UBND TỈNH LÂM ĐỒNG

TRƯỜNG CAO ĐẲNG ĐÀ LẠT

GIÁO TRÌNH

MÔ ĐUN: LẬP TRÌNH WINDOWS 1 NGÀNH/NGHỀ: CÔNG NGHỆ THÔNG TIN (ƯDPM) TRÌNH ĐỘ: CAO ĐẲNG

(Ban hành kèm theo Quyết định số: /QĐ-CĐNĐL ngày …tháng…năm… của Hiệu trưởng Trường Cao đẳng Đà Lạt)

LƯU HÀNH NỘI BỘ

Lâm Đồng, năm 2017

TUYÊN BỐ BẢN QUYỀN

LỜI GIỚI THIỆU

Đây là tài liệu được biên soạn theo chương trình đào tạo Cao đẳng nghề Công nghệ thông tin (ứng dụng phần mềm).

Để học tốt môn học này, người học nên có kiến thức về lập trình căn bản.

Lập trình Windows 1 là một mô đun nhằm giúp người học có kiến thức và kỹ năng lập trình cơ sở trên môi trường Windows. Với phạm vi của tài liệu này, chúng tôi cung cấp cho người học các kiến thức và kỹ năng chính sau:

– Cài đặt và sử dụng được với môi trường VB.NET trên bộ Visual Studio.Net 2010 trở lên;

– Khai báo được lớp đối tượng, các thành phần của lớp đối tượng và sử dụng được lớp đối tượng trên ngôn ngữ VB.Net;

– Cài đặt và xây dựng được chương trình theo phương pháp hướng đối tượng trên một ngôn ngữ lập trình VB.NET;

– Xây dựng các ứng dụng Windows Forms đơn giản kết nối đến cơ sở dữ liệu;

– Nghiêm túc, tỉ mỉ trong quá trình tiếp cận với công cụ mới;

– Chủ động sáng tạo tìm kiếm các ứng dụng viết trên VB.Net.

Trong quá trình biên soạn, chúng tôi có tham khảo nhiều nguồn tài liệu và trên Internet. Mặc dù rất cố gắng biên soạn lại nhưng chắc chắn không tránh khỏi những thiếu sót, tác giả rất mong nhận được những ý kiến đóng góp để tài liệu ngày càng hoàn thiện hơn để cung cấp cho người học những kiến thức và kỹ năng trọng tâm.

Đà Lạt, ngày 07 tháng 07 năm 2017

Tham gia biên soạn

Chủ biên: Ths. Phạm Đình Nam

MỤC LỤC

GIÁO TRÌNH MÔ ĐUN

Tên mô đun: LẬP TRÌNH WINDOWS 1

Mã mô đun: MĐ20

Vị trí, tính chất, ý nghĩa và vai trò của mô đun:

- Vị trí: Mô đun này được học sau các môn Lập trình cơ bản và Tiếng Anh chuyên ngành.

- Tính chất: Là mô đun chuyên môn nghề bắt buộc.

- Ý nghĩa và vai trò của môn học/mô đun: Tài liệu này được thiết kế theo từng mô đun/ môn học thuộc hệ thống mô đun/ môn học của chương trình đào tạo hoàn chỉnh nghề Công nghệ thông tin (Ứng dụng phần mềm) ở trình độ Cao đẳng. Tài liệu dùng làm giáo trình học tập cho sinh viên trong các khóa đào tạo và cũng có thể được sử dụng đào tạo ở Trung tâm để cấp chứng chỉ, đồng thời có thể làm tài liệu tham khảo cho các lập trình viên.

Mục tiêu của mô đun:

Về kiến thức:

- Sử dụng thành thạo ngôn ngữ lập trình Visual Basic .NET (VB.NET) hoặc C Shape .NET (C#.NET).

- Về kỹ năng:

- Cài đặt và sử dụng được bộ Visual Studio.Net 2010 trở lên;

- Khai báo được lớp đối tượng, các thành phần của lớp đối tượng và sử dụng được lớp đối tượng;

- Cài đặt và xây dựng được chương trình theo phương pháp hướng đối tượng trên một ngôn ngữ lập trình VB.NET hoặc C#.NET;

- Xây dựng các ứng dụng Windows Forms;

Về năng lực tự chủ và trách nhiệm:

Có khả năng tự nghiên cứu, tự học, tham khảo tài liệu liên quan đến môn học để vận dụng vào hoạt động hoc tập.

Vận dụng được các kiến thức tự nghiên cứu, học tập và kiến thức, kỹ năng đã được học để hoàn thiện các kỹ năng liên quan đến môn học một cách khoa học, đúng quy định.

Nội dung của mô đun:

BÀI 1. GIỚI THIỆU MICROSOFT VISUAL STUDIO .NET

Mã Bài: MĐ20_01

Giới thiệu:

Với sự phát triển liên tục và đa dạng của thế giới công nghệ thông tin ngày nay, các phần mềm, các hệ điều hành, các môi trường phát triển và các ứng dụng liên tục ra đời. Tuy nhiên, đôi khi việc phát triển không đồng nhất và nhất là do không tương thích về mặt lợi ích của các công ty phần mềm lớn đã làm ảnh hưởng đến công việc của những kỹ sư xây dựng phần mềm.

Mục tiêu:

- Trình bày được cấu trúc Net Framework;
- Hiểu các tính năng của Visual Studio.Net 2010;
- Làm quen với giao diện của VB.Net;
- Viết ứng dụng nhỏ trên VB.net;

Nội dung chính:

1. Giới thiệu Microsoft .NET 2010

1.1. Tình hình trước khi Visual Studio.NET ra đời

Trong giới phát triển ứng dụng trên Internet ta có thể sử dụng các ngôn ngữ Java, PHP, ASP… Khi Java mới được Sun Corporation giới thiệu nó đã có một sức mạnh đáng kể và hướng tới việc chạy trên nhiều hệ điều hành khác nhau, độc lập với các bộ xử lý. Đặc biệt Java rất thích hợp cho việc viết các ứng dụng trên Internet. Tuy nhiên, Java lại có hạn chế về mặt tốc độ và trên thực tế vẫn chưa thịnh hành.

Để làm giảm khả năng ảnh hưởng của Java, bên hãng Microsoft cũng cung cấp ngôn ngữ ASP - chuyên dùng để viết các ứng dụng trên Web. Trong các trang ASP vừa chứa thẻ HTML vừa chứa các đoạn script (VBScript, JavaScript). Trong quá trình xử lý một trang ASP, nếu là thẻ HTML thì sẽ được gửi thẳng tới trình duyệt, còn nếu là các đoạn script thì sẽ được chuyển thành các dòng HTML rồi gửi đi. Khi nhà lập trình muốn đóng gói và sử dụng lại một số chức năng nào đó, thì họ dịch các đoạn chương trình thành ActiveX và đưa nó vào Web Server. Tuy nhiên, vì lý do bảo mật nên các Admin của các trang Web thường rất dè dặt khi cài ActiveX lạ trên máy của họ, ngoài ra việc tháo gỡ các phiên bản của ActiveX này cũng là công việc rất khó khăn.

Còn trong giới phát triển ứng dụng trên Windows ta có thể viết ứng dụng bằng Visual C++, Delphi, Visual Basic… đây là một số công cụ phổ biến và mạnh.

Trong đó Visual C++ là một ngôn ngữ rất mạnh nhưng cũng rất khó sử dụng. Visual Basic thì đơn giản dễ học, dễ dùng nhất nên rất thông dụng nhưng hạn chế là Visual Basic không phải ngôn ngữ hướng đối tượng và không hỗ trợ khả năng phát triển thuật toán.

Tóm lại trong giới lập trình theo Microsoft thì việc lập trình trên desktop cho đến lập trình hệ phân tán hay trên web là những mảng độc lập.

1.2. Sự ra đời của Visual Studio.NET

Đầu năm 1998, sau khi hoàn tất phiên bản Version 4 của Internet Information Server -IIS,đội ngũ lập trình của Microsoft nhận thấy họ còn có rất nhiều sáng kiến để có thể kiện toàn IIS, và họ bắt đầu xây dựng một kiến trúc mới trên nền tảng ý tưởng đó và đặt tên là Next Generation Windows Services - NGWS. Tham vọng của họ là cung cấp một môi trường có thể dùng chung cho tất cả ngôn ngữ lập trình trong bộ Visual Studio cũng như cho các ngôn ngữ lập trình của các công ty khác.

Kết quả là năm 2001 Visual Studio.Net 2001 ra đời đánh dấu cho một môi trường lập trình trên nền .NET Framework 1.0 tiên tiến mới.

Năm 2003, sau 2 năm .NET Framework nâng cấp thêm một bậc với phiên bản 1.1 với đặc điểm ngoài các chương trình Windows truyền thống – là các tệp tin .exe giờ đây Windows còn tồn tại những chương trình khác – những chương trình chạy trên nền .NET. Muốn chạy chương trình .NET ta chỉ cần cài .NET Framework là đủ. Một điểm lý thú và cũng là điều mong đợi của tất cả lập trình viên, từ phiên bản Windows 2003 .NET Framework được cài đặt như một phần mặc định của Windows. Song song đó, môi trường phát triển Visual Studio .NET 2001 được nâng cấp thành Visual Studio .NET 2003 cho phép viết và chạy các ứng dụng trên nền .NET Framework 1.1

Cuối năm 2005, Visual Studio 2005 với nền .NET Framework 2.0 mạnh mẽ và vượt trội hơn so với nền .NET Framwork 1.1 trước đó. Ngay sau đó Microsoft công bố phiên bản Windows Vista, và toàn bộ Windows là .NET, tất cả các hàm API lõi trong những phiên bản Windows trước đây đều đã được thay thế bằng các hàm hay thư viện .NET. Microsoft đã viết lại hoàn toàn lõi API, không còn một lớp API nào nữa.

1.3 Tổng quan về Visual Studio.NET

Visual Studio.NET gồm 2 phần: Framework và Integrated Development Environment– IDE, cho phép lập trình viên khi xây dựng các ứng dụng có thể lựa chọn sử dụng nhiều ngôn ngữ lập trình khác nhau như Visual C++.NET, Visual

C#.NET, Visual J#.NET, Visual Basic.NET… trong cùng một môi trường phát triển IDE thống nhất trên kiến trúc .NET Framework.

Framework là thành phần quan trọng nhất, là cốt lõi và tinh hoa của môi trường .NET, Framework giúp chúng ta biên dịch và thực thi các ứng dụng .NET (cấu trúc của Framework chúng ta sẽ tìm hiểu ở các chương sau của giáo trình).

IDE cung cấp một môi trường phát triển trực quan, giúp các lập trình viên có thể dễ dàng và nhanh chóng xây dựng giao diện cũng như viết mã lệnh cho các ứng dụng dựa trên nền tảng .NET. Nếu không có IDE chúng ta cũng có thể dùng một trình soạn thảo văn bản bất kỳ, ví dụ như Notepad để viết mã lệnh và sử dụng command line để biên dịch và thực thi ứng dụng. Tuy nhiên việc này mất rất nhiều thời gian, tốt nhất là chúng ta nên dùng IDE để phát triển các ứng dụng, và đó cũng là cách dễ sử dụng nhất.

Ngoài ra trong Visual Studio.NET thì lập trình Winform và Webform là tương tự, ví dụ cả Visual C#.NET lẫn Visual Basic.NET đều hỗ trợ khả năng lập trình trên Win và Web…

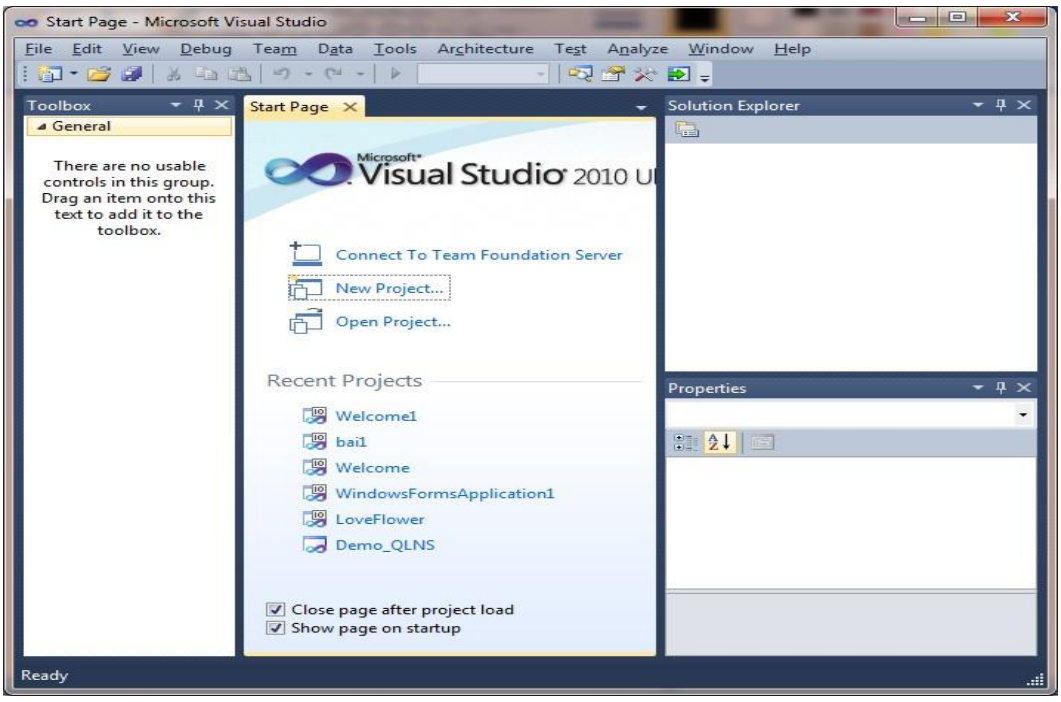
1.4 Trình biên dịch và MSIL

Microsoft Intermediate Language (MSIL) hay Common Intermediate Language (CIL) là một ngôn ngữ trung gian được tạo ra sau quá trình biên dịch từ các loại ngôn ngữ khác trong .Net như C#, C++, VB.Net, J#, …

Tất cả mã nguồn .NET đều được biên dịch thành MSIL. Sau đó MSIL sẽ được chuyển thành mã máy khi phần mềm được cài đặt hoặc khi chạy (run-time) bởi trình biên dịch JIT (Just-In-Time).

2. Khởi động Visual Basic.NET 2010 và giao diện

Để khởi động Visual C# 2010 và giao diện: Vào Start/Programs/Microsoft Visual Studio 2010/Microsoft Visual Studio 2010, xuất hiện cửa sổ *Start Page*.



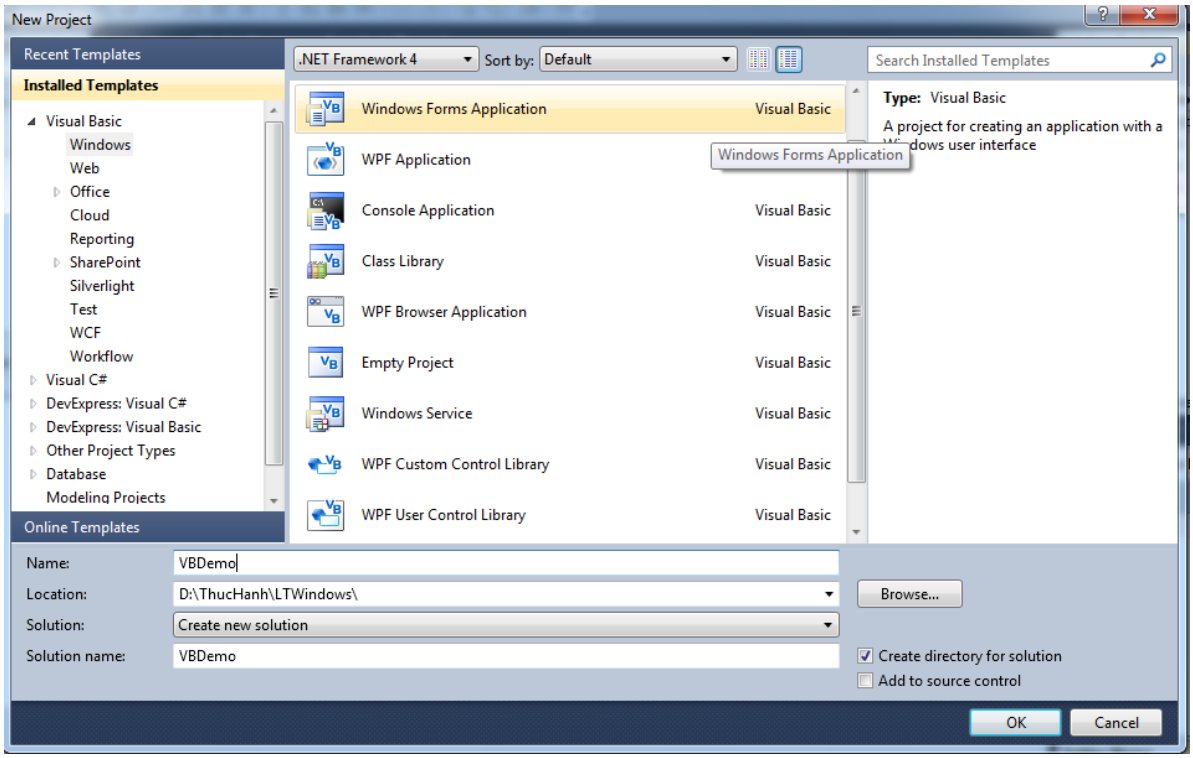
Hình 1. Cửa sổ Start Page

+ New Project: Tạo đồ án mới.

+ Open Project: Mở các đồ án có sẵn.

+ Recent Projects: Danh sách các đồ án gần đây nhất.

Sau đó kích chọn mục New Project hoặc vào File/New/Project hoặc bấm phím tắt Ctrl+Shift+N sẽ xuất hiện cửa sổ *New Project*.



Hình 2. Cửa sổ New Project

+ Chọn ngôn ngữ *Visual Basic* và ứng dụng *Windows*.

+ Đặt tên cho đồ án tại mục *Name*.

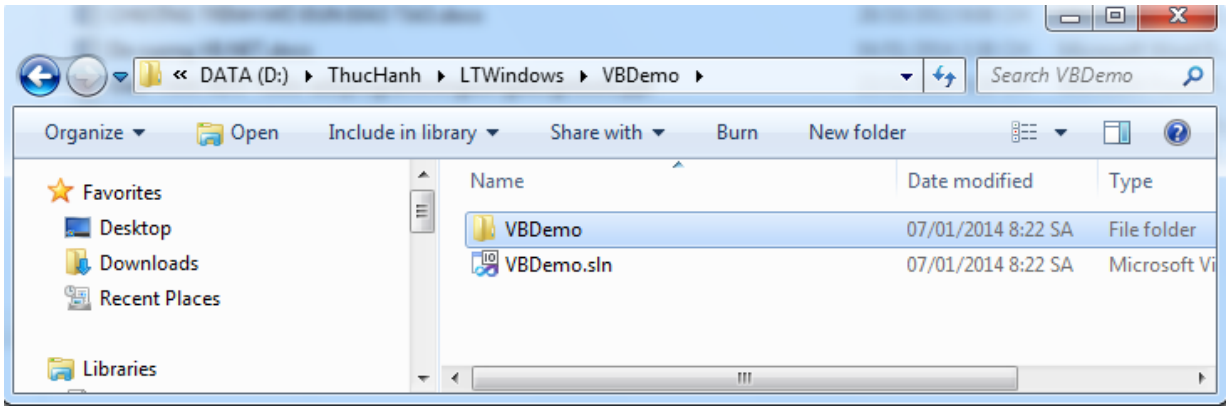
+ Chọn đường dẫn lưu đồ án tại mục *Location*.

+ Chọn *OK* để tạo một đồ án mới.

* Lưu ý:

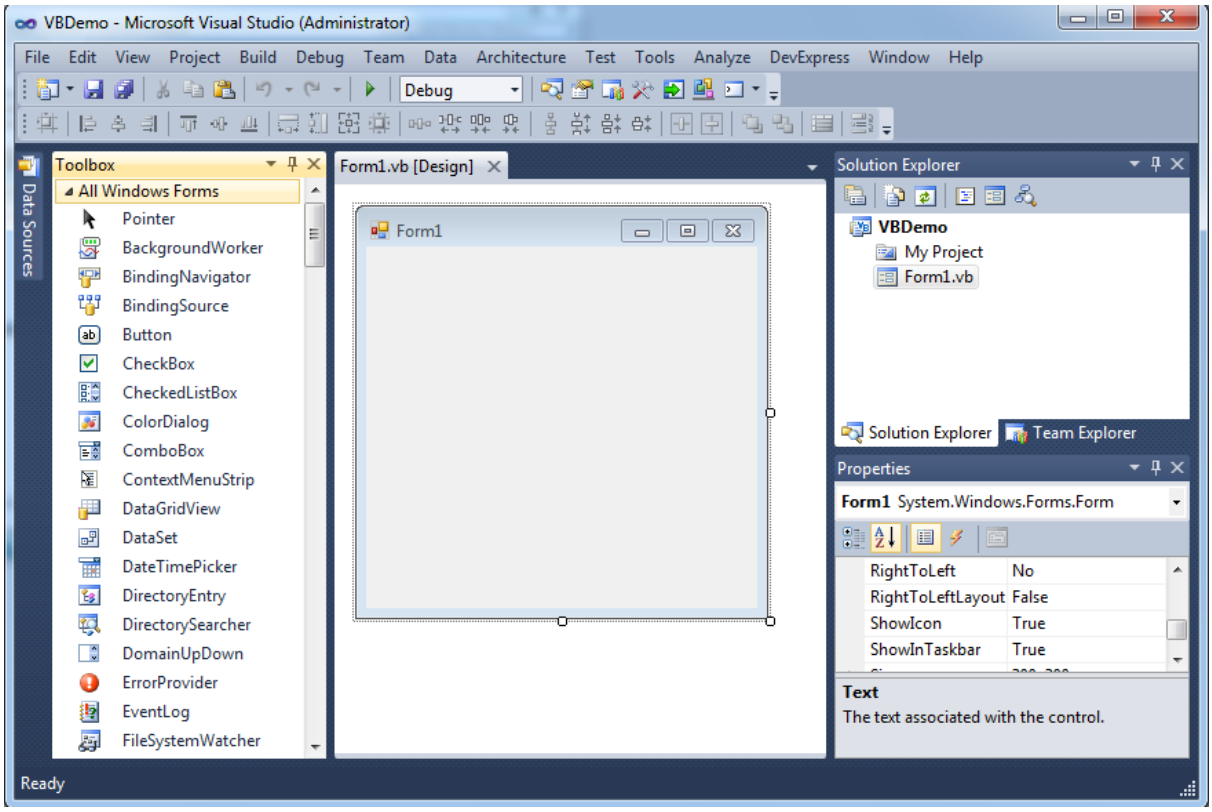
+ Mục *Create directory for solution* cho phép tạo một thư mục tại Location chứa tất cả

các tệp phát sinh của đồ án (nếu không các tệp của đồ án sẽ được lưu tại Location).



Hình 3. Nơi lưu trữ đề án

Kết quả xuất hiện cửa sổ môi trường phát triển tích hợp IDE, với giao diện và các thành phần cơ bản như sau:



Hình 4. Môi trường phát triển tích hợp IDE

Title Bar: Thanh tiêu đề chứa tên đồ án.

Menu Bar: Thanh Menu chứa đầy đủ các công cụ cần để phát triển, thực thi và cài đặt

ứng dụng…

+ *File*: cho phép mở, thêm mới và lưu trữ đồ án…

+ *Edit*: gồm các thao tác hỗ trợ việc soạn thảo mã lệnh như: copy, cắt, dán...

+ *View*: cho phép hiển thị các công cụ hỗ trợ người dùng trong quá trình xây dựng đồ án như:

- Cửa sổ viết mã lệnh - Code
- Form thiết kế - Designer
- Hộp công cụ - Toolbox
- Thanh công cụ - Toolbars
- Cửa sổ thuộc tính - Properties Window…

+ *Project*: cho phép bổ sung các đối tượng khác nhau vào đồ án như: các form, các

component, các modul, các lớp…

+ *Built*: cho phép biên dịch đồ án.

+ *Debug*: cho phép chạy và gỡ rối chương trình.

+ *Data*: cho phép thêm mới và hiển thị cơ sở dữ liệu của đồ án.

+ *Tools*: cung cấp các công cụ cho phép kết nối tới các thiết bị ngoại vi như Pocket PC, Smartphone… hoặc kết nối tới các hệ quản trị cơ sở dữ liệu cũng như kết nối tới máy chủ server…

Toolbar: thanh công cụ gồm một tập hợp các nút lệnh, mỗi nút lệnh chứa một biểu tượng icons và có chức năng tương đương với chức năng của một mục lựa chọn trong thanh menu. Thanh công cụ rất hữu ích và trực quan, giúp người dùng dễ dàng và nhanh chóng thực hiện một chức năng mong muốn chỉ thông qua một cái kích chuột.

Visual Basic 2010 có tới 39 thanh công cụ khác nhau như: Standard, Formatting, Debug, Build... Ví dụ hình ảnh thanh công cụ Standard:

Hình 5. Thanh công cụ Standard

Để gọi các thanh công cụ ta vào *View/Toolbars* khi đó sẽ xuất hiện danh sách tất cả các thanh công cụ. Muốn ẩn/hiện thanh công cụ nào ta kích chọn tại dòng chứa tên thanh công cụ đó.

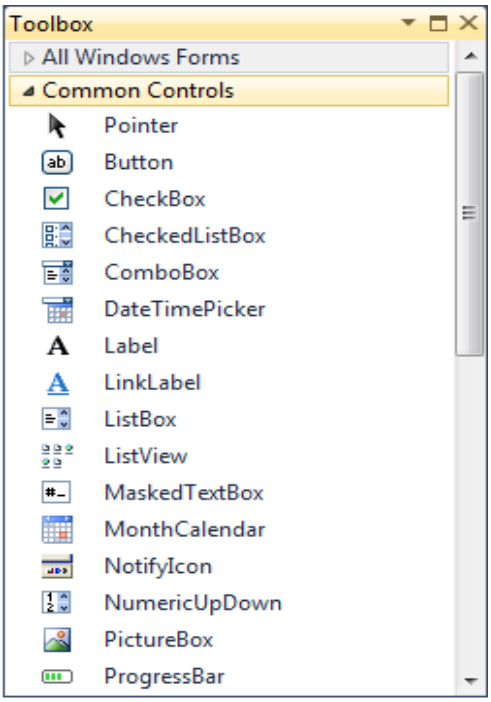
Toolbox: là hộp công cụ chứa các điều khiển – controls được đặt lên Form khi thiết kế giao diện người dùng.

Để hiển thị hộp công cụ ta thực hiện một trong các cách sau:

+ Vào *View*/*Toolbox*

+ Bấm tổ hợp phím *Ctrl+W+X*

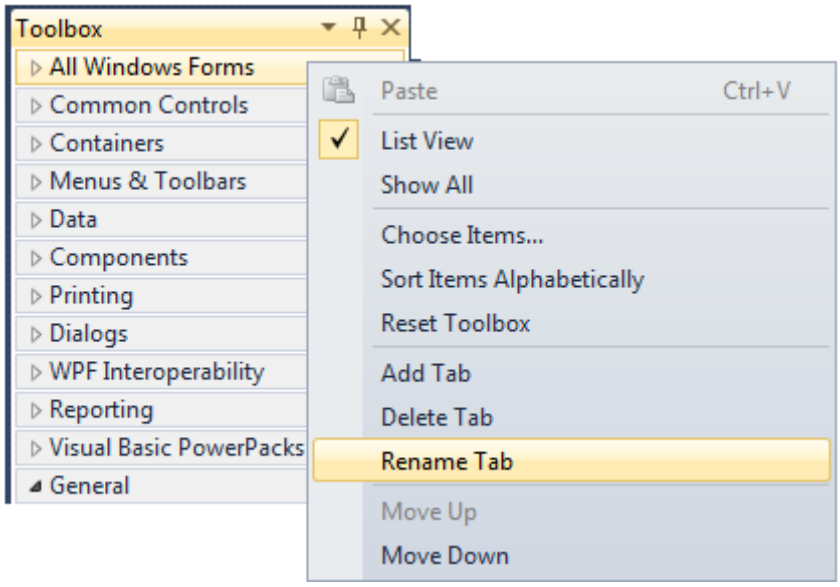
+ Kích chuột tại biểu tượng Toolbox  trên thanh công cụ Standard.



Hình 6. Hộp công cụ Toolbox

Mặc định hộp công cụ được chia thành 11 tab khác nhau như: *All Windows Forms, Common Controls...*

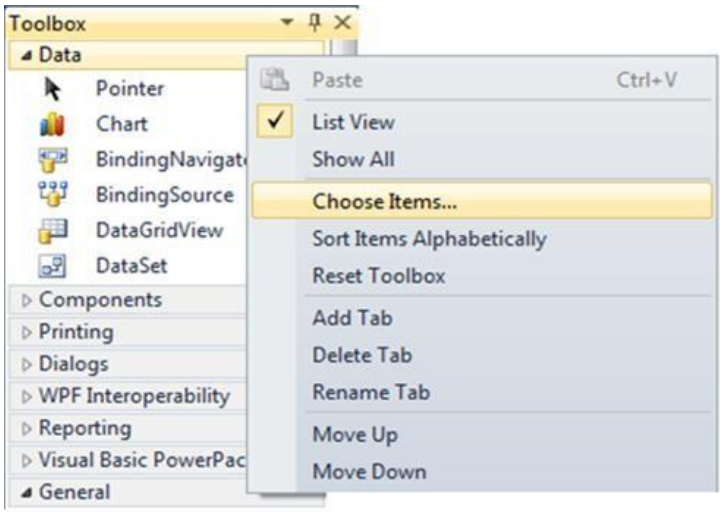
Ta có thể thêm mới, loại bỏ, đổi tên... các tab bằng cách kích chuột phải tại vị trí bất kỳ trên tab, xuất hiện một menu ngữ cảnh cho phép lựa chọn các thao tác cần thực hiện.



Hình 7. Các chức năng làm việc với tab trong Toolbox

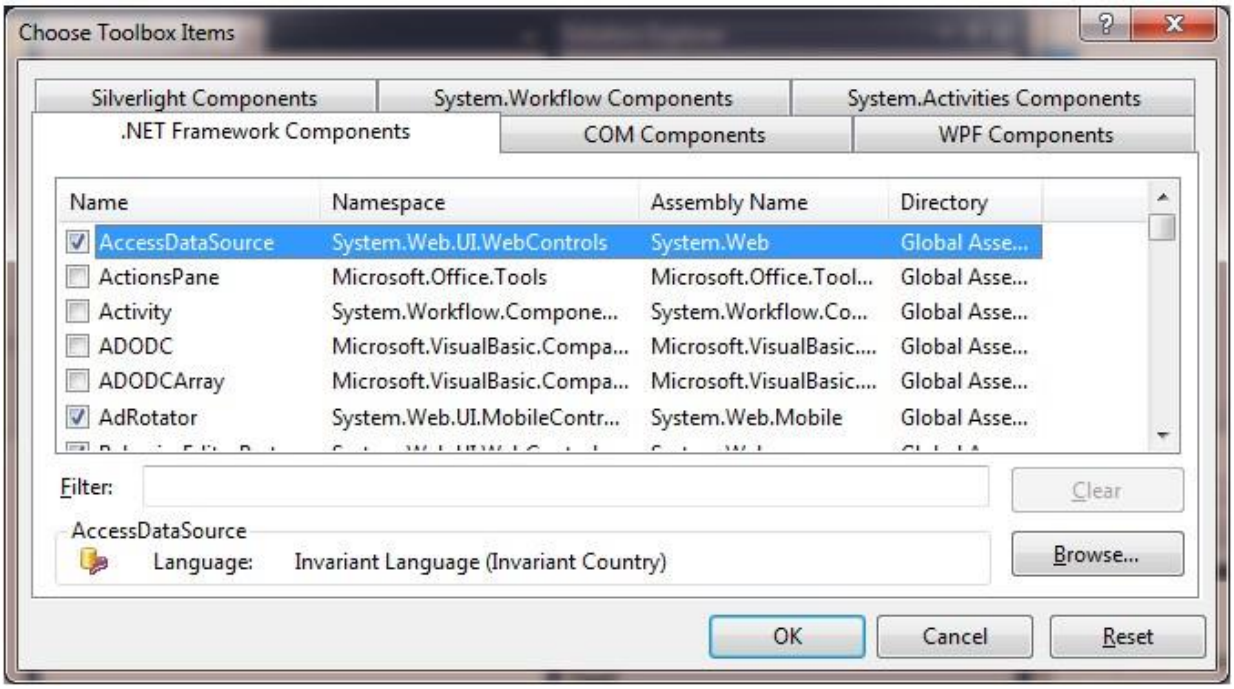
Trong mỗi tab của hộp Toolbox chứa danh sách các loại điều khiển khác nhau, các điều khiển này có thể thêm mới, loại bỏ, thay đổi vị trí… Kích chuột phải tại một điều khiển bất kỳ trên tab, xuất hiện một menu ngữ cảnh cho phép lựa chọn các thao tác cần thực hiện.

Ví dụ để thêm mới một điều khiển vào trong tab Data, ta kích chuột phải tại vị trí bất kỳ trên tab Data, chọn *Choose Items...*



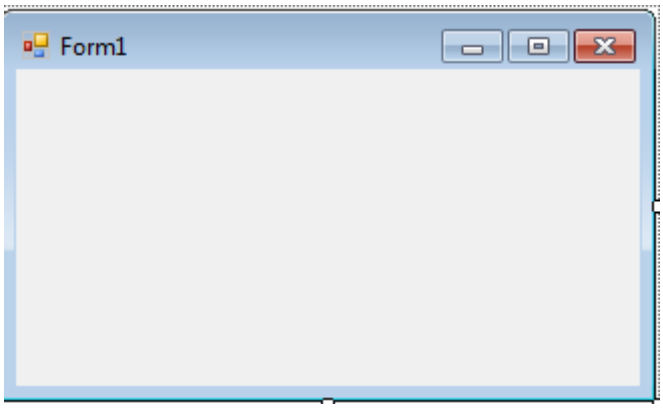
Hình 8. Các chức năng làm việc với từng điều khiển trong tab

Kết quả sẽ xuất hiện cửa sổ *Choose Toolbox Items,* kích chọn các điều khiển mong muốn rồi bấm OK để kết thúc.



Hình 9. Cửa sổ Choose Toolbox Items

Form Designer: cửa sổ thiết kế dùng để thiết kế giao diện cho chương trình, mỗi dự án có thể có một hoặc nhiều Form.



Hình 10. Cửa sổ Form Desigher

Solution Explorer: cửa sổ giải pháp - đây là phần cửa sổ giúp ta quản lý tất cả các tài nguyên và tập tin dự án.

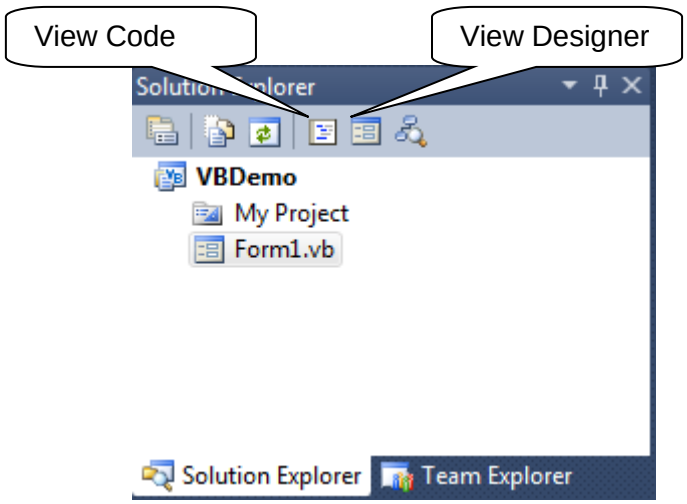
Solution Explorer được tổ chức thành một cấu trúc cây bao gồm những mục khác nhau, như: danh sách các Form của đồ án, danh sách các lớp Class, danh sách các tài nguyên cũng như danh sách cơ sở dữ liệu…

Để hiển thị cửa sổ Solution Explorer ta thực hiện một trong các cách sau:

Vào *View/Solution Explorer*

Bấm tổ hợp phím *Ctrl+W+S*

Kích chuột tại biểu tượng Solution



Hình 11. Cửa sổ Solution Explorer

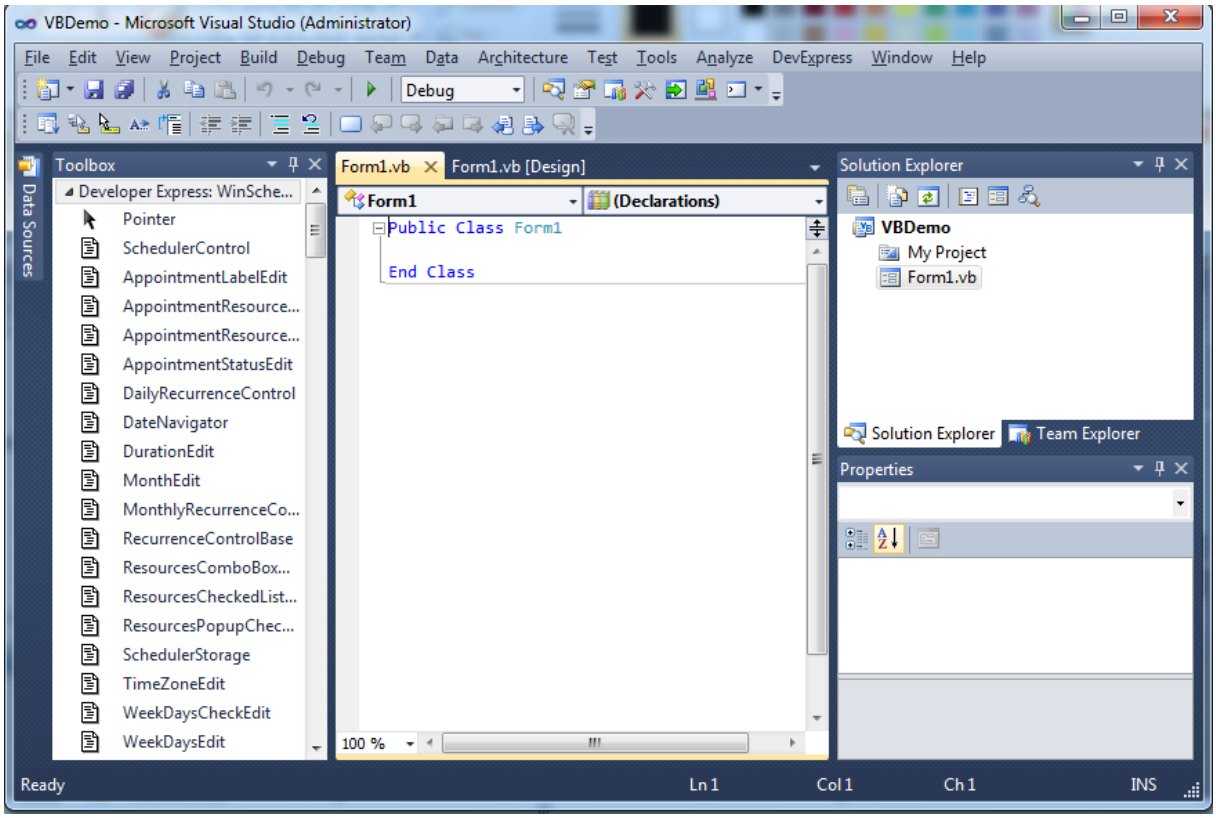
Trong cửa sổ Solution Explorer có hai thành phần hay dùng là *View Code* và *View Designer*.

