm các bước sau :

1. Chắc chắn rằng cần phải tạo ***bản sao dự phòng*** *(backup)* cơ sở dữ liệu mà không bàn cãi. Ví dụ, nếu ta đang làm việc với cơ sở dữ liệu tên là novelty.mdb, ta có thể đặt tên nó là *nmaster.mdb.*
2. Mở cơ sở dữ liệu trong Microsoft Access
3. Từ menu *Tools,* chọn ***Replication.*** Chọn ***Create Replica*** từ menu con.
4. Access phát ra một cảnh báo rằng cơ sở dữ liệu cần được đóng trước khi tạo bản sao từ đó. Nó hỏi có muốn đóng cơ sở dữ liệu và chuyển đổi nó thành *Bản thiết kế gốc* hay không. Chọn ***Yes.***
5. Cơ sở dữ liệu đóng. Kế tiếp, Access cảnh báo nên tạo một *bản sao dự phòng* cho cơ sở dữ liệu và hỏi có muốn thực hiện điều này trước khi tiến hành quy trình sao chụp không. Chọn ***Yes.***
6. Access chuyển đổi cơ sở dữ liệu thành *Bản thiết kế gốc* và tạo một bản sao từ đó, nó yêu cầu ta đặt tên cho bản sao cơ sở dữ liệu. Nếu ta đang dùng cơ sở dữ liệu Novelty, ta có thể đặt tên cho bản sao là *nreplica.mdb.*
7. Khi việc sao chụp thành công, Access hiển thị một thông báo giải thích những gì đã thực hiện.

ii. Thêm các đối tượng hỗ trợ sao chụp vào cơ sở dữ liệu trong Microsoft Access

Ban đầu, khi thêm một đối tượng mới (như một bảng hoặc truy vấn) vào cơ sở dữ liệu *Bản thiết kế gốc,* bởi mặc định, nó không được sao chụp. Trong Microsoft Access, điều này là hiển nhiên bởi vì các loại đối tượng cơ sở dữ liệu khác nhau (hỗ trợ sao chụp và không sao chụp) có các biểu tượng khác nhau trong cửa sổ Database của Access.

Muốn biến một đối tượng cơ sở dữ liệu mới thành đối tượng dữ liệu hỗ trợ sao chụp ta làm như sau:

1. Trong cửa sổ ***Database***, nhấn nút phải chuột trên đối tượng cơ sở dữ liệu mà ta muốn thay đổi trạng thái sao chụp của nó.
2. Từ menu bật ra, chọn *Properties.*
3. Hộp thoại ***Database Object Properties*** xuất hiện.
4. Chọn vào hộp đánh dấu ***Replicable***, nhấn OK. Biểu tượng của đối tượng cơ sở dữ liệu thay đổi phản ánh trạng thái mới của nó là một đối tượng hỗ trợ sao chụp. Sau này, khi tiến hành đồng bộ hoá cơ sở dữ liệu, đối tượng cơ sở dữ liệu mới được copy vào bản sao cơ sở dữ liệu.

Với kỹ thuật này, ta có thể thay đổi trạng thái sao chụp của một đối tượng cơ sở dữ liệu bất kỳ tại một thời điểm bất kỳ. Để biến đối tượng thành cấm sao chụp, đơn giản ta chỉ cần bỏ chọn trong hộp đánh dấu ***replicable*** trong hộp thoại *Database Object Properties.* Tuy nhiên, lưu ý rằng, khi ta chuyển một đối tượng cơ sở dữ liệu đặc biệt thành cấm sao chụp, những đối tượng được copy trước đó vào một bản sao cơ sở dữ liệu sẽ bị xoá khi tiến hành đồng bộ hoá.

17.1.1.3.4.3 Tiến hành đồng bộ hoá (Synchronization) với Microsoft Access

Sau khi tạo cơ sở dữ liệu hỗ trợ sao chụp trong Microsoft Access, ta có thể kiểm nghiệm nó. Cách đơn giản nhất là đưa vào một mẩu tin trong *Bản thiết kế gốc,* sau đó đồng bộ hoá cơ sở dữ liệu. Khi đó, mẩu tin mới được copy vào bản sao :

1. Mở cơ sở dữ liệu *Bản thiết kế gốc* và đưa một mẩu tin vào một trong những bảng. Nếu ta dùng ***Replication ID*** làm kiểm dữ liệu của khoá chính, ta nên chú ý rằng ID sẽ được phát sinh tự động (*AutoNumber)* khi nhập mẩu tin.
2. Đóng bảng. Đối tượng cơ sở dữ liệu phải được đóng để tiến hành ***đồng bộ hoá*** (*synchronization).*
3. Từ menu *Tools* của Microsoft Access, chọn *Replication.* Chọn ***Synchronize*** từ menu con.
4. Hộp thoại ***Synchronize Database*** xuất hiện, hiển thị tên của cơ sở dữ liệu bản sao.

   Lưu ý rằng *Synchronization Database* cho phép ta chọn một bản sao hiện hành hoặc chọn từ danh sách các bản sao (dùng danh sách thả xuống của hộp kết hợp). Vì ở đây ta chỉ mới tạo một bản sao, nên ta có thể chấp nhận tên tập tin đề nghị và nhấn OK.
5. Sau một thoáng dừng, Access thông báo rằng đồng bộ hoá đã hoàn tất. nó cũng hỏi ta muốn đóng hay mở lại cơ sở dữ liệu hiện hành để bảo đảm rằng tất cả dữ liệu được đồng bộ hoá đều được hiển thị. Ở đây, bước thực

hiện này không cần thiết vì không có dữ liệu mới được copy từ bản sao vào *Bản thiết kế gốc.*

6. Để xác nhận rằng mẩu tin mới đã được truyền từ *Bản thiết kế gốc* vào bản sao, đóng cơ sở dữ liệu *Bản thiết kế gốc* và mở bản sao. Ta sẽ thấy rằng dữ liệu đưa vào *Bản thiết kế gốc* đã được copy vào bản sao.

Ta có thể đảo ngược lại quá trình với việc thêm, sửa ở bản sao và đồng bộ hoá lại với bản gốc.

17.1.1.3.4.4 TIẾN HÀNH *REPLICATION* VỚI DAO

Trong Visual Basic, ta có thể điều khiển cách thức sao chụp cơ sở dữ liệu dùng **DAO** (***Đối tượng truy cập dữ liệu –****Data Access Objects).* DAO là một kỹ thuật hướng đối tượng cho phép truy cập cơ sở dữ liệu. Khi ta dùng **DAO** để điều khiển *Replication,* ta sẽ tăng cường khả năng để kiểm soát cách thức thiết lập cơ sở dữ liệu hỗ trợ sao chụp, định cấu hình và đồng bộ hoá bằng chương trình. Sau đây là các đối tượng DAO chứa những thuộc tính hỗ trợ *Replication:*