View Code: có tác dụng hiển thị cửa sổ soạn thảo mã lệnh cho Form đang được chọn. Ngoài ra, để hiển thị cửa sổ soạn thảo mã lệnh ta còn có một số cách khác như sau:

+ Vào *View/Code.*

+ Bấm phím tắt *F7.*

+ Kích đúp chuột tại cửa sổ thiết kế của form. Giao diện cửa sổ soạn thảo như sau:



Hình 12. Cửa sổ soạn thảo

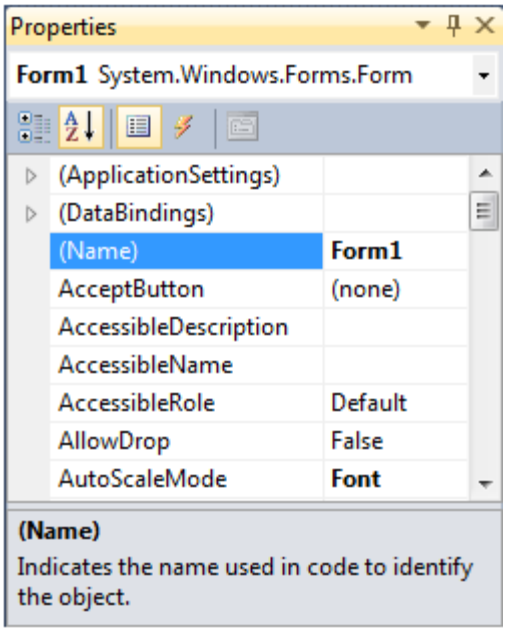
View Designer: có tác dụng hiển thị cửa sổ thiết kế giao diện của Form đang được chọn. Ngoài ra, để hiển thị cửa sổ thiết kế giao diện ta còn có một số cách khác như sau:

+ Vào *View/Designer*

+ Bấm phím tắt *Shift+F7.*

Properties Window: cửa sổ này liệt kê tất cả các thuộc tính, sự kiện của các điều khiển trong form.

Muốn hiển thị thuộc tính của đối tượng nào ta kích chuột chọn đối tượng đó trong cửa sổ thiết kế giao diện, hoặc chọn tên đối tượng trong danh sách thả xuống ở phần đầu của cửa sổ Properties.



Hình 13. Cửa sổ Properties

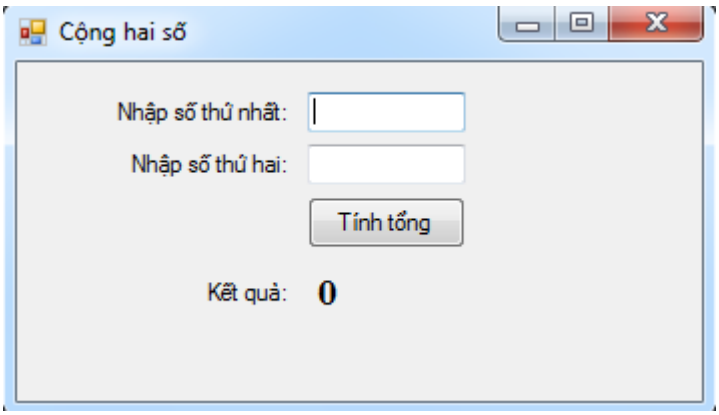
Mỗi thuộc tính có một giá trị mặc định, ta có thể thay đổi giá trị của các thuộc tính trực tiếp tại cửa sổ Properties trong lúc thiết kế, hoặc thay đổi bằng mã lệnh trong lúc thi hành chương trình.

Để hiển thị cửa sổ Properties ta thực hiện theo một trong các cách sau:

+ Vào *View\Properties Window*.
+ Kích chọn biểu tượng Properties Window trên thanh công cụ Standard.
+ Bấm phím tắt *Ctrl+W+P*

3. Tạo ứng dụng đầu tiên

Bây giờ để làm quen với giao diện, chúng ta tạo ứng dụng đầu tiên. Trong ứng dụng này có sử dụng các điều khiển cơ bản nhất là Label, TextBox và Button để thiết kế form nhập vào hai số nguyên, tính tổng của hai số và hiện kết quả.



Hình 14. Giao diện form cộng hai số

Bước 1: Đặt tên cho các điều khiển

Sau khi tạo một đề án mới như phần trên, thay đổi thuộc tính của form Form1.vb và thay đổi thuộc tính của các điều khiển trên form như sau:

Điều khiển	Thuộc tính	Giá trị
Form1.vb	Name	Form1
	StartPosition	CenterScreen
	Text	Cộng hai số
Label1	Text	Nhập số thứ nhất:
Label2	Text	Nhập số thứ hai:
Label3	Text	Kết quả:
Label4	Name	lblKQ
	Text	0
TextBox1	Name	txtSo1
TextBox2	Name	txtSo2
Button1	Name	btnTong
	Text	Tính tổng

Bước 2: Viết lệnh

Nhắp đôi chuột vào nút lệnh btnTong và viết đoạn lệnh sau:

```
Private Sub btnTong_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnTong.Click
        Dim tong As Integer = Convert.ToInt32(txtSo1.Text) +
Convert.ToInt32(txtSo2.Text)
        lblKQ.Text = tong.ToString()
End Sub
```

Ở đoạn lệnh trên, dòng đầu tiên:

```
Private Sub btnTong_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnTong.Click
```

Là sự kiện Click của nút lệnh, tự động sinh ra khi người lập trình nhắp đôi chuột vào nút lệnh btnTong. Đây cũng là sự kiện mặc định của nút lệnh.

Dòng tiếp theo là khai báo biến tên tong (để hiểu rõ hơn về cú pháp khai báo biến chúng ta sẽ xem xét kỹ ở bài 2 của giáo trình này):

```
Dim tong As Integer = Convert.ToInt32(txtSo1.Text) +
Convert.ToInt32(txtSo2.Text)
```

Có thể thay thế bằng dòng lệnh:

```
Dim tong As Integer = Integer.Parse(txtSo1.Text) +
Integer.Parse(txtSo2.Text)
```

Dùng để chuyển kiểu giá trị chuỗi nhập vào hai điều khiển TextBox thành kiểu số nguyên, sau đó cộng hai số và gán kết quả cuối cùng cho biến tong.

Dòng lệnh:

```
lblKQ.Text = tong.ToString()
```

Dùng để hiện tổng của hai số ra điều khiển Label.

4. Cấu trúc của ứng dụng Visual Basic.NET

4.1 Namespaces là gì?

Namespaces giúp tổ chức các đối tượng của một Assembly thành một cấu trúc để dễ hiểu hơn, chúng nhóm các đối tượng liên quan lại với nhau để dễ truy cập bằng code:

+ Ví dụ namespace **SQLClient** được định ngĩa trong **System.Data**

Namespaces tạo phải đầy đủ tên của đối tượng, tránh sự nhập nhằng và các tên xung đột với các class.

4.2 Tạo một Namespace

Để tạo một Namespace, chúng ta dùng câu lệnh **Namespace … End Namespace**

Ví dụ:

```
Namespace KhachHang
'Tạo các lớp, module hay interface liên quan đến thông tin khách hàng
End Namespace
```

Assembly thường định nghĩa Namespace gốc cho Project, được thiết lập trong hộp thoại Project Properties. Assemply có Namespace gốc là **MyAssembly**.

Ví dụ:

```
Namespace Top
        'Tên đầy đủ là MyAssembly.Top
        Public Class Inside
            'Tên đầy đủ là MyAssembly.Top.Inside
          ...
        End Class
        Namespace InsideTop
            'Tên đầy đủ là MyAssembly.Top.InsideTop
```

```
            Public Class Inside
                'Tên đầy đủ là MyAssembly.Top.InsideTop.Inside
             ...
            End Class
        End Namespace
End Namespace
```

Để gọi code trong cùng Assembly, chúng ta có thể bỏ qua tên Namespace.

Ví dụ:

```
Public Sub Perform( )
Dim x As New Top.Inside( )
Dim y As New Top.InsideTop.Inside( )
...
End Sub
```

Khi gọi code phải tham chiếu đầy đủ đến tên của Namespace, điều này làm cho code khó đọc:

```
Dim x As New MyAssembly.Top.InsideTop.Inside
```

Vì vậy, chúng ta có thể dùng câu lệnh Imports để code đơn giản hơn như sau:

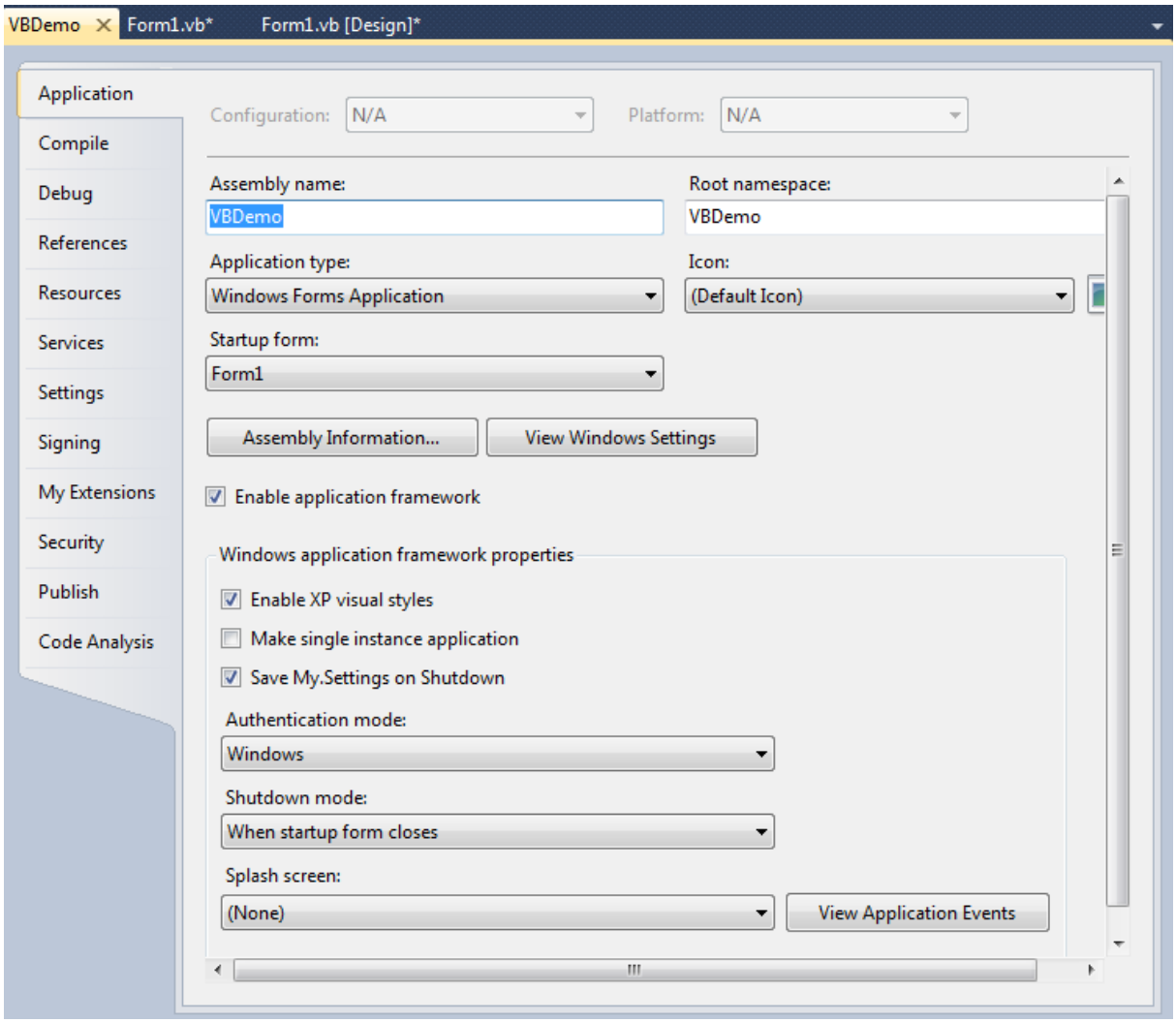
```
Imports MyAssembly.Top.InsideTop
…
Dim x As New Inside( )
```

Chúng ta cũng có thể Import một tên bí danh cho một Namespace hoặc một kiểu.

Ví dụ:

```
Imports IT = MyAssembly.Top.InsideTop
…
Dim x As New IT.Inside
```

* Lưu ý: Để thiết lập các thuộc tính thông thường cho một Project, chúng ta vào menu Project\<tên đề án> Properties… hộp thoại Properties cho Project sẽ xuất hiện cho phép bạn thay đổi các tham số mặc định.



Hình 15. Cửa sổ Project Properties

5. Bài tập

Bài 1: Cải tiến ứng dụng "Cộng hai số" ở trên bằng cách bổ sung thêm các nút lệnh Tính Hiệu, Tích, Thương của hai số.

Bài 2: Thiết kế form nhập vào họ, tên của một người. Hiện thông báo ra màn hình để chào người có họ và tên được nhập.

Hướng dẫn:

- Để nối hai chuỗi chúng ta sử dụng phép nối &.

  Ví dụ: `"Nguyễn" & " " & "Bình" = "Nguyễn Văn Bình"`

BÀI 2. NỀN TẢNG CỦA NGÔN NGỮ VB.NET

Mã Bài: MĐ20_02

Giới thiệu:

Bài này sẽ tập trung trình bày về các kiểu dữ liệu, các toán tử và câu lệnh của ngôn ngữ lập trình Visual Basic .NET.

Mục tiêu:

- Hiểu về các nền tảng của VB.Net như: kiểu dữ liệu, biến, mảng,...;
- Hiểu về cú pháp cấu trúc điều khiển trong VB.Net;
- Viết ứng dụng nhỏ trên VB.net;
- Nghiêm túc, tỉ mỉ trong quá trình tiếp cận với công cụ mới.

Nội dung chính:

1. Các kiểu dữ liệu

Kiểu dữ liệu	Kích thước	Phạm vi	Ví dụ
Short	16-bit	-32,678 - 32,767	Dim S as Short S = 12500
Integer	32-bit	-2,147,483,648 đến 2,147,483,647	Dim I as Integer S = 4000
Long	64-bit	-9,233,372,036,854,775,808 đến 9,233,372,036,854,775,807	Dim L as Long L = 3988890343
Single	32-bit (dấu phảy động)	-3.402823E38 đến 3.402823E38	Dim Sg as Single Sg = 899.99
Double	64-bit (dấu phảy động)	-1.797631348623E308 đến 1.797631348623E308	Dim D as Double D=3.1.4159265
Decimal	128-bit	Trong khoảng +/- $79,228x10^{24}$	Dim Dc as Decimal Dc=7234734.5
Byte	8-bit	0-255	Dim B as Byte B=12
Char	16-bit	0-65,536	Dim Ch As Char Ch=”L”
String	Nhiều ký tự	Chứa 0 đến 2 tỷ ký tự	Dim St As String St=”Đức Lập”
Boolean	16-bit	Hai giá trị True hay False	Dim Bl As Boolean Bl = True

Date	64-bit	Từ 1/1/1 đến 31/12/9999	Dim Da As Date Da=#16/07/1984
Object	32-bit	Bất kỳ kiểu đối tượng nào	Dim Obj As Object

2. Biến

2.1 Khái niệm

Mỗi ứng dụng thường xử lý nhiều dữ liệu, ta dùng khái niệm "biến" để lưu trữ dữ liệu trong bộ nhớ máy tính, mỗi biến lưu trữ một dữ liệu của chương trình.

Mặc dù VB không đòi hỏi, nhưng ta nên định nghĩa rõ ràng từng biến trước khi truy xuất nó để code của chương trình được trong sáng, dễ hiểu, dễ bảo trì và phát triển.

Biến (Variable) là vùng lưu trữ được đặt tên để chứa dữ liệu tạm thời trong quá trình tính toán, so sánh và các công việc khác.

Biến thường có hai đăc điểm:

+ Mỗi biến có một tên.

+ Mỗi biến có thể chứa duy nhất một loại dữ liệu.

2.2 Khai báo biến

Cú pháp đơn giản của lệnh định nghĩa biến:

[Static|Public|Private|Dim] <tên biến> As <kiểu dữ liệu>[= <Biểu thức>]

Trong đó:

<tên biến>: là một tên được đặt giống quy tắc đặt tên điều khiển. Nếu cần khai báo nhiều biến trên một dòng thì mỗi khai báo cách nhau dấu phẩy (,).

<kiểu dữ liệu>: là một trong các kiểu dữ liệu đã tìm hiểu ở trên.

Nếu khai báo biến không xác định kiểu dữ liệu thì biến đó có kiểu Variant.

Khai báo ngầm: Đây là hình thức không cần phải khai báo một biến trước khi sử dụng. Cách dùng này có vẻ thuận tiện nhưng sẽ gây một số sai sót, chẳng hạn khi ta đánh nhầm tên biến, VB.NET sẽ hiểu đó là một biến mới dẫn đến kết quả chương trình sai mà rất khó phát hiện.

Vì vậy trong VB.NET bạn cần khai báo biến trước khi sử dụng nó.

Việc khai báo biến có thể đặt ở bất kỳ đâu nhưng thường được đặt ở đầu mỗi thủ tục, nơi cần dùng biến.

Biến cục bộ: là biến được khai báo trong một khối lệnh (Dim)

Ví dụ: Tìm giá trị nghịch đảo của x

```
If x <> 0 Then
```

```
Dim rec As Integer
rec = 1/x
End If
MsgBox CStr(rec)
```

Biến cấp module: là biến được khai báo trong phần khai báo toàn cục của một module (Public, Friend, Private).

Private: là biến chỉ có hiệu lực trong module đó (mặc định).

Friend: là biến chỉ có hiệu lực trong dự án đó.

Public: biến có hiệu lực không chỉ trong dự án nó được khai báo mà còn trong các dự án khác có tham chiếu đến dự án này.

Ví dụ:

```
    Dim LastName As String
```

Phát biểu trên khai báo một biến tên là *LastName* có kiểu dữ liệu là *String*. Sau khi đã khai báo biến thì bạn có thể gán hay lưu thông tin vào biến.

ví dụ:

```
    LastName = "Đình Nam"
```

Và có thể gán nội dung biến cho thuộc tính của đối tượng,

ví dụ:

```
    Label1.Text = LastName
```

2.3 Khởi tạo giá trị cho biến

Ví dụ sau đây vừa khai báo vừa khởi tạo giá trị cho các biến:

```
Dim x As Integer = 5
Dim y As Integer = 6, z As Integer = 9
```

3 Mảng

3.1 Khái niệm

Mảng là tập hợp các phần tử có cùng một kiểu. Dùng mảng sẽ làm cho chương trình đơn giản và gọn hơn vì ta có thể sử dụng vòng lặp. Mảng sẽ có biên trên và biên dưới, trong đó các thành phần của mảng là liên tiếp trong khoảng giữa hai biên này.

Có hai loại biến mảng: mảng có chiều dài cố định và mảng có chiều dài thay đổi lúc thi hành.

3.2 Khai báo

3.2.1 Mảng có chiều dài cố định:

Dim <Tên biến mảng>(<Kích thước>) [As <Kiểu dữ liệu>]

Lúc này phần tử đầu tiên có chỉ số là 0 & phần tử cuối cùng có chỉ số là <Kích thước >.

Hoặc:

Dim <Tên biến mảng>(<Chỉ số đầu> To <Chỉ số cuối>) [As <Kiểu>]

Ví dụ:

```
' Khai báo một biến mảng 15 phần tử kiểu Integer
Dim  Counters(14) As Integer
' Khai báo một biến mảng 21 phần tử kiểu Double
Public Sums(20)As Double
' Khai báo một biến mảng 10 phần tử kiểu chuỗi ký tự
Dim List (1 To 10) As String * 12
```

Hàm UBound trả về biên trên của một mảng.

Ví dụ:

UBound(List) sẽ trả về giá trị là 10.

LBound(List) sẽ trả về giá trị là 1.

* *Lưu ý:* Chúng ta có thể khai báo một mảng nhiều chiều như sau:

Dim Multi3D (3, 1 To 10, 9) As Double

Khai báo này tạo ra một mảng 3 chiều với kích thước 4 x 10 x 10.

3.2.2 Mảng động

Đây là mảng có kích thước thay đổi, đó là một trong những ưu điểm của mảng động vì nó giúp ta tiết kiệm tài nguyên hệ thống. Ta có thể sử dụng một mảng có kích thước lớn trong một thời gian nào đó rồi xoá bỏ để trả lại vùng nhớ cho hệ thống.

Khai báo một mảng động bằng cách cho nó một danh sách không theo chiều nào cả.

Cú pháp: *Dim <Tên mảng> () [As <Kiểu>]*

Ví dụ:

```
Dim DynArray() As Integer
```

Sau đó ta có thể cấp phát số phần tử thật sự bằng lệnh ReDim:

ReDim <*Tên mảng*>(N) ' Trong đó N là một biểu thức kiểu Integer.

ReDim dùng để xác định hay thay đổi kích thước của một mảng động. Ta có thể dùng ReDim để thay đổi số phần tử, số chiều của một mảng nhiều lần nhưng không thể thay đổi kiểu dữ liệu của mảng ngoại trừ kiểu mảng là kiểu Variant.

Mỗi lần gọi ReDim tất cả các giá trị chứa trong mảng sẽ bị mất. VB.NET khởi tạo lại giá trị cho chúng (Empty đối với mảng Variant, 0 cho mảng kiểu số, chuỗi rỗng cho mảng chuỗi hoặc Nothing cho mảng các đối tượng). Nhưng đôi khi ta muốn tăng kích cỡ của mảng nhưng không muốn làm mất dữ liệu, ta dùng ReDim đi kèm với từ khoá Preserve. Ta xem ví dụ dưới đây:

```
ReDim Preserve DynArray (UBound(DynArray) +10)
```

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

3.2 Một số thao tác trên mảng

Truy xuất từng phần tử trong mảng: <Tên mảng>(<Vị trí>)

Sao chép mảng: Ta có thể gán một mảng cho một mảng khác, hoặc kết quả trả về của một hàm có thể là một mảng.

Ví dụ:

```
Sub ByteCopy (cu() As Byte, moi() As Byte)
moi = cu
End Sub
```

Tuy nhiên, cách này cũng chỉ áp dụng được cho mảng khai báo động mà thôi.

Khi gán biến, có một số quy luật mà ta cần lưu ý: Đó là quy luật về kiểu dữ liệu và quy luật về kích thước và số chiều của mảng.

Lỗi khi gán mảng có thể xảy ra lúc biên dịch hoặc khi thi hành. Ta có thể thêm bẫy lỗi để đảm bảo rằng hai mảng là tương thích trước khi gán.

Mảng còn có thể là kết quả trả về của hàm. Chẳng hạn như:

```
Public Function ArrayFunction (b As Byte) As Byte()
Dim x(2) As Byte
x(0) = b
x(1) = b + 2 x(2) = b + b
ArrayFunction = x
End Function
```

Khi gọi hàm trả về mảng, biến giữ giá trị trả về phải là một mảng và có kiểu như kiểu của hàm, nếu không nó sẽ báo lỗi "*không tương thích kiểu*".

4. Toán tử

4.1 Khái niệm

Toán tử hay phép toán (Operator): là từ hay ký hiệu nhằm thực hiện phép tính và xử lý dữ liệu.

Toán hạng: là giá trị dữ liệu (biến, hằng…).

Biểu thức: là tập hợp các toán hạng và các toán tử kết hợp lại với nhau theo quy tắc nhất định để tính toán ra một giá trị nào đó.

4.2 Các loại phép toán

Các phép toán số học: thao tác trên các giá trị có kiểu dữ liệu số.

Phép toán	Ý nghĩa	Kiểu của đối số	Kiểu kết quả
-	Phép lấy sô đôi	Kiểu số (Integer, Single…)	Như kiểu đối số
+	Phép cộng hai số	Kiểu số (Integer, Single…)	Như kiểu đối số
-	Phép trừ hai số	Kiểu số (Integer, Single…)	Như kiểu đối số
*	Phép nhân hai số	Kiểu số (Integer, Single…)	Như kiểu đối số
/	Phép chia hai số	Kiểu số (Integer, Single…)	Single hay Double
\	Phép chia lấy phần nguyên	Integer, Long	Integer, Long
Mod	Phép chia lấy phần dư	Integer, Long	Integer, Long
^	Tính lũy thừa	Kiểu số (Integer, Single…)	Như kiểu đối số

Các phép toán quan hệ: là các phép toán mà giá trị trả về của chúng là một giá trị kiểu Boolean (TRUE hay FALSE).

Phép toán , Ý nghĩa

= So sánh bằng nhau

<> So sánh khác nhau

> So sánh lớn hơn

< So sánh nhỏ hơn

>= So sánh lớn hơn hoặc bằng

<= So sánh nhỏ hơn hoặc bằng

Các phép toán Logic: là các phép toán tác động trên kiểu Boolean và cho kết quả là kiểu Boolean. Các phép toán này bao gồm AND (và), OR (hoặc), NOT (phủ định). Sau đây là bảng giá trị của các phép toán:

X	**Y**	**X AND Y**	**X OR Y**	**NOT X**
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE
FALSE	TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	TRUE

5. Câu lệnh điều khiển

Một câu lệnh (statement) xác định một công việc mà chương trình phải thực hiện để xử lý dữ liệu đã được mô tả và khai báo. Các câu lệnh được ngăn cách với nhau bởi ký tự xuống dòng. *Ký tự xuống dòng báo hiệu kết thúc một câu lệnh*.

5.1 Câu lệnh gán

Cú pháp:

<Tên biến> = <Biểu thức>

Ví dụ:

Giả sử ta có khai báo sau:

```
Dim TodayTemp As Single, MinAge As Integer
Dim Sales As Single, NewSales As Single, FullName As String
```

Các lệnh sau gán giá trị cho các biến trên:

```
TodayTemp = 30.5
MinAge = 18
Sales = 200000
NewSales = Sales * 1.2
```

Giả sử người dùng cần nhập họ và tên vào ô nhập liệu TextBox có thuộc tính Name là txtName, câu lệnh dưới đây sẽ lưu giá trị của ô nhập liệu vào trong biến FullName:

```
FullName = txtName.Text
```

* Lưu ý: Kiểu dữ liệu của biểu thức (vế phải của lệnh gán) phải phù hợp với biến ta cần gán trị.

5.2 Câu lệnh rẽ nhánh If

o Một dòng lệnh:

If <điều kiện> **Then** <*dòng lệnh*>

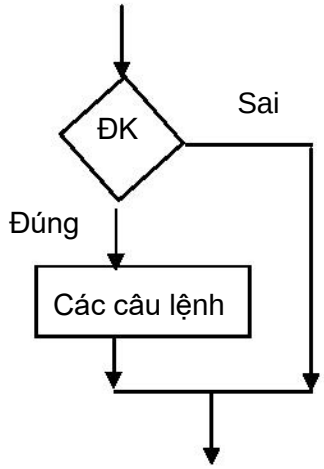
o Nhiều dòng lệnh:

If <điều kiện> **Then**

Các dòng lệnh

End If

Lưu đồ cú pháp:



Trong đó, <điều kiện>: biểu thức mà kết quả trả về kiểu Boolean.

Ý nghĩa câu lệnh: *Các dòng lệnh* hay *dòng lệnh* sẽ được thi hành nếu như điều kiện là đúng. Còn nếu như điều kiện là sai thì câu lệnh tiếp theo sau cấu trúc If ... Then được thi hành.

Dạng đầy đủ: **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

VB.NET sẽ kiểm tra các điều kiện, nếu điều kiện nào đúng thì khối lệnh tương ứng sẽ được thi hành. Ngược lại nếu không có điều kiện nào đúng thì khối lệnh sau từ khóa Else sẽ được thi hành.

Ví dụ:

```
If (TheColorYouLike = vbRed) Then
MsgBox "You are a lucky person"
ElseIf (TheColorYouLike = vbGreen) Then
```

```
MsgBox "You are a hopeful person"
ElseIf (TheColorYouLike = vbBlue) Then
MsgBox "You are a brave person"
ElseIf (TheColorYouLike = vbMagenta) Then
MsgBox "You are a sad person"
Else
MsgBox "You are an average person"
End If
```

5.3 Câu lệnh lựa chọn Select Case

Trong trường hợp có quá nhiều các điều kiện cần phải kiểm tra, nếu ta dùng cấu trúc rẽ nhánh **If**…**Then** thì đoạn lệnh không được trong sáng, khó kiểm tra, sửa đổi khi có sai sót. Ngược lại với cấu trúc **Select**…**Case**, biểu thức điều kiện sẽ được tính toán một lần vào đầu cấu trúc, sau đó VB sẽ so sánh kết quả với từng trường hợp (**Case**). Nếu bằng nó thi hành khối lệnh trong trường hợp (**Case**) đó.

Select Case <biểu thức kiểm tra>

Case <Danh sách kết quả biểu thức 1>

[Khối lệnh 1]

Case <Danh sách kết quả biểu thức 2>

[Khối lệnh 2] **...**

[**Case Else**

[Khối lệnh n]]

End Select

Mỗi danh sách kết quả biểu thức sẽ chứa một hoặc nhiều giá trị. Trong trường hợp có nhiều giá trị thì mỗi giá trị cách nhau bởi dấu phẩy (**,**). Nếu có nhiều Case cùng thỏa điều kiện thì khối lệnh của Case đầu tiên sẽ được thực hiện.

Ví dụ của lệnh rẽ nhánh **If…Then** ở trên có thể viết như sau:

```
Select Case TheColorYouLike
Case vbRed
MsgBox "You are a lucky person"
Case vbGreen
MsgBox "You are a hopeful person"
Case vbBlue
MsgBox "You are a brave person"
Case vbMagenta
```

```
MsgBox "You are a sad person"
Case Else
MsgBox "You are an average person"
End Select

MsgBox "You are an average person"
End Select
```

5.4 Toán tử Is & To

Toán tử Is: Được dùng để so sánh <Biểu thức kiểm tra> với một biểu thức nào đó.

Toán tử To: Dùng để xác lập miền giá trị của <Biểu thức kiểm tra>.

Ví dụ:

```
Select Case Tuoi
Case Is < 18
      MsgBox "Vi thanh nien"
Case 18 To 30
      MsgBox "Ban da truong thanh, lo lap than di"
Case 31 To 60
      MsgBox "Ban dang o lua tuoi trung nien"
Case Else
      MsgBox "Ban da lon tuoi, nghi huu duoc roi day!"
End Select
```

* *Lưu ý:* Trong ví dụ trên không thể viết **Case** Tuoi < 18.

5.5 Cấu trúc lặp

5.5.1 Lặp không biết trước số lần lặp

5.5.1.1 Câu lệnh Do ... Loop

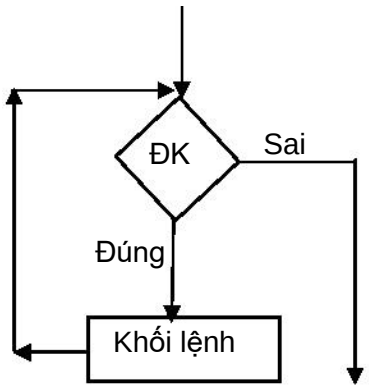
Đây là cấu trúc lặp không xác định trước số lần lặp, trong đó, số lần lặp sẽ được quyết định bởi một biểu thức điều kiện. Biểu thức điều kiện phải có kết quả là True hoặc False. Cấu trúc này có 4 kiểu:

Kiểu 1:

Do While <điều kiện>

<khối lệnh>

Loop



Khối lệnh sẽ được thi hành đến khi nào điều kiện không còn đúng nữa.

Do biểu thức điều kiện được kiểm tra trước khi thi hành khối lệnh, do đó có thể khối lệnh sẽ không được thực hiện một lần nào cả.

Kiểu 2:

Do

<khối lệnh>

Loop While <điều kiện>

Khối lệnh sẽ được thực hiện, sau đó biểu thức điều kiện được kiểm tra, nếu điều kiện còn đúng thì, khối lệnh sẽ được thực hiện tiếp tục. Do biểu thức điều kiện được kiểm tra sau, do đó khối lệnh sẽ được thực hiện ít nhất một lần.

Kiểu 3:

Do Until <điều kiện>

<khối lệnh>

Loop

Cũng tương tự như cấu trúc Do While ... Loop nhưng khác biệt ở chỗ là

khối lệnh sẽ được thi hành khi điều kiện còn sai.

Kiểu 4:

Do

<khối lệnh>

Loop Until <điều kiện>

Khối lệnh được thi hành trong khi điều kiện còn sai và có ít nhất là một lần lặp.

Ví dụ: Đoạn lệnh dưới đây cho phép kiểm tra một số nguyên N có phải là số nguyên tố hay không?

```
Dim i As Integer
i=2
Do While (i <= Sqr(N))   And (N Mod i = 0)
      i=i+1
Loop
If (i > Sqr(N)) And (N   <> 1) Then
      MsgBox Str(N) & "   la so nguyen to"
Else
      MsgBox Str(N) & "   khong la so nguyen to"
End If
```

Trong đó, hàm Sqr: hàm tính căn bậc hai của một số.

5.5.1.2 Câu lệnh While ... End While

Cú pháp:

While < điều kiện>

<khối lệnh>

End While

Cách thức hoạt động của câu lệnh này hoàn toàn giống với Kiểu 1 của câu lệnh Do…Loop.

5.5.2 Lặp biết trước số lần lặp với câu lệnh For…Next

Đây là cấu trúc biết trước số lần lặp, ta dùng biến đếm tăng dần hoặc giảm dần để xác định số lần lặp.

Cú pháp:

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

[khối lệnh]

Next

Biến đếm, điểm đầu, điểm cuối, bước nhảy là những giá trị số (Integer, Single,…). Bước nhảy có thể là âm hoặc dương. Nếu bước nhảy là số âm thì điểm đầu phải lớn hơn điểm cuối, nếu không khối lệnh sẽ không được thi hành.

Khi Step không được chỉ ra, VB.NET sẽ dùng bước nhảy mặc định là một.

Ví dụ: Đoạn lệnh sau đây sẽ hiển thị các kiểu chữ hiện có của máy bạn.

```
Private Sub     Form_Click( )
     Dim i     As       Integer
     For    i = 0 To Screen.FontCount
          MsgBox Screen.Fonts(I)
     Next
End   Sub
```

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.

Cú pháp:

For Each <phần tử> **In** <nhóm>

<khối lệnh>

Next <phần tử>

Lưu ý:

- Phần tử trong tập hợp chỉ có thể là biến Variant, biến Object, hoặc một đối tượng trong Object Browser.

- Phần tử trong mảng chỉ có thể là biến Variant.

- Không dùng For Each ... Next với mảng chứa kiểu tự định nghĩa vì Variant không chứa kiểu tự định nghĩa.

6. Xử lý lỗi

Lỗi có thể phát sinh bất cứ lúc nào. Ví dụ như khi bạn nạp một file mà không có thực trong đĩa thì chương trình sẽ gặp lỗi. VB.NET có khả năng xử lý nhưng nhiệm vụ của bạn là phải thông báo cho VB.NET biết. Chính vì thế khối lệnh *Try…Catch* sẽ bao bọc đoạn mã lệnh có khả năng gây ra lỗi cho chương trình. Thông thường có các lỗi xảy ra do nhập xuất dữ liệu, phép chia cho 0, thiết bị ngoại vi không sẵn sàng.

6.1 Cú pháp Try…Catch

Try

Các phát biểu có thể gây lỗi

Catch

Các phát biểu xử lý nếu có lỗi phát sinh

Finally

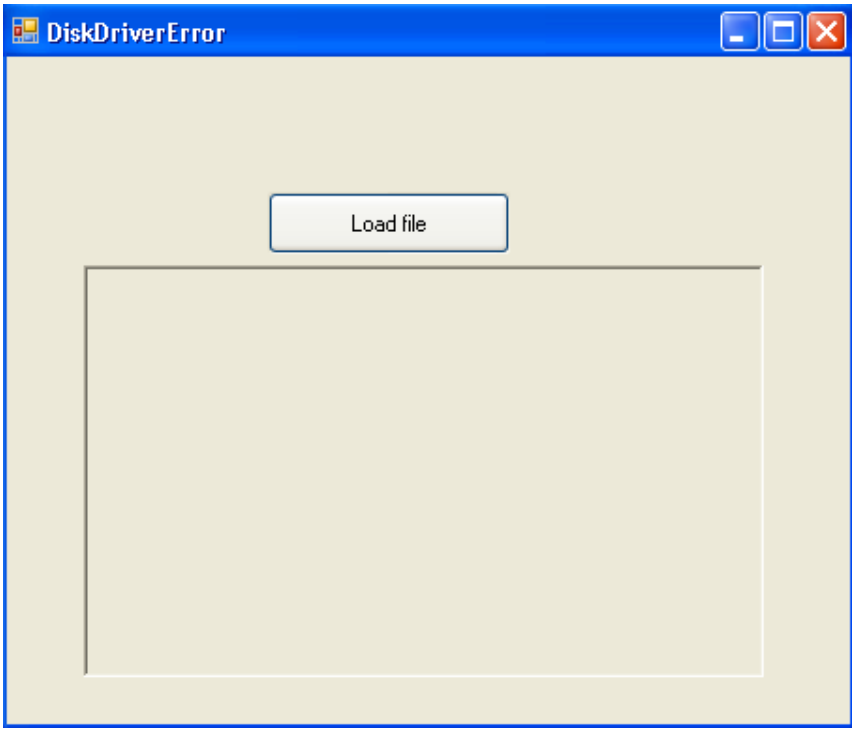
Các phát biểu được gọi ngay cả khi có hay không có lỗi

End Try

Trong đó *Finally* là tùy chọn, các từ khóa còn lại là bắt buộc.

Ví dụ sau *DiskDriverError* sẽ minh họa tình huống xử lý lỗi runtime thường thấy nhất. Chúng ta tạo một form có nút nhấn và một ô ảnh PictureBox. Khi click vào nút thì ảnh trong một đĩa mềm có tên 6_82MELINH.ico sẽ load vào ô ảnh. Nếu bỏ đĩa mềm ra khỏi ổ mềm thì chạy chương trình sẽ báo lỗi không tìm thấy đĩa trong ổ A:\ ngay.

Để minh họa cho việc này, chúng ta mở mới một dự án và thiết kế form như hình:

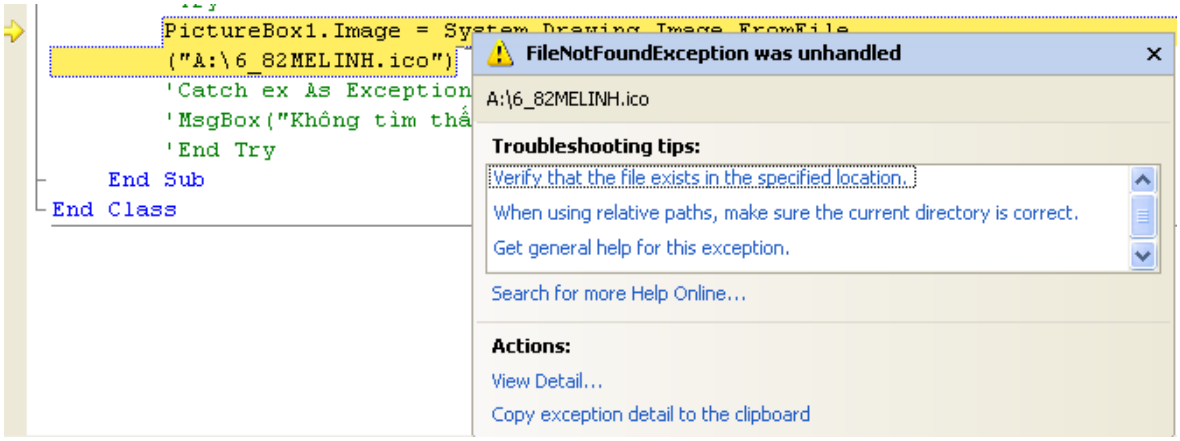


Hình 16

Trong sự kiện Button1_Click, gõ mã như sau:

```
PictureBox1.Image = System.Drawing.Image.FromFile _
        ("A:\6_82MELINH.ico")
```

Lúc này trong ổ mềm không có đĩa nên khi chạy chương trình sẽ có thông báo lỗi xảy ra:



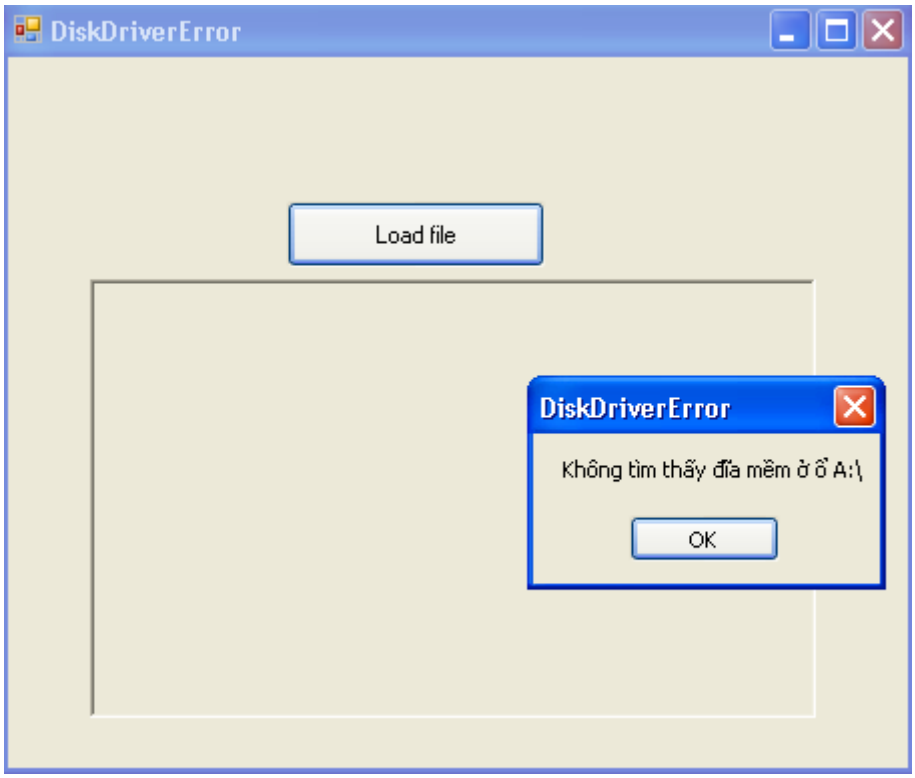
Hình 17

Để khắc phục ta đặt thêm khối try … catch vào sự kiện Button1_click như sau:

```
Try
        PictureBox1.Image = System.Drawing.Image.FromFile _
```

```
            ("A:\6_82MELINH.ico")
    Catch ex As Exception
           MsgBox("Không tìm thấy đĩa mềm ở ổ A:\")
    End Try
```

Lúc này phát biểu gây lỗi PictureBox1.Image = System.Drawing.Image.FromFile _ đã được đặt ở trong khối *Try…Catch* nên khi chạy chương sẽ thực thi hiện thông báo thay vì phát sinh lỗi như trên:



Hình 18

6.2 Sử dụng mệnh đề Finally

Mệnh đề này sẽ cho phép dùng các phát biểu sau nó dù có hay không có lỗi xảy ra. Nó thuận tiện khi bạn muốn dọn dẹp lỗi, giá trị của biến, thuộc tính khi bạn thực thi đoạn mã bảo vệ xong. Trở lại ví dụ trên, ta thêm vào đoạn mã như sau:

```
Try
      PictureBox1.Image = System.Drawing.Image.FromFile _
           ("A:\6_82MELINH.ico")
Catch ex As Exception
      MsgBox("Không tìm thấy đĩa mềm ở ổ A:\")
Finally
      MsgBox("Đã bắt lỗi thành công.")
End Try
```

Và chạy lại chương trình, lúc này thay vì phát sinh lỗi không mong muốn từ chương trình chúng ta sẽ nhận được thông báo lỗi do mình kiểm soát.

6.3 Cài đặt Try…Catch phức tạp hơn

Khi chương trình phức tạp thì việc bắt lỗi cũng trở nên phức tạp hơn. Với Try…Catch bạn có thể:

- Đặt một khối hay nhiều khối phát biểu giữa các từ khóa.
- Cho phép sử dụng mệnh đề lọc lỗi *Catch When*
- Cho phép sử dụng khối *Try…Catch* lồng nhau
- Cùng với đối tượng *Err* cho phép xác định lỗi phát sinh

Err là đối tượng đặc biệt cung cấp chi tiết thông tin lỗi phát sinh. Các thuộc tính thông dụng Err.Number, Err.Description chứa thông tin mã lỗi, mô tả chi tiết lỗi. Phương thức Err.Clear cho phép xóa bỏ lỗi hiện hành. Bảng sau đây liệt kê các lỗi Runtime thường gặp trong VB.NET:

Mã lỗi (Err.Number)	**Mô tả**
5	Gọi hàm hay truyền đối số không đúng
6	Tràn
7	Hết bộ nhớ
9	Truy xuất vượt chỉ số mảng
11	Chia cho 0
13	Kiểu không hợp lệ
48	Lỗi nạp thư viện DLL
51	Lỗi nội bộ
52	Tên File hay số không hợp lệ
53	Không tìm thấy File
55	File đang mở
57	Lỗi thiết bị xuất nhập
58	File đã tồn tại
61	Đĩa đầy
62	Con trỏ file vượt quá điểm cuối file
67	File mở quá nhiều
68	Thiết bị chưa sẵn sàng
70	Không cho phép truy xuất
71	Ổ đĩa chưa sẵn sàng
75	Truy cập đường dẫn và file không đúng
76	Không thấy đường dẫn
91	Biến đối tượng thiếu từ khóa truy xuất *With*

321	Định dạng file không hợp lệ
322	Không thể tạo file tạm
380	Giá trị thuộc tính không hợp lệ
381	Chỉ số thuộc tính không hợp lệ
422	Thuộc tính không tìm thấy
423	Thuộc tính hay phương thức không có
424	Yêu cầu về đối tượng
429	Không thể tạo đối tượng ActiveX
430	Lớp đối tượng không hỗ trợ Automation
440	Không thể tạo đối tượng Automation
460	Định dạng trong Clipboard không hợp lệ
461	Phương thức hay biến thành viên không tìm thấy
462	Server không sẵn sàng
463	Lớp không đăng ký trên máy cục bộ
481	Ảnh không hợp lệ
482	Máy in bị lỗi

Bây giờ vẫn dùng ví dụ trên nhưng ta thêm thuộc tính Err.Number, Err.Description đồng thời ta cũng tìm hiểu thêm về mệnh đề đọc lỗi *Catch When*.

Bạn sửa lại thủ tục Button1_Click như sau:

```
Try
    PictureBox1.Image = System.Drawing.Image.FromFile _
    ("A:\6_82MELINH.ico")
Catch When Err.Number = 53 'nếu không thấy file
    MsgBox("Kiểm tra lại đường dẫn và tên file")
Catch When Err.Number = 7  'Hết bộ nhớ
    MsgBox("File ảnh quá lớn - hết bộ nhớ", ,
     Err.Description)
Catch ex As Exception
    MsgBox("Không tìm thấy đĩa mềm ở ổ A:\", ,
     Err.Description)
Finally
    MsgBox("Đã bắt lỗi thành công.")
End Try
```

Trong khối lệnh trên ta sử dụng mệnh đề *Catch When* hai lần, mỗi lần ta sử dụng thêm các thuộc tính Number của đối tượng Err để phát hiện lỗi cụ thể hơn.

Bạn chạy lại chương trình xem nó hoạt động ra sao.

6.4 Tự mình phát sinh lỗi

Trong một số trường hợp bạn có thể tự kiểm tra lỗi trong mệnh đề *Try* và muốn nhảy ngay đến mệnh đề *Catch* để lỗi được xử lý. Khi đó VB.NET cung cấp phương thức *Err.Raise* để làm điều đó. Ví dụ ta có thể tự phát hiện ra lỗi không tìm thấy *File* ở ví dụ trên (lỗi 53) và thực hiện phát biểu trong mệnh đề Catch:

```
Try
     PictureBox1.Image = System.Drawing.Image.FromFile _
     ("A:\6_82MELINH.ico")
     If Err.Number = 53 Then Err.Raise(53)
Catch When Err.Number = 53
     MsgBox("Không tìm File")
End Try
```

Xác định số lần thử lại

Một trong những đặc sắc của *Try…Catch* là cho phép bạn thử lại một số thao tác gây ra lỗi trước khi đưa ra quyết định không thực hiện thao tác này nữa. Ví dụ ta có thể xem số lần người dùng click vào nút "Load File" bao nhiêu lần, nếu vượt quá số lần cho phép thì không cho người dùng click tiếp nữa:

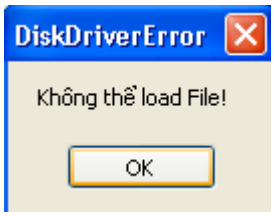
Khai báo thêm biến *dem* ở dưới dòng public class form1:

```
  Dim dem As Short = 0
```

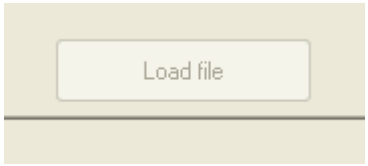
Sửa lại thủ tục Button1_Click như sau:

```
        Try
           PictureBox1.Image = System.Drawing.Image.FromFile _
            ("A:\6_82MELINH.ico")
        Catch ex As Exception
            dem += 1
            If dem <= 2 Then
                MsgBox("Không tìm thấy đĩa mềm ở ổ A:\")
            Else
                MsgBox("Không thể load File!")
                Button1.Enabled = False
            End If
        End Try
```

Và bây giờ khi người dùng click vào nút "Load File" quá hai lần thì thông báo xuất hiện:



Và nút "Load File" sẽ bị mờ đi không cho người dùng click nữa như thế này:



6.5 Sử dụng các khối Try…Catch lồng nhau

Bạn có thể sử dụng các khối Try…Catch lồng nhau để kiểm tra kép các thao tác có thể gây lỗi. Ví dụ bây giờ ta sửa lại ví dụ trên để người dùng phải đưa đĩa mềm vào ổ A:\ ngay từ lần thông báo lỗi đầu tiên, nếu không nút "Load File" lập tức sẽ bị vô hiệu hóa:

```
Try
    PictureBox1.Image = System.Drawing.Image.FromFile _
        ("A:\6_82MELINH.ico")
Catch
    MsgBox("Không tìm thấy đĩa mềm ở ổ A:\, cho đĩa mềm vào")
    Try
       PictureBox1.Image = System.Drawing.Image.FromFile _
                ("A:\6_82MELINH.ico")
    Catch ex As Exception
       MsgBox("Không thể load file!")
       Button1.Enabled = False
    End Try
End Try
```

Bạn nên sử dụng việc lồng hai phát biểu Try…Catch lồng nhau trong trường hợp kiểm tra lại lỗi 2 lần. Còn nếu kiểm tra nhiều lần thì bạn nên sử dụng kết hợp với các biến đếm và vòng lặp *For, Do Loop*.

6.6 So sánh cơ chế xử lý lỗi với các kỹ thuật phòng vệ lỗi

Bạn có thể đoán trước xem lỗi nào có thể xảy ra để phòng trước thay vì xử lý lỗi bằng Try…Catch. Ví dụ trong bài tập trên, thay vì dùng Try ta sẽ dùng

phương thức của hệ thống là *File.Exists* kiểm tra xem có tồn tại file hay không rồi mới gọi phương thức nạp ảnh *FromFile*:

Để dùng được phương thức này, bạn cần khai báo sử dụng thư viện IO bằng từ khóa *Imports* ở đầu khối lệnh:

```
Imports System.IO
```

Rồi sửa lại mã lệnh trong thủ tục Button1_Click như sau:

```
  'Phòng vệ lỗi
If File.Exists("A:\6_82MELINH.ico") Then
         PictureBox1.Image = System.Drawing.Image.FromFile _
          ("A:\6_82MELINH.ico")
Else
          MsgBox("Không tồn tại file này!")
End If
```

Việc sử dụng phương thức nào là do bạn quyết định và trong hoàn cảnh nào thì dùng phương thức nào cho hợp lý.

6.7 Sử dụng phát biểu thoát Exit Try

Phát biểu này là tùy chọn trong khối Try…Catch. Nó giúp bạn thoát khỏi khối Try…Catch khi muốn.

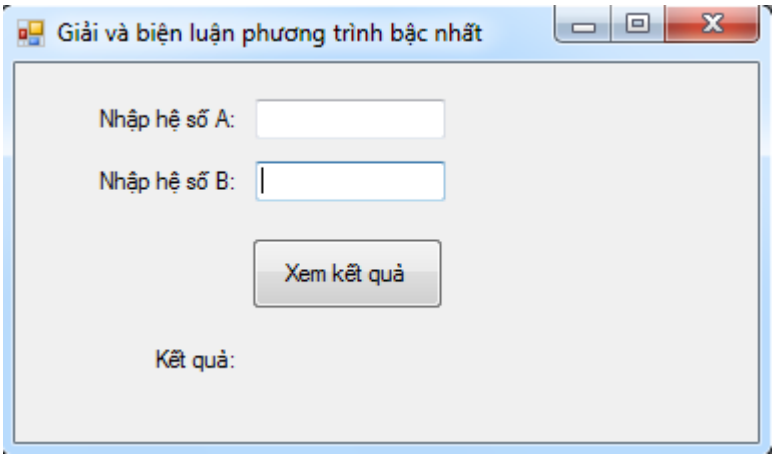
Tuy nhiên nếu trong khối Try…Catch có phát biểu Finally thì chương trình sẽ thực thi các phát biểu trong phần Finally trước khi thoát khỏi khối Try theo yêu cầu của Ext Try.Ví dụ như sau:

```
'Thoát Try với Exit Try
Try
      If PictureBox1.Enabled = False Then Exit Try
      PictureBox1.Image = System.Drawing.Image.FromFile _
            ("A:\6_82MELINH.ico")
Catch ex As Exception
       MsgBox("Không tìm thấy File này!")
End Try
```

Trong đoạn mã trên, nếu chương trình kiểm tra xem điều khiển PictureBox1 mà chưa sẵn sàng thì lập tức thoát khỏi khối Try…Catch mà không thực hiện đưa ra thông báo nào.

7. Bài tập

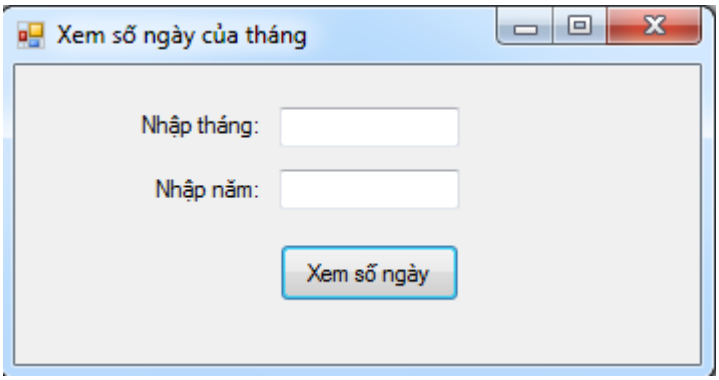
Bài 1: Thiết kế form nhập các hệ số A, B của phương trình bậc nhất $Ax^2+B=0$. Giải và biện luận phương trình theo các hệ số A, B.



Hình 19

Yêu cầu: Nghiệm của chương trình hiện ra ở nhãn (Label) kết quả.

Bài 2: Thiết kế form nhập vào tháng, năm bất kỳ (năm có giá trị từ 1900 đến nay). Sau đó thông báo số ngày trong tháng của năm vừa nhập.



Hình 20

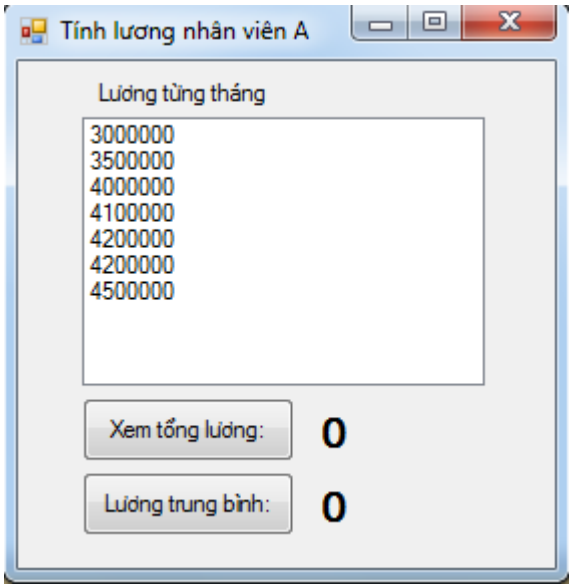
Yêu cầu:

+ Tháng phải là một số từ 1 đến 12.

+ Năm bắt buộc phải nhập số từ 1900 đến 2099.

+ Kết quả hiện ở hộp thoại thông báo riêng.

Bài 3: Thiết kế form nhập vào danh sách lương từng tháng của một nhân viên trong một ListBox. Tính tổng lương và lương trung bình từng tháng của nhân viên đó.



Hình 21

Yêu cầu:

+ Số tháng lương có thể thay đổi mà không ảnh hưởng đến việc xem tổng lương, lương trung bình.

+ Kết quả thể hiện ở nhãn (Label) khi nhấn vào nút lệnh Xem tổng lương hoặc Lương trung bình.

BÀI 3. LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG VISUAL BASIC .NET

Mã Bài: MĐ20_03

Giới thiệu:

Bài này tập trung việc khai báo và sử dụng lớp đối tượng cũng như cách kế thừa trong lớp đối tượng.

Mục tiêu:

- Hiểu đặc điểm lập trình hướng đối tượng trong VB.Net;
- Xây dựng các lớp xử lý, chuyển tải số liệu giữa các form trong VB.Net.

Nội dung chính:

1. Khái niệm hướng đối tượng

1.1 Định nghĩa

Lớp đối tượng (Class) là một khuân mẫu hoặc một bản thiết kế mà định nghĩa các thuộc tính và các phương thức của đối tượng.

Đối tượng (Object) là một bản sao chạy được của một class, sử dụng bộ nhớ và có hạn chế về thời gian.

1.2 Đặt điểm

1.2.1 Tính trừu tượng

Khi bạn mua một tủ lạnh, bạn quan tâm tới kích thước, độ bền và các đặc điểm của nó, chứ không quan tâm tới máy móc của nó được làm như thế nào, đây gọi là sự trừu tượng. VB.Net cũng cung cấp tính trừu tượng qua class và objects.

Mỗi đối tượng có các đặc điểm hoặc thuộc tính gọi là thuộc tính (property) của đối tượng và có thể thực hiện hành động gọi là phương thức (method).

VB.Net cho phép bạn có khả năng tạo các thuộc tính và các phương thức cho các đối tượng khi tạo các class.

Với một lập trình viên, dùng tính trừu tượng để giảm độ phức tạp của đối tượng, chỉ hiện ra các thuộc tính và các phương thức cần thiết cho đối tượng.

Tính trừu tượng cho phép tổng quát hóa một đối tượng như một kiểu dữ liệu.

1.2.2 Tính đóng gói

Được hiểu như việc ẩn thông tin. Nó ẩn những chi tiết không cần thiết của đối tượng.

Ví dụ: Khi bạn bật tủ lạnh, chức năng start bắt đầu nhưng bạn không thể nhìn thấy trong tủ hoạt động như thế nào.

Tính đóng gói là một cách thi hành tính trừu tượng.

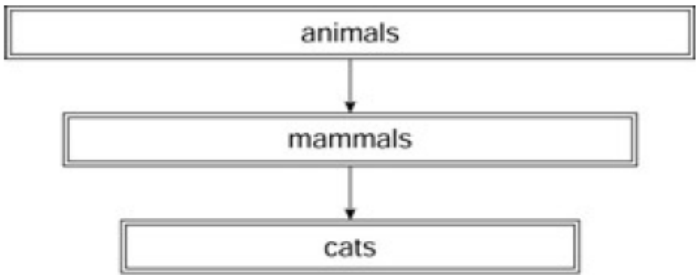
Tính đóng gói ẩn việc thi hành của class đối với người sử dụng. Hay nói cách khác, nó chỉ hiển thị các thuộc tính và các phương thức của đối tượng.

1.2.3 Tính thừa kế

Một class có thể thừa kế từ một class tồn tại. Lớp thừa kế gọi là lớp con (subclass) và lớp được thừa kế gọi là lớp cơ sở (base class).

Tất cả các lớp trong VB.Net đều xuất phát từ lớp Object. Lớp con thừa kế các thuộc tính và các phương thức từ lớp cơ sở. Chúng ta cũng có thể thêm các thuộc tính và phương thức cho lớp con. Bạn cũng có thể chồng các phương thức của lớp cơ sở. Tính thừa kế cho phép bạn tạo phân cấp các đối tượng.

Ví dụ: phân cấp class



Hình 22

Mặc định, tất các các class bạn tạo trong VB.Net có thể được thừa kế. Thừa kế cho phép bạn dùng lại code và tạo các đối tượng phức tạp hơn từ các đối tượng đơn giản. VB.Net cung cấp nhiều từ khóa cho phép bạn thi hành việc thừa kế.

1.2.4 Tính đa hình

Để chỉ một đối tượng tồn tại nhiều khuôn dạng khác nhau.

Ví dụ: Khi bạn mua tủ lạnh có 2 cách, bạn phải liên hệ với người bán hoặc nhà sản xuất. Khi bạn liên hệ với ngươi bán, người bán sẽ đặt hàng và liên hệ với công ty. Khi bạn liên hệ với công ty, công ty sẽ liên hệ với người bán ở vùng của bạn để sắp đặt việc phân phát tủ lạnh.

Như vậy, người bán và công ty là hai class khác nhau. Mỗi class đều có cách phản hồi khác nhau về cùng việc đặt hàng. Chúng ta có thể hiểu như là tính đa hình trong lập trình hướng đối tượng.

Tính đa hình cho phép bạn tạo cùng phương thức nhưng thi hành các công việc khác nhau. Bạn cũng có thay đổi cách thực thi các phương thức của lớp cơ sở.

2. Lập trình hướng đối tượng trong VB.NET

2.1 Tạo một class

Tính trừu tượng được thể hiện bằng việc dùng class, cú pháp tạo class:

[AccessModifier][Keyword] **Class** _ ClassName [Implements InterfaceName]

' Định nghĩa các thuộc tính và phương thức

End Class

AccessModifier định nghĩa khả năng truy cập của class, sử dụng một trong các từ khóa : **Public, Private, Protected, Friend,Protected Friend**.

Keyword chỉ rõ các lớp có được thừa kế hay không, từ khóa **Inherit**, **NotInheritable** hoặc MustInherit.

Class đánh dấu bắt đầu một class

Classname: tên của một class

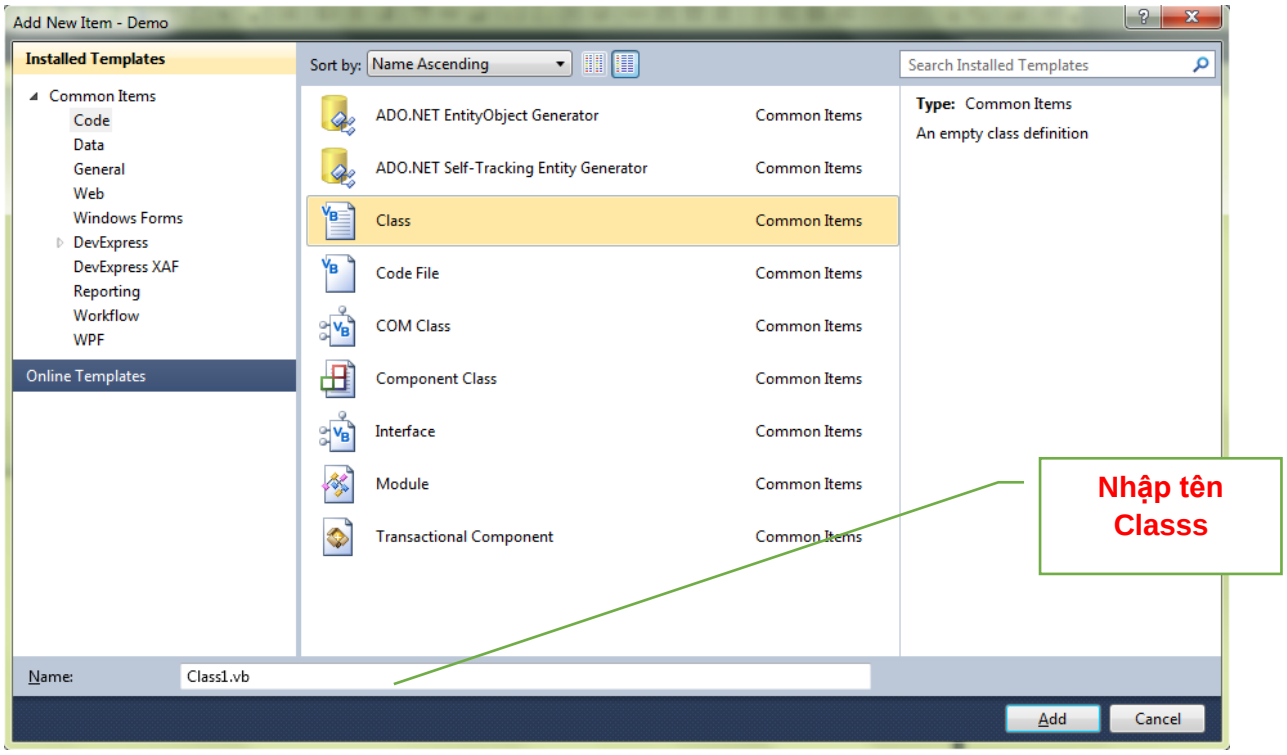
Implements chỉ rõ class thực thi trên giao diện nào.

InterfaceName miêu tả tên giao diện.Một class có thể thực thi trên một hoặc nhiều giao diện.

End Class đánh dấu kết thúc khai báo của một class

Các bước tạo class trong vb.net:

Vào Project \ Add New Item …



Hình 23

Bảng AccessModifier

Access Modifier	Dùng trong	Mô tả
Public	module, class, structure	Được truy cập từ cùng project, từ project khác hoặc từ thành phần khác
Private	module, class, structure	Chỉ được truy cập trong cùng module, class , structure
Protected	Classes, class member	Được truy cập trong cùng class , hoặc class được kế thừa
Friend	module, class, structure	Truy cập được trong cùng project
Protected Friend	Classes, class member	Truy cập được trong cùng project Và từ các class được kế thừa

2.2 Tạo class kế thừa

2.2.1 Tính thừa kế (Inherits)

Cú pháp: **Inherits**

Ví dụ:

```
Public Class ThisClass
Inherits OtherClass
'Property and method declarations
'Other code
End Class
```

Lớp *ThisClass* kế thừa từ lớp *OtherClass*

VB.Net cung cấp các từ khóa khác nhau để thực hiện việc thừa kế.

Bảng Keyword:

Keyword	Được dùng với	Mục đính
Inherits	Classes	Thừa kế tất cả các thành viên của lớp thừa kế (trừ private)

MustInherit	Classes	Chỉ rõ lớp này chỉ sử dụng như lớp cơ sở
NotInheritable	Classes	Chỉ rõ lớp này không được sử dụng như lớp cơ sở
Overridable	Procedures	Chỉ rõ thủ tục có thể viết chồng trong class được thừa kế.
NotOverridable	Procedures	Chỉ rõ thủ tục không được viết chồng trong class được thừa kế.
MustOverride	Procedures	Chỉ định các thủ tục phải viết chồng trong tất cả các lớp được kế thừa
Overrides	Procedures	Chỉ định một thủ tục được viết chồng từ lớp cơ sở
MyBase	Code	Gọi code của lớp cơ sở từ lớp được thừa kế
MyClass	Procedures	Gọi code của chính class đó
Protected	Procedures, fields	Chỉ định các thủ tục và các trường được truy cập trong cùng class và các class được thừa kế

Ví dụ:

```
Public MustInherit Class Communication
    Public Sub New()
        MyBase.New()
```

```
        MsgBox("Constructor of Communication class",
MsgBoxStyle.OKOnly)
    End Sub

    Public MustOverride Function Send() As Boolean
End Class

Public Class Email
    Inherits Communication
    Public Sub New()
        MyBase.New()
        MsgBox("Constructor of Email class",
MsgBoxStyle.OKOnly)
    End Sub

    Overrides Function Send() As Boolean
        MsgBox("Send function of Email class",
MsgBoxStyle.OKOnly)
        'Code specific to the Email class
        Return True
    End Function
End Class
```

Ví dụ 2:

```
Public Class Fax
    Inherits Communication
    Public Sub New()
        MyBase.New()
        MsgBox("Constructor of Fax class", MsgBoxStyle.OKOnly)
    End Sub

    Overrides Function Send() As Boolean
        MsgBox("Send function of Fax class",
MsgBoxStyle.OKOnly)
        'Code specific to the Fax class
        Return True
    End Function
```

```
End Class
```

Ví dụ 3:

```
Private Sub Button1_Click(ByVal sender As Object, ByVal e As
EventArgs) Handles Button1.Click
        Dim int1 As Integer
        Dim communicate As Communication
        int1 = InputBox("Enter 1 to send an e-mail message and
2 to send a faxmessage.")
        Select Case (int1)
            Case "1"
                communicate = New Email()
                communicate.Send()
            Case "2"
                communicate = New Fax()
                communicate.Send()
        End Select
    End Sub
```

Giải thích:

Lớp *Email* và *Fax* kế thừa từ lớp cơ sở *Communication*

Tính đa hình thể hiện thể hiện ở chỗ bạn có thể chồng nhiều phương thức.

Có thể ghi chồng phương thức của lớp cơ sở, nó có thể thực hiện các hành động khác.

2.3 Constructor (Thủ tục khởi tạo)

Được dùng để khởi tạo đối tượng

VB.Net, Thủ tục Sub New thực hiện như một Contructor

Thủ tục Sub New được thực hiện khi đối tượng của class đươc tạo, bạn có thể thực hiện các công việc cần thiết trước khi dùng đối tượng.

Ví dụ khi bạn kết nối với Database, và các biến được khởi tạo trong Sub New

Tất cả các class trong VB.Net đều xuất phát từ class Object. Vì vậy khi tạo một class, bạn cần gọi Contructor của class Object. Để làm việc này bạn thêm câu lệnh MyBase.New() vào dòng đầu tiên của Contructor class của bạn.

```
Public Class MyNewClass
        Public Sub New()
```

```
            MyBase.New()
            'Code for Initializing objects and variables
        End Sub
        'Other class members
    End Class
```

Contructor cũng có thể mang đối số:

```
Public class Employer
    Public Sub New(Optional ByVal iempcode As Integer = 0)
        'code initiation here
    End Sub
    'code here
End Class
```

Sau đó, chúng ta có thể tạo các đối tượng Employer:

```
Dim Emp1 As New Employee(1001)
```

Hoặc

```
Dim Emp2 As Employee = New Employee(1001)
```

2.4 Destructors(Thủ tục khởi hủy)

VB.Net cũng cung cấp thủ tục khởi hủy Sub Finalize. Thủ tục khởi hủy thực hiện ngược lại với hàm khởi tạo, thủ tục sẽ tự động giải phóng bộ nhớ và các tài nguyên đã được sử dụng của đối tượng. Thủ tục Sub Finalize có phạm vi là Protected.

Ví dụ:

```
Protected Overrides Sub Finalize()
    MyBase.Finalize()
    'Add code here
End Sub
```

Từ khóa Overrides được sử dụng bởi vì Thủ tục Sub Finalize() kế thừa từ lớp Object. Thủ tục Sub Finalize() không được gọi khi chạy chương trình, .Net framework sẽ gọi thủ tục này khi đối tượng kết thúc để giải phóng bộ nhớ và tài nguyên đối tượng sử dụng, .Net Framework sẽ tự động thu thập rác, vì vậy người lập trình không cần thêm công việc giải phóng bộ nhớ và tài nguyên.

Ngoài ra .Net Framework cung cấp giao diện IDisposable giúp bạn quản lý tài nguyên. IDisposable cung cấp phương thức Dispose. Không giống thủ tục Sub Finalize(), bạn có thể gọi phương thức Dispose. Bạn có thể thêm code trong

phương thức Dispose để giải phóng tài nguyên. Thủ tục Sub Finalize() đảm bảo rác được dọn dẹp nếu phương thưc Dispose không được gọi.

2.5 Phương thức (Methods)

Phương thức bao gồm thủ tục (Sub) và hàm (Function) được khai báo trong class.

Thủ tục không trả về giá trị. Khi thủ tục được gọi, tất cả các câu lệnh trong thủ tục được thực hiện cho đến khi gặp các câu lệnh End Sub, Exit Sub hoặc Return.

Khai báo phương thức là thủ tục:

Public Sub <tên thủ tục>([danh sách tham số])

//các câu lệnh

End Sub

Hàm thường được sử dụng trong các biểu thức hoặc phép so sánh. Ngoài việc thực hiện các câu lệnh như thủ tục, hàm còn phải trả về một giá trị.

Khai báo phương thức là hàm:

Public Function <tên hàm>([danh sách tham số]) As <kiểu dữ liệu>

//các câu lệnh

Return <giá trị>

End Function

2.6 Trường (Fields) và thuộc tính (Properties)

Fields là các biến được khai báo trong class mà có thể truy cập từ class khác.

Ví dụ:

```
Public Class MyClass1
    Public MyField As Integer 'Declaring a field
    'Other declarations and code
End Class
Dim MyObject As New MyClass1()
MyObject.MyField = 6
```

Properties định nghĩa các thuộc tính của đối tượng, cú pháp tạo Property:

```
Public Property NamProperty() As DataType
    Get
        Return PropertyValue
        'Where PropertyValue is the property's value
    End Get
```

```
    Set(ByVal value As DataType)
        PropertyValue = value
        'Where PropertyValue is the new value to be assigned
    End Set
End Property
```

Mặc định Property có cả hai thuộc tính đọc và ghi. VB.Net cũng định nghĩa thuộc tính chỉ đọc hoặc chỉ ghi.

+ Thuộc tính chỉ đọc dùng từ khóa ReadOnly

+ Thuộc tính chỉ ghi dùng từ khóa WriteOnly

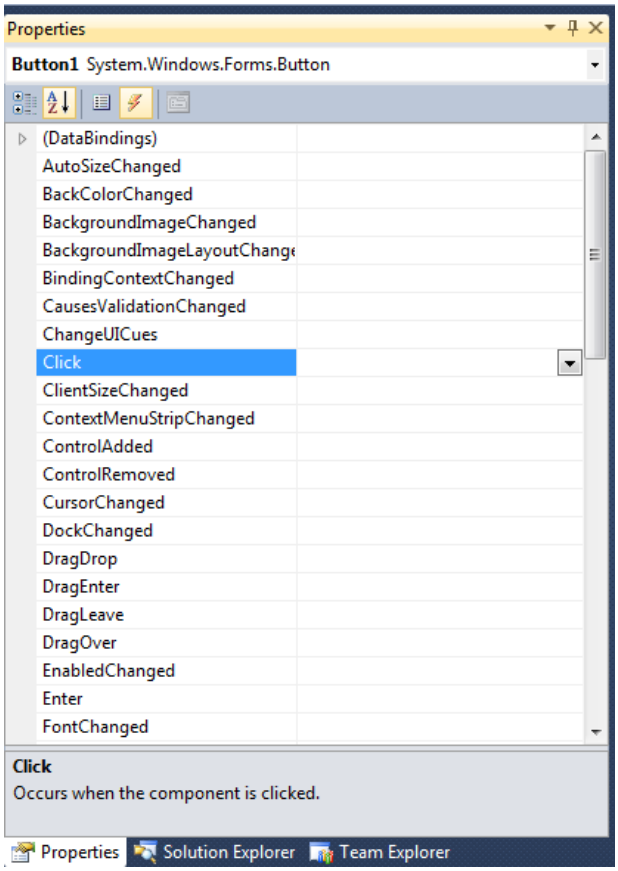
Các từ đặt trước từ khóa AcessModifier.

2.7 Khai báo sự kiện (Event)

- Event là hành động của người dùng tác động vào ứng dụng đang thực thi.

Chẳng hạn khi chúng ta nhấn phím bất kỳ trên bàn phím, khả năng đáp ứng lại sự kiện của ứng dụng phụ thuộc người lập trình. Người lập trình phải viết đoạn mã lệnh, đoạn lệnh này sẽ thực thi khi sự kiện tương ứng xảy ra. Để khai báo sự kiện, chúng ta vào mục Properties của đối tượng.