Chức năng *Replication*	Đối tượng	Mô tả
Thuộc tính ***KeepLocal***	***TableDef*** và ***QueryDef*** (cũng như các đối tượng Microsoft Access khác như là biểu mẫu và báo cáo)	Xác định đối tượng có được tạo để hỗ trợ sao chup hay không
Thuộc tính ***Replicable***	Đối tượng ***Database, TableDef*** và ***QueryDef*** (cũng như các đối tượng cơ sở dữ liệu Access như báo cáo và mô-dun chương trình).	Xác định đối tượng (và cho ***TableDefs***, dữ liệu mà nó chứa) có được sao chụp vào lúc đồng bộ hoá hay không.
Phương thức ***MakeReplica***	***Database***	Tạo một bản sao từ *Bản thiết kế gốc.*
Phương thức ***Synchronize***	***Database***	*Đồng bộ hoá* một bản sao cơ sở dữ liệu với *Bản thiết kế gốc.*
Thuộc tính ***ReplicaFilter***	***TableDef***	Cho phép cung cấp một mệnh đề WHERE trong câu SQL để kiểm soát các mẩu tin trong bảng được sao chụp (thuộc tính này hỗ trợ *Sao chụp một phần).*
Thuộc tính ***PartialReplica***	***Relation***	Đối với *Sao chụp một phần,* nó cho phép ta xác định những quan hệ nào kiểm soát việc sao chụp mẩu tin.
Thuộc tính ***ReplicableBool***	***Database***	Tương đương với thuộc tính ***Replicable,*** nhưng dễ cài hơn.

i. Thiết lập Replication với DAO

Để bắt đầu dùng ***Replication*** với lập trình DAO, ta phải cho phép cơ sở dữ liệu hỗ trợ *Replication.* Để thực hiện điều này, thêm một thuộc tính động vào đối tượng *Database* thể hiện cơ sở dữ liệu. Khái niệm thuộc tính động chủ yếu để hiểu cách thực hiện *Replication* với DAO; các thuộc tính hiệu chỉnh cho phép mở rộng mô hình đối tượng của DAO.

Các thuộc tính động của cơ sở dữ liệu Jet được giới thiệu trong chương 13 "Đối tượng truy cập dữ liệu ". Dùng DAO, ta có thể lập trình để xác định các thuộc tính của một đối tượng cơ sở dữ liệu đặc biệt. Ví dụ sau đây trình bày liệt kê tất cả các thuộc tính của một bảng trong *Bản thiết kế gốc.*

```
Option Explicit

' References DAO 3.51.

Dim db As Database
Dim pr As Property
Dim td As TableDef

Public Sub ListProps()
   Set db = OpenDatabase("..\..\DB\nmaster.mdb")
   Set td = db.TableDefs("tblCustomer")

   For Each pr In td.Properties
      Debug.Print pr.Name
   Next
End Sub
```

Biến một cơ sở dữ liệu thành cơ sở dữ liệu hỗ trợ sao chụp bằng cách tạo một thuộc tính ***Replicable,*** sau đó quy định giá trị cho nó là chuỗi "T". Biến một cơ sở dữ liệu thành cơ sở dữ liệu hỗ trợ sao chụp sẽ chuyển đổi nó thành *Bản thiết kế gốc;* điều này khác với khi tạo các đối tượng riêng rẽ ở trong cơ sở dữ liệu hỗ trợ sao chụp. Làm việc với **DAO** khác với khi ta làm việc với Microsoft Access, bởi vì tự Access đã thực hiện việc sao chụp bên trong. Khi ta yêu cầu Access tạo một bản sao, nó tự làm thêm việc chuyển cơ sở dữ liệu thành cơ sở dữ liệu hỗ trợ sao chụp trên đối tượng cơ sở dữ liệu mà ta chọn.

Dùng chương trình viết với DAO để biến cơ sở dữ liệu thành cơ sở dữ liệu hỗ trợ sao chụp :

```
Option Explicit

' References DAO 3.51.

Dim db As Database
Dim pr As Property
Dim td As TableDef

Const DBPath = "..\..\DB\novelty.mdb"
Const MasterDBPath = "..\..\DB\nmaster.mdb"
Const ReplicaDBPath = "..\..\DB\nreplica.mdb"
```

```
Private Sub cmdMakeDBRep_Click()
On Error GoTo ErrHandler

   ' The "True" parameter in the OpenDatabase method
   ' tells Jet to open the database for exclusive
   ' access, which is required for creating properties.

   FileCopy DBPath, MasterDBPath

   Set db = OpenDatabase(MasterDBPath, True)

   With db
      Set pr =.CreateProperty("Replicable", dbText, "T")
     .Properties.Append pr
     .Properties("Replicable") = "T"
   End With

   MsgBox "The database has been copied to " & _
        db.Name & _
        " and its Replicable property is now " & _
        db.Properties("Replicable").Value

   db.Close
   Set db = Nothing    ' Release exclusive lock on db.

   Exit Sub
ErrHandler:
   Select Case Err.Number
      Case 3367       ' Replicable property already exists
      Exit Sub        ' So ignore the error and exit

      Case Else       ' Something unforseen happened
      MsgBox "Error: " & Err & " - " & Error

   End Select

End Sub
```

Lưu ý rằng đoạn chương trình này sẽ chạy được dù cho cơ sở dữ liệu hiện hành có thuộc tính động ***Replicable*** hay không. Nếu cơ sở dữ liệu hiện hành có thuộc tính *Replicable,* chương trình sẽ không tạo ra thuộc tính này nữa.

Tuy nhiên, có một rắc rối khi tạo một thuộc tính *Replicable* cho một cơ sở dữ liệu – Sau khi tạo ra, ta không thể loại bỏ nó. Thử dùng đoạn chương trình sau để loại bỏ :

```
db.Properties.Delete “Replicable”
```

Nhưng không thực hiện được. Đoạn chương trình phát ra báo lỗi số 3607, thông báo rằng đây là thuộc tính chỉ đọc và không thể loại bỏ. Vì vậy, như đã cảnh báo ở

phần trước, việc tạo bản dự phòng cho cơ sở dữ liệu là điều đặc biệt quan trọng khi ta thao tác với DAO để thực hiện *Replication.*

ii. Tạo đối tượng trong cơ sở dữ liệu hỗ trợ sao chụp

Sau khi tạo cơ sở dữ liệu hỗ trợ sao chụp, ta phải tạo các đối tượng riêng rẽ (như bảng) trong cơ sở dữ liệu này. Thực hiện điều này bằng cách tạo đối tượng ***Replicable*** cho chúng, tương tự như khi ta thực hiện với cơ sở dữ liệu. Các đối tượng cơ sở dữ liệu có thuộc tính *Replicable* là "T" sẽ được sao chụp đồng bộ hoá.

Để thực hiện điều này, ta dùng đoạn chương trình sau đây. Giả sử rằng đây là cơ sở dữ liệu hỗ trợ sao chụp; ta lấy một bảng hiện hành trong cơ sở dữ liệu và biến nó thành bảng hỗ trợ sao chụp.

```
Private Sub cmdMakeTable_Click()
On Error GoTo ErrHandler

   Set db = OpenDatabase(MasterDBPath, True)
   Set td = db.TableDefs("tblCustomer")
   td.Properties("Replicable") = "T"

   On Error GoTo 0

   MsgBox "The Replicable property of " & _
        td.Name & _
        " has been set to " & _
        td.Properties("Replicable")

   Set db = Nothing    ' Release exclusive lock on DB

   Exit Sub

ErrHandler:

   If Err.Number = 3270 Then
      Set pr = td.CreateProperty("Replicable", dbText, "T")
      td.Properties.Append pr
   Else
      MsgBox "Error " & Err & " - " & Error
   End If

End Sub
```

Đoạn chương trình này tương tự đoạn chương trình thiết lập thuộc tính ***Replicable*** của cơ sở dữ liệu. Ở đây, bẫy lỗi đảmnhiệm tình huống thuộc tính *Replicable* chưa được tạo cho đối tượng. Đoạn chương trình bẫy lỗi để xác định thuộc tính *Replicable* của đối tượng cơ sở dữ liệu có tồn tại không; nếu chưa có, nó tạo ra thuộc tính đó và quy định thuộc tính hiệu chỉnh là chuỗi "T". Giá trị "T"

làm cho đối tượng (và các dữ liệu chứa bên trong nó) được copy vào bản sao cơ sở dữ liệu vào lúc đồng bộ hoá.