Ví dụ các sự kiện của một button:



Hình 24

Mỗi mục ứng với các hành động riêng biệt, chẳng hạn sự kiện khi Click vào một button:

```
    Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        'Đặt lệnh ở đây
    End Sub
```

2.8 Từ khóa Me, MyBase, MyClass

2.8.1 Từ khóa Me

Từ khóa **Me** được dùng khi ta muốn nói rõ (explicitly) rằng ta muốn dùng các phương thức của chính Class đang chứa code ấy.

Cũng có trường hợp ta phải dùng từ khóa Me để nói ta muốn dùng biến ở cấp độ lớp (class-level) chứ không phải cấp độ thủ tục (procedure-level) có cùng tên. Một biến procedure-level, tức là biến cục bộ (local variable) của một phương thức, có cùng tên với một biến class-level. Ví dụ:

```
Public Class TheClass
   Private strName As String
   Public Sub DoSomething()
      Dim strName As String
      strName = "Quang"
   End Sub
End Class
```

Ở đây, biến strName được định nghĩa ở class-level và bên trong Sub DoSomething. Bên trong phương thức ấy, local variables sẽ được dùng vì chúng che đậy các biến class-level trừ khi ta nói rõ rằng phải dùng biến của class-level bằng cách dùng từ khóa Me:

```
Public Class TheClass
   Private strName As String
   Public Sub DoSomething()
      Dim strName As String
      strName = "Quang" ' thay đổi value của local (shadowed)
variable
      Me.strName = "Kim" ' thay đổi value của class-level
variable
   End Sub
End Class
```

2.8.2 Từ khóa MyBase

Từ khóa Me rất tiện dụng khi ta muốn dùng các thành viên của chính Class chứa code. Tương tự như vậy, đôi khi ta muốn dùng phương thức của BaseClass (cũng gọi là SuperClass).

Từ trong một SubClass, nếu muốn gọi một phương thức của BaseClass ta dùng từ khóa **MyBase** như sau:

```
Public Class ClassCon
   Inherits ClassCha
   Public Overrides Sub ChàoHỏi()
      MessageBox.Show("Thưa các Bác", "Class Con")
      MyBase.ChàoHỏi()
   End Sub
End Class
```

Bây giờ nếu ta chạy Sub ChàoHỏi của ClassCon ta sẽ có hai thông báo, một cái từ ClassCon theo sau bởi một cái từ ClassCha.

MyBase chỉ nói đến BaseClass trực tiếp, tức là Class cha thôi chớ không nói đến Class ông nội. Không có cách nào để nói đến hơn một thế hệ.

Dầu vậy, từ khóa Mybase có thể được dùng cho bất cứ thứ gì đã được khai báo là Public, Friend hay Protected trong ParentClass. Điều này kể luôn cả những thứ mà ParentClass thừa kế từ các thế hệ trước trong gia đình, tức là ClassÔngNội, ClassÔngCố .v.v..

2.8.3 Từ khóa MyClass

Vì lý do virtual method, ta sẽ gặp những trường hợp rắc rối như khi code của ParentClass lại chạy code của SubClasses.

Khi viết code của một class, từ phương thức này chúng ta thường gọi những phương thức khác nằm trong cùng class. Thí dụ như:

```
Public Class ClassCha
   Public Sub VôĐề()
      ChàoHỏi()
   End Sub
   Public Overridable Sub ChàoHỏi()
      MessageBox.Show("Chào các cháu", "Class Cha")
   End Sub
End Class
```

Trong trường hợp nầy, **VôĐề** gọi **Sub ChàoHỏi** để đón tiếp. Để ý là vì ChàoHỏi được định nghĩa chồng nên rất có thể một SubClass sẽ thực thi phương thức ChàoHỏi và lấn quyền nó. Thí dụ:

```
Public Class ClassCon
   Inherits ClassCha
   Public Overrides Sub ChàoHỏi()
      MessageBox.Show("Thưa các Bác", "Class Con")
   End Sub
End Class
```

Vì đặc tính ảo (virtual) của ChàoHỏi nên ta tưởng ClassCha thực hiện chính thủ tục ChàoHỏi của nó nhưng thật ra ra nó lại thực hiện code của ChàoHỏi trong ClassCon. Trong code dưới đây, một đối tượng ClassCon gọi thủ tục VôĐề của ClassCha:

```
Private Sub BtnSubClassObject_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
BtnSubClassObject.Click
   Dim obj As New ClassCon()
   obj.VôĐề()
End Sub
```

Trong ClassCha, thủ tục VôĐề gọi ChàoHỏi của chính nó, tuy nhiên thủ tục ChàoHỏi ấy bị đè bởi thủ tục ChàoHỏi trong ClassCon. Do đó, chương trình sẽ hiển thị thông báo "**Thưa các Bác**".

Nếu ta không muốn như vậy, ta muốn VôĐề thực hiện chính code của ChàoHỏi trong ClassCha thì phải dùng từ khóa **MyClass** như sau:

```
Public Class ClassCha
   Public Sub VôĐề()
      MyClass.ChàoHỏi()
   End Sub
   Public Overridable Sub ChàoHỏi()
      MessageBox.Show("Chào các cháu", "Class Cha")
   End Sub
End Class
```

2.9 Giao diện (Interface)

Interface là một phương pháp lập trình giúp phân lập phần vận hành và phần giao tiếp. Nói một cách đơn giản, interface là bộ mặt của một thành phần để

"giao tiếp" với các thành phần khác ở ngoài. Các thành phần bên ngoài chỉ cần biết thành phần này cung cấp chức năng gì, xài ra sao, mà không cần quan tâm là nội bộ nó thế nào.

Cũng như cái đầu máy, đầu máy nào cũng có nút Play, nút Stop, người sử dụng không cần biết là ở trong mạch điện nó chạy làm sao. Người ta chỉ cần biết là có nút Play bấm thì hát, Stop bấm thì im. Bộ nút Play, Stop đó chính là interface mà tất cả đầu máy đều cài đặt.

Ví dụ:

```
' Giao diện cho bộ nút bấm
Interface IDauMay
    Sub PressPlay()
    Sub PressStop()
End Interface
' Đầu máy Sony cài đặt bộ nút bấm
Class DauMaySony
    Implements IDauMay
    Public Sub PressPlay() Implements IDauMay.PressPlay
        ' lệnh Play
    End Sub
    Public Sub PressStop() Implements IDauMay.PressStop
        ' lệnh Stop
    End Sub
End Class
' Đầu máy Panasonic cài đặt bộ nút bấm
Class DauMayPanasonic
    Implements IDauMay
    Public Sub PressPlay() Implements IDauMay.PressPlay
        ' lệnh Play
    End Sub
    Public Sub PressStop() Implements IDauMay.PressStop
        ' lệnh Stop
    End Sub
End Class
Class Demo
    Public Shared Sub Main()
        Dim dauMay As IDauMay = New DauMaySony()
```

```
        Test(dauMay)
    End Sub
  Shared Sub Test(Dim dauMay As IDauMay)
        ' Kiểm tra chức năng PressPlay.
        dauMay.PressPlay()
    End Sub
End Class
```

3. Xây dựng các lớp xử lý

Trong phần này, chúng ta sẽ tìm hiểu về mô hình lập trình 03 tầng/lớp (3 layers)

và cách xây dựng ứng dụng trên mỗi tầng/lớp.

3.1 Mô hình đa tầng

Trong phát triển ứng dụng, để dễ quản lý các thành phần của hệ thống, cũng như không bị ảnh hưởng bởi các thay đổi, người ta hay nhóm các thành phần có cùng chức năng lại với nhau và phân chia trách nhiệm cho từng nhóm để công việc không bị chồng chéo và ảnh hưởng lẫn nhau. Ví dụ trong một công ty bạn có từng phòng ban, mỗi phòng ban sẽ chịu trách nhiệm một công việc cụ thể nào đó, phòng này không được can thiệp vào công việc nội bộ của phòng kia như Phòng tài chính thì chỉ phát lương, còn chuyện lấy tiền đâu phát cho các anh phòng Marketing thì các anh không cần biết. Trong phát triển phần mềm, người ta cũng áp dụng cách phân chia chức năng này. Bạn sẽ nghe nói đến thuật ngữ kiến trúc đa tầng/nhiều lớp, mỗi lớp sẽ thực hiện một chức năng nào đó, trong đó mô hình 3 lớp là phổ biến nhất. 3 lớp này là **Presentation**, **Business Logic**, và **Data Access**. Các lớp này sẽ giao tiếp với nhau thông qua các dịch vụ(services) mà mỗi lớp cung cấp để tạo nên ứng dụng, lớp này cũng không cần biết bên trong lớp kia làm gì mà chỉ cần biết lớp kia cung cấp dịch vụ gì cho mình và sử dụng nó mà thôi.

Mô hình 3 lớp mà Microsoft đề nghị dùng cho các hệ thống phát triển trên nền .NET như sau:

3.1.1 Presentation Layer

Lớp này làm nhiệm vụ giao tiếp với người dùng cuối để thu thập dữ liệu và hiển thị kết quả/dữ liệu thông qua các thành phần trong giao diện người sử dụng. Lớp này sẽ sử dụng các dịch vụ do lớp Business Logic cung cấp. Trong .NET thì bạn có thể dùng**Windows Forms**, **ASP.NET** hay **Mobile Forms** để hiện thực lớp này.

Trong lớp này có 2 thành phần chính là **User Interface Components** và **User Interface Process Components**.

UI Components là những phần tử chịu trách nhiệm thu thập và hiển thị thông tin cho người dùng cuối. Trong ASP.NET thì những thành phần này có thể là các TextBox, các Button, DataGrid…

UI Process Components: là thành phần chịu trách nhiệm quản lý các qui trình chuyển đổi giữa các UI Components. Ví dụ chịu trách nhiệm quản lý các màn hình nhập dữ liệu trong một loạt các thao tác định trước như các bước trong một Wizard…

Lưu ý *: lớp này không nên sử dụng trực tiếp các dịch vụ của lớp Data Access mà nên sử dụng thông qua các dịch vụ của lớp Business Logic vì khi bạn sử dụng trực tiếp như vậy, bạn có thể bỏ qua các ràng buộc, các logic nghiệp vụ mà ứng dụng cần phải có.*

3.1.2 Business Logic Layer

Lớp này thực hiện các nghiệp vụ chính của hệ thống, sử dụng các dịch vụ do lớp **Data Access** cung cấp, và cung cấp các dịch vụ cho lớp **Presentation**. Lớp này cũng có thể sử dụng các dịch vụ của các nhà cung cấp thứ 3 (3rd parties) để thực hiện công việc của mình(ví dụ như sử dụng dịch vụ của các cổng thanh tóan trực tuyến như VeriSign, Paypal…).

Trong lớp này có các thành phần chính là **Business Components**, **Business Entities** và **Service Interface**.

Service Interface là giao diện lập trình mà lớp này cung cấp cho lớp **Presentation** sử dụng. Lớp **Presentation** chỉ cần biết các dịch vụ thông qua giao diện này mà không cần phải quan tâm đến bên trong lớp này được hiện thực như thế nào.

Business Entities là những thực thể mô tả những đối tượng thông tin mà hệ thống xử lý. Trong ứng dụng chúng ta các đối tượng này là các chuyên mục(**Category**) và bản tin(**News**). Các business entities này cũng được dùng để trao đổi thông tin giữa lớp**Presentation** và lớp **Data Access**.

Business Components là những thành phần chính thực hiện các dịch vụ mà **Service Interface** cung cấp, chịu trách nhiệm kiểm tra các ràng buộc logic(constraints), các qui tắc nghiệp vụ(business rules), sử dụng các dịch vụ bên ngoài khác để thực hiện các yêu cầu của ứng dụng.

Trong ứng dụng của chúng ta, lớp này sẽ chứa các thành phần là **CategoryService** và **NewsService** làm nhiệm vụ cung cấp các dịch vụ quản lý chuyên mục và các bản tin (thêm, xóa, sửa, xem chi tiết, lấy danh sách…).

3.1.3 Data Access Layer

Lớp này thực hiện các nghiệp vụ liên quan đến lưu trữ và truy xuất dữ liệu của ứng dụng. Thường lớp này sẽ sử dụng các dịch vụ của các hệ quản trị cơ sở dữ liệu như SQL Server, Oracle,… để thực hiện nhiệm vụ của mình. Trong lớp này có các thành phần chính là **Data Access Logic**, **Data Sources**, **Servive Agents**).

Data Access Logic components (DALC) là thành phần chính chịu trách nhiệm lưu trữ vào và truy xuất dữ liệu từ các nguồn dữ liệu – Data Sources như RDMBS, XML, File systems…. Trong .NET Các **DALC** này thường được hiện thực bằng cách sử dụng thư viện ADO.NET để giao tiếp với các hệ cơ sở dữ liệu hoặc sử dụng các O/R Mapping Frameworks để thực hiện việc ánh xạ các đối tượng trong bộ nhớ thành dữ liệu lưu trữ trong CSDL. Chúng ta sẽ tìm hiểu các thư viện O/R Mapping này trong một bài viết khác.

Service Agents là những thành phần trợ giúp việc truy xuất các dịch vụ bên ngòai một cách dễ dàng và đơn giản như truy xuất các dịch vụ nội tại.

Chúng ta đã tìm hiểu qua các lớp của mô hình 3 lớp. Lý thuyết hơi nhiều một chút có thể làm bạn khó hiểu vì khả năng trình bày có hạn, nên bây giờ thử tìm hiểu một qui trình cụ thể hơn để biết các lớp này giao tiếp với nhau như thế nào. Ví dụ trong ứng dụng của chúng ta có thao tác tạo một chuyên mục mới, thì các lớp sẽ tương tác với nhau như sau:

Lớp Presentation

- Trình bày một form, có các text box cho phép người sử dụng nhập tên và mô tả cho chuyên mục.

- Khi người dùng nhấn nút tạo trên form này, ứng dụng sẽ thực hiện việc tạo một Business Entity **Category** mới như đoạn code sau minh họa:

```
Public Sub CreateNewCategory()
Dim theloai  As New Category()
theloai.Name = txtName.Text
theloai.Description = txtDescription.Text
' sử dụng dịch vụ do lớp Business cung cấp để tạo chuyên mục
CategoryService.CreateCategory(theloai)
End Sub
```

Lớp Business Logic

Để cung cấp dịch vụ tạo một chuyên mục, thành phần **CategoryService** sẽ cung cấp hàm sau:

```
Public  Sub TaoTheLoai(theloai as Category)
' kiểm tra xem tên khóa của chuyên mục đã tồn tại chưa?
…
' kiểm tra tên khóa của chuyên mục có hợp lệ không?
…
' sử dụng DV của lớp Data Access để lưu chuyên mục mới này vào
CSDL
Dim theloaiDB As new CategoryDAO()
theloaiDB.CreateCategory(theloai)
End Sub
```

Lớp Data Access

Tương tự, để cung cấp dịch vụ lưu một chuyên mục mới vào CSDL, thành phần **CategoryDAO** sẽ cung cấp hàm sau (sử dụng ADO.NET để kết nối với CSDL):

```
Public Sub CreateCategory(Byval theloai As Category)
' tạo connection
…
' tạo command, khởi tạo các tham số…
…
' lưu dữ liệu
cmd.ExecuteNonQuery()
End Sub
```

3.2 Phân tích và thiết kế

Chúng ta đã tìm hiểu qua các thành phần chính trong mô hình 3 lớp, giờ đến lúc bắt tay vào thiết kế các thành phần đó cho ứng dụng tin tức của chúng ta. Trong ứng dụng tin tức mà chúng ta đã tìm hiểu yêu cầu qua bài viết trước, chúng ta thấy có hai đối tượng thông tin chính mà chúng ta cần quản lý là các chuyên mục(category) và tin tức(news). Ứng dụng quản lý của chúng ta sẽ thu thập những đối tượng dữ liệu này từ người dùng(phóng viên, biên tập viên) và trình bày lại cho người sử dụng khác xem(độc giả). Giờ chúng ta bắt tay vào thiết kế các thành phần **Business Entities**.

3.2.1 Business Entities

Ứng dụng của chúng ta sẽ bao gồm hai thực thể (entity) chính là Category và News.

Trước hết là Category, một chuyên mục (Category) sẽ gồm những thông tin sau:

CategoryId: Mã chuyên mục – sẽ được sinh tự động khi tạo mới.

Name: Tên chuyên mục. VD: Vi tính, Kinh tế…

KeyName: Tên gợi nhớ dùng để phân biệt chuyên mục với nhau (không được trùng nhau). Chẳng hạn Vi-tinh, Suc-khoe…

Description: Mô tả cho chuyên mục. Ví dụ: Description cho Vi-tinh là: thông tin mới nhất về công nghệ thông tin của Việt Nam và thế giới…

Picture: Hình ảnh đại diện cho chuyên mục.

Trong ứng dụng đơn giản này, chúng ta chỉ thiết kế chuyên mục có một cấp, không có các chuyên mục con, cháu…

Tiếp theo là **News**. Mỗi một bản tin sẽ có các thông tin sau:

NewsId: mã bản tin. Sẽ được sinh tự động khi tạo mới.

Title: tiêu đề chính của bản tin. Ví dụ: Microsoft tuyên bố phá sản!

Subtitle: tiêu đề phụ của bản tin. Ví dụ: Bill Gates từ chức!

Excerpt: phần giới thiệu ngắn gọn của bản tin

Authors: danh sách tác giả bản tin. Ví dụ: Nguyễn Văn A, Hoàng Thị B

Keywords: danh sách từ khóa chính trong bản tin dùng để tìm kiếm. Ví dụ: Microsoft, broken

Body: Đây là phần nội dung chính của bản tin.

Picture: Hình ảnh minh họa cho bản tin.

CreationTime: Ngày giờ tạo bản tin

LastModificationTime: Ngày giờ chỉnh sửa cuối cùng của bản tin

PublishedTime: Ngày giờ bản tin được đăng

TotalViews: Tổng số lượt người xem bản tin

TotalRates: Tổng số lượt người đánh giá bản tin

Rate: Điểm đánh giá trung bình của bản tin

Status: Trạng thái hiện tại của bản tin.

Business Service Components

Bước tiếp theo chúng ta sẽ phân tích và thiết kế các Business Service Components. Các thành phần này sẽ làm nhiệm vụ chính cung cấp các dịch vụ cho lớp Presentation dùng để lấy và lưu trữ thông tin.

3.2.2 Lớp CategoryService

Chúng ta cần những thao tác chính trên đối tượng dữ liệu Category:

Tạo mới – **CreateCategory(Category category)**

Cập nhật – **UpdateCategory(Category category)**

Xóa – **DeleteCategory(int categoryId)**

Lấy thông tin chi tiết – **GetCategory(int categoryId)**

Lấy danh sách các category – **GetCategories()**

Kiểm tra một Key xem có trong database chưa – **CheckKey(string keyName)**.

Thao tác này dùng để kiểm tra xem khi tạo mới một category thì KeyName đã tồn tại trong hệ thống chưa. Thao tác này có thể dùng trên lớp Presentation để kiểm tra và thông báo lỗi cho người dùng khi họ nhập một tên khóa đã có trong hệ thống.

Lưu ý: Chúng ta sẽ thực hiện các business logic của hệ thống trong lớp này.

Tương tự đối với lớp **NewsService**, dựa trên những gì yêu cầu chúng ta phân tích ở bài viết đầu tiên, chúng ta cần những thao tác chính sau đây trên đối tượng News:

Tạo mới – **CreateNews(News news)**

Cập nhật – **UpdateNews(News news)**

Xóa – **DeleteNews(int newsId)**

Lấy thông tin chi tiết – **GetNews(int newsId)**

Lấy danh sách các bản tin thuộc một chuyên mục nào đó, sắp xếp theo tin mới nhất –**GetNewsOfCategory(int categoryId, int page, int pageSize, out int totalRecords)**

Cập nhật số lần xem của một bản tin – **UpdateTotalViews(int newsId)**

Cập nhật đánh giá cho một bản tin – **UpdateRate(int newsId, int rate)**

Tìm bài viết dựa trên từ khóa – **SearchNews(string keyWords, int page, int pageSize, out int totalRecords)**

Trong các hàm trên, các bạn chú ý đến hàm GetNewsOfCategory. Trong hàm này có các tham số dùng để phân trang các bản tin. Chúng ta cần đến chức năng này vì khi trình bày trên trang tin, chúng ta chỉ trình bày một số lượng có hạn các bản tin của một chuyên mục nào đó chứ không thể trình bày tất cả trên màn hình được. Khi người dùng muốn xem thêm, họ có thể chọn trang tiếp theo hoặc nhấp vào link **Xem tiếp**, lúc đó ứng dụng sẽ trình bày các bản tin ở các trang tiếp theo. Tham số **totalRecords** cho chúng ta biết được tổng số bản tin thật sự có trong chuyên mục đó.

3.2.3 Data Access Components

Bây giờ chúng ta sẽ thiết kế các lớp dùng để truy xuất và cập nhật dữ liệu. Các hàm của các lớp DAO cũng khá đơn giản, chỉ làm nhiệm vụ cập nhật dữ liệu vào database và truy xuất dữ liệu từ database. Các bạn cũng thấy chức năng nó giống như trên lớp Business Logic, nhưng ở đây chúng ta không có bất kỳ ràng buộc logic gì, chỉ đơn giản thực hiện việc truy xuất dữ liệu mà thôi. Các business logic đã được kiểm tra trên lớp Business Logic.

Chúng ta tách biệt 2 lớp CategoryDAO (DAO – Data Access Object) và NewsDAO để dễ quản lý và thay đổi khi cần thiết. Ví dụ nếu bạn muốn thêm một thao tác truy xuất dữ liệu mới trên đối tượng News, bạn sẽ biết ngay mình phải thay đổi lớp NewsDAO. Nhưng có những thao tác bạn phải cân nhắc nên để nó lớp nào vì nó liên quan đến nhiều đối tượng, lúc đó bạn phải xét xem mục đích chính của thao tác đó là gì, thao tác trên đối tượng dữ liệu chính nào để đưa thao tác đó vào lớp phù hợp.

3.3 Thiết kế cơ sở dữ liệu

Do ứng dụng của chúng ta đơn giản nên chỉ có 2 bảng dữ liệu ánh xạ gần như 1-1 với các thực thể trên lớp Business Logic như sau:

3.3.1 Hiện thực lớp Business Logic & Data Access

Sau khi đã xong bước thiết kế, chúng ta sẽ tiến hành hiện thực 2 lớp Business và Data Access. Các bạn có thể xem source code đính kèm để biết chi tiết cách hiện thực 2 lớp này như thế nào. Sau đây tôi sẽ trình bày một số điểm chính trong cách hiện thực.

3.3.2 Hiện thực Data Access Components

Do ứng dụng của chúng ta đơn giản nên được giới hạn sẽ dùng với CSDL SQL Server 2005/2008 nên lớp này không được thiết kế để chạy cùng lúc với nhiều loại database khác nhau. Chúng ta sẽ dùng cái Stored Procedures để truy xuất dữ liệu an toàn và dễ thay đổi hơn, tránh bị các lỗi như SQL Injection(không thể tránh hoàn toàn nếu bạn không hiện thực đúng)

```
Public Sub CreateCategory(Byval theloai As Category)
Dim conn As SqlConnection= GetConnection()
Dim cmd As New SqlCommand ("spCategoriesCreate", conn)
cmd.CommandType = CommandType.StoredProcedure
cmd.Parameters.AddWithValue("@KeyName", theloai.KeyName)
cmd.Parameters.AddWithValue("@Name", theloai.Name)
cmd.Parameters.AddWithValue("@Description",
theloai.Description)
```

```
cmd.Parameters.AddWithValue("@Picture", theloai.Picture)
conn.Open()
cmd.ExecuteNonQuery()
End Sub
```

Stored Procedure **spCategoriesCreate** đơn giản được viết như sau:

```
CREATE PROCEDURE dbo.spCategoriesCreate
@Name nvarchar(50),
@KeyName varchar(30),
@Description ntext,
@Picture varchar(256),
@CategoryId int output
AS
INSERT INTO Categories
(
Name,
KeyName,
Description,
Picture
)
VALUES
(
@Name,
@KeyName,
@Description,
@Picture
)
SELECT @CategoryId = SCOPE_IDENTITY()
```

3.3.3 Hiện thực lớp Business Logic

Hiện thực lớp **Business Logic** đòi hỏi bạn phải nắm rõ các business logic của ứng dụng. Ví dụ đối với ứng dụng tin tức của chúng ta thì khi tạo một chuyên mục mới, bạn phải kiểm tra xem **KeyName** của chuyên mục đó đã có trong hệ thống chưa? Nếu có rồi thì phải báo lỗi, và nếu chưa có thì chúng ta kiểm tra **KeyName** đó có hợp lệ hay không?

```
Public Sub ThemTheLoai(Byval theloai As Category)
If CheckKey(theloai.KeyName)=False Then
```

```
        'Thông báo lỗi
        Exit Sub
End If
If ValidateKey(category.KeyName)=False Then
        'Thông báo lỗi
        Exit Sub
End If
Dim theloaiDB As New CategoryDAO()
theloaiDB.CreateCategory(theloai)
End Sub
```

4. Bài tập

Bài tập 1: Thông tin về một cuốn sách được khai báo với cú pháp như sau:

```
Public Class Sach
        Private _SoLuong As Integer      'Kiểu số nguyên
        Private _DonGia As Double        'Kiểu số thực
Public SoTrang As Integer                'Kiểu số nguyên
Public Property SoLuong() As Integer
        Set  (ByVal Value As Integer)
          _SoLuong = Value
         End Set
         Get
          SoLuong = _SoLuong
        End Get
        End Property
Public Property DonGia() As Double
        Set  (ByVal Value As Double)
          _DonGia = Value
         End Set
         Get
          DonGia = _DonGia
        End Get
        End Property
        Public Function ThanhTien()  As Double
                ThanhTien = ( _SoLuong * _DonGia )
         End Sub
End Class
```

a) Hãy tạo lớp đối tượng sách trên.

b) Xây dựng lớp đối tượng TapChi (tạp chí) kế thừa lớp Sách và có thêm các thành phần:

+ Trường: Ngày phát hành

+ Thuộc tính: Hình bìa, Khổ giấy

+ Phương thức: Năm phát hành, Hiển thị

Bài tập 2: Xây dựng và hiện thực mô hình 3 lớp (3-tier) để thực hiện công việc thêm mới một tạp chí vào cơ sở dữ liệu.

BÀI 4. LÀM VIỆC VỚI DỰ ÁN CÓ NHIỀU FORM

Mã Bài: MĐ20_04

Giới thiệu:

Việc khai thác các thuộc tính của một số công cụ xây dựng dựng sẵn trong .NET giúp người lập trình sử dụng hiệu quả các điều khiển để thiết kế giao diện trực quan và đẹp mắt.

Mục tiêu:

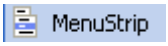
- Hiểu được đặc tính của các điều khiển hiển thị dữ liệu;
- Biết quy trình thiết kế các dạng biểu mẫu;
- Sử dụng được các điều khiển cơ bản;
- Khai báo và sử dụng được các thành phần của Module
- Kết nối và hiển thị được cơ sở dữ liệu Access bằng DataGridview;
- Nghiêm túc, sáng tạo, chủ động trong việc thiết kế và kế thừa các dạng biểu mẫu khác nhau.

Nội dung chính:

1. Thiết kế thực đơn bằng MenuStrip

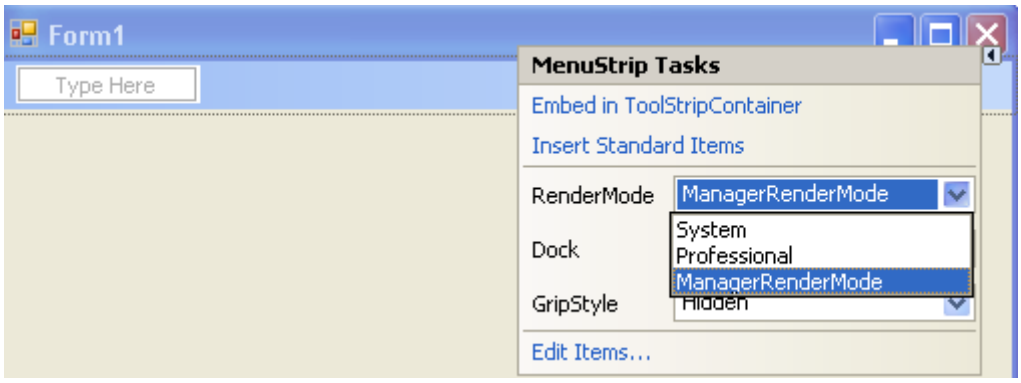
1.1 Tạo Menu

Tạo mới một giải pháp mang tên *MyMenu* và thêm vào đó một dự án mới cùng tên như đã biết trong các bài tập trước.

Tại giao diện thiết kế, các bạn đưa điều khiển *MenuStrip* 

vào trong Form bằng cách double click hay kéo thả như đã biết.

Chúng ta không cần quan tâm đến vị trí của menu trên form vì VS sẽ tự động đặt nó sao cho phù hợp. Các bạn có thể thay đổi các thuộc tính sao cho phù hợp bằng cách click mở *Smart Tags* là nút mũi tên tam giác màu đen bên góc phải điều khiển *Menu.*



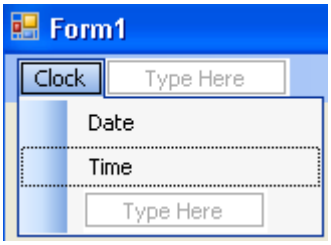
Hình 25

Khi được đặt vào form thì điều khiển menu sẽ được đặt tại một vùng như trên hình gọi là *khay công cụ - Component tray* và VS sẽ hiển thị trực quan menu trên đầu cửa sổ Form.

Chuỗi *Type Here* là nơi bạn có thể click chọn và nhập vào các mục chọn cho menu, chúng ta sẽ tạo ra menu đơn giản như sau:

Nhắp chuột vào chuỗi *Type Here* và gõ vào chuỗi "Clock" và ấn enter.

Nhắp chuột vào chuỗi *Type Here* con ở dưới rồi gõ Date, Time như hình



Hình 26

Để đóng phần thiết kế menu, chúng ta click vào một vùng nào đó trên form, để hiển thị lại click vào menu *Clock* như trên.

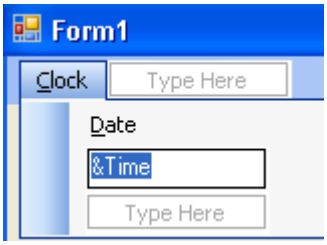
1.2 Một số tùy biến cho Menu

1.2.1 Thêm phím truy cập vào các mục chọn lệnh trên menu

Trong một số phần mềm hay ngay trình duyệt Windows Explorer của hệ điều hành các bạn có thể ấn tổ hợp *Alt + phím tắt* để mở nhanh một thực đơn nào đó. Các phím tắt ấy được gọi là phím truy cập – *Access Key*. Phím này có dấu gạch chân ở dưới.

Trong .NET, để tạo phím này ở menu chúng ta chỉ việc gõ thêm dấu '&' trước ký tự nào muốn hiển thị gạch chân trong phần *Type Here*.

Ví dụ sau đây tạo ra các phím tắt cho các mục chọn của menu *Clock* như hình:



Hình 27

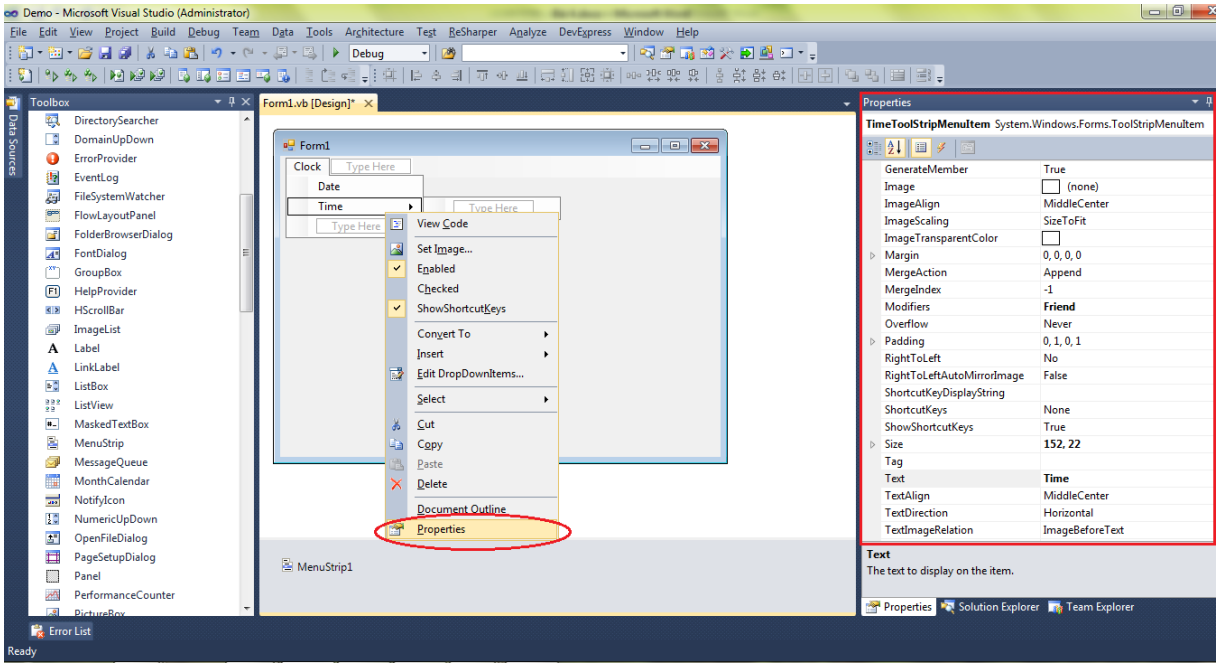
1.2.2 Thay đổi thứ tự các mục chọn

Để thay đổi thứ tự các mục chọn chúng ta mở chế độ thiết kế menu rồi nhắp chọn mục chọn nào đó và kéo nó đến vị trí mong muốn.

Chẳng hạn để kéo mục chọn *Time* lên thay cho vị trí mục chọn *Date*, trước tiên chúng ta nhắp và giữ chuột mục chọn *Time* và kéo thả lên phía trên mục chọn *Date* trong menu.

1.2.3 Đặt tên và thuộc tính cho menu

Để thay đổi thuộc tính tên cho 1 menu, chọn Properties của menu đó.



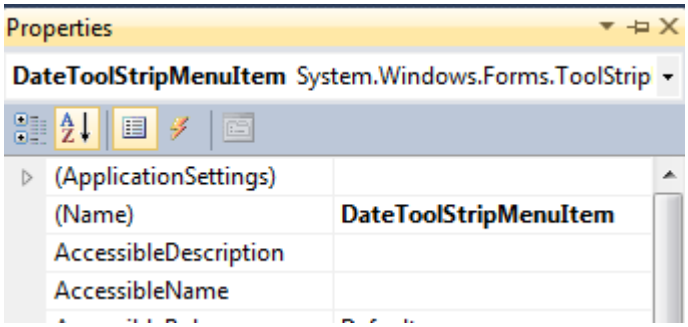
Hình 28

Ví dụ:

- Thay đổi tên menu: vào thuộc tính Text.
- Ví trí tên menu : TextAlign.
- Màu nền: BackColor.
- Màu chữ: ForeColor, …

Để thay đổi thuộc tính nào chúng ta chọn vào thuộc tính đó và thay đổi giá trị mới tùy theo kiểu giá trị của thuộc tính.

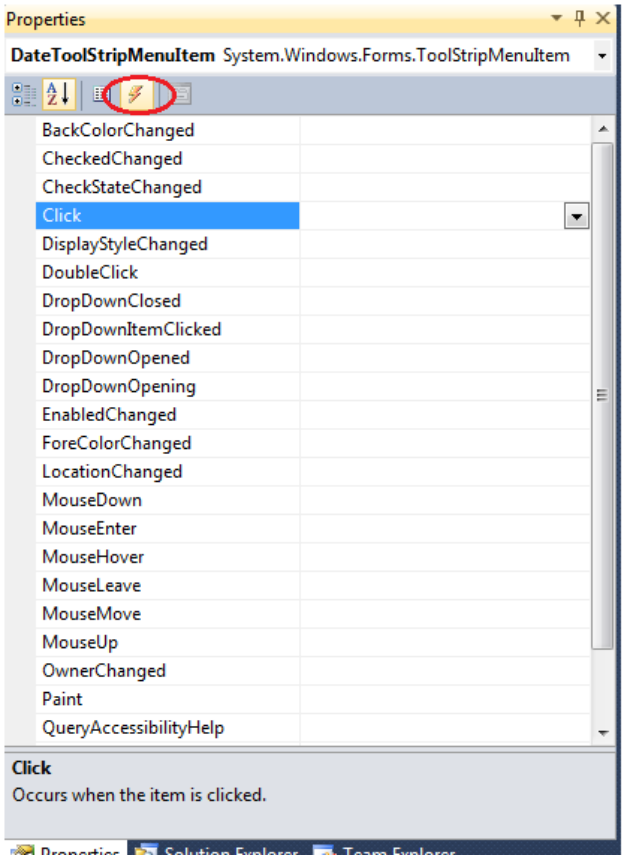
Vì vậy, để đặt tên cho control hay một menu cụ thể chúng ta vào thuộc tính Name:



Tên hiện tại là "DateToolStripMenuItem" có thể thay đổi theo ý muốn của người lập trình. Thuộc tính Name được gọi trong quá trình lập trình các sự kiện để thực thi ứng dụng.

1.3 Viết lệnh cho sự kiện của menu

Các Sự kiên của menu nằm trong mục Event của cửa sổ thuộc tính (Properties).

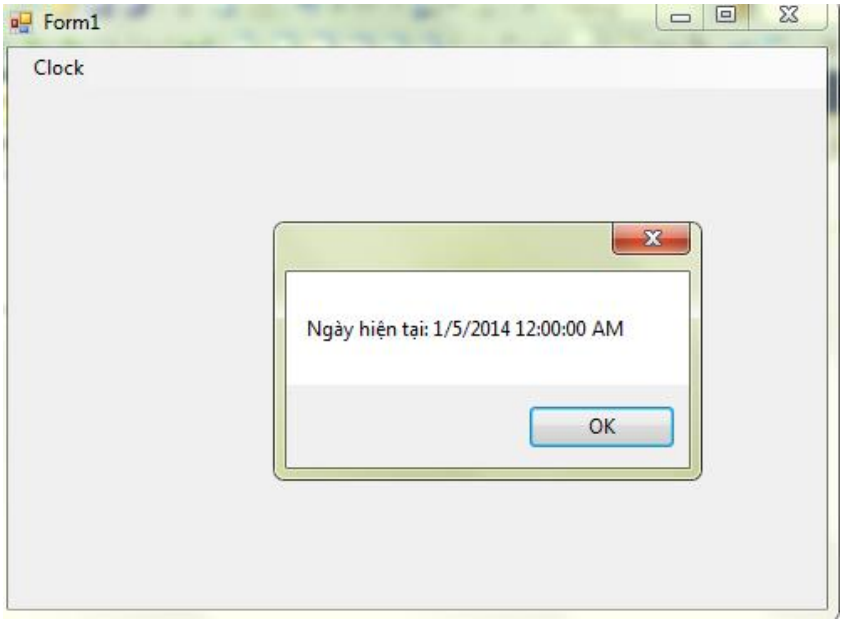


Hình 29

Để minh họa, chúng ta viết lệnh cho sự kiện nhấp chuột vào menu bằng cách nhấp đôi chuột vào sự kiện Click để viết lệnh, một sửa sổ code sẽ hiện ra, người lập trình chỉ cần viết phần thực hiện công việc mình mong muốn giữa thủ tục sự kiện Private Sub…End Sub.

```
Private Sub DateToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
DateToolStripMenuItem.Click
      MessageBox.Show("Ngày hiện tại: " + Today.ToString())
End Sub
```

Khi người dùng ấn vào menu "Date" chương trình sẽ xuất hiện hộp thoại thông báo ngày tháng hiện tại.

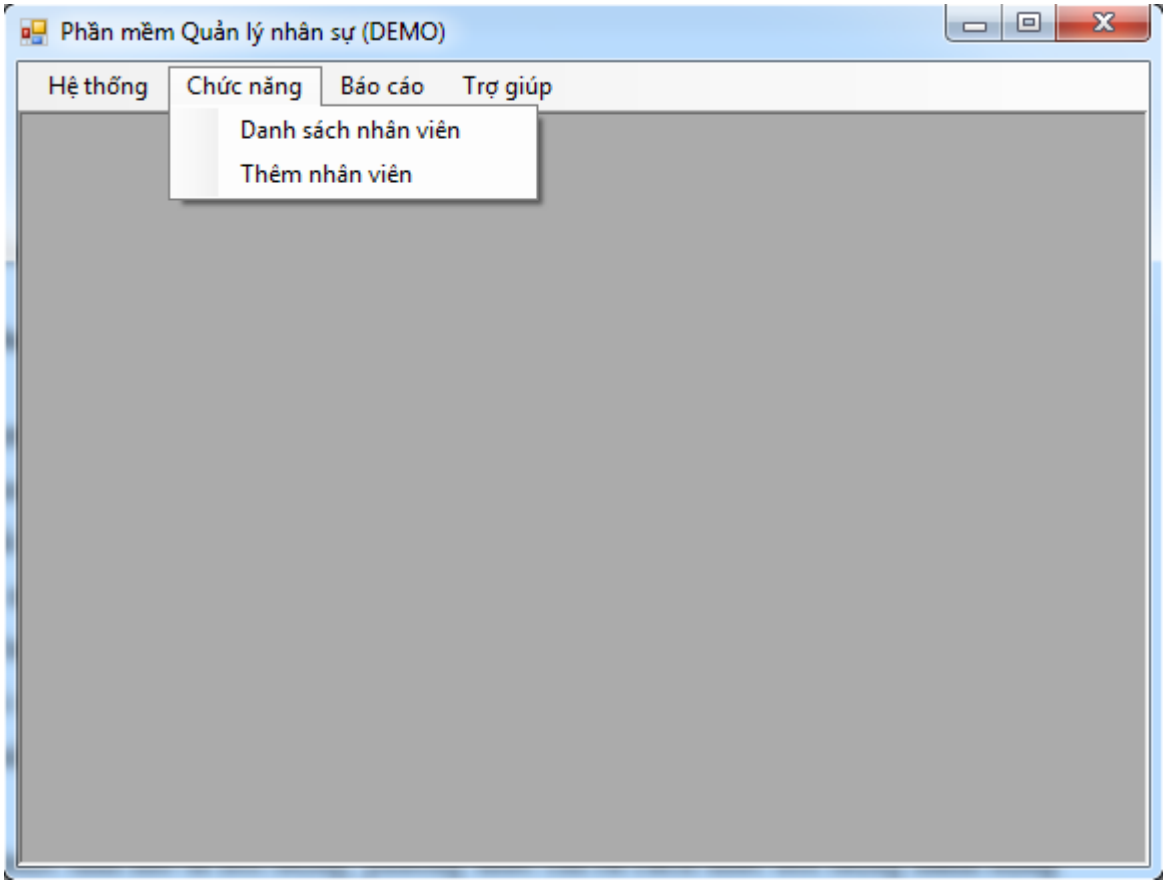


Hình 30

2. Thiết kế các dạng form

2.1 Form cha

Giả sử có 1 giao diện chính của một phần mềm quản lý nhân sự như sau:

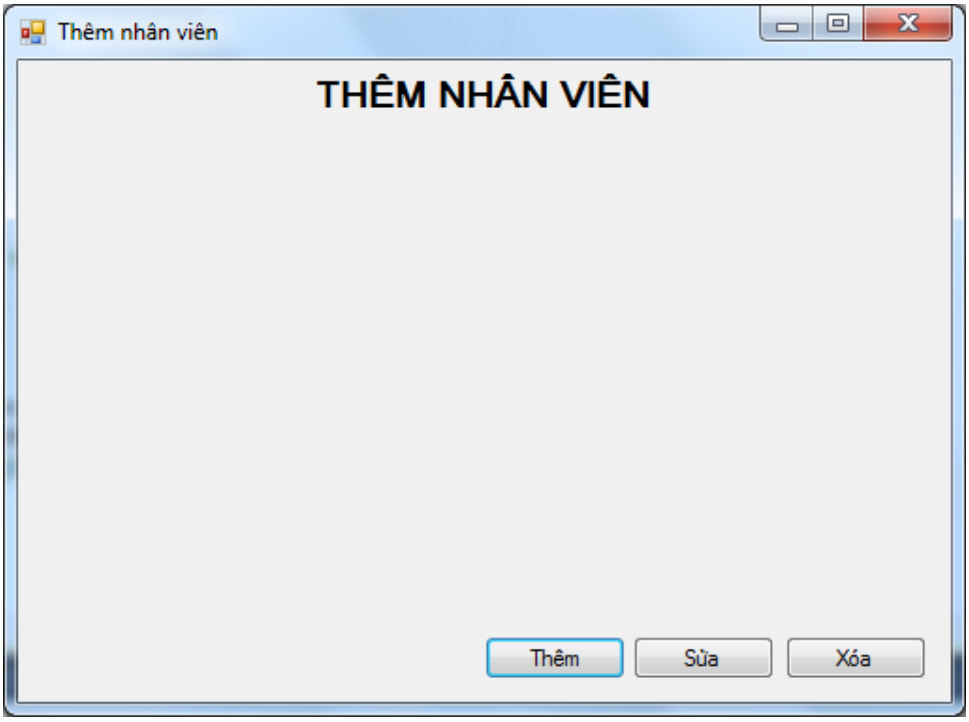


Hình 31

Giả sử tên form là frmMain, để form trên trở thành form cha chúng ta hãy chắc chắn rằng đã chọn thuộc tính: IsMdiContainer = True để thiết lập đây là Form cha.

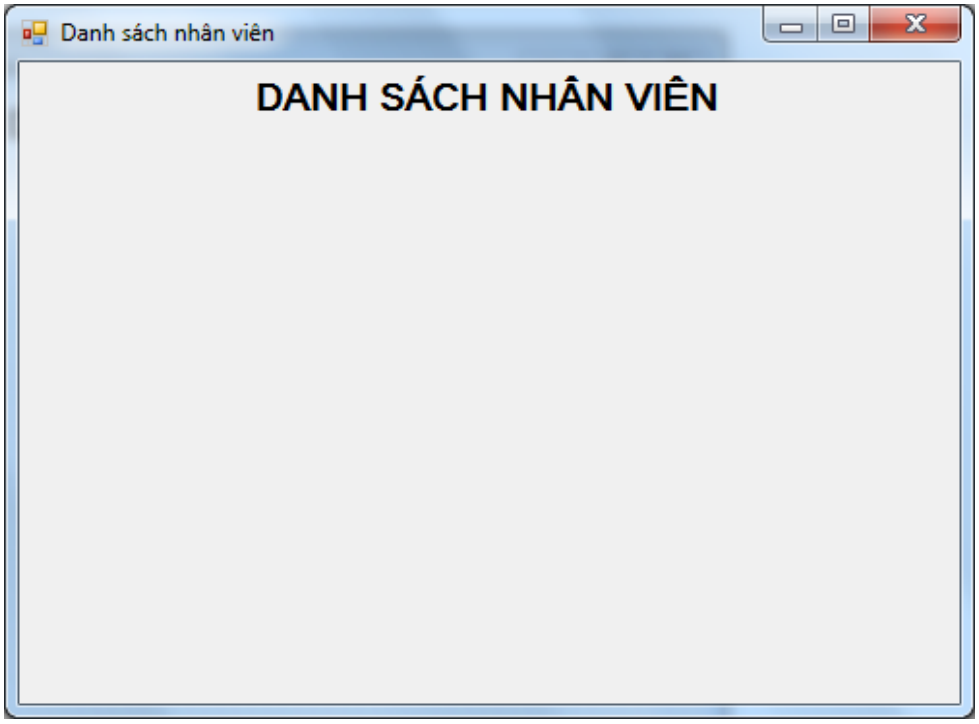
2.2 Form con

Thêm vào Project hai form mới là "Thêm nhân viên" (frmEmployee) và "Danh sách nhân viên" (frmListEmpoyee).



Hình 32

Form "Thêm nhân viên" (frmEmployee)



Hình 33

Form “Danh sách nhân viên” (frmListEmpoyee).

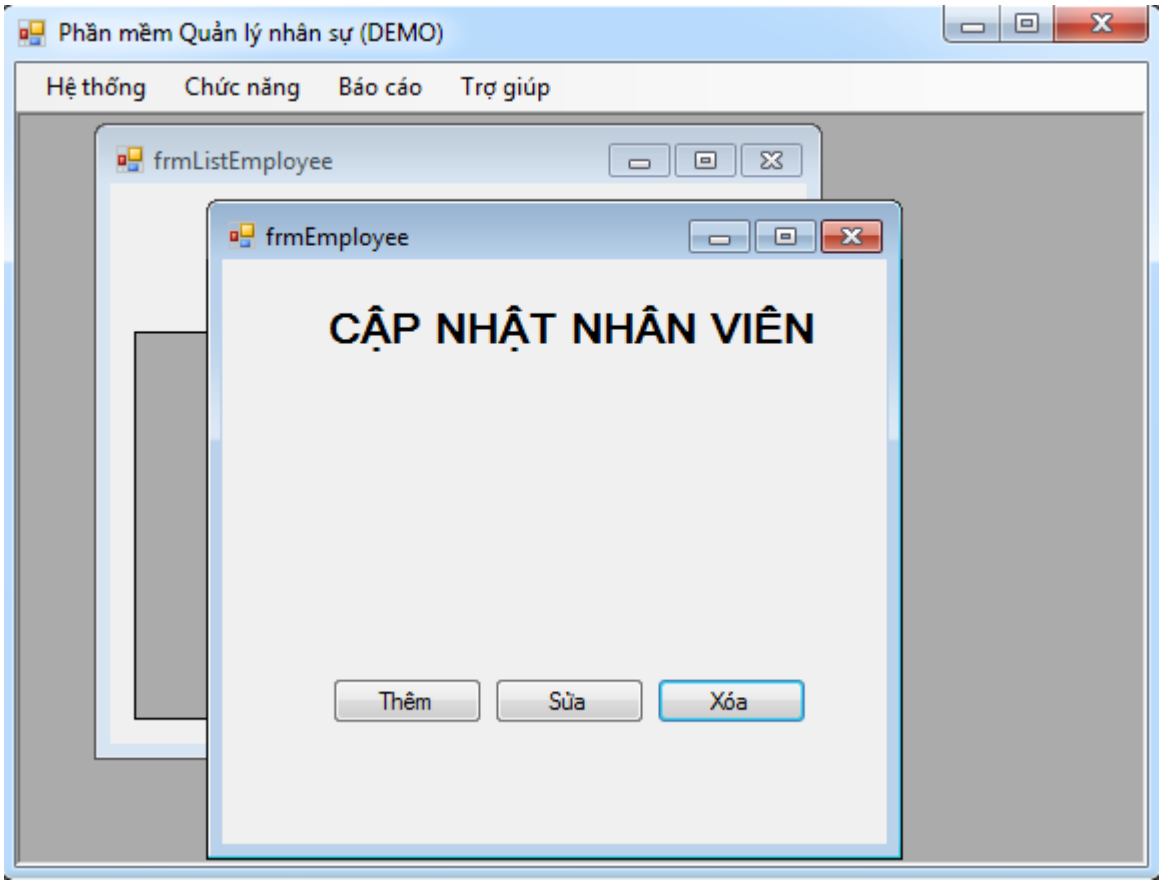
Viết sự kiện Click cho 2 nút “Thêm nhân viên” và “Danh sách nhân viên” trên thanh MenuStrip của Form Cha để hiển thị form tương ứng.

```
frmEmployee.MdiParent = Me
frmEmployee.Show()
```

và

```
frmListEmployee.MdiParent = Me
frmListEmployee.Show()
```

Khi chọn chức năng từ menu tương ứng, các form con sẽ “nằm gọn” trong form cha như hình dưới đây:



Hình 34

Chú ý, thuộc tính MdiParent dùng để thiết lập Form Cha của các form này là frmMain.

3. Sử dụng các điều khiển cơ bản

3.1 Mối quan hệ giữa thuộc tính, phương thức 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 thì một số thuộc tính như Top, Height, Left, Width sẽ thay đổi theo theo, khi đó kích cỡ của điều khiển thay đổi tức là sự kiện Resize xảy ra.

Phụ thuộc lẫn nhau còn có nghĩa là ta có thể thực hiện một công việc bằng nhiều cách: xử lý trên thuộc tính hoặc xử lý bằng phương thức.

Ví dụ: ta có 2 cách để làm hộp văn bản textBox1 xuất hiện và biến mất trên màn hình:

+ Thực hiện bằng thuộc tính:

Xuất hiện: `TextBox1.Visible = True`

Biến mất: `TextBox1.Visible = False`

+ Thực hiện bằng phương thức:

Xuất hiện: `TextBox1.Show()`

Biến mất: `TextBox1.Hide()`

3.2 Thuộc tính, phương thức, sự kiện của một số điều khiển cơ bản

3.2.1 Form

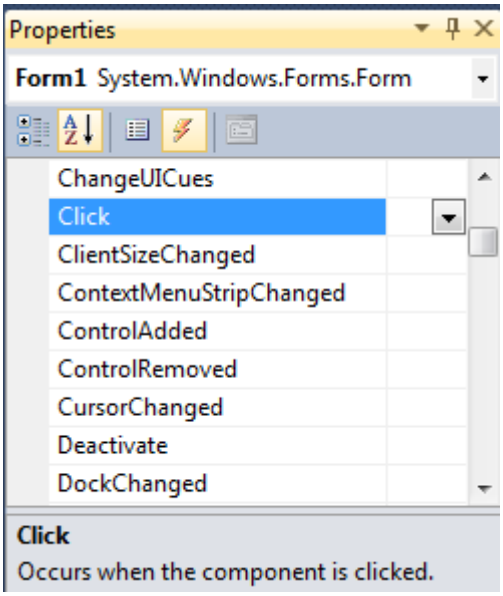
3.2.1.1. Thuộc tính

Name	Tên form, bắt đầu bởi tiếp đầu ngữ **frm**
BackColor	Thiết lập mầu nền cho Form.
BackgroundImage	Thiết lập ảnh nền cho Form.
BackgroundImageLayout	Thiết lập chế độ hiển thị ảnh nền trên Form. Tile: hiển thị ảnh từ trên xuống, Center: hiển thị ảnh từ giữa ra, Stretch: dãn đều ảnh trên Form.
AcceptButton	Thiết lập nút lệnh Accept. Sự kiện Click của nút lệnh này được gọi bất cứ khi nào người dùng bấm phím Enter.

CancelButton	Thiết lập nút lệnh Cancel. Sự kiện Click của nút lệnh này được gọi bất cứ khi nào người dùng bấm phím Esc.
Cursor	Thiết lập chế độ hiển thị con trỏ trên Form.
Enabled	Nếu nhận giá trị True thì cho phép người dùng tác động lên Form, Ngược lại thì nhận giá trị False.
Font	Thiết lập kiểu chữ, cỡ chữ cho các điều khiển trên Form.
ForeColor	Thiết lập mầu chữ cho các điều khiển trên Form.
FormBorderStyle	Thiết lập kiểu đoờng viền cho Form. Fixed Single: không thể thay đổi kích thước của Form, Sizable: có thể phóng to thu nhỏ và thay đổi kích thước của Form, Sizable ToolWindow: có thể thay đổi kích thước của Form…
Icon	Thiết lập biểu tượng cho Form (các tệp ảnh có đuôi .ico).
MainMenuStrip	Gắn kết Form với Menu.
Opacity	Thiết lập độ trong suốt cho nền của Form, nếu độ trong suốt <100% thì có thể nhìn xuyên thấu những gì nằm bên dưới Form.
ShowIcon	Nếu nhận giá trị True thì cho phép hiển thị biểu tượng đã được thiết lập ở thuộc tính Icon, ngược lại thì nhận giá trị False.
StartPosition	Thiết lập vị trí xuất hiện của Form trên màn hình. Manual: xuất hiện ở góc trên bên trái màn hình, CenterScreen: giữa màn hình…
Text	Thiết lập dòng tiêu đề của Form.
Window State	Thiết lập trạng thái của Form khi chạy chương trình. Normal: hiểnthị Form đúng theo kích cỡ thiết kế, Maximized: phóng to Form bằng màn hình, Minimized: thu nhỏ Form trên thanh Taskbar

3.2.1.2. Sự kiện

Để hiển thị danh sách các sự kiện của các điều khiển, ta kích chuột tại biểu tượng trên cửa sổ Properties:



Hình 35. Các sự kiện của Form

Muốn gọi sự kiện nào thì ta kích đúp chuột vào tên sự kiện đó, kết quả VISUAL BASIC sẽ tự động tạo ra dòng tiêu đề của phương thức chứa sự kiện trong cửa sổ code.

Form có một số sự kiện thông dụng như sau:

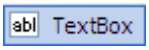
Load	được kích hoạt khi Form được nạp vào bộ nhớ, nó thường được dùng để khởi tạo các giá trị và trạng thái cho các biến, các điều khiển… trên Form.
Click	được kích hoạt khi người dùng kích chuột trên Form.
FormClosed	được kích hoạt khi người dùng kích chuột vào nút Close **x** ở góc trên bên phải để đóng Form.
FormClosing	Cũng được kích hoạt khi người dùng kích chuột vào nút Close **x,** nhưng xảy ra trước sự kiện FormClosed tức là được phát sinh trước khi cửa sổ Form chuẩn bị đóng lại.

Ví dụ: Nếu không muốn người dùng đóng Form bằng cách bấm chọn biểu tượng Close thì trong thủ tục **FormClosing** ta đặt thuộc tính Cancel = True như sau:

```
Private Sub Form1_FormClosing(ByVal sender As System.Object,
ByVal e As System.Windows.Forms.FormClosingEventArgs) Handles
MyBase.FormClosing
```

```
        e.Cancel = True
End Sub
```

3.2.2 Hộp văn bản - TextBox

Hộp văn bản  là điều khiển rất thông dụng, dùng để nhập dữ liệu đầu vào từ phía người sử dụng và hiển thị các kết quả đã tính toán được.

3.2.2.1. Thuộc tính

Name	Tên Textbox, bắt đầu bởi tiếp đầu ngữ **txt**
BackColor	Thiết lập mầu nền cho hộp TextBox.
Enabled	Enabled=False: không cho phép người dùng truy cập vào TextBox (Hộp Textbox bị mờ đi), ngược lại thì bằng True.
Font	Thiết lập kiểu chữ và cỡ chữ cho hộp văn bản.
ForeColor	Thiết lập mầu chữ cho hộp văn bản.
Locked	Locked = True: khóa không cho phép dịch chuyển vị trí của hộp văn bản trên Form, ngược lại thì nhận giá trị False.
MaxLength	Quy định chiều dài tối đa được chấp nhận của hộp văn bản, giá trị mặc định là 32767 hoặc 0, tức là có thể chứa 32767 ký tự. Mọi xác lập khác 0, ví dụ 5 thì chỉ cho phép người dùng nhập tối đa 5 ký tự vào hộp văn bản.
Multiline	Multiline = False: chỉ cho phép hiển thị văn bản trên một dòng, và khi thiết kế ta chỉ thay đổi được độ dài của hộp văn bản. Multiline = True: cho phép văn bản được hiển thị trên nhiều dòng, và có thể thay đổi cả độ dài lẫn độ rộng của hộp văn
PasswordChar	Thuộc tính này cho phép người sử dụng bảo mật được thông tin nhập vào Textbox. Ví dụ đặt thuộc tính này bằng ký tự „*“ khi đó toàn bộ dữ liệu nhập vào sẽ được hiển thị dưới dạng dấu hoa thị
ReadOnly	ReadOnly = True: hộp văn bản vẫn được truy cập nhưng người dùng không thể thay đổi được nội dung bên trong.
ScrollBars	Thiết lập thanh cuốn ngang, dọc cho hộp văn bản (có hiệu lực khi Multiline = True). Chú ý: thanh cuốn ngang chỉ có hiệu lực khi WordWrap = False.
TabIndex	Thứ tự truy cập của hộp văn bản khi người dùng bấm phím Tab, thứ tự đầu tiên là 0.

Text	Chứa nội dung của hộp văn bản.
TextAlign	Thiết lập chế độ căn chỉnh: trái, phải hoặc giữa của dữ liệu trong hộp TextBox.
Visible	Visible = True: hiển thị hộp văn bản, Visible = False: ẩn hộp văn bản.
WordWrap	WordWrap = True: dòng văn bản được tự động cuộn xuống dòng khi gặp lề bên phải của hộp TextBox, ngược lại thì nhận giá trị False. Chỉ có hiệu lực khi Multiline = True

3.2.2.2 Sự kiện

Hộp văn bản có một số sự kiện cơ bản sau:

TextChanged	được kích hoạt khi người dùng thực hiện sự thay đổi bất kỳ trong hộp văn bản như: thêm, xoá, sửa, dán văn bản.
Click	được kích hoạt khi người dùng kích chuột vào hộp văn bản.
DoubleClick	được kích hoạt khi người dùng kích đúp chuột vào hộp văn bản.
GotFocus	được kích hoạt khi người dùng chuyển tiêu điểm tới hộp văn
KeyPress	Trả về ký tự (trừ các ký tự đặc biệt như phím Delete, Home, Ctrl, F1…)mà người sử dụng gõ vào hộp văn bản thông qua thuộc tính KeyChar.
KeyDown	Trả về mã Ascii của tất cả các ký tự mà người sử dụng gõ vào hộp văn bản thông qua thuộc tính KeyCode.
LostFocus	được kích hoạt khi hộp văn bản mất tiêu điểm.
MouseMove	được kích hoạt khi người dùng di chuyển chuột qua hộp văn
MouseLeave	được kích hoạt khi người dùng dời chuột ra khỏi hộp văn bản.

Ví dụ 1: Để hiển thị mã Ascii của một ký tự bất kỳ được gõ vào hộp văn bản ***TextBox1*** ta có

đoạn chương trình như sau:

```
Private Sub TextBox1_KeyDown(ByVal sender As System.Object,
ByVal e As System.Windows.Forms.KeyEventArgs) Handles
TextBox1.KeyDown
        MessageBox.Show(e.KeyCode)
```

```
End Sub
```

Ví dụ 2: Dùng sự kiện **KeyPress** để kiểm tra việc nhập dữ liệu: chỉ cho phép nhập vào hộp văn bản ***TextBox1*** các số từ 0 tới 9, dấu âm - , dấu chấm thập phân . , phím Del (có mã Ascii=13) và phím Backspace (có mã Ascii = 8) để xóa dữ liệu. Ta có đoạn chương trình như sau:

```
Private Sub TextBox1_KeyPress(ByVal sender As System.Object,
ByVal e As System.Windows.Forms.KeyPressEventArgs) Handles
TextBox1.KeyPress
        If (e.KeyChar >= "0" And e.KeyChar <= "9") Or
e.KeyChar = "-" Or e.KeyChar = "." Or
Convert.ToInt32(e.KeyChar) = 8 Or Convert.ToInt32(e.KeyChar) =
13 Then
            e.Handled = False
        Else
            e.Handled = True
        End If
End Sub
```

Nếu mỗi ký tự được nhập vào hộp Textbox không thoả mãn điều kiện if thì sẽ bị hủy bỏ

bằng cách đặt thuộc tính Handled là true.

3. 2.3 Nút lệnh – Button

Nút lệnh 

cho phép người dùng thực hiện một hành động nào đó.

3.2.3.1. Thuộc tính

Name	Tên nút lệnh, bắt đầu bởi tiếp đầu ngữ **btn**
BackColor	Thiết lập mầu nền cho nút lệnh.
BackgroundImag	Thiết lập ảnh nền cho nút lệnh.
Enabled	Enabled=False: người dùng không thể tác động lên nút lệnh, ngược lại thì bằng True.
Font	Xác lập kiểu chữ và cỡ chữ cho nút lệnh.
ForeColor	Thiết lập mầu chữ cho nút lệnh.
Image	Thiết lập ảnh hiển thị trên nút lệnh.

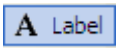
Locked	Locked = True: khóa không cho phép dịch chuyển vị trí của nút lệnh trên Form, ngược lại thì nhận giá trị False.
TabIndex	Thứ tự truy cập của nút lệnh khi người dùng bấm phím
Text	Tiêu đề của nút lệnh. Ta có thể quy định phím nóng cho nút lệnh bằng cách đặt dấu "&" trước một ký tự của Text. Ví dụ &Quit sẽ được hiển thị là Quit, khi người sử dụng bấm Alt+Q chương trình sẽ kích hoạt nút lệnh Quit.
Visible	Visible = True: hiển thị nút lệnh, Visible = False: ẩn nút

3.2.3.2. Sự kiện

Nút lệnh có một số sự kiện cơ bản sau:

Click	được kích hoạt khi người dùng kích chuột vào nút lệnh.
GotFocus	được kích hoạt khi người dùng chuyển tiêu điểm tới nút
LostFocus	được kích hoạt khi nút lệnh mất tiêu điểm.
MouseDown	được kích hoạt khi người dùng đặt chuột vào nút lệnh.
MouseUp	được kích hoạt khi người dùng đọa chuột ra khỏi nút lệnh.
MouseMove	được kích hoạt khi người dùng di chuyển chuột trên nút
MouseLeave	được kích hoạt khi người dùng dời chuột ra khỏi nút lệnh.

3.2.4 Nhãn – Lable

Nhãn 

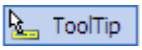
dùng để hiển thị những thông tin có tính chất cố định người sử dụng không có khả năng thay đổi ví dụ như dòng thông báo, hợớng dẫn …

Nhãn có một số thuộc tính hay dùng sau:

Name	Tên nhãn, bắt đầu bởi tiếp đầu ngữ **lbl**
BackColor	Thiết lập mầu nền cho nhãn, nếu thiết lập BackColor = Transparent (mụclựa chọn đầu tiên trong tab Web) thì nhãn sẽ có nền giống với nền của Form.
BorderStyl	Thiết lập kiểu đợờng viền cho nhãn.
Font	Thiết lập kiểu chữ và cỡ chữ cho nhãn.
ForeColor	Thiết lập mầu chữ cho nhãn.
Image	Thiết lập ảnh hiển thị trên nhãn.

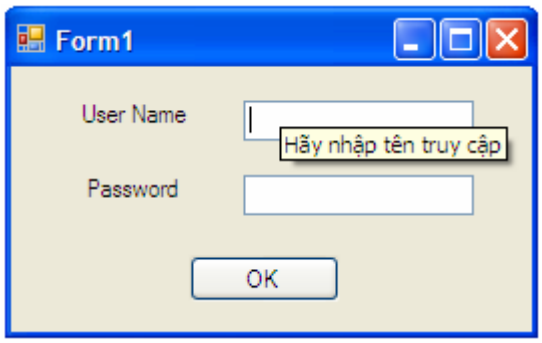
Locked	Locked = True: khóa không cho phép dịch chuyển vị trí của nhãn trên Form, ngược lại thì nhận giá trị False.
TabIndex	Thứ tự truy cập của nhãn khi người dùng bấm phím Tab.
Text	Tiêu đề của nhãn.
TextAlign	Thiết lập chế độ căn chỉnh: trái, phải hoặc giữa của tiêu đề nhãn.
Visible	Hiện hoặc ẩn nhãn.

3.2.5 Dòng mách nước - ToolTip

Điều khiển mách nước 

cho phép hiển thị các thông tin chú thích khi người dùng đưa chuột qua điều khiển có thiết lập ToolTip.

Ví dụ dòng mách nước "Hãy nhập tên truy cập" như hình dưới đây.



Hình 36

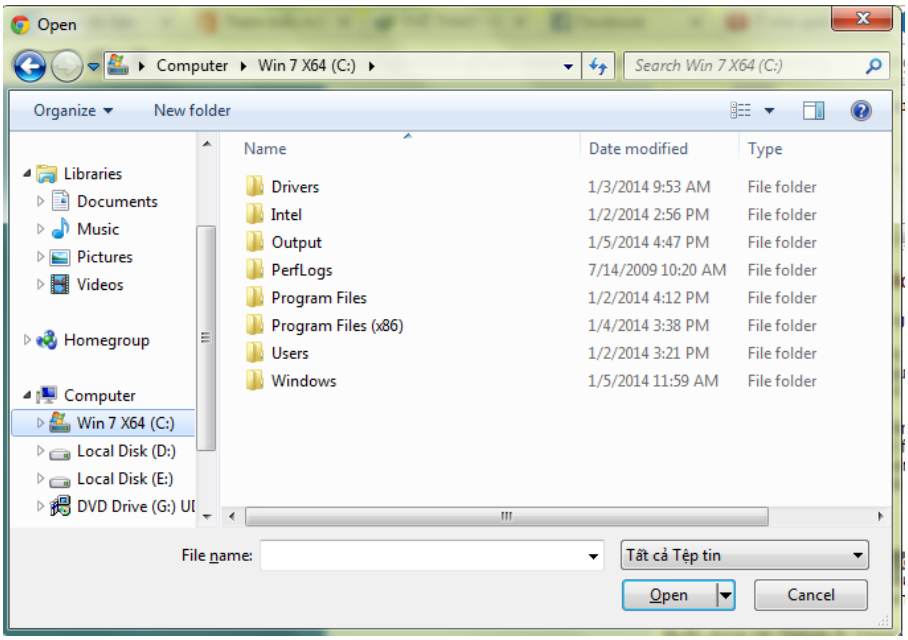
Để tạo dòng mách nước cho một điều khiển ta thực hiện như sau:

+ Kéo điều khiển ToolTip vào Form, điều khiển ToolTip không được hiển thị ở trên Form mà được hiển thị ở thanh ngang cuối Form và được dùng chung cho mọi điều khiển trên form.

+ Kích chuột chọn điều khiển muốn tạo ToolTip, trong cửa sổ Window Properties gõ nội dung dòng ToolTip tại thuộc tính ToolTip on ToolTip1.

3.3 Các hộp thoại thông dụng

3.3.1 Hộp thoại mở tập tin (OpenFileDialog)

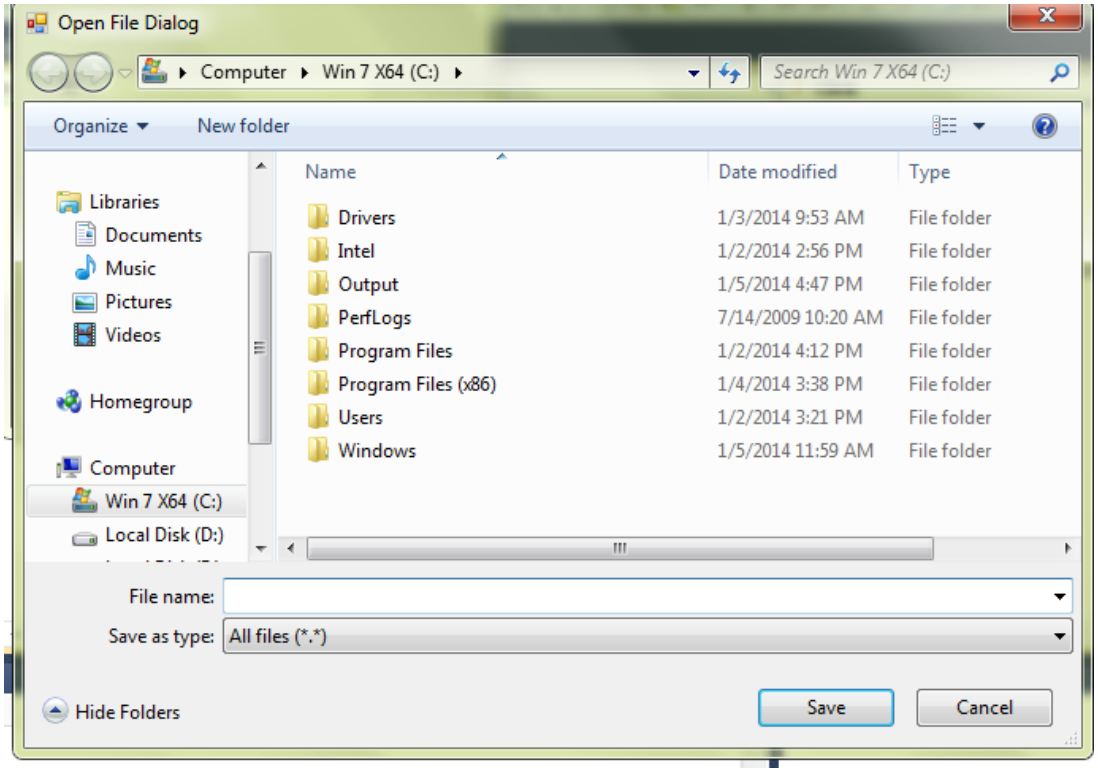


Hình 37

Đoạn lệnh sau đây cho phép người lập trình tùy biến các thuộc tính và hiện hộp thoại mở tập tin:

```
Dim fd As OpenFileDialog = New OpenFileDialog()
Dim strFileName As String
fd.Title = "Chọn tập tin cần mở"
fd.InitialDirectory = "C:\"
fd.Filter = "Tập tin Word(*.doc)|*.doc|All files (*.*)|*.*"
fd.FilterIndex = 2
fd.RestoreDirectory = True
If fd.ShowDialog() = DialogResult.OK Then
     strFileName = fd.FileName
End If
```

3.3.2 Hộp thoại lưu tập tin (SaveFileDialog)

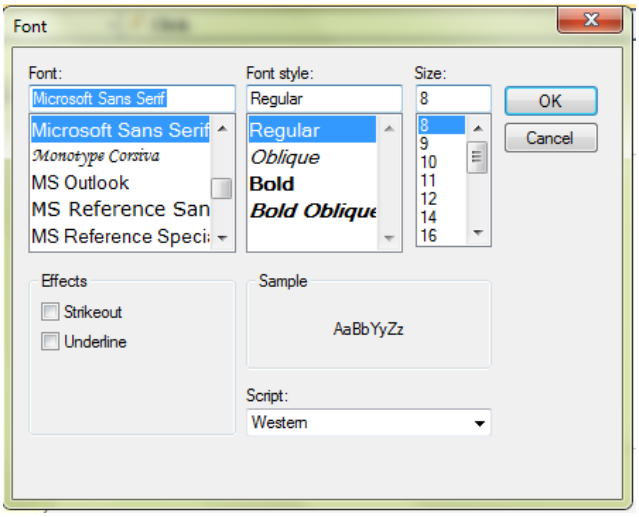


Hình 38

Đoạn lệnh sau cho phép người lập trình tùy biến các đặc điểm của một hộp thoại:

```
Dim fd As SaveFileDialog = New SaveFileDialog()
Dim strFileName As String
fd.Title = "Chọn tập tin cần lưu"
fd.InitialDirectory = "C:\"
fd.Filter = "Tập tin Word(*.doc)|*.doc|All files (*.*)|*.*"
fd.FilterIndex = 2
fd.RestoreDirectory = True
If fd.ShowDialog() = DialogResult.OK Then
  strFileName = fd.FileName
End If
```

3.3.3 Hộp thoại font

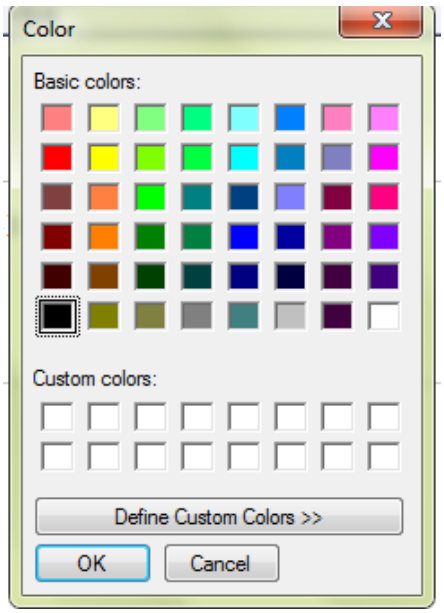


Hình 39

Để hiện hộp thoại Font chúng ta sử dụng lệnh sau :

```
Dim fd As FontDialog = New FontDialog()
If fd.ShowDialog() = DialogResult.OK Then
' Thực hiện lệnh
End If
```

3.3.4 Hộp thoại màu



Hình 40

Đoạn lệnh sau sẽ hiện hộp thoại màu để người sử dụng lựa chọn:

```
Dim fd As ColorDialog = New ColorDialog()
If fd.ShowDialog() = DialogResult.OK Then
' Thực hiện lệnh
End If
```

4. Làm việc với Module

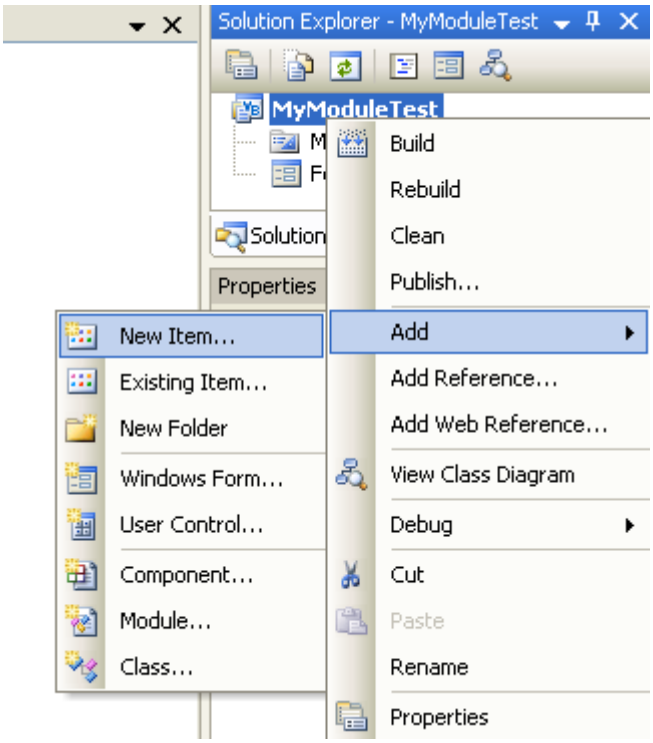
Khi dự án của bạn rất lớn thì việc có nhiều form là điều đương nhiên. Có điều bạn không thể sử dụng những hàm, biến khai báo trong form này cho form kia được.