Giá trị "T" của thuộc tính *Replicable* của đối tượng cơ sở dữ liệu tự động đổi thuộc tính *KeepLocal* thành "F".

iii. Dùng thuộc tính ReplicableBool để biến đối tượng thành đối tượng hỗ trợ sao chụp

Chú ý rằng các thuộc tính hiệu chỉnh ta dùng đến giờ đều có giá trị là ký tự - ký tự T hay F thể hiện giá trị Boolean là *True* hay *False*. Bởi vì khi chúng được giới thiệu cho Jet, các thuộc tính cơ sở dữ liệu hiệu chỉnh chỉ có thể chứa giá trị là ký tự. Trong Jet 3.51, các thuộc tính hiệu chỉnh có thể chứa một tập hợp rất phong phú gồm nhiều kiểu dữ liệu. Nhờ đó, nó giúp ta tạo các thuộc tính gắn liền với *Replication* trong DAO một cách dễ dàng hơn nếu như ta thích dùng *True / False* thay vì T / F.

Nếu muốn thuộc tính *Replicable* của đối tượng cơ sở dữ liệu dùng giá trị Boolean, ta tạo và định thuộc tính *ReplicableBool* thay vì *Replicable.ReplicatonBool* dùng giá trị Boolean, thay vì chuỗi ký tự.

```
Private Sub cmdMakeTableBool_Click()
On Error GoTo ErrHandler
   Set db = OpenDatabase(MasterDBPath, True)
   Set td = db.TableDefs("tblCustomer")
   td.Properties("ReplicableBool") = True
   On Error GoTo 0
      MsgBox "The Replicable property of " & _
        td.Name & _
        " has been set to " & _
        td.Properties("Replicable")
   Set db = Nothing    ' Release exclusive lock on DB
   Exit Sub
ErrHandler:
   If Err.Number = 3270 Then
      Set pr = td.CreateProperty("ReplicableBool", dbBoolean, True)
      td.Properties.Append pr
   Else
      MsgBox "Error " & Err & " - " & Error
   End If
End Sub
```

Khi tạo và dùng thuộc tính *ReplicableBool,* thuộc tính *Replicable* cũng trở thành có sẵn, và 2 thuộc tính trả về cùng giá trị, nghĩa là nếu ta quy định *ReplicableBool* là *True*, thì đọc thuộc tính *Replicable,* ta thấy nó là "T".

iv. Lập trình với DAO để tạo một cơ sở dữ liệu bản sao

Tạo một cơ sở dữ liệu bản sao từ *Bản thiết kế gốc* bằng cách thi hành phương thức *MakeReplica* của đối tượng *Database.*

Cú pháp :

db.MakeReplica *<tên tập tin>,[<mô tả>], [<tùy chọn>]*

<Tên tập tin> là tập tin bản sao cơ sở dữ liệu mới mà ta muốn tạo.
<Mô tả> là chuỗi ký tự của bản sao mới. Tham số này là tuỳ chọn.
Tham biến *<Tuỳ chọn>* có thể là một hay hai giá trị. Tuỳ chọn thứ nhất, *dbRepMakePartial,* tạo bản sao một phần, nghĩa là ta có thể kiểm soát những mẩu tin nào được copy từ Bản thiết kế gốc vào bản sao. Tuỳ chọn thứ hai, *dbRepMakeReadOnly,* cho phép tạo bản sao chỉ được đọc với những người sử dụng bản sao. (Tuy nhiên, ta có thể vẫn gửi dữ liệu và các đối tượng dữ liệu mới đến bản sao thông qua đồng bộ hoá).

```
Private Sub cmdSpawn_Click()
  Dim db As Database
  Set db = OpenDatabase(MasterDBPath, True)
  db.MakeReplica ReplicaDBPath, "MyReplica"
  db.Close
  Set db = Nothing
End Sub
```

v. Lập trình với DAO để tiến hành đồng bộ hoá

1. Trong chương trình, tạo một đối tượng *Database* thể hiện bản sao cơ sở dữ liệu.
2. Thi hành phương thức *Synchronize* của đối tượng *Database* để tiến hành đồng bộ hoá bản sao với *Bản thiết kế gốc.* Phương thức *Synchronize* dùng tên tập tin của cơ sở dữ liệu bản thiết kế gốc làm tham biến.

Trong đoạn chương trình sau, *nreplica.mdb* là bản sao, *nmaster.mdb* là *Bản thiết kế gốc*.

```
Private Sub cmdSynch_Click()
  Dim db As Database
  Set db = OpenDatabase(MasterDBPath)
 Screen.MousePointer = vbHourglass
  db.Synchronize ReplicaDBPath
```

```
    Screen.MousePointer = vbNormal
End Sub
```

Thuộc tính *MousePointer* của đối tượng *Screen* dùng để báo cho người sử dụng rằng *Replication* đang tiến hành.

vi. Sao chụp một phần

Thông thường, ta không muốn truyền toàn bộ dữ liệu từ *Bản thiết kế gốc* đến bản sao. Nhất là khi việc này không thích hợp và không thực tế khi phải gửi toàn bộ nội dung của cơ sở dữ liệu gốc đến người sử dụng ở xa.

Những gì không nên truyền đi là những thông tin bí mật trong kinh doanh như lương bổng, cũng như những thông tin bảo mật của cơ sở dữ liệu nhưu ID và mật khẩu người sử dụng. Cơ sở dữ liệu ta cần là những dữ liệu gắn liền với cá nhân một người sử dụng sẽ được sao chụp vào máy tính của họ. Kiểu sao chụp này hiệu quả hơn kiểu sao chụp toàn phần bởi vì nó chỉ tạo bản sao của những dữ liệu mà người sử dụng cần đến.

Một bản sao một phần là một cơ sở dữ liệu không sao chụp toàn bộ dữ liệu của *Bản thiết kế gốc.* Để tạo một bản sao một phần, ta theo các bước sau:

1. Tạo một bản sao một phần dùng phương thức *MakeReplica* của đối tượng *Database,* chỉ ra tuỳ chọn *dbRepMakePartial.*
2. Quy định thuộc tính *ReplicaFilter* của đối tượng *TableDef* là những mẩu tin được copy vào bản sao một phần. *ReplicaFilter* cho ta quy định mệnh đề Where của câu SQL đến số mẩu tin cần copy vào bản sao một phần từ cơ sở dữ liệu nguồn.
3. Ngoài ra, có thể quy định thêm thuộc tính *PartialReplica* của đối tượng *Relation* là số mẩu tin được copy vào bản sao dựa trên kết nối giữa 2 bảng.
4. Thi hành phương thức *PopulatePartial* của đối tượng *Database* để copy dữ liệu từ *Bản thiết kế gốc* vào bản sao một phần.

Đối với bản sao một phần, ta có thể lọc ra các mẩu tin, nhưng không thể lọc ra các trường. Để hạn chế số cột hiển thị đối với người sử dụng, ta xem xét bảo mật cơ sở dữ liệu để hạn chế quyền truy cập đối tượng cơ sở dữ liệu.

17.1.1.3.4.5 Dùng phương thức MakeReplica để tạo bản sao một phần

Ta có thể tạo bản sao một phần từ *Bản thiết kế gốc* hoặc từ một bản sao toàn phần khác. (Ta không thể tạo một bản sao một phần từ một bản sao một phần khác).

```
Option Explicit
' References DAO 3.51
Private db As Database
Private td As TableDef
Const MasterDBPath = "..\..\DB\nmaster.mdb"
Const ReplicaPath = "..\..\DB\npartial.mdb"
```

```
Private Sub cmdMakePartial_Click()
  Set db = OpenDatabase(MasterDBPath)
  db.MakeReplica ReplicaPath, "Partial", dbRepMakePartial
  db.Close
  Set db = Nothing
End Sub
```

Bản sao một phần tạo ra từ chương trình trên đây chỉ mới chứa cấu trúc cơ sở dữ liệu bản sao, chưa có dữ liệu.

Khi tạo một bản sao một phần, ta không thể chuyển đổi nó thành bản sao toàn phần. Tuy nhiên, có thể sao chụp toàn bộ dữ liệu trong *Bản thiết kế gốc* vào bản sao bằng cách quy định thuộc tính *ReplicaFilter* là *True.*

17.1.1.3.4.6 Tiến hành sao chụp một phần

Sau khi tạo một bản sao một phần, ta có thể copy các mẩu tin từ bản sao toàn phần hoặc *Bản thiết kế gốc* theo các bước sau đây :

1. Trong chương trình sử dụng DAO, khai báo và quy định giá trị cho biến đối tượng *Database* là cơ sở dữ liệu bản sao một phần.
2. Khai báo biến *TableDef* cho các bảng trong bản sao một phần để chứa dữ liệu sao chụp.
3. Quy định thuộc tính *ReplicaFilter* của từng đối tượng *TableDef* là một điều kiện WHERE của câu SQL. Nó xác định các mẩu tin sẽ được copy vào bản sao một phần.
4. Thi hành phương thức *PopulatePartial* của đối tượng *Database,* chỉ ra đường dẫn và tên tập tin của *bản thiết kế gốc* hoặc bản sao toàn phần mà ta muốn copy từ đó.

Có thể dùng thuộc tính *ReplicaFilter* của đối tượng *TableDef* để sao chụp tập con các mẩu tin từ cơ sở dữ liệu đến cơ sở dữ liệu bản sao một phần. Thuộc tính *ReplicaFilter* có thể chứa một trong 3 giá trị :

Nếu *ReplicaFilter* là *True,* toàn bộ các mẩu tin trong cơ sở dữ liệu nguồn được copy vào cơ sở dữ liệu bản sao một phần.

Nếu *ReplicaFilter* là *False,* không có mẩu tin nào từ cơ sở dữ liệu nguồn được copy vào bản sao một phần.

Nếu *ReplicaFilter* là một chuỗi, bộ máy cơ sở dữ liệu xem đó là mệnh đề WHERE của câu SQL.

Để gán một bộ lọc bản sao và copy các mẩu tin từ *Bản thiết kế gốc* hoặc bản sao toàn phần vào bản sao một phần, dùng đoạn chương trình sau :

```
Option Explicit
' References DAO 3.51
Private db As Database
```

```
Private td As TableDef
Const MasterDBPath = "..\..\DB\nmaster.mdb"
Const ReplicaPath = "..\..\DB\npartial.mdb"
Private Sub cmdMakePartial_Click()
   Set db = OpenDatabase(MasterDBPath)
   db.MakeReplica ReplicaPath, "Partial", dbRepMakePartial
   db.Close
   Set db = Nothing
End Sub

Private Sub cmdReplicate_Click()
   ' Open partial replica in exclusive mode
   Set db = OpenDatabase(ReplicaPath, True)
   Set td = db.TableDefs("tblCustomer")
   td.ReplicaFilter = "State = 'CA'"
   ' Populate with data from design master
   db.PopulatePartial MasterDBPath
   ' Release exclusive lock on database
   Set db = Nothing
End Sub
```

Khi ta thi hành đoạn chương trình trên, chỉ những khách hàng sống ở tiểu bang California được copy vào bản sao một phần.

TỔNG KẾT

Chương này đề cập 2 vấn đề chủ yếu khi phân phát dữ liệu qua mạng đến nhiều người sử dụng. Trong phần đầu, bạn đã tìm hiểu cách thức sử dụng các thành phần ActiveX để truy cập dữ liệu theo hướng đối tượng. Nó cũng thảo luận về triển khai DCOM với thành phần ActiveX, cho phép ta triển khai các đối tượng kinh doanh qua mạng LAN.

Trong phần thứ hai, ta tìm hiểu về cách thức sao chụp cơ sở dữ liệu Jet qua mạng để phân phát dữ liệu qua 2 hay nhiều tập tin cơ sở dữ liệu.

Mặc dù chủ đề về các thành phần tầng trung gian ActiveX và sao chụp cơ sở dữ liệu nói ở 2 phần riêng, nhưng không có nghĩa là chúng loại trừ lẫn nhau. Trong chương trình, bạn có thể phối hợp cả hai kỹ thuật để phân phát dữ liệu theo diện xa và rộng. Dùng kỹ thuật nào là tuỳ thuộc vào cấu trúc chương trình, số người sử dụng cần hỗ trợ, và mức độ yêu cầu giữ cho dữ liệu nhất quán.

HỎI VÀ ĐÁP

***Hỏi** : Tôi vừa tạo các thư viện ActiveX DLL dùng để truy cập dữ liệu. Tôi có cần biên dịch lại thành Active EXE để truy cập chúng qua mạng từ xa không ?*

Đáp: Không. Nếu bạn dùng *Microsoft Transaction Server,* bạn có thể đóng gói các thư viện *ActiveX DLL* ngay và truy cập chúng từ xa. Triển khai các thành phần từ xa dùng kỹ thuật MTS đã được trình bày ở đầu chương.

***Hỏi** : Khi tôi đang tiến hành sao chụp cơ sở dữ liệu, hệ thống cơ sở dữ liệu chủ yếu là off-line. Vậy làm sao để biết nó sẽ tốn bao lâu ?*

***Đáp**:* Bởi vì có rất nhiều yếu tố ảnh hưởng, lượng dữ liệu ta cần thao tác, băng thông trên mạng, tốc độ máy tính xử lý các mẩu tin. Nếu gặp phải rắc rối này khi tiến hành sao chụp cơ sở dữ liệu, bạn có thể thử quá trình tự động nếu được (viết một ứng dụng Visual Basic để đồng bộ hoá dữ liệu lúc cơ sở dữ liệu không được sử dụng nhiều – như vào ban đêm chẳng hạn). Bạn cũng có thể thử dùng bản sao một phần để giảm thiểu lượng dữ liệu được copy.