Để chia sẻ biến và các hàm, thủ tục giữa các form trong dự án chúng ta có thể khai báo chúng trong một module của dự án. Module là một file có đuôi mở rộng .vb chỉ chứa các mã lệnh.

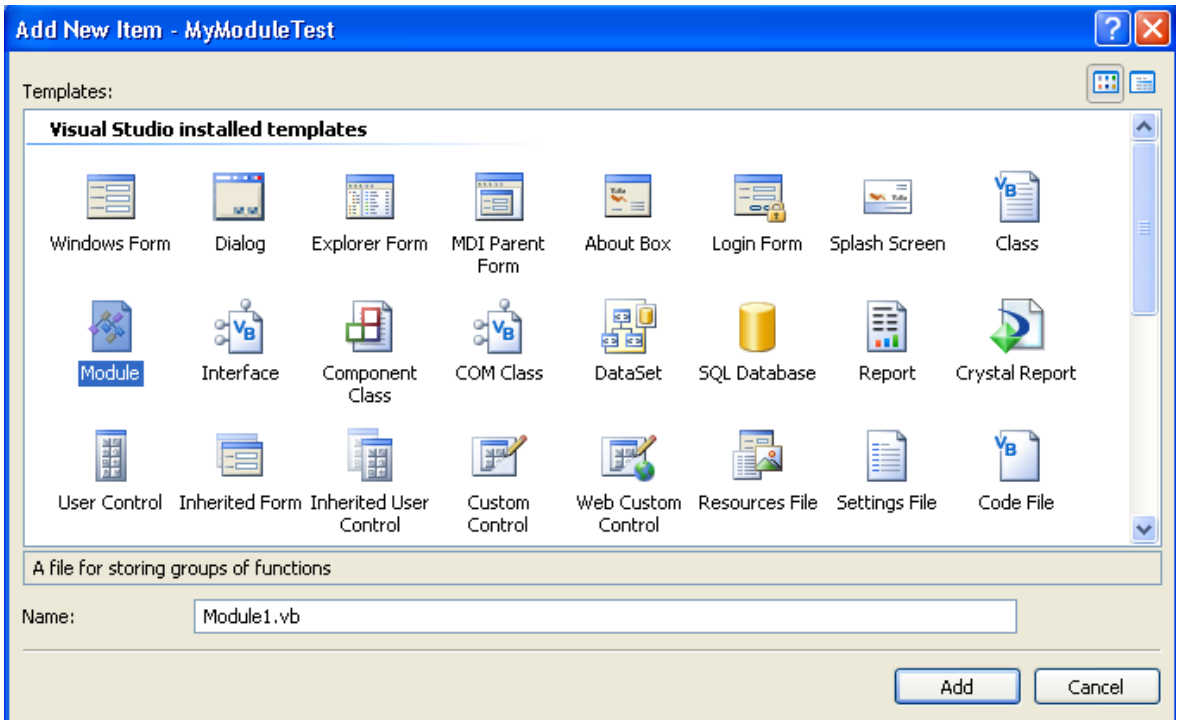
4.1 Tạo và lưu module chuẩn

Bây giờ chúng ta tạo một module với ví dụ *MymoduleTest* sau đây:

Tạo một dự án tên *MyModuleTest*, tại cửa sổ Solution Explorer, Right-Click vào tên dự án và chọn Add | New Item… như hình:



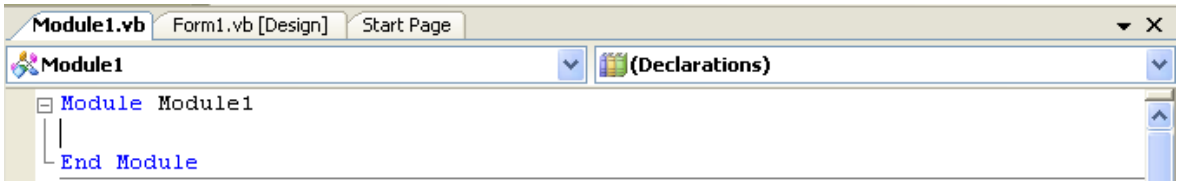
Hình 41



Hình 42

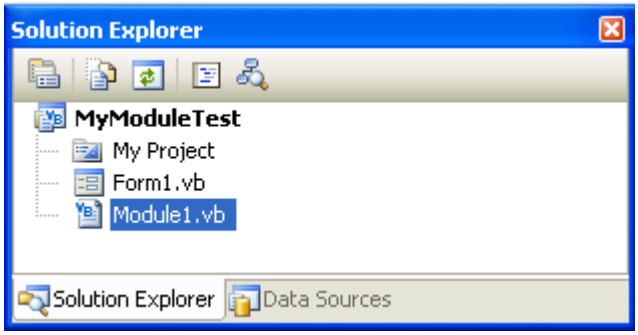
Chọn mẫu *Module* và nhấn Add, ở cửa sổ này có thể để tên mặc định là Module1.vb hay có thể đặt lại tên module. Nếu để tên mặc định thì việc thay đổi tên sau này có thể dùng phương thức File | Save Module1 As.

Khi nhấn Add, một cửa sổ ở chế độ Code Editor hiện ra cho phép ta thao tác mã.



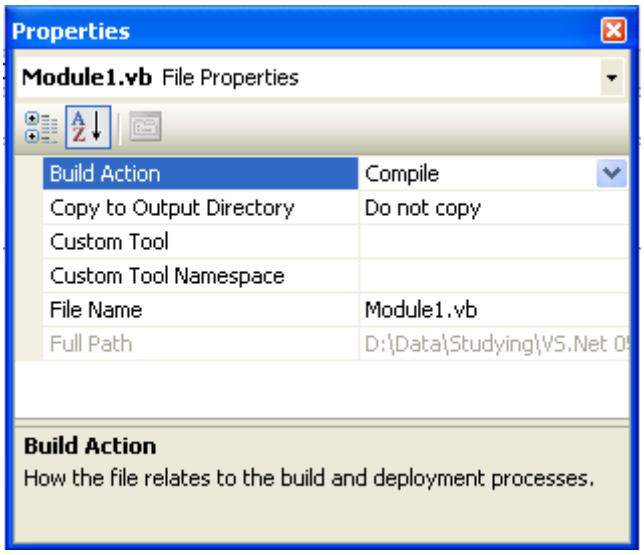
Hình 43

Chúng ta có thể xem liệt kê các thành phần của dự án bao gồm cả module1 ta vừa tạo trong cửa sổ Solution Explorer:



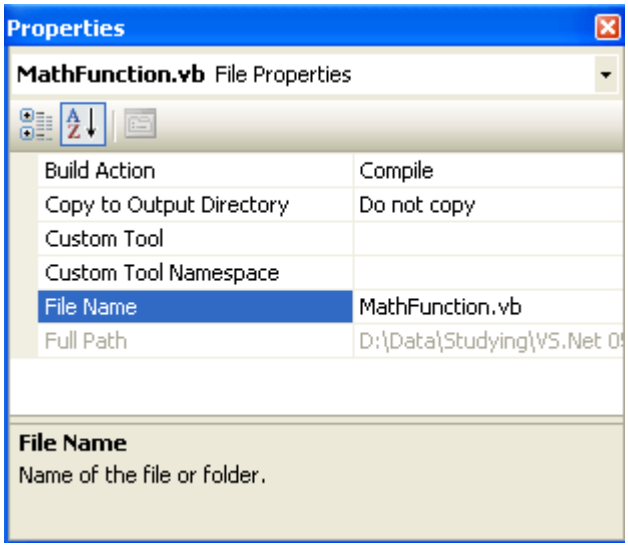
Hình 44

Để xem thuộc tính của module, Right-Click vào module và chọn Properties:



Hình 45

Để thay đổi tên của module, chúng ta có thể đặt lại bằng thuộc tính *File Name*. Ở đây giả sử ta thay tên thành *MathFuction*:



Hình 46

Để xóa module: Right-click vào tên module và chọn Delete.

Để tạm loại bỏ nó ra khỏi dự án: Right-Click chọn Exclude From Projects (có thể chọn Project | Exclude From Project).

Muốn thêm trở lại: chọn Add | Exist Item.

4.2 Sử dụng các biến Public

4.2.1 Làm việc với các biến Public (biến toàn cục)

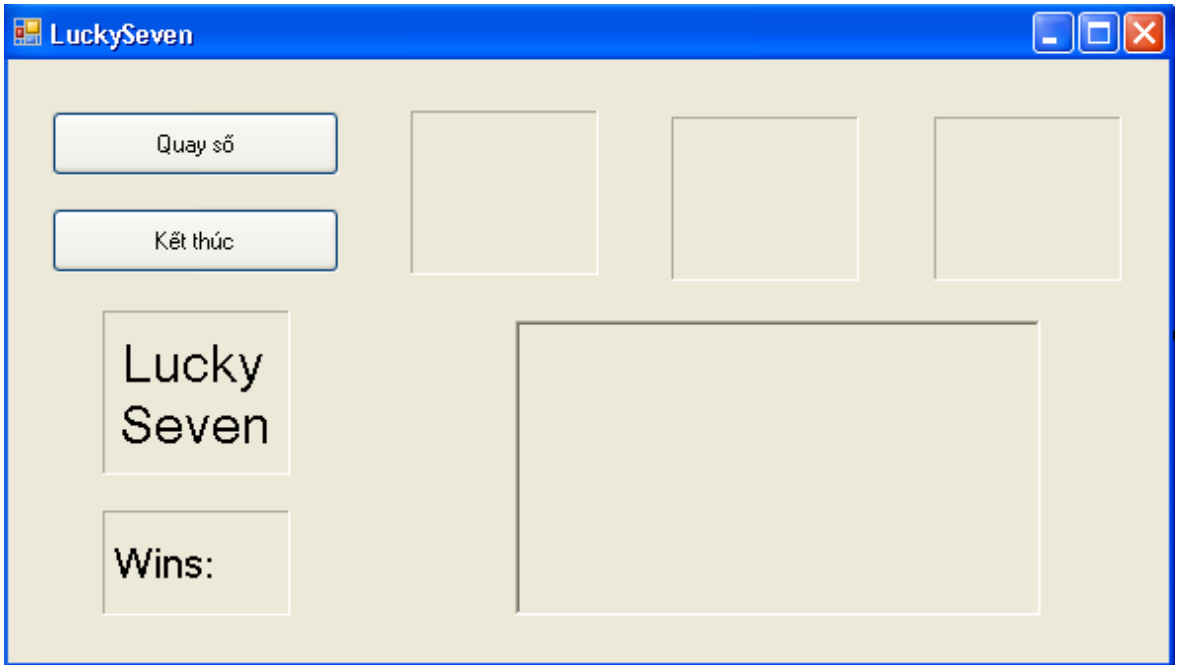
Biến toàn cục là biến được khai báo với từ khóa *Public* ở trước. Biến này cho phép chúng ta gọi xử lý ở bất cứ nơi nào trong chương trình. Ví dụ:

```
Public toancuc As Integer
```

Khai báo này khai báo một biến tên *toancuc* có kiểu dữ liệu là Integer.

Bây giờ ta sẽ mô phỏng bằng một ứng dụng quay số trúng thưởng "Số 7 may mắn", trong ví dụ này ta sử dụng một biến toàn cục có tên *solanthang* để lưu lại số lần người chơi chiến thắng và cho hiển thị nó lên trong một nhãn.

Bạn lưu lại dự án trên đây và đóng nó lại. Chọn tạo mới một giải pháp và thêm vào một dự án cùng tên *LuckySeven* như đã biết. Bạn thiết kế Form như hình:



Hình 47

Chương trình bao gồm ba nhãn hiển thị 3 số ngẫu nhiên, hai nút cho phép click quay số và kết thúc chương trình, một ô PictureBox hiển thị ảnh khi chiến thắng, một nhãn ghi tên chương trình *LuckySeven*. Một nhãn (Label5) hiển thị số lần chiến thắng của người chơi.

Bây giờ ta thêm vào một module – module *module1* và gõ vào trong đó một khai báo biến như sau:

```
Public solanchienthang As Integer
```

Sau đó chúng ta sẽ sử dụng biến này trong thủ tục Button1_Click như sau:

```
Private Sub Button1_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles Button1.Click
    PictureBox1.Visible = False
    Label1.Text = CStr(Int(Rnd() * 10))
    Label2.Text = CStr(Int(Rnd() * 10))
    Label3.Text = CStr(Int(Rnd() * 10))

    If (Label1.Text = "7") Or (Label2.Text = "7") _
    Or (Label3.Text = "7") Then
        PictureBox1.Visible = True
        Beep()
        solanchienthang += 1
        Label5.Text = "Wins: " & solanchienthang
    End If
End Sub
```

Chúng ta cũng dùng hàm Randomize() trong sự kiện Form_Load để tạo ngẫu nhiên ba số. Bây giờ bạn hãy chạy chương trình để xem nó hoạt động như thế nào.

Hình 48

4.2.2 Biến Public ở phạm vi form

Biến phạm vi form là biến khai báo ở đầu chương trình dưới dòng khai báo form. Với chương trình này thì cả hai cách đều cho kết quả như nhau. Nhưng những biến khai báo ở mức độ form chỉ có thể sử dụng trong các hàm, các thủ tục ở form đó.

Trong trường hợp chúng ta muốn các form khác truy xuất được các biến ở phạm vi form thì có thể sử dụng từ khóa Public thay vì từ khóa Dim. Tuy nhiên, để truy xuất các biến này ở các form khác chúng ta phải chỉ ra:

<Tên form>.<tên biến Public ở phạm vi form>

Mặc khác, giá trị của các biến ở phạm vi form cũng sẽ bị mất đi khi form đóng lại.

4.3 Tạo thủ tục (Procedure)

4.3.1. Khai báo thủ tục

Cú pháp:

```
Sub ProcedureName([Arguments])
    'Procedure Statement
End Sub
```

Trong đó:

- ProcedureName: tên của thủ tục.
- Arguments: các đối số.
- Procedure Statements: Các phát biểu cài đặt cho phần nội dung của thủ tục.

Ví dụ: Tạo ví dụ *BestWishesForBirthday* để minh họa cách tạo và dùng hàm Sub. Thiết kế form1 chỉ có một nút "End" và nhập End vào thủ tục Button1_Click.

Thêm một module vào chương trình và khai báo trong đó một thủ tục như sau:

```
Sub BirthdayGreeting(ByVal Person As String)
        Dim msg As String
        If Person <> "" Then
            msg = "Happy Birthday " & Person & "!"
        Else
            msg = "Name no specified"
        End If
        MsgBox(msg, , "BestWhished")
End Sub
```

Thủ tục này cho phép hiện một lời chào với người có tên là đối số truyền vào cho biến *Person*.

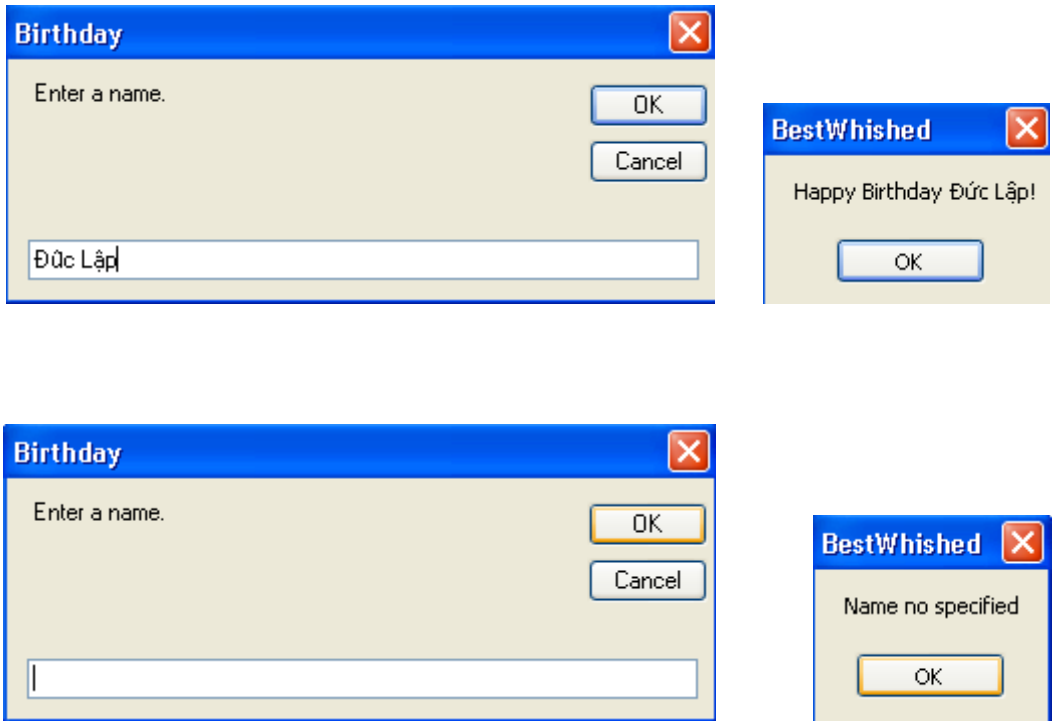
4.3.2 Sử dụng các thủ tục - Sub

Bây giờ trở về cửa sổ soạn mã cho form1 và tạo sự kiện Form1_Load nhập đoạn mã sau:

```
Private Sub Form1_Load(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles MyBase.Load
    Dim name As String
    Do
```

```
            name = InputBox("Enter a name.", "Birthday")
            BirthdayGreeting(name)
        Loop Until name = ""
    End Sub
```

Đoạn mã này sẽ hiện một hộp thoại để người dùng nhập tên của họ cho chương trình gọi thủ tục trên chúc sinh nhật người đó. Khi không nhập vào thì chương trình hiện form1 với nút "End" cho phép kết thúc chương trình:



Hình 49

4.3.3 Truyền đối số theo tham trị và tham biến

Trong thủ tục đối số có thể truyền theo tham trị (Byval) hay tham chiếu (Byref). Mặc định khi không khai báo thì nó truyền theo tham trị.

Để truyền theo tham trị ta khai báo đối số trong thủ tục bằng từ khóa *Byval*. Khi đó bất cứ thay đổi nào trong thủ tục cũng không làm thay đổi giá trị của biến sau khi thủ tục chấm dứt.

Ngược lại, để truyền theo tham chiếu, ta khai báo đối số bằng từ khóa *Byref*. Khi đó những thay đổi trong hàm hay thủ tục sẽ làm thay đổi giá trị của biến.

4.4 Khai báo hàm (Function)

Hàm được khai báo bằng từ khóa *Function* và kết thúc bằng từ khóa *End Function*. Việc thực thi hay gọi hàm bằng cách dùng tên hàm cùng các đối số trong ngoặc đơn nếu có.

Khi hàm được khai báo trong module được mặc định là hàm toàn cục và có thể được gọi từ bất cứ nơi nào của dự án.

4.4.1 Cú pháp khai báo hàm

Cú pháp của hàm:

```
Function FunctionName([argument]) As Type
    Function_statements()
    [Return value]
End Function
```

Trong đó:

- FunctionName: tên của hàm mà mình muốn tạo, dùng để gọi hàm sau này

- As Type: định nghĩa kiểu dl trả về của hàm sau khi tính toán xong

- Argument: danh sách đối số truyền cho hàm. Mặc định VB.NET sẽ thêm từ khóa Byval tức truyền theo tham trị, tất cả những thay đổi trong hàm lên đối số không làm thay ảnh hưởng đến đối số truyền vào khi hàm chấm dứt.

- Function_Statement: các khối phát biểu cài đặt cho hàm.

- Return: cho phép trả lại kết quả sau cùng của hàm cho nơi gọi hàm.

Ví dụ:

```
Function TotalTax(ByVal Cost As Single) As Single
    Dim StateTax, CityTax As Single
    StateTax = Cost * 0.05
    CityTax = Cost * 0.015
    TotalTax = StateTax + CityTax
End Function
```

Hàm trên trả về giá trị thuế tổng *TotalTax* bằng câu lệnh ở phát biểu cuối cùng. Chúng ta có thể dùng phát biểu *Return* như ví dụ sau để trả về kết quả cho hàm:

```
Function TotalTax(ByVal Cost As Single) As Single
    Dim StateTax, CityTax As Single
    StateTax = Cost * 0.05
    CityTax = Cost * 0.015
    Return (StateTax + CityTax)
End Function
```

4.4.2. Gọi hàm

Việc gọi hàm theo cú pháp sau:

```
Label1.Text = TotalTax(500)
```

Phát biểu trên sẽ tính thuế dựa trên chi phí đầu vào là $500 và gán kết quả cho thuộc tính Text của nhãn label1.

4.4.3. Sử dụng hàm thực hiện tác vụ tính toán

Trong ví dụ sau ta sẽ quay trở lại ví dụ *LuckySeven* và tính tỷ lệ chiến thắng rồi gán vào thuộc tính của nhãn mới *Label6*.

Bạn thiết kế lại giao diện như sau:

Hình 50

Trước hết ta khai báo thêm biến public trong module1 như sau:

```
Public solanchienthang As Short
Public Spins As Short
```

Sau đó tạo thủ tục HitRate tính số phần trăm chiến thắng. Số phần trăm này được tính bằng tỉ số giữa số lần chiến thắng trên tổng số lần quay:

```
Function HitRate(ByVal Hits As Short, _
ByVal Tries As Short) As String
    Dim percent As Single
    percent = Hits / Tries
    Return Format(percent, "0.0%")
End Function
```

Bây giờ chúng ta trở lại thủ tục Button1_Click và nhập như sau:

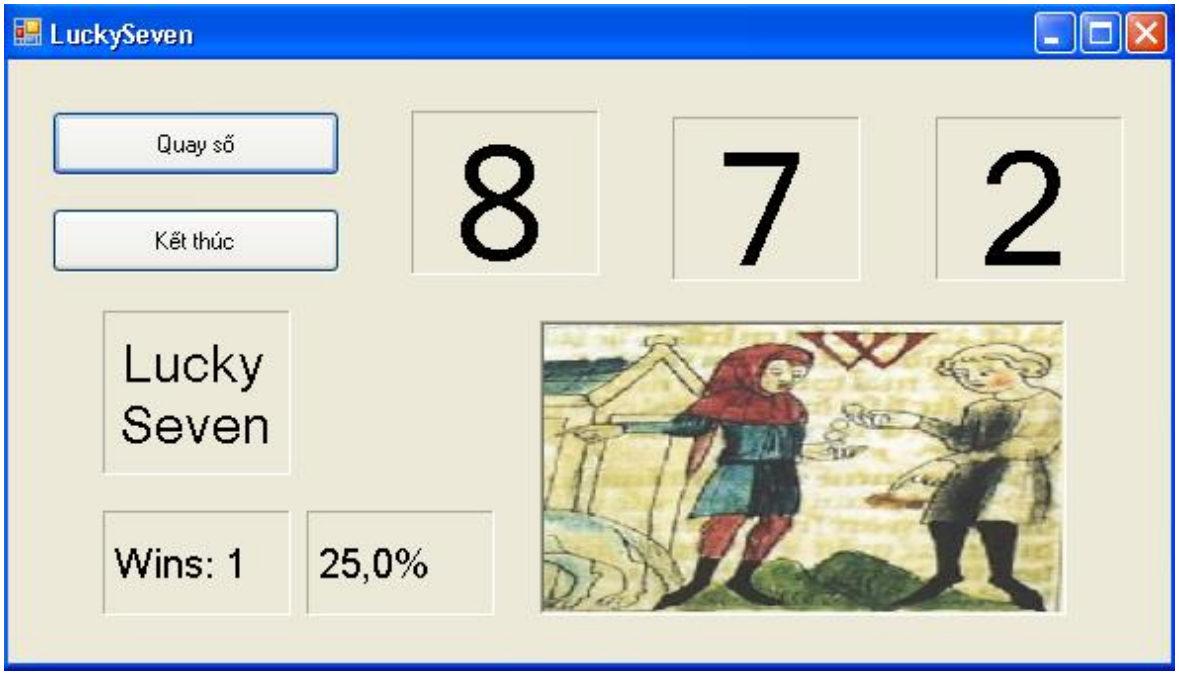
```
Private Sub Button1_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles Button1.Click
    PictureBox1.Visible = False
    Label1.Text = CStr(Int(Rnd() * 10))
    Label2.Text = CStr(Int(Rnd() * 10))
    Label3.Text = CStr(Int(Rnd() * 10))
    Spins += 1
    If (Label1.Text = "7") Or (Label2.Text = "7") _
    Or (Label3.Text = "7") Then
        PictureBox1.Visible = True
        Beep()

        solanchienthang += 1
        Label5.Text = "Wins: " & solanchienthang
    End If
    Label6.Text = HitRate(solanchienthang, Spins)
End Sub
```

Giá trị của hàm HitRate đã được gán cho thuộc tính Text của nhãn Label6.

4.4.5 Chạy chương trình:

Bạn chọn Save All và chạy bằng cách ấn F5 và theo dõi kết quả.



Hình 51

5. Làm quen với ADO.NET

5.1 Lập trình với ADO.NET

Cơ sở dữ liệu rất quan trọng trong việc lưu trữ thông tin. Dữ liệu có rất nhiều nguồn và đa dạng. VB.NET được thiết kế với mục đích truy xuất, hiển thị, phân tích CSDL. Với ADO.NET, bạn có thể truy xuất đến mọi hệ CSDL theo cùng cách thức và mã chương trình như nhau.

5.1.1 Thuật ngữ về cơ sở dữ liệu (CSDL)

Chúng ta hãy làm quen với một số thuật ngữ về CSDL trước khi thực sự thao tác với nó.

CSDL là một hệ thống dữ liệu được tổ chức thông tin thành các bảng gọi là Table.

Mỗi bảng lại bao gồm nhiều hàng và cột. Cột thường được gọi là trường (field) và dòng được gọi là mẩu tin (record).

Mô hình truy xuất CSDL trong ADO.NET có thể nói như sau: trước hết là thiết lập kết nối đến CSDL. Tiếp theo đối tượng điều phối (data adapter) được tạo ra để truy vấn dữ liệu từ các bảng. Sau đó tạo các đối tượng DataSet chứa bảng dữ liệu bạn muốn trích dữ liệu.

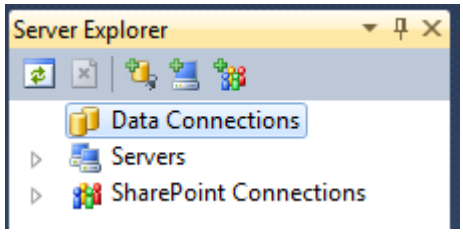
DataSet sẽ tạo bản sao của bảng dữ liệu. Cuối cùng là gán thông tin trong DataSet vào các đối tượng hiển thị trên Form như TextBox, Label, Button, DataGrid,…

5.1.2 Làm việc với cơ sở dữ liệu Access

Trong phần tiếp theo chúng ta sẽ sử dụng Server Explorer để thiết lập kết nối đến CSDL của MS Access có tên Students.mdb. Sau khi đã biết cách kết nối và đưa dữ liệu vào dataset, chúng ta sẽ bắt đầu xây dựng và tích hợp chúng vào giao diện của form.

Bạn tạo mới một Solution có tên MyADOForm và thêm vào một dự án cùng tên.

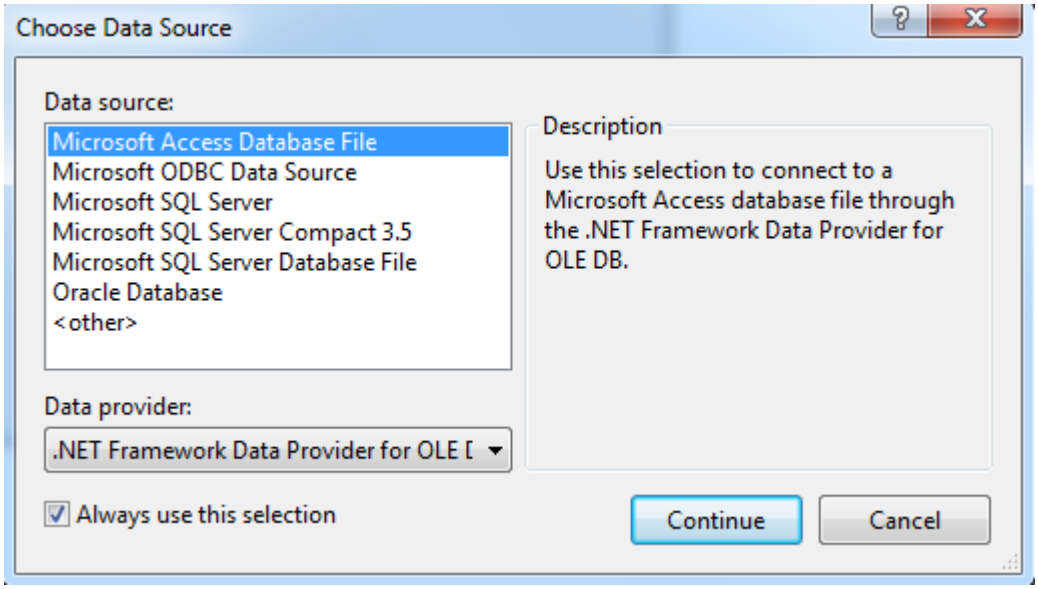
Bạn chọn View | Server Explorer từ menu để hiện cửa sổ Server Explorer như hình:



Hình 52

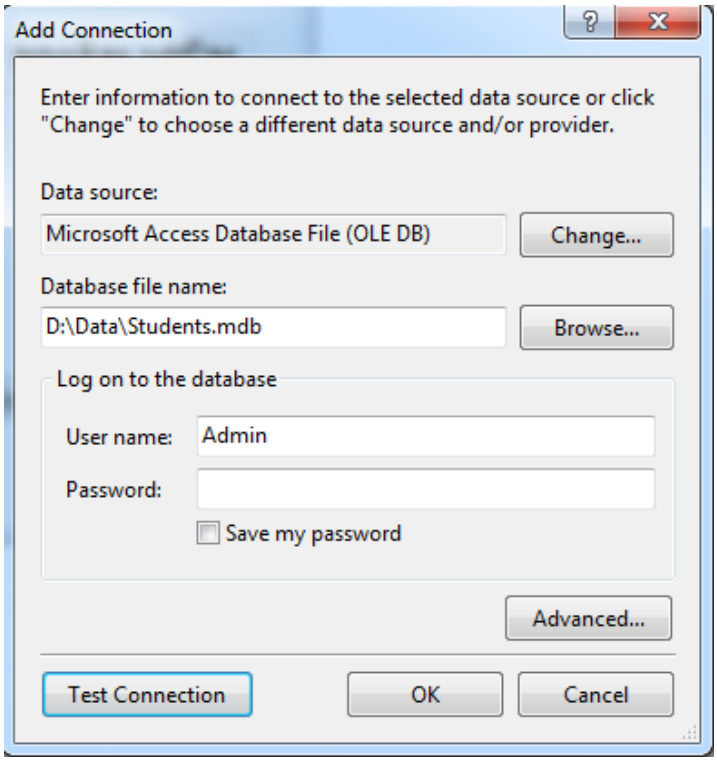
Đây là công cụ đồ họa cho phép kết nối đến CSDL cục bộ, trên server theo mô hình client – server. Ta cũng có thể sử dụng nó để xem cấu trúc trong CSDL, xem thuộc tính của bảng, kiểu dữ liệu của trường và mẩu tin trong CSDL. Chúng ta có thể nắm kéo các kết nối và bảng dữ liệu trong cửa sổ này để tạo ra đối tượng dữ liệu cho chương trình.

Tiếp theo tạo kết nối đến CSDL bằng cách click vào nút Connect To DataBase  trong cửa sổ Server Explorer. Một hộp thoại Choose Data Source hiện ra cho phép ta chọn nguồn dữ liệu như hình:



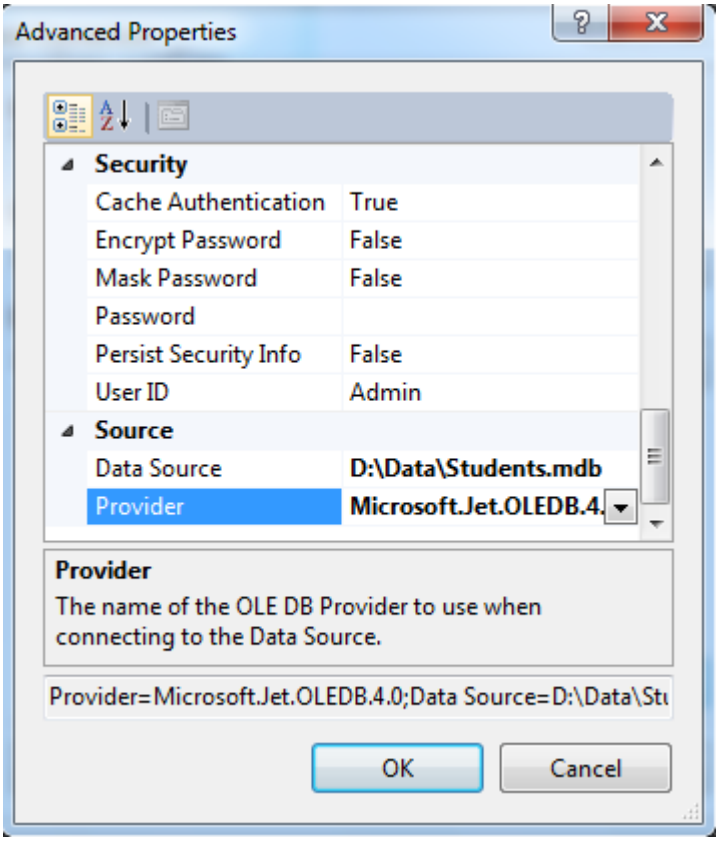
Hình 53

Bạn chọn Microsoft Access DataBase File và nhấn vào nút Continue để làm xuất hiện hộp thoại Add Connection như hình:



Hình 54

Bạn chọn đường dẫn đến CSDL bằng cách nhắp vào nút Browse… và chọn CSDL Students.mdb như hình. Bạn có thể kiểm tra xem kết nối có thành công không bằng cách click vào nút Test Connection, bạn cũng có thể tùy chỉnh kết nối bằng cách click vào nút Advanced:

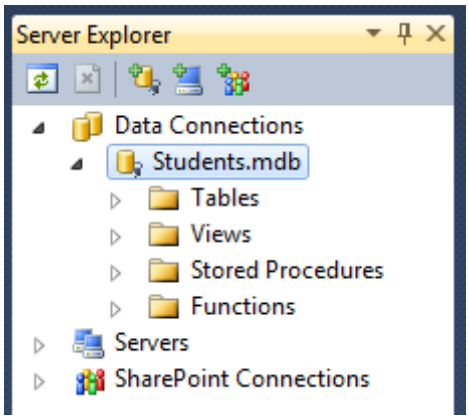


Hình 55

Bạn có thể thấy dòng mã kết nối ở ô cuối cùng như hình, dòng mã có nội dung: “Provider=Microsoft.Jet.OLEDB.4.0;Data Source="D:\Data\Students.mdb"”

Nhấn OK để thêm kết nối vào Server Explorer.

Bạn có thể mở rộng tất cả các mục bằng cách click vào biểu tượng bên cạnh trái của từng mục để mở rộng như hình:



Hình 56

5.1.3 Tạo bộ điều phối dữ liệu Data Adapter

Bước hai trong thao tác CSDL như ta đã biết đó là tạo bộ điều phối Data Adapter. Data Adapter sẽ định nghĩa chính xác những thông tin mà bạn muốn lấy trong CSDL, là nền tảng để tạo DataSet.

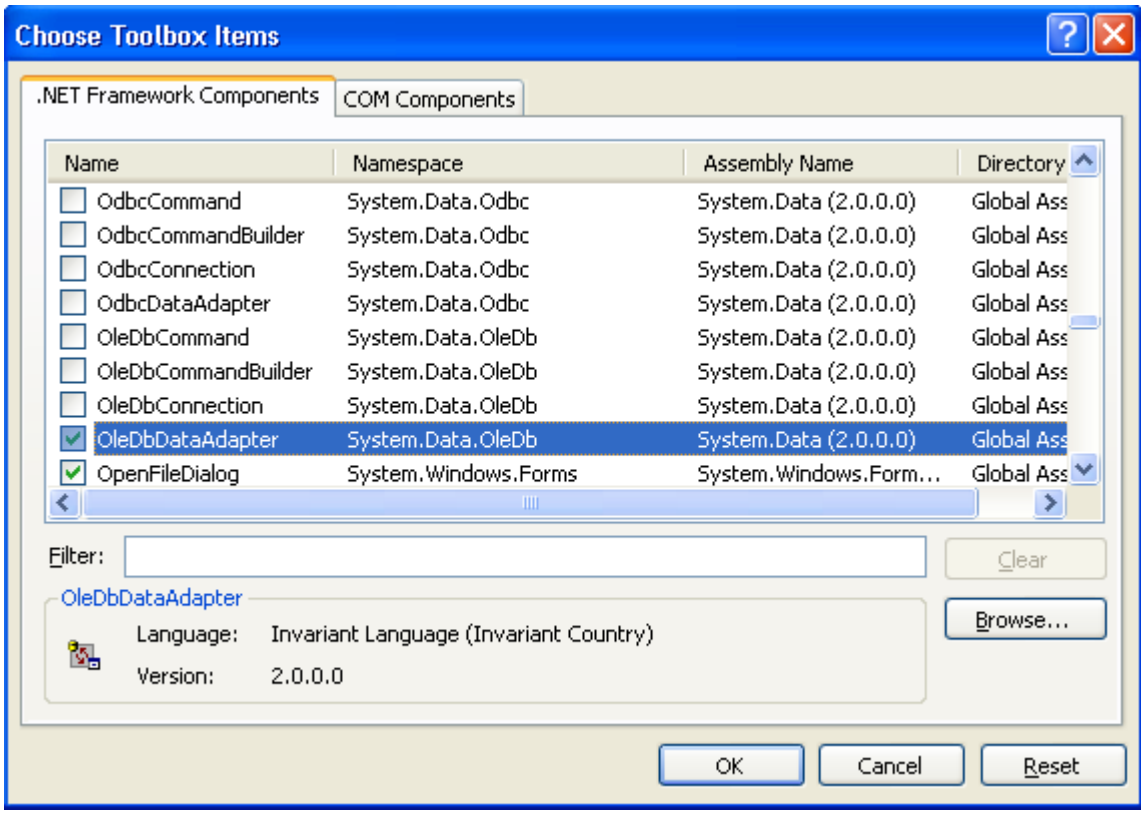
VB.NET cung cấp rất nhiều cách tạo bộ điều phối. Cách đơn giản nhất là ta kéo các biểu tượng bảng trong Server Explorer vào cửa sổ form trong chế độ thiết kế. Ta cũng có cách thứ hai là dùng công cụ Data Adapter Configuration Winzard. Ta gọi đến công cụ này bằng cách chọn đối tượng OledbDataAdapter trên tab Data của ToolBox và đặt nó lên form. Trong bài tập này chúng ta sẽ sử dụng cách thứ hai này.