***Hỏi**: Bởi vì sao chụp một phần cho phép tuỳ chọn không sao chụp, hoặc sao chụp một phần hoặc sao chụp toàn bộ mẩu tin; trong thực tế, ta không thể chuyển đổi qua lại giữa sao chụp một phần và toàn phần, tại sao không để tất cả chỉ là sao chụp một phần ?*

***Đáp**:* Bạn chỉ có thể đồng bộ hoá giữa *Bản thiết kế gốc* và bản sao toàn phần, không cho phép đồng bộ hoá giữa bản sao một phần và bản sao khác. Điều này cung cấp sự linh hoạt trong những trường hợp mà bạn muốn đồng bộ hoá nhiều bản sao với nhau để giảm bớt áp lực phải xử lý trên máy tính chứa *Bản thiết kế gốc.*

17.2

18 Đối tượng dữ liệu ActiveX

Xây dựng ứng dụng Visual basic với ADO
Sử dụng dịch vụ dữ liệu từ ADO

Cho đến Visual basic 5.0, ADO (Dữ liệu đối tượng ActivateX - ActivateX Data Object) trở thành nền tảng của ký thuật truy cập dữ liệu Internet. Trong Visual basic 6.0, ADO 2.0 cang quan trọng hơn - mạnh mẽ hơn. Ta có thể dùng ADO không chỉ để truy cập cơ sở dữ liệu thông qua trang web, mà còn có thể dùng nó để lấy dữ liệu từ ứng dụng viết bằng Visual basic. ADO là giao diện dựa trên đối tượng cho công nghệ dữ liệu mới nổi gọi là OLE DB.

OLE DB được thiết kế để thay thế ODBC như một phương thức truy cập dữ liệu. ODBC hiện thời là tiêu chuẩn phía Client sử dụng Windows rất phổ biến để truy cập dữ liệu quan hệ bởi vì nó thiết lập các Server cơ sở dữ liệu quan hệ càng tổng quát càng tốt đến các ứng dụng Client. OLE DB đi sâu hơn một bước, bằng cách làm cho tất cả nguồn dữ liệu trở thành tổng quát đối với ứng dụng Client.

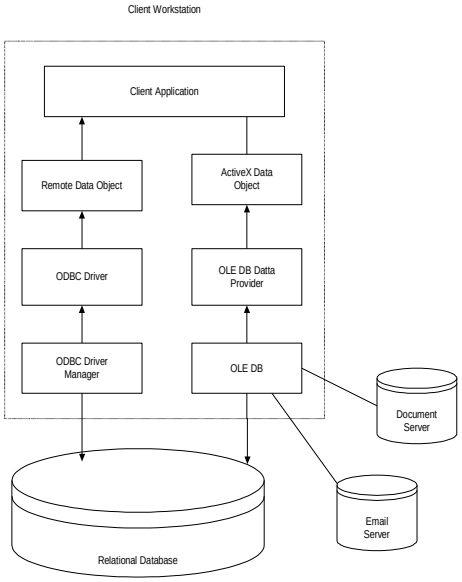
18.1 Xây dựng ứng dụng Visual basic với ADO

ADO là công nghệ truy cập cơ sở dữ liệu hướng đối tượng tương tự như DAO và RDO.

ADO hiện nay được Microsoft xem là kỹ thuật để truy cập cơ sở dữ liệu từ Web server. Bởi vì ADO được cung cấp dưới dạng thư viện ActivateX Server (tương tự DAO và RDO), ta có thể thoải mái dùng ADO trong ứng dụng Visual basic. Trong thực tế, bằng nhiều cách, ta sẽ thấy rằng sử dụng ADO để làm việc với cơ sở dữ liệu Client/Server thì dễ hơn các kỹ thuật khác.

18.1.1 Tìm hiểu cấu trúc OLE DB / ADO

Phần lớn các nhà lập trình viên Visual basic không thao tác trực tiếp với OLE DB. Thay vào đó, họ lập tình với ADO, mô hình đối tượng cung cấp giao diện với OLE DB.



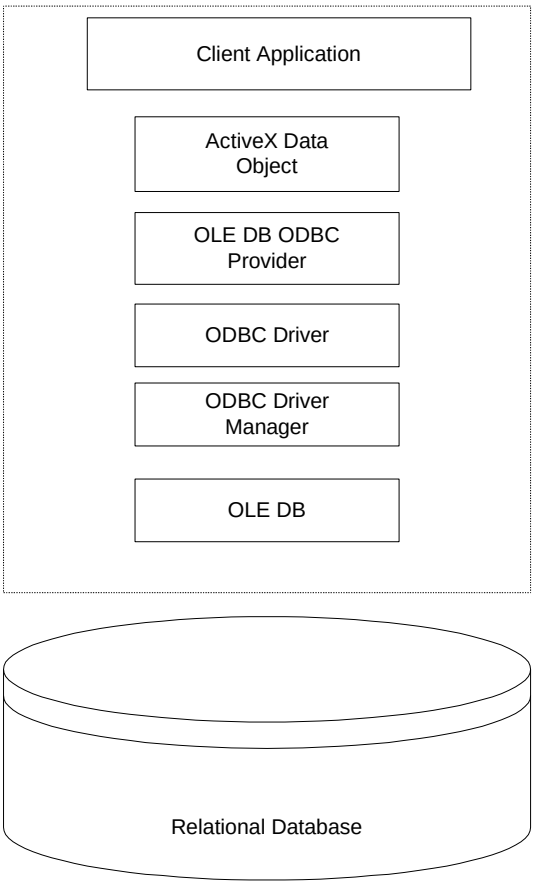
Hình: Sử dụng ADO và OLE DB để tăng cường truy cập thông tin trong một cơ sở dữ liệu

Trình cung cấp OLE DB không nhiều như trình cung cấp ODBC nhưng số lượng này đã tăng lên đáng kể từ khi ADO 2.0 được phát hành vào năm 1998. Phiên bản này đã được đưa vào Visual basic 6.0 bao gồm các trình cung cấp cục bộ cho SQL Server, Oracle và Microsoft Jet/ access.

Có nhiều khả năng bạn có thể dùng ADO và OLE DB để đạt được nguồn dữ liệu quan hệ ngay khi không có trình cung cấp OLE DB cụ bộ. Bởi vì đã có một tình cung cấp OLE DB tổng quát cho cơ sở dữ liệu quan hệ ODBC.

Ta chỉ cần lập trình với phần giao diện Người sử dụng ở phía Client. Bởi vì việc truy cập dữ liệu trên cả trình duyệt Web và ứng dụng Visual basic được chuyển hết về phía ActivateX Server, ta có thể bảo đảm rằng logic chương trình luôn nhất quán, bất kể loại ứng dụng nào đang được dùng.

Sau đây là cấu trúc truy cập cơ sở dữ liệu ODBC dùng trình cung cấp ODBC OLE DB.



Cấu trúc này cho phép ta dùng thành phần lập trình ActivateX thông dụng trên cả trình duyệt Web và ứng dụng Client Visual basic.

18.1.2 Cài đặt và thiết lập tham chiếu đến ADO trong ứng dụng Visual basic

ADO được cài đặt như một phần của Visual basic 6.0

Phiên bản mới nhất của của ADO cho phép tải xuống miễn phí từ địa chỉ *http://www.Microsoft.com/data/ado*.

Sau khi cài đặt xong, ta bắt đầu sử dụng nó bằng cách thiết lập tham chiếu đến thư viện ADO trong ứng dụng Visual basic, tương tự khi ta thiết lập tham chiếu đến thư viện DAO hay RDO.

- Trong project Visual basic chọn references, hộp thoại references xuất hiện
- Chọn vào hộp đánh dấu "Microsoft ActivateX Data Objects 2.0 Library" rồi nhấn OK

Chú ý rằng khi tham chiếu ADO thì phải đảm bảo là không có tham chiếu đến thư viện "Microsoft ActivateX Data Objects 2.0 Recordset". Đây là phiên bản loại nhẹ của thư viện ADO được thiết kế để sử dụng phía Client. Nó chỉ hỗ trợ Recordset và Field.

18.1.3 Sử dụng ADO với các thư viện đối tượng truy cập dữ liệu khác

Nếu bạn tạo ứng dụng được thiết kế để sử dụng ADO kết hợp với thư viện đối tượng truy cập dữ liệu khác, như là DAO, cần phải phân biệt giữa, ví dụ như đối tượng RecordSet của DAO và Recordset của ADO. Chúng không thể đổi chỗ cho nhau được.

Nếu ta tham chiếu đến vừa ADO vừa DAO và khai báo một biến Recordset, làm thế nào để phân biệt Recordset của DAO hay ADO? Câu trả lời là thứ tự tham chiếu vào đề án. Nếu thêm tham chiếu đối tượng DAO trước thì Recordset là của DAO và ngược lại. Để tránh nhầm lẫn ta nên khai báo tường minh như sau: ADODB.Recorset

Nếu không muốn tham chiếu trực tiếp đến thư viện đối tượng trong chương trình, ta có cách khác. Ta có thể kiểm soát thư viện đối tượng nao được truy cập mặc định bằng cách dùng giá trị mực ưu tiên trong hộp References.

18.1.4 Dùng đối tượng connection của ADO để kết nối với nguồn dữ liệu

Vị trí của đối tượng Connection trong mô hình của ADO:(Xem hình dưới):

Dùng phương thức Open của đối tượng Connection để thiết lập kết nối với nguồn dữ liệu. Để thông báo cho ADO cách nối với nguồn dữ liệu ta phải cung cấp thông tin dưới dạng chuối kết nối của ODBC. Ta dùng thuộc tính ConnectionString để thực hiện điều này. Ta còn có khả năng tuỳ chọn để chọn trình cung cấpnào sẽ được dùng bằng cách quy định giá trị thuộc tính Provider của đối tượng Connection.

18.1.4.1 Chỉ ra trình cung cấp OLE DB và chuỗi kết nối

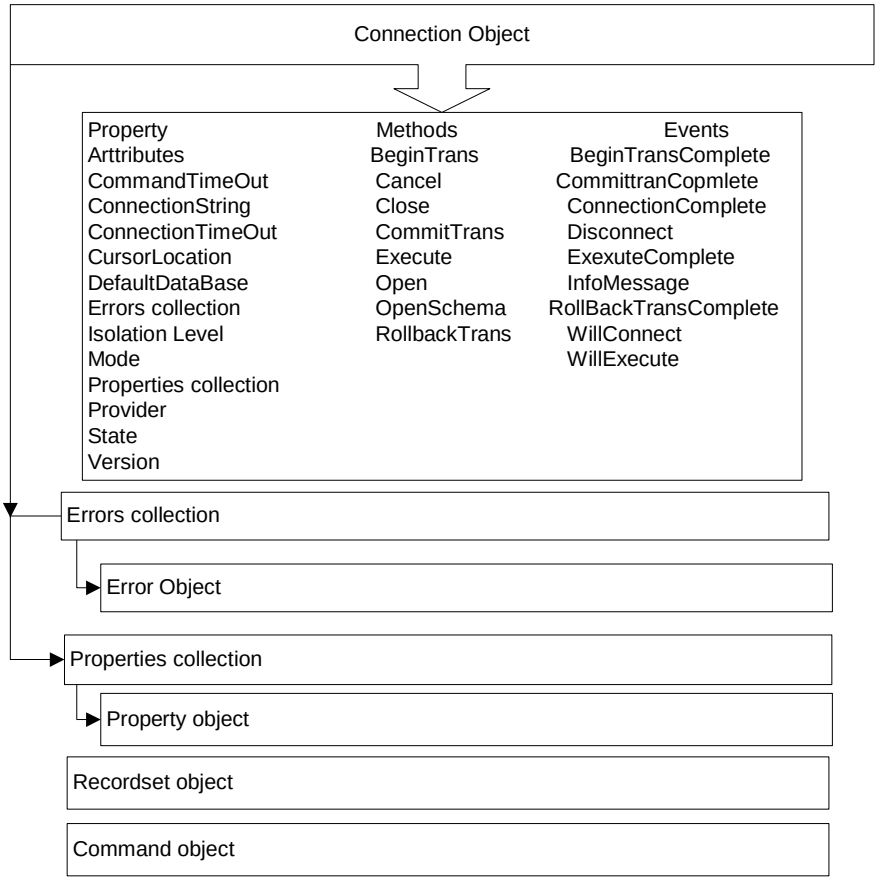
Nếu không chỉ ra trình cung cấp, hoặc ta không dùng đối tượng Connection, ta sẽ có một trình cung cấp mặc định, là trình cung cấp ODBC, MSDASQL.

Thuộc tính Provider của đối tượng Connection là chuỗi ký tự chỉ ra kết nối mà trình cung cấp OLE DB sẽ dùng.

Dùng chuỗi kết nối trong ADO để cung cấp thông tin về cách thức kết nối với Server cơ sở dữ liệu. Khi ta dùng trình cung cấp ODBC cho OLE DB, chuỗi kết nối tương tự chuỗi kết nối ODBC. Điều này có nghĩa là thông tin chính xác được mong

chờ bởi trình điều khiển ODBC có thể thay đổi tuỳ theo cách thực hiện. Đối với các trình cung cấp khác, chuỗi kết nối có thể có một cú pháp hoàn toàn khác.

Khi ta dùng trình cung cấp ODBC, thuộc tính ConnectionString có thể là một DSN (tên nguồn dữ liệu) hay nó là kết nối không có DSN. Đây là một ví dụ của một kết nối đến cơ sở dữ liệu dùng trình cung cấp ODBC với DSN:



Cn.Provider = "MSDASQL"

Cn.ConnectionString = "DSN = Novelty;"

Sử dụng DSN trong chuỗi kết nối dĩ nhiên yêu cầu một DSN tên là Novelty phải thực sự tồn tại trên máy Client.