5.1.4 Sử dụng đối tượng điều khiển OleDbDataAdapter

Chọn tab Data trong cửa sổ ToolBox. Tab này chứa các điều khiển để thao tác với CSDL. Trong tab này có hai đối tượng OleDbConnection và sqlConnection đều cho phép tạo kết nối đến CSDL. Nhưng chúng ta đã kết nối bằng Server Explorer nên không cần hai đối tượng này nữa.

Kéo đối tượng OleDbDataAdapter 

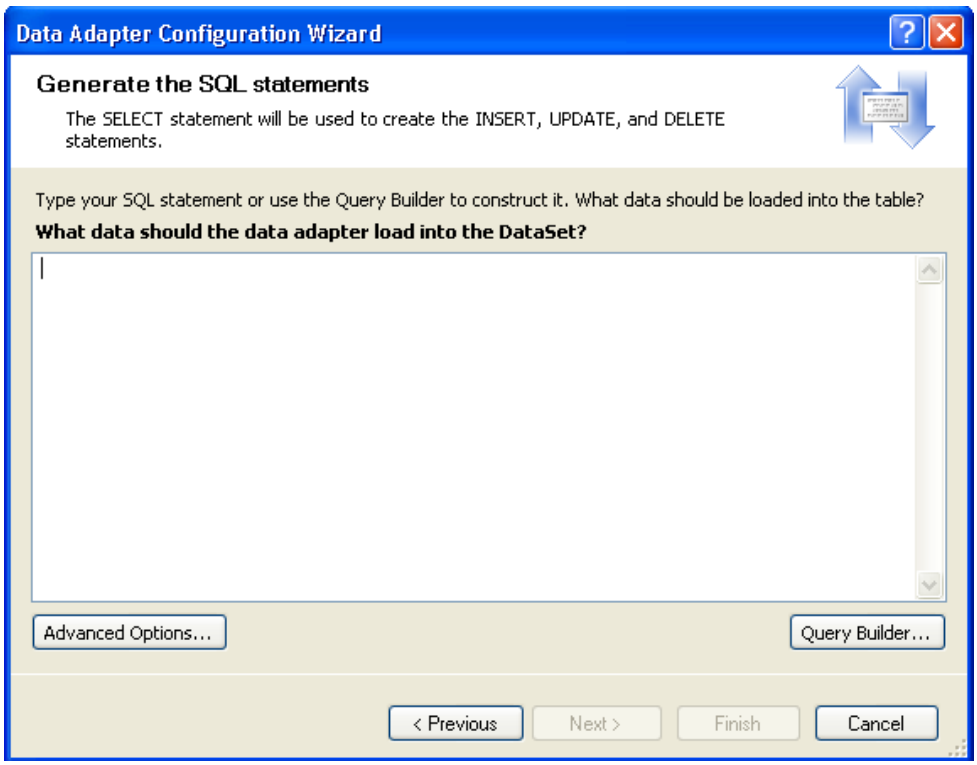
vào trong form. Nếu đối tượng này không xuất hiện, bạn có thể thêm nó vào bằng cách Right-Click vào tab Data chọn Choose Item… để làm xuất hiện cửa sổ Choose ToolBox Items. Chọn tab .Net Framework Components và chọn OleDbAdapter như hình:



Hình 57

Nhắp OK để hoàn thiện việc thêm Item này cho ToolBox. Bạn cũng có thể làm tương tự với các đối tượng khác.

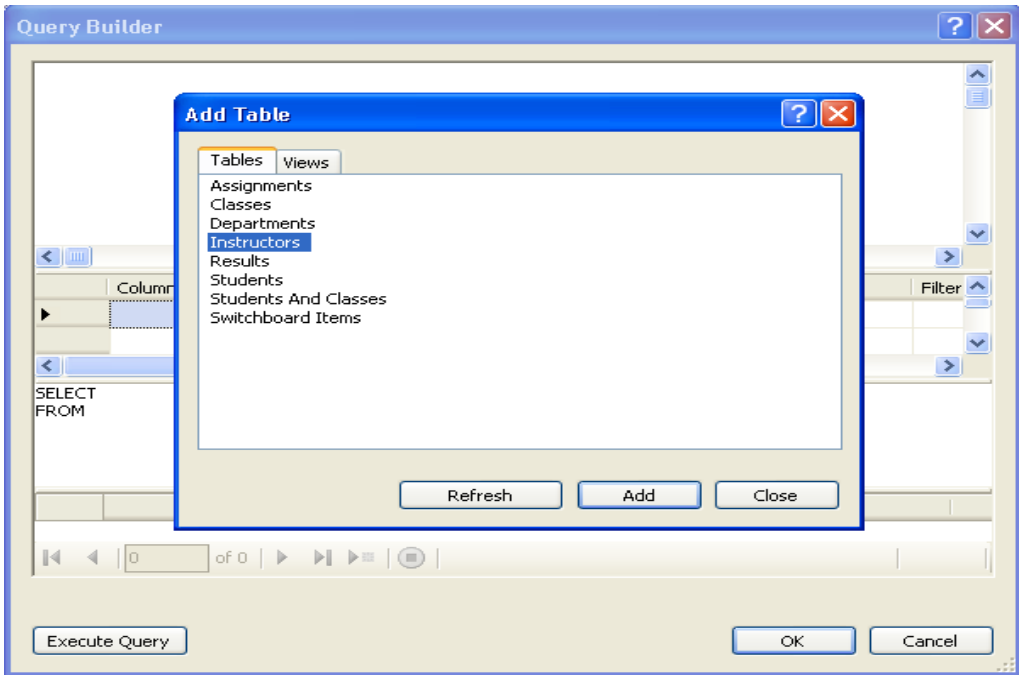
OleDbAdapter được thiết kế để kết nối đến CSDL Access. Khi kéo thả đối tượng này vào form thì VS.NET sẽ tao trình Data Adapter Configuration Winzard. Một màn hình khởi đầu, Bạn nhấn Next hai lần để xuất hiện màn hình soạn thảo câu lệnh SQL như hình dưới.



Hình 58. Cửa sổ soạn thảo mã SQL

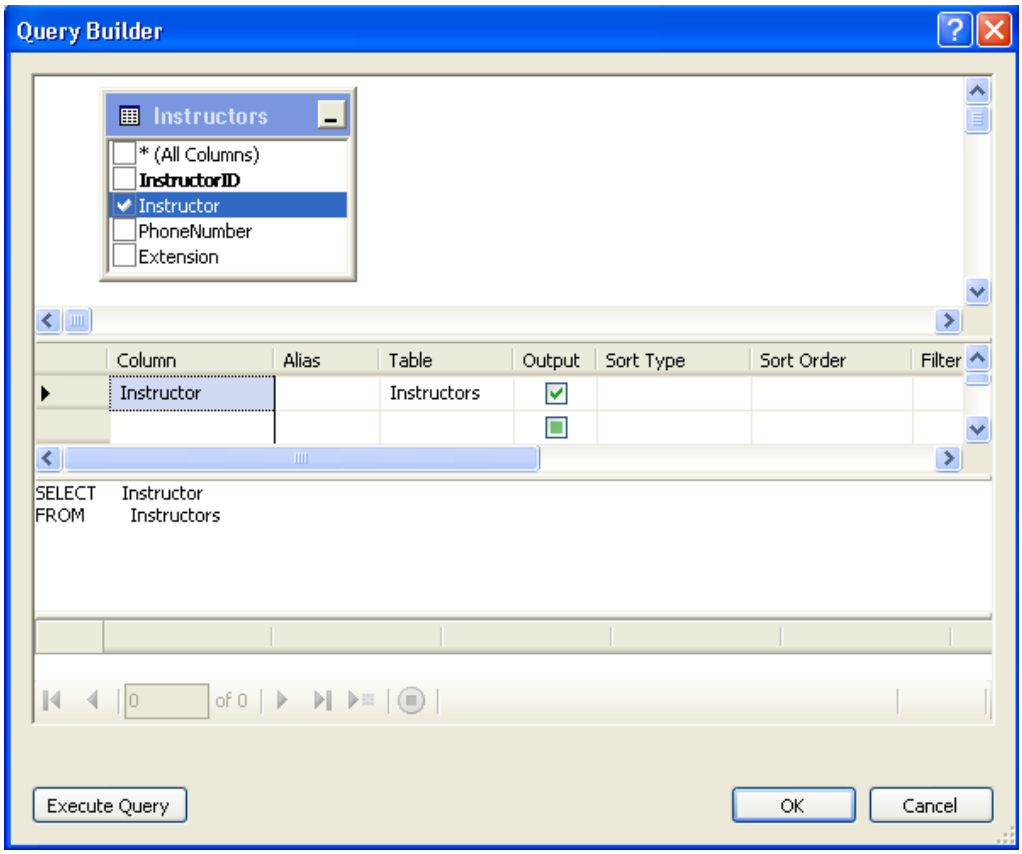
Nếu bạn chưa biết đến các câu lệnh SQL, có thể nhấn vào nút Query Builder… để VS.NET liệt kê các bảng của CSDL để bạn chọn.

Bạn hãy nhấn vào bảng Instructors như hình dưới và nhấn Add, rồi ấn Close để đóng cửa sổ này lại.



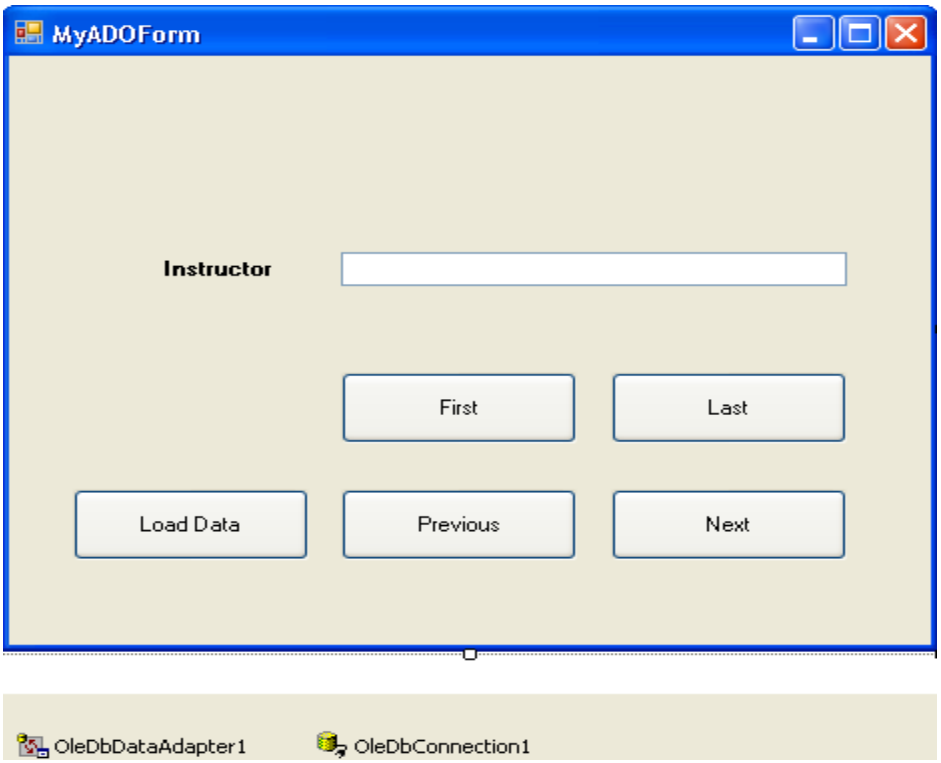
Hình 59. Query Builder

Bạn thấy trong bảng Instructors có các ô CheckBox tương ứng với các trường. Query sẽ tạo câu lệnh tương ứng để rút thông tin của bảng. Trong bài tập này chúng ta chỉ rút thông tin từ một cột trong bảng. Bạn nhấn vào cột Instructor để chọn nó như hình tiếp theo và nhấn OK. Chúng ta đã tạo xong câu lệnh SQL để rút dữ liệu.



Hình 60. Chọn trường để xây dựng câu lệnh SQL

Sau khi nhấn OK, một cửa sổ Generate The SQL Statement hiện ra hiển thị câu lệnh SQL ta vừa tạo. Bạn nhấn Finish đê hoàn thành việc tạo đối tượng điều phối. Lúc này giao diện có dạng như hình sau:



Hình 61. Giao diện Form

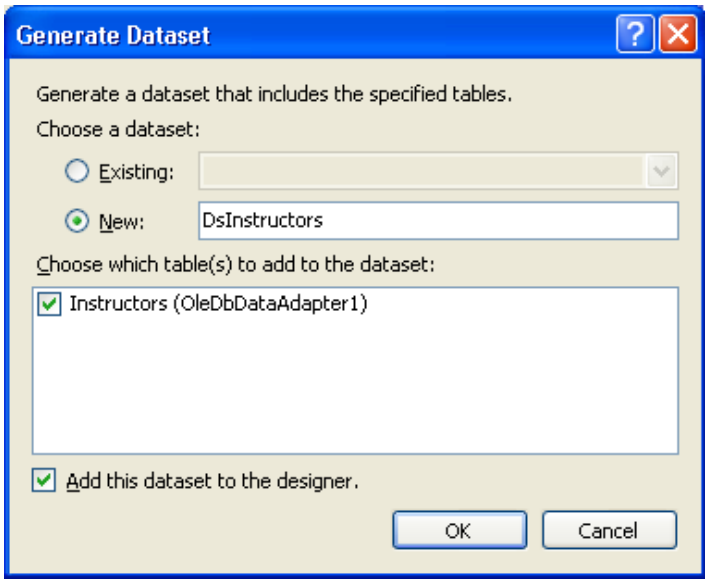
5.1.5 Làm việc với DataSet

Tiếp theo ta tạo ra đối tượng trình diễn dữ liệu cho người dùng thao tác. Đối tượng này là DataSet. Nó là hình ảnh có được từ DataAdapter. Nó chỉ là ảnh của CSDL nên mọi thao tác của người dùng sẽ chưa ảnh hưởng đến CSDL cho đến khi có yêu cầu cập nhật.

Trong phần tiếp theo của bài tập này chúng ta sẽ tạo đối tượng DataSet trình diễn thông tin trong cột Instructor của bảng Instructors trong CSDL Students.mdb.

Bạn nhấp chuột lên form1 để chọn nó. Nếu không chọn nó thì các lệnh tạo DataSet sẽ không hiển thị trên menu.

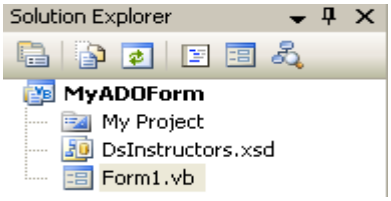
Chọn Data | Generate DataSet từ menu để làm xuất hiện hộp thoại Generate DataSet như hình:



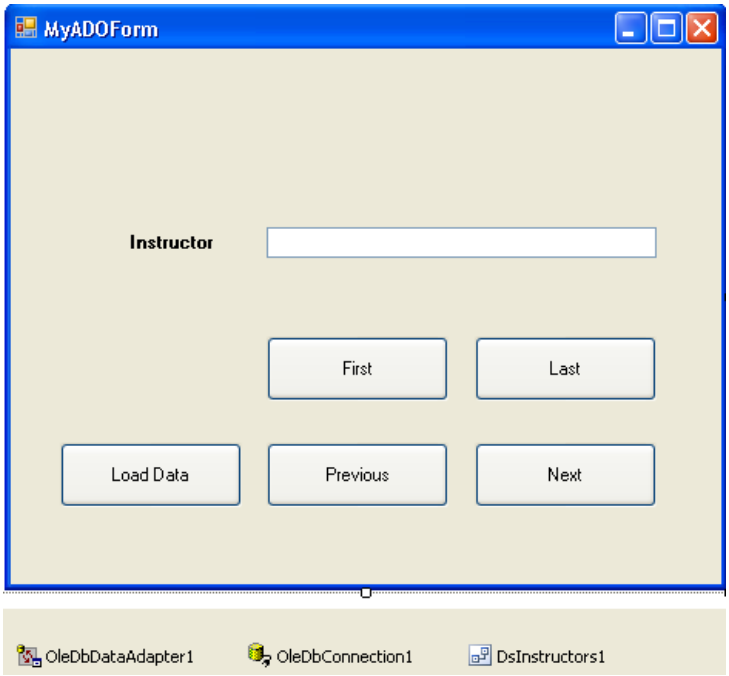
Hình 62

Đặt tên nào tùy thích tại ô New, trong trường hợp này đặt tên là DsInstructors. Chọn ô checkBox Add this dataset to the designer để VS đưa dataset vào khay công cụ.

Nhấn OK và đối tượng DataSet DsINstructors được tạo trên khay công cụ. Lúc này VB.NET sẽ tự thêm vào một file có tên DsInstructors.xsd trong cửa sổ Solution Explorer. File này chứa các thông tin về dữ liệu theo khuôn dạng XML:



Hình 63



Hình 64. Đối tượng DsInstructors1 được tạo trên khay hệ thống

Tiếp theo ta sẽ hiển thị dữ liệu trong dataset này lên form.

5.2 Sử dụng các điều khiển ràng buộc dữ liệu

Tiếp theo ta sẽ dùng các điều khiển quen thuộc như Textbox, Label, Button để trình bày cơ sở dữ liệu lên form. Để trình bày được như thế ta cần phải làm một thao tác gọi là ràng buộc dữ liệu (data binding), nghĩa là dữ liệu hiển thị lên trong các điều khiển sẽ phụ thuộc vào nguồn dữ liệu có trong DataSet hay DataAdapter.

Bạn có thể ràng buộc dữ liệu với các điều khiển sau: TextBox, Label, ListBox, ComboBox, RadioButon, DataGrid và PictureBox. Trong đó đặc biệt và hữu ích nhất có lẽ là DataGrid vì nó cho phép bạn hiển thị toàn bộ nội dung của DataSet.

Trong bài tập này, chúng ta sẽ ràng buộc dữ liệu vào TextBox để hiển thị thông tin trong bảng Instructors của CSDL Students.mdb.

Bạn thiết kế giao diện form như hình trên. Trong đó thuộc tính của các điều khiển như sau:

Button First: Name – btnFirst, enable – False

Button Last: Name – btnLast, enable – False

Button Next: Name – btnNext, enable – False

Button Previous: Name – btnPrevious, enable – False

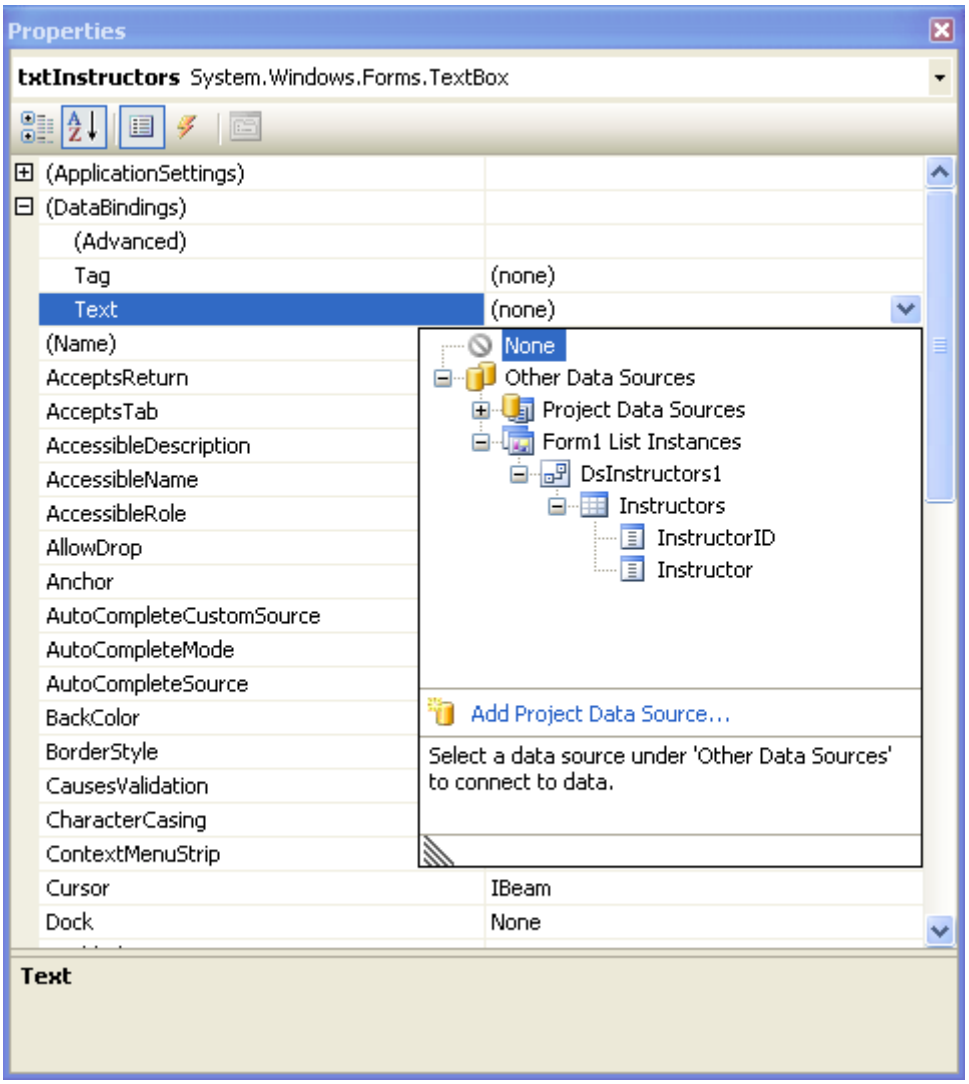
Button Load Data: Name – btnLoadData

TextBox1: Name - txtInstructors

Các điều khiển còn lại có thuộc tính như hình.

Bây giờ ta sẽ tiến hành ràng buộc dữ liệu là các trường (cột dữ liệu – field) vào textbox txtInstructors.

Để làm điều này, bạn chọn ô textbox và mở Properties của nó ra. Click vào biểu tượng bên cạnh nhánh thuộc tính DataBindings và chọn ô text, Click vào nút mũi tên đi xuống chúng ta có thể nhìn thấy nguồn dữ liệu DsInstructors1 hiển thị trong danh sách:



Hình 65

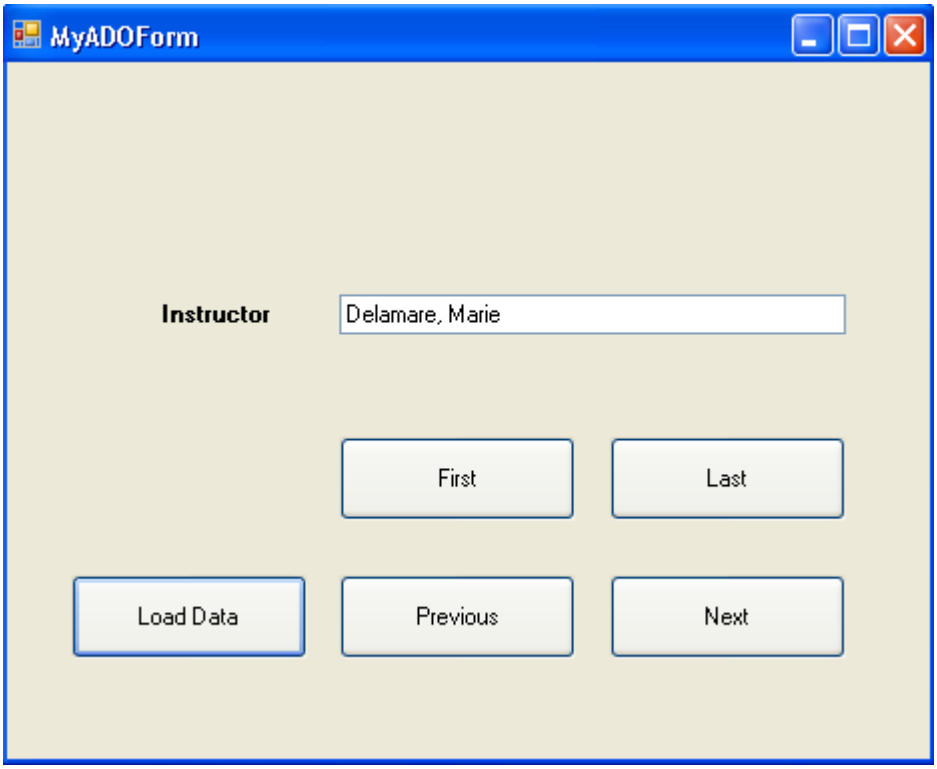
Nhấn chọn cột Instructor để chỉ định trường này sẽ hiển thị trong ô textbox txtInstructor. Như vậy ta đã ràng buộc xong, tiếp theo cần viết mã để xuất dữ liệu khi chương trình thực thi.

Để làm được điều đó, chúng ta tạo thủ tục btnLoadData_Click bằng cách trở lại cửa sổ thiết kế form và double click vào nút Load Data rồi nhập đoạn mã sau:

```
DsInstructors1.Clear()
OleDbDataAdapter1.Fill(DsInstructors1)
btnFirst.Enabled = True
btnLast.Enabled = True
btnNext.Enabled = True
btnPrevious.Enabled = True
```

Chúng ta viết mã để xóa sạch dữ liệu mà DataSet DsInstructors1 nắm giữ trước đây. Tiếp theo khiến bộ điều phối DataAdapter1 điền dữ liệu vào đối tượng DataSet DsInstructors1 mà chúng ta đã tạo ra ở bước 3 bằng phương thức Fill().

Để chạy kiểm thử chương trình chúng ta nhấn phím F5. Khi chương trình chạy, bạn nhắp vào nút Load Data để chương trình hiển thị bản ghi đầu tiên của trường Instructor trong bảng dữ liệu:



Hình 66

Tiếp theo chúng ta sẽ mở rộng một số chức năng khác của ứng dụng cơ sở dữ liệu thuần túy như duyệt qua các bản ghi, đếm và hiển thị số bản ghi hiện hành.

Như vậy quá trình thao tác CSDL có thể tóm tắt như sau: thứ nhất, tạo kết nối đến CSDL cần truy xuất; thứ hai tạo đối tượng điều phối DataAdapter; thứ ba, tạo đối tượng trình diễn DataSet; cuối cùng là ràng buộc dữ liệu vào các điều khiển cho phép ràng buộc.

5.3 Tạo các điều khiển duyệt xem dữ liệu

Trong bài tập này, chúng ta mới chỉ dừng lại ở việc ràng buộc dữ liệu và hiển thị được bản ghi đầu tiên vào ô textbox mà thôi. Trong phần tiếp theo chúng ta sẽ tạo ra các nút cho phép duyệt qua các bản ghi khác nhau, xem bản ghi đầu tiên cũng như cuối cùng.

ADO.NET cho phép quản lý và duyệt qua các bản ghi (record) bằng đối tượng CurrentManager. Với đối tượng này bạn có thể biết được vị trí hiện hành, đi đến mẩu tin sau cùng hay trở về mẩu tin đầu tiên cũng như đến mẩu tin kế tiếp hay ở trước. Mỗi DataSet đều có sẵn đối tượng CurrentManager và mỗi đối tượng form đều có thuộc tính BindingContext theo dõi tất cả đối tượng CurrentManager trên form.

Bây giờ trở lại bài tập của chúng ta. Trong phần trước chúng ta đã tạo ra bốn nút nhấn mang tên First, Last, Next, Previous. Giờ chúng ta sẽ viết mã cho chúng sử dụng đối tượng BindingContext, CurrentManager để duyệt qua các bản ghi.

Trước hết tạo thủ tục btnFirst_Click với nội dung như sau:

```
Me.BindingContext(DsInstructors1, _
"Instructors").Position = 0
btnFirst.Enabled = False
btnNext.Enabled = True
btnLast.Enabled = True
```

Cú pháp này hiển thị bản ghi đầu tiên của DsInstructors1 sử dụng đối tượng BindingContext. Nó gán giá trị 0 cho thuộc tính Position để con trỏ hiện hành của dữ liệu chuyển đến bản ghi đầu tiên.

Tạo thủ tục btnLast_Click và nhập đoạn mã sau:

```
'Đếm tổng số bản ghi
Dim tongsobanghi As Integer = Me.BindingContext _
(DsInstructors1, "Instructors").Count
'Chuyển con trỏ đến bản ghi cuối cùng
Me.BindingContext(DsInstructors1, _
"Instructors").Position = tongsobanghi - 1
btnLast.Enabled = False
btnFirst.Enabled = True
btnPrevious.Enabled = True
btnNext.Enabled = False
```

Tạo thủ tục btnNext_Click và nhập vào đoạn mã sau:

```
'Đếm số bản ghi hiện hành
Dim tongsobanghi As Integer = Me.BindingContext _
(DsInstructors1, "Instructors").Count
'Nếu chưa phải là bản ghi cuối thì next lên 1
```

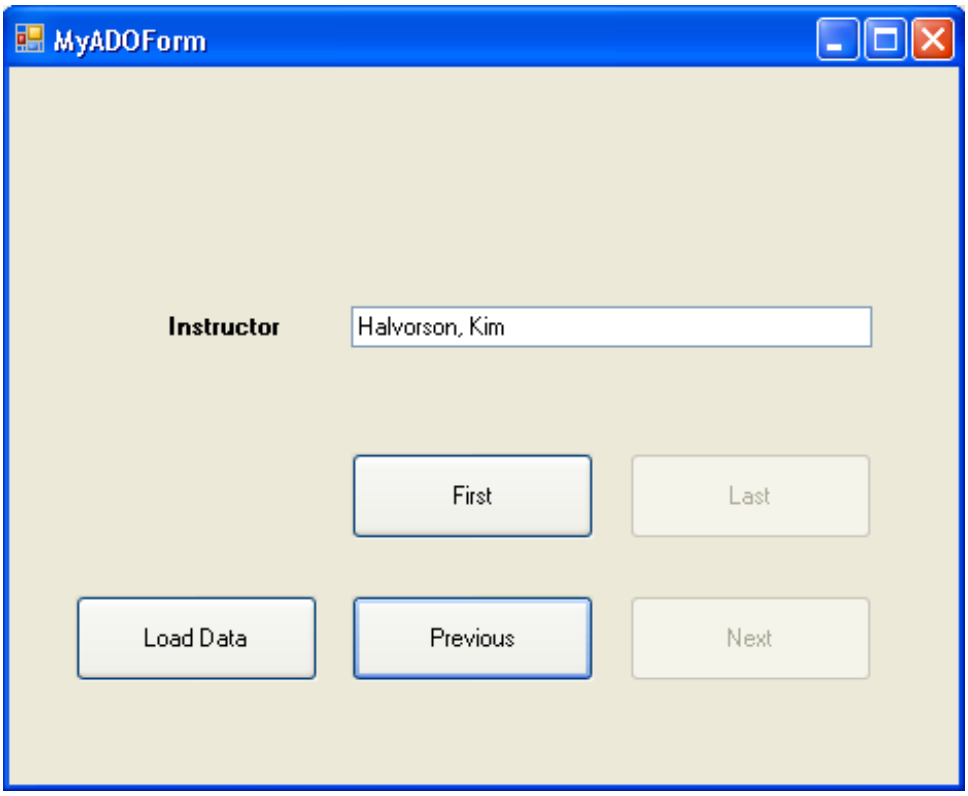
```
        If Me.BindingContext(DsInstructors1, _
        "Instructors").Position < tongsobanghi - 1 Then
            Me.BindingContext(DsInstructors1, _
            "Instructors").Position += 1
            btnFirst.Enabled = True
            btnPrevious.Enabled = True
            btnLast.Enabled = True
        Else
            btnNext.Enabled = False
            btnLast.Enabled = False
            btnFirst.Enabled = True
            btnPrevious.Enabled = True
        End If
```

Thủ tục btnPrevious_Click:

```
        'Nếu chưa phải là bản ghi đầu thì lùi lại 1
        If Me.BindingContext(DsInstructors1, _
        "Instructors").Position > 0 Then
            Me.BindingContext(DsInstructors1, _
            "Instructors").Position -= 1
            btnFirst.Enabled = True
            btnLast.Enabled = True
            btnNext.Enabled = True
        Else
            btnFirst.Enabled = False
            btnPrevious.Enabled = False
        End If
```

Vậy là chúng ta đã tạo xong các nút cho phép duyệt qua các bản ghi. Bây giờ chúng ta chạy thử chương trình.

Bây giờ chung ta nhấn F5 để chạy chương trình, nhấn nút Load Data để hiển thị dữ liệu vào textbox. nhấn các phím để duyệt qua các bản ghi trong cơ sở dữ liệu.



Hình 67

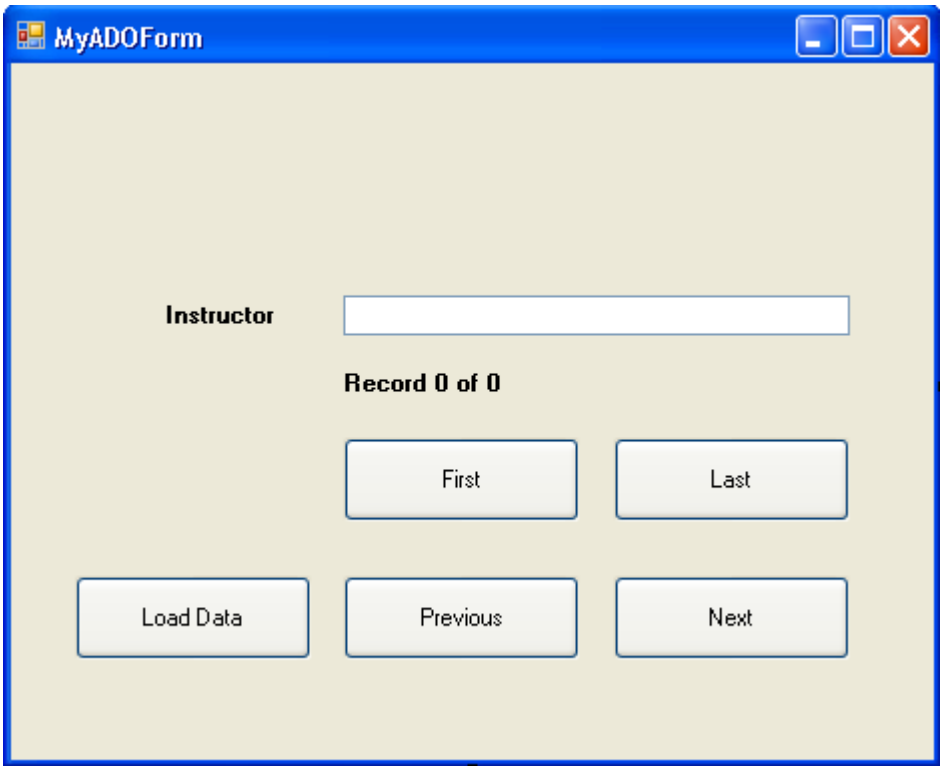
Bạn nhấn nút Close ở góc phải trên của form để đóng chương trình lại.

Để cụ thể hơn nữa chúng ta sẽ tạo điều khiển label cho hiển thị vị trí bản ghi hiện hành để người dùng tiện quan sát.

5.4. Hiển thị vị trí của bản ghi hiện hành

Ngoài việc cung cấp cơ chế duyệt xem các bản ghi, ta cũng cần cho người dùng biết đó là bản ghi thứ mấy. Bây giờ chúng ta sẽ thêm một nhãn Label để hiển thị thứ tự của bản ghi.

Bạn mở thiết kế form và thêm vào một nhãn label1 có thuộc tính Name là lblIndexOfRecord, thuộc tính Text của nhãn là "Record 0 of 0". Giao diện như hình:



Hình 68

Ta tạo một thủ tục có tên count() ở ngay dưới phát biểu khai báo form1 như sau:

```
    Private Sub Count()
        Dim tongsobanghi, banghihienhanh As Integer
        tongsobanghi = Me.BindingContext _
        (DsInstructors1, "Instructors").Count
        banghihienhanh = Me.BindingContext _
        (DsInstructors1, "Instructors").Position + 1
        lblIndexOfRecord.Text = "Record " & _
        banghihienhanh.ToString & "Of " &
tongsobanghi.ToString
    End Sub
```

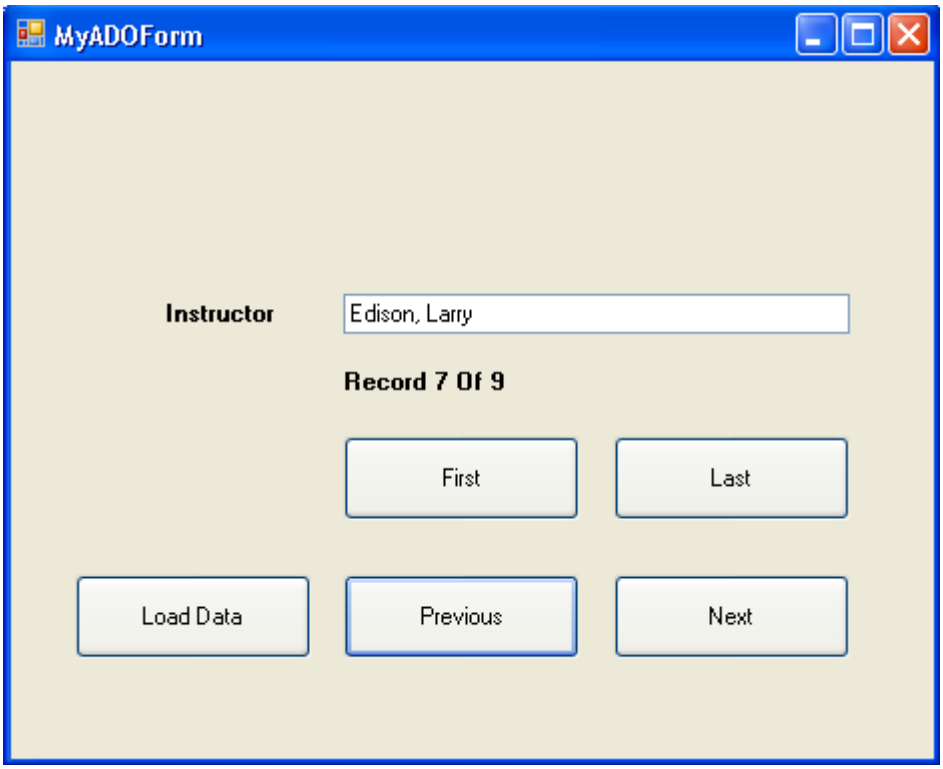
Thủ tục này sẽ gán thuộc tính count của đối tượng BindingContext vào biến *tongsobanghi* và thuộc tính *Position* của nó cho biến *banghihienhanh* nhưng cộng thêm 1 vì thứ tự bản ghi trong bảng dữ liệu được tính từ 0. Sau đó hai giá trị của hai biến trên được gán cho thuộc tính Text của điều khiển Label lblIndexOfRecord.