Trường hợp kết nối không có DSN:

Cn.Provider = "MSDASQL"

Cn.ConnectionString = "DRIVER = {SQL Server}; DATABASE = Novelty; UID = sa ; PD = ;"

Kết nối này sẽ nối kết Server nhanh hơn bởi vì nó không cần đọc thông tin DSN từ bảng đăng ký của Windows. Tuy nhiên, nó kém linh hoạt vì nó gắn chặt thông tin với chương trình đã được biên dịch

18.1.5 Làm việc với con trỏ

Tương tự RDO và DAO, ADO hỗ trợ một số kiểu con trỏ. Ngoài việc cung cấp hỗ trợ duyệt qua từng bản ghi tại một thời điểm, các kiểu con trỏ khác nhau cho phép ta điều khiển cách quản lý của một Recordset.

Quy định vị trí của con trỏ bằng cách gán giá trị cho thuộc tính Recordset. Sau đây là các kiểu con trỏ của đối tượng Connection:

Kiểu con trỏ	Hằng	Mô tả

Phía Client	adUseClient	Tạo con trỏ phía Client
Phía Server	adUseServer	Tạo con trỏ phía Server

Chọn con trỏ kiểu Client nghĩa là ADO và OLE DB xử lý các hoạt động của con trỏ. Con trỏ Client thường không có sẵn trên server. Ví dụ, trong ADO, ta có thể tạo một Recordset không kết nối, cho phép ta thao tác với các bản ghi mà không có kết nối thường xuyên đến server. Khả năng này là một chức năng của thư viện con trỏ phía Client.

Trong ADO, thuộc tính CursorLocation áp dụng cho cả đối tượng Recordset và Connection. Nếu ta gán thuộc tính CursorLocation của đối tượng Connection, tất cả Recordset mà ta tạo từ kết nối đó đều có cùng vị trí con trỏ như đối tượng Connection.

Ngoài việc chỉ ra vị trí con trỏ, ta có khả năng tạo 4 kiểu con trỏ khác nhau trong ADO. Việc chọn lựa con trỏ tuỳ theo sự cân đối giữa chức năng và khả năng hoạt động.

Chỉ ra kiểu con trỏ bằng cách gán thuộc tính CursorType của đối tượng Recordset. Sau đây là các kiểu con trỏ có thể tạo trong ADO:

Kiểu con trỏ	Hằng	Mô tả
Forward-only	adOpenForwardOnly	Không dùng con trỏ - ta chỉ có thể chuyển về phía trước Recordset; sử dụng phương thức MovePrevious và MoveFirst sẽ sinh lỗi.
Keyset(Trong ADO được gọi là dynaset)	adOpenKeyset	Ta không thể tháy các bản ghi mới do người dùng khác thêm vào, nhưng khi họ sửa đổi hay xoá tin sẽ làm ảnh hưởng đến Recordset ta đang làm việc; đây là kiểu con trỏ hiệu quả nhất, đặc biệt là khi Recordset khá lớn.
Dynamic	adOpenDynamic	Ta có thể thấy toàn bộ thay đổi trên dữ liệu do những người sử dụng khác thực hiện trong khi ta đang mở Recordset; đây là kiểu con trỏ ít hiệu quả nhất nhưng mạnh mẽ.
Static(Trong DAO gọi là snapshot)	adOpenStatic	Bản sao của toàn bộ dữ liệu của một Recordset; kiểu này đặc biệt hữu dụng khi ta đang tìm kiếm dữ liệu hay khi thi hành báo cáo; kiểu con trỏ này rất hữu dụng cho các Recordset nhỏ.

Dĩ nhiên, lý do để ta chọn con trỏ kiểu forward-only thay vì keyset hay dynamic là khả năng hoạt động của - nếu ta chỉ hiển thị dữ liệu chứa trong cơ sở dữ liệu – con trỏ kiểu forward-only sẽ làm khả năng hoạt động của ứng dụng hiệu quả hơn.

Lưu ý rằng, nếu trình cung cấp dữ liệu không thể tạo ra điều khiển con trỏ mà ta yêu cầu, nó sẽ tạo ra con trỏ mà nó có thể. Nói chung, nó sẽ không báo lỗi trừ phi ta cố thi hành một tác vụ nào đó vốn bị cấm đối với kiểu con trỏ.

Xác định con trỏ và các tính năng khác được hỗ trợ bởi một trình cung cấp

Bởi vì OLE DB và ADO được thiết kế để cho phép truy cập đến nhiều nguồn dữ liệu, ứng dụng cần xác định các tính năng do một trình cung cấp nhất định hỗ trợ. Có thể là trong khi một hệ cơ sở dữ liệu quan hệ cho phép tạo con trỏ kiểu *forword-Only* ở phía server, hệ cơ sở dữ liệu trên máy cá nhân hoặc cơ sở dữ liệu dựa trên tập tin có thể không có tính năng này.

Phương thức supports của đối tượng ADO Recordset xác định kiểu con trỏ do trình cung cấp dữ liệu hỗ trợ.

Bảng sau đây liệt kê các giá trị truyền vào phương thức Supports để xác định tính năng được hỗ trợ bởi đối tượng Recordset:

Hằng	Mô tả
adAddnew	Thêm bản ghi vào Recordset
adApproxPosition	Thuộc tính AbsolutePage và AbsolutePosition có sẵn; chúng được dùng để kết hợp với thuộc tính Pagesize và PageCount của đối tượng Recordset để cho phép xác định trang chứa bản ghi hiện hành.
adBookmark	Có thể quy định đánh dấu trang sách trong Recordset
adDelete	Bản ghi được xoá trong Recordset
adHoldRecords	Bản ghi được trả về từ cơ sở dữ liệu mà không cần ghi sửa đổi hiện có vào server.
adMovePrevious	Có thể cuộn tới lui trong Recordset
adResync	Phương thức này có sẵn
adUpdate	Recordset cập nhật được
adUpdateBatch	Recordset có thể cập nhật được hàng loạt với phương thức UpdateBatch. Ta có thể nạp các sửa đổi trên nhiều mẫu tin trong một hoạt động duy nhất, cải tiến hiệu quả Client/Server

18.1.6 Khoá bản ghi trong ADO

Tương tự các mô hình đối tượng truy cập cơ sở dữ liệu khác, ADO cho phép quy định các kiểu khoá bản ghi(Record - locking) khác nhau. Ta dùng tính năng này khi cần kiểm soát cách thức cập nhật các bản ghi với nhiều người sử dụng trong cơ sở dữ liệu.

Quy định chế độ khoá cho đối tượng Recordset của ADO thông qua thuộc tính LockType. Dưới đây là danh sách 4 kiểu khoá bản ghi.

Hằng	Mô tả
adLockReadOnly	Cấm cập nhật bản ghi
adLockPressimistic	Bản ghi trong Recordset bị khoá khi bắt đầu sửa đổi, và tiếp tục bị khoá đến khi thi hành phương thức Update hay chuyển sang bản ghi khác.
adlockOptimistic	Bản ghi bị khoá ngay khi thi hành phương thức Update hay di chuyển sang bản ghi khác.
adlockBatchOptimistic	Hỗ trợ cập nhật nhiều bản ghi cùng lúc.

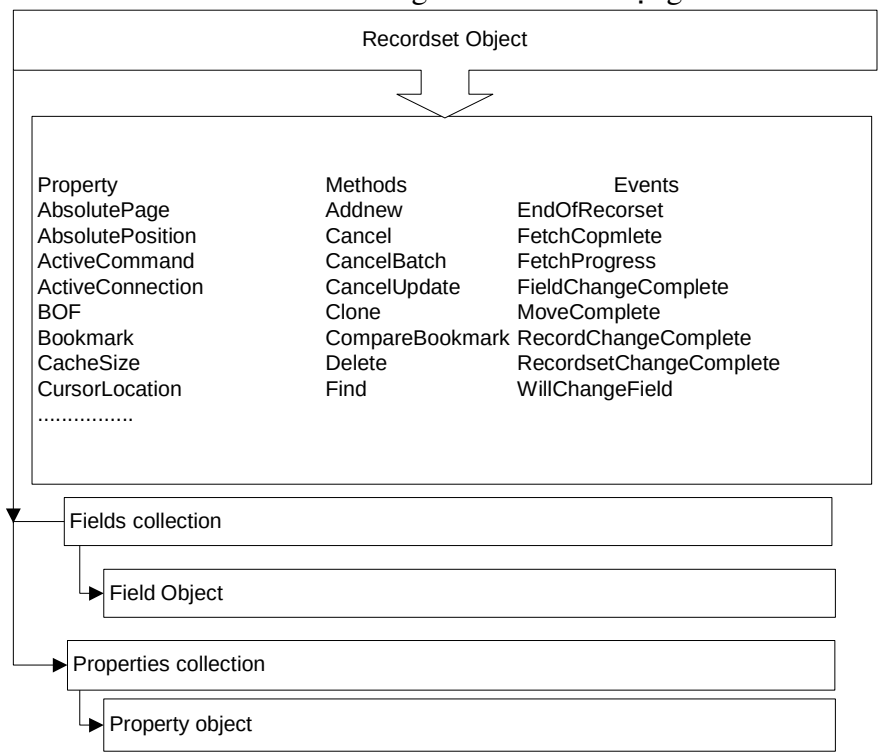
Điểm quan trọng cần lưu ý là phương thức khoá mặc định trong ADO là adLockReadOnly. Đây là một trong các điểm khác biệt đáng kể giữa lập trình ADO và DAO, vì trong DAO bởi mặ định Recordset được phép sửa đổi. Điều này có nghĩa là nếu ta không đổi thuộc tính LockType và CursorType, các Recordset của DAO luôn là chỉ đọc.

Sự hiện diện của các kiểu khoá bản ghi phụ thuộc vào những gì trình cung cấp dữ liệu hỗ trợ. Ta có thể dùng phương thức support của đối tượng Recordset để xác định trình cung cấp có hỗ trợ kiểu khoá bản ghi hay không.

18.1.7 Sử dụng đối tượng Recordset của ADO để thao tác với dữ liệu

Đối tượng Recordset của ADO, tương tự Recordset của DAO và rdoresultset của RDO, là phương pháp truy cập thông tin được trả về từ trình cung cấp dữ liệu. Recordset của ADO có nhiều thuộc tính và phương thức trùng với Recordset của các mô hình khác, vì thế có thể làm việc với chúng tương tự các Recordset khác.

Vị trí của Recordset của ADO trong mô hình đối tượng ADO:



Thủ tục tạo Recordset cảu ADO tương tự tạo rdorseultset của RDO. Tuy nhiên, ADO thêm một thay đổi lý thú: khả năng tạo đối tượng Recordset không đòi hỏi một đối tượng Connection ngầm.

18.1.7.1 Dùng đối tượng Recordset để cập nhật và thêm bản ghi mới

Thêm mới và cập nhật bản ghi trong ADO hầu như tương tự như trong DAO.

Thêm mới bản ghi:

- Mở Recordset
- Thi hành phương thức AddNew
- Gán giá trị cho các trường đối tượng Recordset

- Lưu bản ghi bằng phương thức Update của Recordset

Cập nhật bản ghi:

- Mở Recordset
- Gán giá trị cho các trường trong Recordset
- Lưu bản ghi bằng thi hành phương thức Update

18.1.8 Tạo Recordset ngắt kết nối

Khi dùng con trỏ phía Client của ADO, ta có khả năng ngắt kết nối với server và tiếp tục làm việc với dữ liệu. Cách này làm cho ứng dụng trở nên linh hoạt, bởi vì nhiều người sử dụng có thể làm việc với dữ liệu nếu họ không cần kết nối đến Server.

Để ngắt kết nối với Server trong ADO, ta quy định thuộc tính ActiveConnection của đối tượng Recordset là Nothing. Client sẽ tiếp tục làm việc với dữ liệu thậm chí khi nó không kết nối với server.

18.2 Sử dụng dịch vụ dữ liệu từ xa của ADO

Dịch vụ dữ liệu từ xa (Remote Data Service - RDS) dùng để lấy các Recordset của ADO từ web server. Thư viên này đi kèm với ADO, chủ yếu cho phép ta dùng HTTP làm vận chuyển trên mạng cho ứng dụng cơ sở dữ liệu. Chủ yếu được dùng trong các ứng dụng trình duyệt web, nó cũng hoạt động tốt với các Client của Visual basic.

Lưu ý RDS có tên cũ là *Nối dữ liệu nâng cao* (Advance Data Connector) bạn có thể tìm hiểu thêm tại địa chỉ: http://www.Microsoft.com.data/ado/rds

Ta cũng có thể dùng đối tượng DataControl của RDS để lấy về một đối tượng Recordset của ADO trên Internet. Đối tượng này, được phục vụ từ một máy tính chạy với IIS 3.0(microsoft Internet Infomation Server) trở lên, có khả năng trả về đối tượng Recordset của ADO đến bất kỳ Client qua HTTP. Cách dễ nhất để minh hoạ cách dùng RDS là thi hành một truy vấn trên Web server dùng đối tượng DataControl của RDS.

- Tạo một đề án Standard EXE mới
- Trong menu Project Referances, lập một tham chiếu đến Microsoft ActivateX Data Objects 2.0 Recordset Library
- Tạo một hộp văn bản, một nút lệnh, và một danh sách trên biểu mẫu.
- Đưa đoạn chương trình sau:

```
Option Explicit
Private rdc As RDS.DataControl

Private Sub cmdQuery_Click()

   Screen.MousePointer = vbHourglass

   Set rdc = New RDS.DataControl

   rdc.SQL = "select * from tblCustomer where state = 'IN'"
   rdc.ExecuteOptions = adcExecAsync
   rdc.Connect = "DSN=JetNovelty;"
```

```
    rdc.Server = "http://localhost/"
    rdc.Refresh

    While rdc.ReadyState = adcReadyStateLoaded  ' busy
      DoEvents
    Wend
    Do Until rdc.Recordset.EOF
       With rdc.Recordset
          lstCustomer.AddItem.Fields("FirstName") & " " & _
                        .Fields("LastName")
         .MoveNext
       End With
    Loop

    Set rdc = Nothing

    Screen.MousePointer = vbNormal

End Sub
```

Trong ví dụ này, server "localhost" được dùng. Đây là một cách gọi tắt của Web server chứa trên cùng một máy. Nó rất tiện dụng khi cần kiểm nghiệm chương trình.Vòng lặp While Wend dùng để chờ đáp ứng của server. Đây là tính năng bất đồng bộ của lênh gọi HTTP. Ta không thể chắc chắn mất bao nhiêu lâu thì server đáp ứng, vì vậy ta phải chờ.