Để thủ tục này phát huy tác dụng thì bạn sẽ thêm lời gọi thủ tục này trong các thủ tục khác như btnFirst_Click, btnLast_Click, btnPrevious_Click, btnNext_Click như sau:

```
            Count()
```

Chương trình của chúng ta đến đây là hoàn thiện. Bạn có thể chạy thử để kiểm tra.

Nhấn F5 để chạy chương trình, nhấn nút Load Data để hiển thị dữ liệu. Sau đó bạn hãy nhấn các nút di chuyển để duyệt qua các bản ghi và xem thứ tự của bản ghi đó trong bảng dữ liệu, kết quả:



Hình 69

5.5 Trình diễn dữ liệu sử dụng điều khiển DataGrid

5.5.1 Sử dụng DataGrid để hiển thị dữ liệu trong bảng:

Trong phần này chúng ta sẽ dùng DataGrid để hiển thị dữ liệu của bảng trong CSDL Students.mdb. Ta sẽ điền đầy đủ nội dung khung lưới bằng dữ liệu của bảng ở dạng chuỗi sau đó thực hiện một số thao tác định dạng, sắp xếp và ghi lại những thay đổi trong DataGrid trở lại CSDL.

Cũng giống như TextBox, bạn có thể ràng buộc dữ liệu trong DataSet vào DataGrid. Việc ràng buộc này thông qua hai thuộc tính là DataSource và DataMember.

Trong bài tập MyDataGridBinding sau chúng ta sẽ đưa toàn bộ nội dung của bảng Instructors có trong DsInstructors1 hiển thị trong khung lưới DataGrid.

Để minh họa, chúng ta cùng làm một bài tập MyDataGridBinding như sau:

Bước 1: Tạo mới một Solution và thêm vào một dự án cùng tên là MyDataGridBinding.

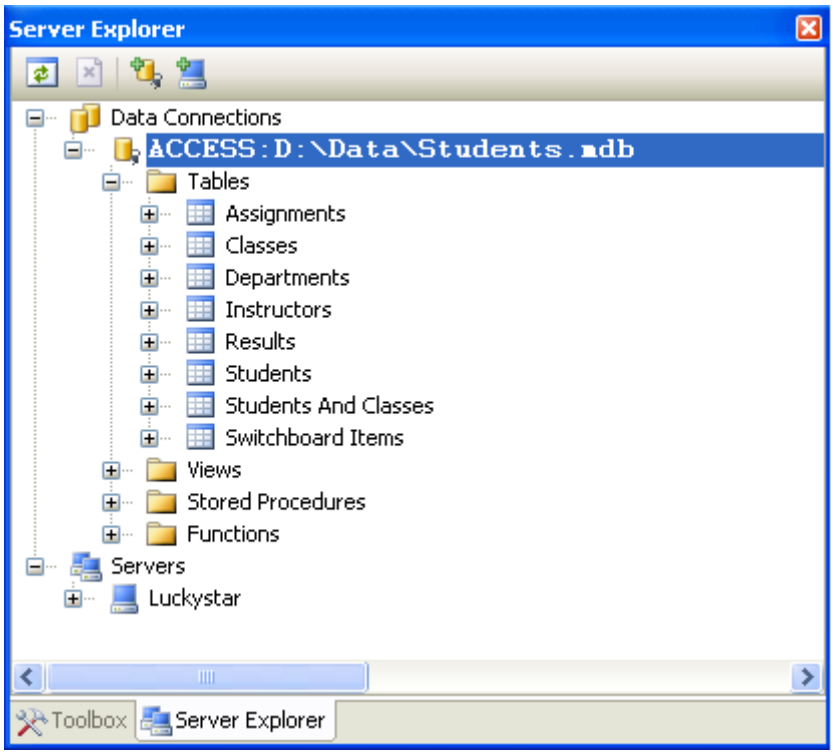
Bước 2: Kết nối CSDL

Nếu trong bài trước chúng ta đã hoàn thành kết nối với CSDL thì bây giờ trong cửa sổ Server Explorer sẽ có một kết nối đến CSDL đó nhưng có thêm một gạch đỏ ở kết nối đó. Nếu muốn sử dụng lại kết nối này bạn chỉ việc ấn vào nút Refresh là xong. Trong bài tập này tôi chép file CSDL Students.mdb vào cùng thư mục với dự án để tiện thao tác.

Bạn chọn nút để thực hiện kết nối đến CSDL như đã biết. Chọn CSDL mà chúng ta vừa chép vào thư mục chứa dự án.

Nhấn OK để hoàn thành kết nối.

Bạn có thể xem chi tiết các bảng có trong CSDL này bằng cửa sổ Server Explorer:



Hình 70

Bước 3: Tạo đối tượng điều phối DataAdapter:

Bạn tạo thêm đối tượng OleDbDataAdapter vào trong form bằng cách kéo nó từ ToolBox ở tab data vào trong form. Khi đó một cửa sổ Data Adapter Configuration xuất hiện.

Nhấn Next hai lần để hiện cửa sổ Generate SQL Statements. Tại đây bạn có thể tự gõ câu lệnh SQL hay sử dụng nút nhấn Query Builder… Ở đây mình dùng cách nhập trực tiếp câu lệnh SQL. Bạn nhập câu lệnh sau:

```
SELECT    Extension, PhoneNumber, Instructor, InstructorID
FROM      Instructors
```

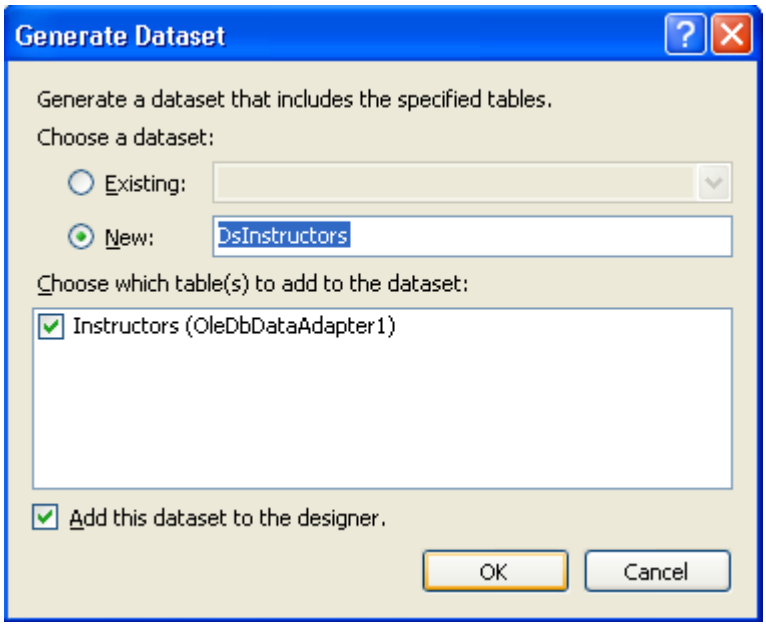
Phát biểu này sẽ trích rút dữ liệu ở cả bốn trường trong bảng Instructors. Bạn nhấn Next để xem kết quả của Winzard. Lúc này, trình Winzard tự tạo ra các câu lệnh khác là Update (cập nhật), Select, Insert (chèn), Delete (xóa).

Nhấn Finish để kết thúc quá trình xây dựng tạo đối tượng điều phối DataAdapter có tên OleDbDataAdapter1.

Bước 4: Tạo đối tượng trình diễn DataSet

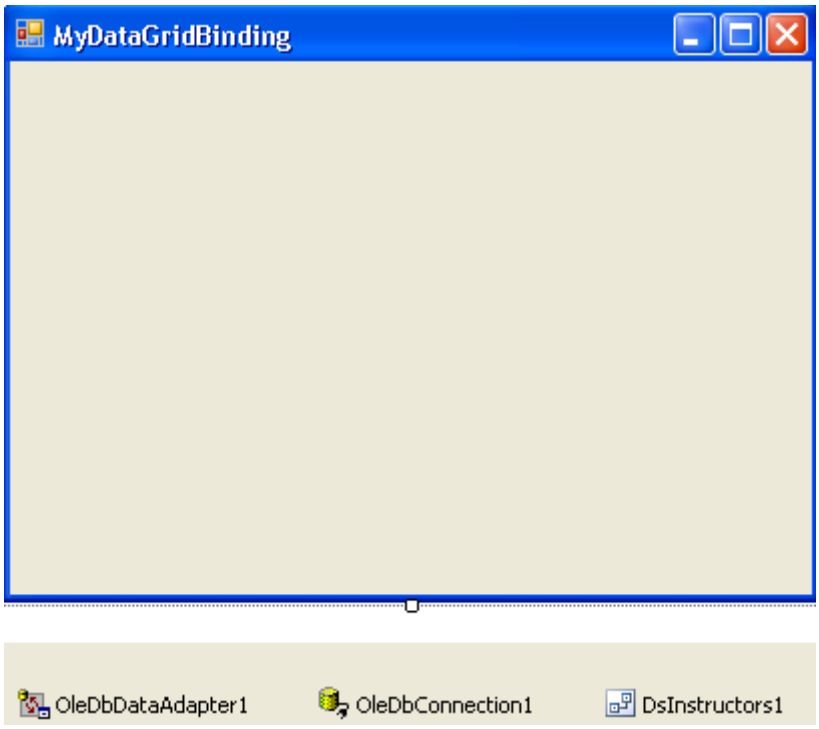
Nhấn Form để chọn nó.

Chọn Data | Generate DataSet từ menu làm hiện hộp thoại Generate DataSet như đã biết. Tại ô New bạn nhập vào tên DsInstructors và đánh dấu vào ô checkBox Add this DataSet To The Designer để VS tạo ra đối tượng DataSet và đưa nó vào khay hệ thống như hình:



Hình 71

Nhấn OK để VS tạo đối tượng DataSet cho bảng Instructors trong CSDL Students.mdb. Lúc này cửa sổ form có thêm các đối tượng như hình:



Hình 72

Chúng ta đã hoàn thành ba bước đầu của thao tác với CSDL. Bây giờ chúng ta sử dụng DataGrid để trình bày dữ liệu.

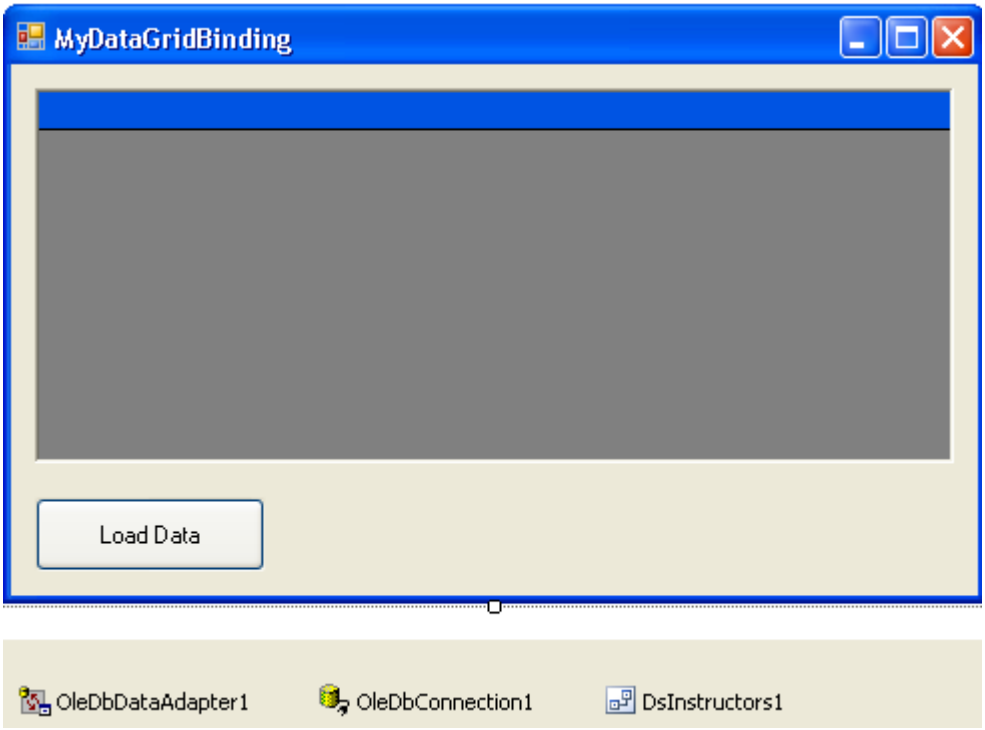
Bước 5: Tạo đối tượng DataGrid

Kéo form cho kích thước rộng ra để chứa đủ khung lưới DataGrid với 4 cột và 10 dòng.

Đưa điều khiển DataGrid DataGrid trên ToolBox vào trong form. Kéo chiều dài của nó cho phù hợp với chiều kích thước của form.

Tạo thêm một nút nhấn nữa vào form. Đặt thuộc tính Name là btnLoad và text là "Load Data".

Mở Properties của DataGrid và đặt thuộc tính Anchor của nó là cả Left, Right, Top, Bottom. Giao diện của form lúc này như hình:



Hình 73

Tiếp theo ta sẽ dùng thuộc tính DataSource và DataMember để ràng buộc dữ liệu trong DsInstructors1 vào khung lưới DataGrid.

Bạn cho hiển thị các tùy chọn của thuộc tính DataSource trong cửa sổ Properties. Một chương trình có thể có rất nhiều DataSet nhưng tại một thời điểm khung lưới chỉ có thể thể hiện một DataSet mà thôi. Bạn chọn DsInstructors1 như hình H.1.

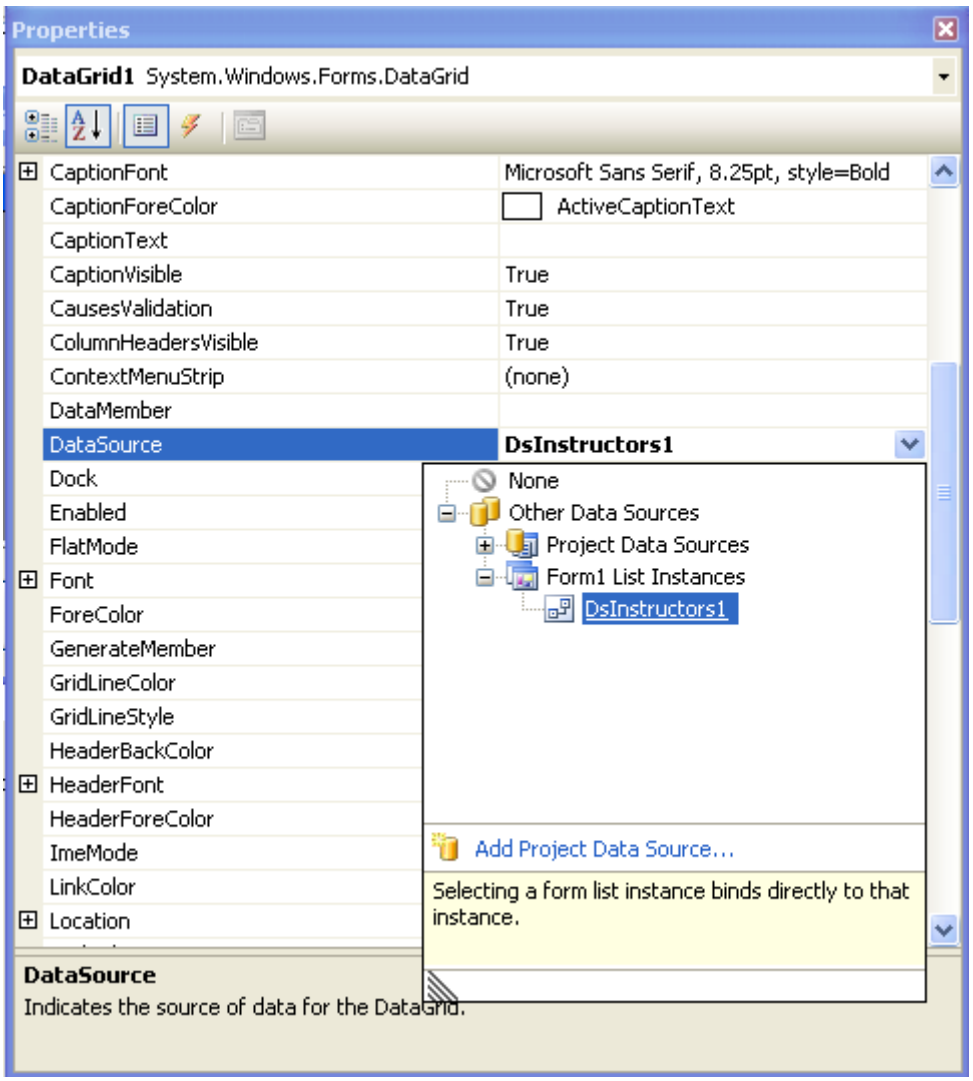
Tiếp theo bạn chọn thuộc tính DataMember là Instructors như hình H.2.

Ngay sau khi bạn chọn xong hai thuộc tính DataSource và DataMember thì khung lưới sẽ hiển thị các cột dữ liệu dù chưa có dòng dữ liệu nào hiển thị. Dữ liệu sẽ được đưa vào khung lưới khi chương trình thực thi.

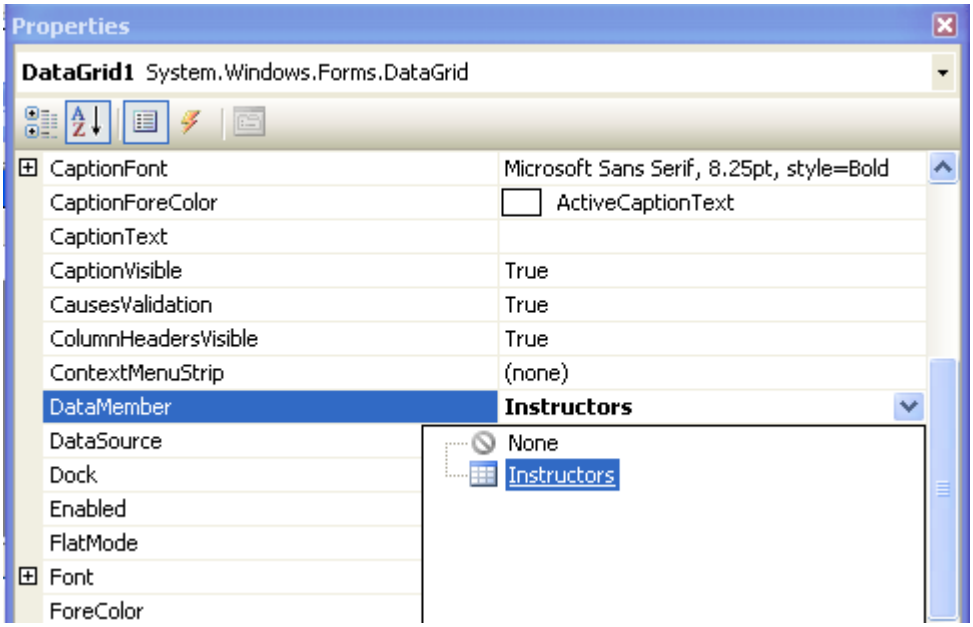
Bạn chọn nút Load Data và đặt thuộc tính Anchor của nó là Bottom, Left.

Lúc này giao diện form thiết kế sẽ như hình H.3.

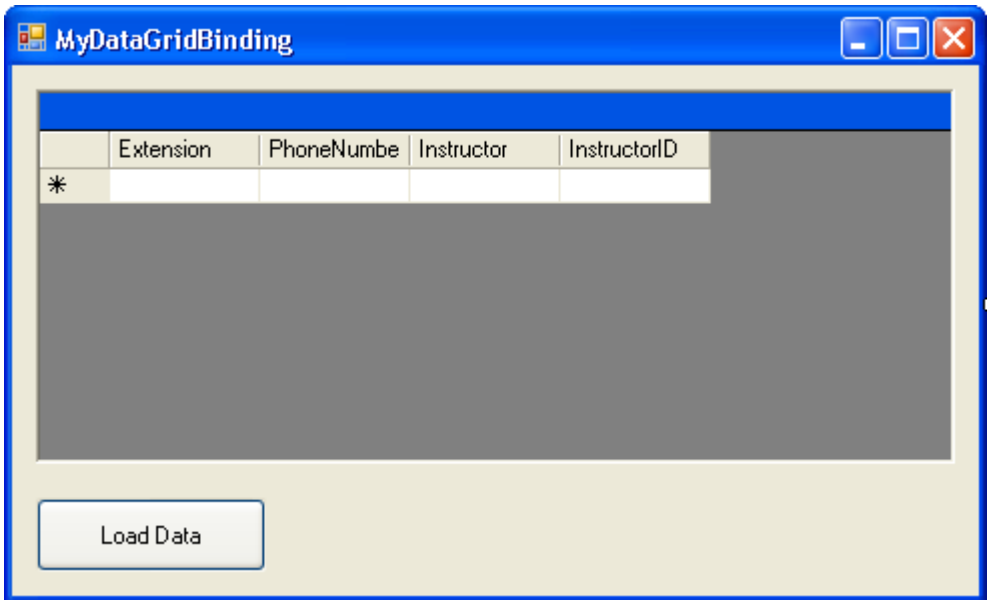
Tiếp theo chúng ta cần viết mã để đổ dữ liệu vào khung lưới bằng phương thức Fill như bạn đã biết trong chương trước.



Hình 74. Chọn DsInstructors1 cho thuộc tính DataSource



Hình 75. Chọn Instructors cho thuộc tính DataMember



Hình 76. Cửa sổ form khi thiết kế xong

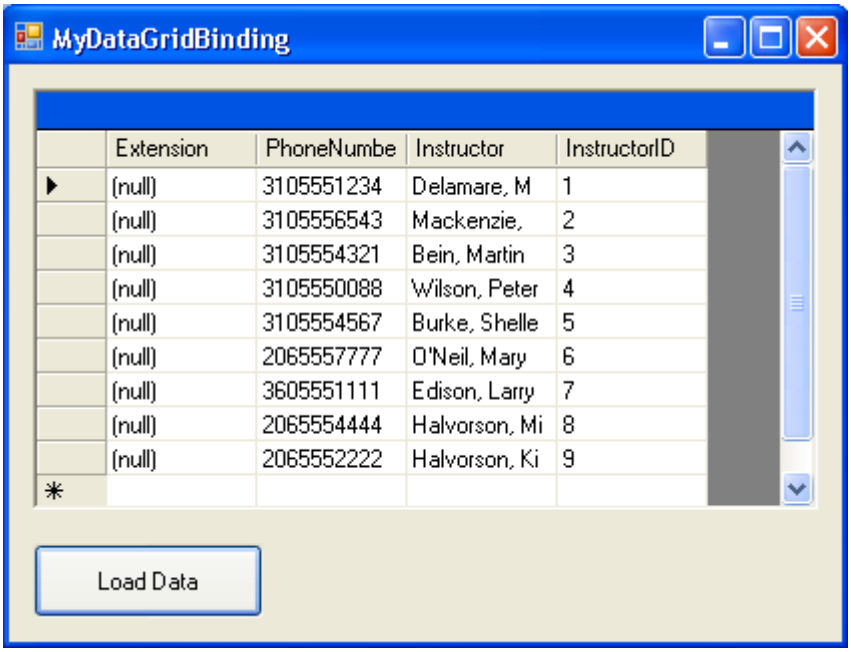
Tạo thủ tục btnLoad_Click bằng cách double click vào nó và nhập đoạn mã sau:

```
DsInstructors1.Clear()
    OleDbDataAdapter1.Fill(DsInstructors1)
```

Nhấn nút Save All để lưu lại các thay đổi và chạy thử chương trình.

Bước 6: Chạy chương trình

Nhấn nút F5 để chạy chương trình. Nhấn nút Load Data để nạp dữ liệu vào trong khung lưới DataGrid:



Hình 77

Bạn có thể kéo để thay đổi kích thước form sao cho các thông tin về CSDL xuất hiện đầy đủ. Bạn cũng có thể sắp xếp dữ liệu trong khung lưới bằng cách click vào tiêu đề một cột nào đó. Nhấn nút Close để đóng chương trình.

5.5.2 Định dạng các ô lưới trong DataGrid

Bạn có thể định dạng các thành phần trong DataGrid thông qua thuộc tính của nó lúc thiết kế hay khi thực thi chương trình.. Chúng ta sẽ làm điều này với bài tập trên.

Bạn trở lại cửa sổ thiết kế form và mở thuộc tính Properties của khung lưới DataGrid.

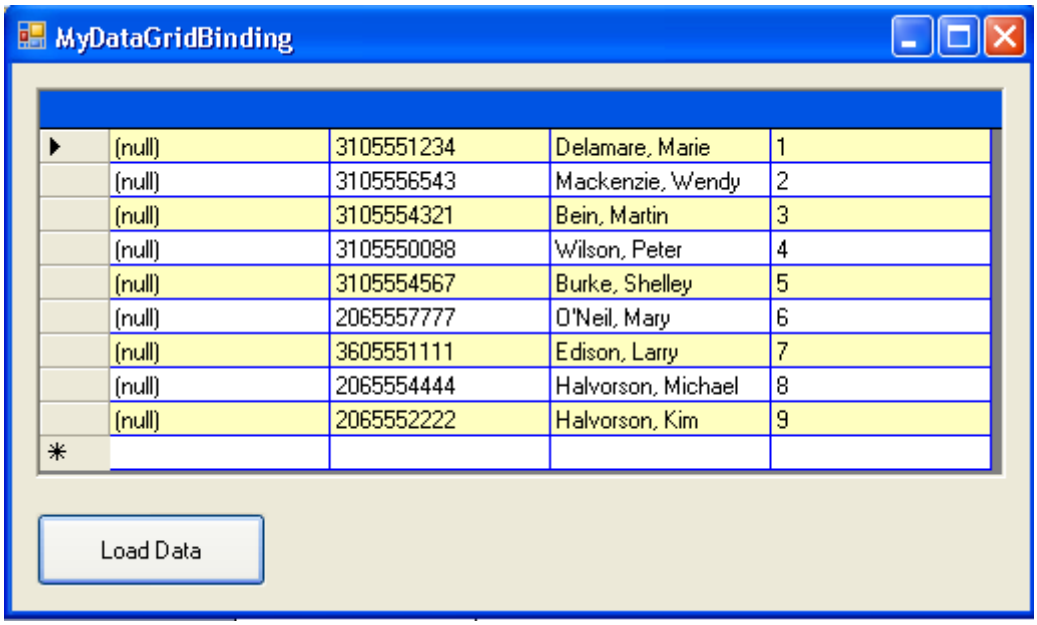
Đặt thuộc tính PreferredColumnWidth là 110 (rộng 110 đơn vị đo Pixel).

Bạn đặt thuộc tính ColumnHeadersVisible là False. Với thiết lập này thì phần tiêu đề của các cột sẽ không hiển thị.

Nhấn chọn thuộc tính BackColor, chọn màu vàng nhạt hiển thị cho nội dung chuỗi chứa trong ô lưới tạo các dòng xen kẽ nhau.

Đặt thuộc tính GridLineColor, chọn màu xanh.

Còn rất nhiều thuộc tính khác bạn có thể tìm hiểu thêm. Giờ bạn hãy chạy chương trình để xem những thay đổi:



Hình 78

5.5.3 Cập nhật cơ sở dữ liệu trở lại bảng

DataSet chỉ tiến hành sao chép bảng của CSDL chứ không làm thay đổi nội dung CSDL cho đến khi có yêu cầu cập nhật bằng phương thức Update. Cùng với thuộc tính ReadOnly của DataSet sẽ cho phép có thay đổi hay không với CSDL.

Bây giờ chúng ta sẽ tiến hành tìm hiểu những điều đó.

Trở lại cửa sổ thiết kế form và mở thuộc tính properties của DataGrid và thiết lập giá trị TRUE đối với thuộc tính ReadOnly cho phép có những thay đổi dữ liệu trong khung lưới.

Tiến hành đặt một nút nhấn nữa lên form. Thuộc tính như sau: Name – btnUpdate, Text – "Update".

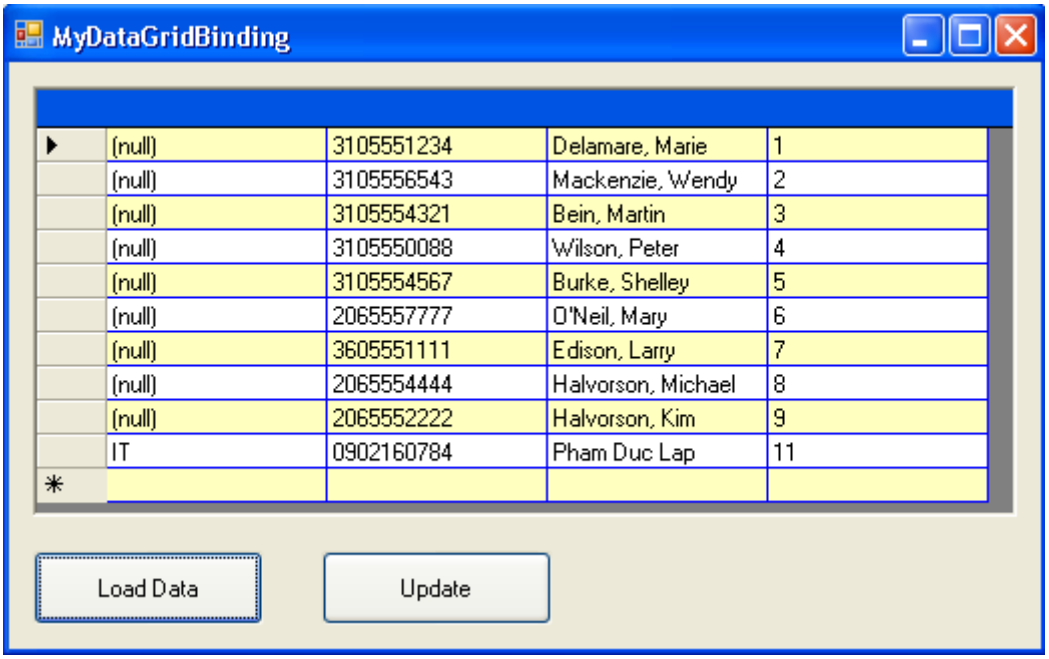
Nút nhấn Update sẽ hiển thị khi có những thay đổi trong DataGrid và tiến hành cập nhật trở lại cơ sở dữ liệu khi người dùng click vào nó.

Tạo thủ tục btnUpdate_Click và nhập nội dung như sau:

```
Try
     OleDbDataAdapter1.Update(DsInstructors1)
Catch ex As Exception
     MsgBox(ex.ToString)
End Try
```

Thủ tục này sử dụng phương thức Update của OleDbDataAdapter1 để yêu cầu các thay đổi trong tập DataSet DsInstructors1 trở lại bảng CSDL.

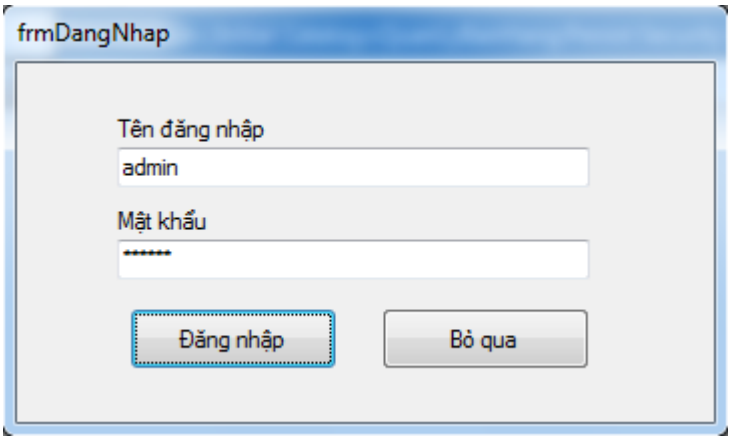
Nhấn F5 để chạy chương trình. Bạn thay đổi nội dung một cột nào đó hay có thể thêm một bản ghi nữa và click vào nút Update để cập nhật vào CSDL. Sau đó lại click vào nút Load Data để xem CSDL có thay đổi gì không.



Hình 79

6. Bài tập

Bài 1. Tạo một form đăng nhập có giao diện như sau:



Hình 80

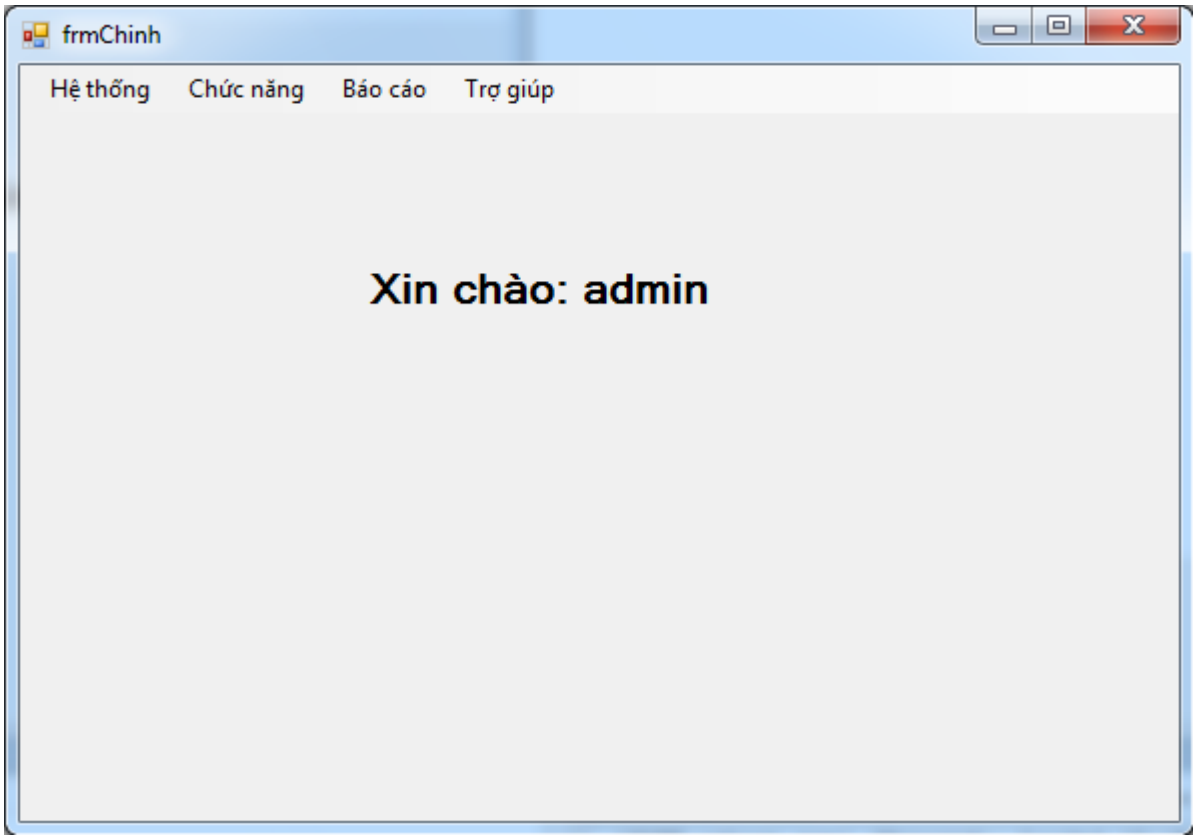
Yêu cầu:

+ Khi người đăng nhập đúng với tên và mật khẩu của một trong hai thông tin sau:

Tên đăng nhập	Mật khẩu
admin	123456
Nam	123

thì chương trình sẽ cho phép mở form chính. Ngược lại thông báo tên đăng nhập hoặc mật khẩu không hợp lệ.

+ Khi vào form chính, giữa form sẽ xuất hiện dòng chữ: Xin chào: + "tên đăng nhập" như hình dưới đây:



Hình 81

Bài 2. Tạo cơ sở dữ liệu QuanLyVatTu.mdb bằng Microsoft Access gồm các bảng sau và thiết lập quan hệ cho các bảng:

NHACUNGCAP

Tên trường	Diễn giải
MaNCC	Mã nhà cung cấp
TenNCC	Tên nhà cung cấp
DiaChiNCC	Địa chỉ nhà cung cấp
SDT	Số điện thoại
SoFax	Số Fax

VATTU

Tên trường	Diễn giải
MaVatTu	Mã vật tư
TenVatTu	Tên vật tư
QuyCach	Quy cách
DVT	Đơn vị tính
MaNCC	Mã nhà cung cấp

a) Thiết kế form nạp danh sách nhà cung cấp bằng công cụ DataGridview.

b) Thiết kế form sử dụng các TextBox để hiển thị thông tin của một mẫu tin trong bảng VATTU. Trên form có các chức năng (nút lệnh) di chuyển đến mẫu tin đầu tiên, mẫu tin đứng trước, mẫu tin đứng sau và mẫu tin cuối cùng.

Tài liệu tham khảo:

[1] Đoàn Văn Ban, Phân tích - thiết kế và lập trình hướng đối tượng, NXB Thống kê, 1997.

[2] Nguyễn Ngọc Bình Phương, Các giải pháp lập trình Visual basic. Net, NXB Giao thông vận tải, 2006.

[3] Phạm Hữu Khang, Ví dụ và bài tập Visual Basic.Net: Lập trình windows form và tập tin, NXB Lao động xã hội, 2006.

[4] John Connell, Coding Techniques for Microsoft Visual Basic .NET, 2012.

[5] Microsoft Visual Studio 2010 Documentation, Copyright © 2010 by Microsoft Corporation.

[6] Website http://duriangroup.wordpress.com, Lập trình với Mô hình 3 lớp (3 layers) – n-tiers , 3-tiers, multi tiers, 2013.

[7] Website http://congnghephanmem.vn, Lập trình hiển thị và quản lý các Form Con trong From Cha, 2013.